Learning to Extract Entities from Labeled and Unlabeled Text

Rosie Jones

May 2005

CMU-LTI-05-191

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213

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

This research was supported in part by the National Science Foundation under LIS Grant REC-9720374 and by DARPA under research contract F30602-97-1-0215.

Keywords: Information Extraction, Machine Learning, Semi-supervised Learning, Graph Properties of Labeled and Unlabeled Data

Abstract

Imagine trying to build a system to identify *people*, *locations* and *organizations*, or other arbitrary types, in a human language you are not familiar with. If we knew what kinds of words represent the classes *people*, *locations* and *organizations*, by examining enough text data they occur in, we could learn to recognize the contexts they occur in. And if we knew what kind of contexts they occur in, we could recognize instances of these classes themselves. In this work we address this chicken-and-egg problem by assigning it to a computer, and giving it a small number of examples of the class as initial examples to learn from. We explore several algorithms in which alternating looking at noun-phrases and their local contexts allows us to learn to recognize members of a semantic class in context. We examine active learning algorithms for eliciting useful labels from an expert to improve learning performance, customized to this domain. Finally we explore the graph structure of the underlying labeled and unlabeled data, showing how properties of this graph structure explain performance and inform design choices we have to make when applying these methods to new tasks.

Acknowledgements

I would like to thank my mentors: my advisor at Carnegie Mellon, Tom Mitchell, and my manager at Yahoo!, Dan Fain, for all they have taught me about research. I also owe a great deal to my co-authors and collaborators Rayid Ghani, Ellen Riloff and Dunja Mladenic. Thanks to my committee members Alex Hauptmann and Roni Rosenfeld for making suggestions to improve the thesis, and thanks for the valuable input on the thesis I received from Fernando Diaz, Rayid Ghani, Seán Slattery, Dave Hershberger, Chris Leger and Omid Madani. I also benefitted from two *coadvising* groups, in which students discuss the state of their theses with one another. Seán Slattery, Joseph O'Sullivan and Belinda Thom formed the core of one, while LTI students Laura Mayfield Tomokiyo, Greg Aist, Jade Goldstein, Yan Qu, Klaus Zechner and Kathy Baker formed another. Thanks to Tom Pierce, Jacob Sisk and Benjamin Rey for providing me with seed-words at short notice. I also appreciate general advice, encouragement and support I've had along the way, from Phyllis Reuther, Richard Plunkett, Orna Raz, Nathaniel Daw, Francisco Pereira, Paul Bennett and Paul Ogilvie. My various Pittsburgh hosts: Rachel Collins, Jimi Shanahan, Ann Funge, Kathrin Probst and Guy Lebanon also helped me get from there to here. And I'd like to thank Paul O'Gorman and Cathy Jones for everything.

Contents

List of Tables

List of Figures

List of Algorithms

Chapter 1

Introduction

1.1 Problem Description

Imagine being able to find all of the `organizations` mentioned in a document, together with their `locations`, and the `people` who are involved in the organizations. Tasks of this type are known as *information extraction* problems. The type of information to be extracted is defined in advance, and the goal is to extract the information from new, previously unseen documents.

In this dissertation we describe approaches for training such information extractors, using as input example words which may belong to the target class, and a collection of unlabeled text documents. We give more details about this task in Chapter 2.

1.2 Thesis Statement

The thesis of this research is that we can *efficiently* automate information extraction, that is, learn from tens of examples of labeled training data instead of requiring thousands, by exploiting redundancy and separability of the two features: (1) *noun-phrases* and (2) *contexts*. We exploit this redundancy and separability in two ways: (1) in the algorithms for learning semantic classes, and (2) in novel algorithms for active learning, leading to better extractors for a given amount of user labeling effort.

1.3 Summary of Results

Our goal is to learn to extract noun-phrases of particular semantic types or classes from sentences in text documents. We perform in-depth experiments with identifying the three semantic classes `locations`, `people`, and `organizations` in sentences in text documents. We represent instances of semantic classes with an ordered pair, consisting of a noun-phrase and the local lexico-syntactic context. We identify these noun-phrases and contexts in documents using Sundance, a shallow parser. We take a small set of words which the user believes may be examples of the target class (which we will call *seeds*), and use them to perform *weak labeling* (providing noisy/partial labels to a relatively small number of the examples) on a collection of documents, by identifying any noun-phrase containing those words as a positive example. We weakly label all other examples as negative examples, and perform *semi-supervised learning* with this weakly labeled data.

We describe the algorithms *metabootstrapping*, *coEM* and *EM* and show how they are applicable to semi-supervised learning of this information extraction task. We find that *cotraining* does not work well using our representation on our task. We break the algorithms down into those that employ a separation of the feature sets, and those that combine the feature sets. We show that performance is affected by the set of seeds chosen, with more frequently occurring seeds leading to better extraction performance. However, correcting errors in the weak labeling performed with seeds does not lead to substantial performance improvement. We also show that we should use stopwords as part of the model, for best performance across classes. More details are given in Chapter 3.

Given that our goal is optimizing the trade-off between user training time and algorithm performance, it is important to see how we can improve results with active learning. We describe novel active learning algorithms which are algorithmically coupled with the separable feature sets. In addition, we describe a novel labeling technique, *single feature-set labeling*. In *single feature-set labeling*, the user sees only a partial example, for example the noun-phrase in isolation from the context. We then use the labeled noun-phrase to label all contexts it cooccurs with. This technique provides for economy in labeling, and we show that in some cases it is more effective than standard whole example labeling.

The novel active learning algorithms are *feature-set disagreement*, in which we select examples for labeling when the features disagree strongly on the target label (that is, when the noun-phrase and context disagree on the target label), and *context disagreement*, when we select noun-phrases for single feature-set labeling when their contexts disagree (for example, the noun-phrase occurs with several different contexts, which differ in their confidence of representing a positive example). We also compare against selecting the most frequent examples, which can be important when some classes are represented well by pronouns. Overall we show that judicious selection of examples for labeling can lead to greatly increased accuracy, without greatly increasing the burden on the user. In particular, we show that using feature set redundancy allows selection of examples for labeling which are much more effective than examples chosen randomly, or without use of feature set redundancy. In addition, these results show that active learning can compensate for a bad choice of initial seeds and that the labeling effort is better spent *during* the active learning process rather than at the beginning. More details of these experiments can be found in Chapter 4.

Finally, we perform a deeper analysis of the results. We measure properties of the noun-phrase context connectivity graph, and show that it exhibits small-world graph structure rather than random graph structure. We analyze how this explains the failure of cotraining on our tasks, and how pronouns and certain very common nouns form the hubs of the connectivity graph. We measure the mutual information between noun-phrases and contexts in each class, in order to test our conditional independence hypothesis. We also perform Spearman rank correlation tests over multiple experiments, finding correlations between algorithm breakeven point and features including the number of contexts labeled by initial seeds and the percent of examples labeled positive in active learning. These features correlated with learning performance can help us pinpoint the important properties of active learning and bootstrapping algorithms for information extraction. Comparing these across classes also highlights the different desiderata for active learning algorithms for classes with sparse feature sets and extremely small priors. More detail of this analysis are given in Chapter 5.

1.4 Contributions

The contributions of this research are

- In-depth experiments with bootstrapping algorithms across multiple semantic classes.

- Adaptation of existing semi-supervised learning algorithms for the task of information extraction.
- Novel active learning algorithms that take into account the feature set split into two sets.
- Analysis of the noun-phrase context co-occurrence graph to show that it exhibits small-world and power-law structure.
- Demonstration of the correlation between graph features and algorithm performance.
- Suggestions for seed selection for bootstrapping algorithms informed by the graph structure of the data.
- Suggestions for active learning for bootstrapping algorithms informed by the graph structure of the data.

1.5 Dissertation Roadmap

In Chapter 2 we give background to terminology used in this dissertation, describe the kind of machine learning we perform, and define the information extraction problem in more detail. In Chapter 3 we describe algorithms for learning to perform this information extraction task, as well as describing the data we use. We make explicit the assumptions these algorithms make, and measure some properties of the data to see how well these assumptions hold. We give results of the bootstrapping algorithms on our task. In Chapter 4 we describe active learning extensions to the algorithms described in Chapter 3. In Chapter 5 we analyze our results in terms of the underlying graph structure of the data. In Chapter 6 we describe an experiment learning to extract a new class, using some of the insights gained throughout this dissertation. In Chapter 7 we summarize our conclusions and give suggestions for future work.

Chapter 2

Background

In this chapter we provide background for this dissertation. First we define some linguistic syntactic categories. Then we introduce the data representation, in which semantic classes are represented by an ordered pair, consisting of a noun-phrase and the local syntactic context. We then describe Sundance, the parser used to extract noun-phrases and syntactic contexts from documents. We give some background on work in machine learning on supervised, unsupervised, and semi-supervised learning, and define weak labeling, which we will employ in this dissertation. We then give a general overview of the information extraction task we are tackling in this dissertation, which is learning to extract noun-phrases of particular semantic types from sentences in text documents.

2.1 Linguistic Terminology

In this section we give a brief introduction to some linguistic terminology that will be used throughout this dissertation. The general method of describing linguistic syntactic categories used here is derived from the descriptions in Radford (1988). In general it is difficult to give precise definitions of linguistic syntactic categories, as there will be exceptions, but the descriptions below should give sufficient detail for someone fluent in English to recognize the category.

2.1.1 Noun

A noun is, roughly speaking, a word which describes an object or thing, such as “cat”, “computer” and “city”, as well as more abstract things, such as “love”. Morphologically, nouns are words which become plural by adding ‘s’ (apart from a few exceptions). They can appear as the first word in sentences like ‘X is fun’ or ‘The X is fun’.

2.1.2 Noun-phrase

Frequently nouns do not occur on their own, but as part of longer groups of words called noun-phrases. A noun-phrase can be modified with the possessive ‘s’, as in “this food is **the fat cat**’s” or “this mouse is **the cat in the hat**’s. Noun-phrases can be replaced by pronouns in sentences, as in “I really like **the job I am working in now**”, becoming “I really like **it**”. Noun-phrase is abbreviated *NP*.

2.1.3 Head of noun-phrase

In this dissertation we will often refer to the “head” of a noun-phrase. The *head* of a noun-phrase is the noun which contains the core meaning of the phrase. Other words can be deleted from the phrase and the sentence will still make sense, but the head-noun is the most essential word. Some examples of noun-phrases, with the head shown in bold, are below:

- the **job** that I am working in now
- the fat **cat**
- a few of my favorite **things**
- the **person** who I met last week
- a shiny new telephone **handset**
- **dogs** in coats

If we remove relative clauses, which start with words such as “that” and “who”, and prepositional phrases, which start with words like “in” and “of”, the head of a noun-phrase is generally the right-most word in English. For example if we take the noun-phrase "the person who I met last week", and remove the phrasal modifier "who I met last week", we are left with "the person". The right-most word "person" is the head of the noun-phrase.

2.1.4 Verb

A verb can be described as a "doing, being or having" word. Examples of verbs include "eat", "read", "are", "buy", "think", "give" and "keep". In English, verbs change form according to whether the action or event they refer to is in the past, present or future. For example in the present we would say "I write", whereas to refer to the past we say "I wrote", and to refer to the future we say "I will write".

2.1.5 Verb Phrase

A verb phrase contains a verb, plus any modifiers. A verb phrase may also contain a noun-phrase. For example the verb phrase "will eat the pistachios quickly" contains:

- a modal verb "will"
- the main verb "eat"
- the object of the verb "the pistachios" (which is also a noun-phrase)
- an adverb "quickly".

Generally the noun-phrase which is the subject of a sentence (the person or thing "doing" the action described) is not part of a verb phrase, but the noun-phrase which is the object of the sentence is part of the verb-phrase.

In this dissertation we will be concerned primarily with the modal + verb parts of verb phrases, for example "will eat", which we will describe as "eat (future tense, active verb)". We will ignore any adverbs.

2.1.6 Prepositional Phrase

A prepositional phrase is used to situate an object or situation in space or time or manner, and is introduced with a preposition. Table 2.1 gives examples of common

about, above, across, after, against, along, amidst, among, around, as, at, before, behind, below, beneath, beside, between, beyond, by, despite, down, during, except, for, from, in, inside, into, like, near, of, off, on, onto, opposite, out, outside, over, past, since, through, till, to, throughout, towards, under, underneath, unlike, until, up, upon, via, with, within, without

Table 2.1: Common prepositions in English

English prepositions.

The other part of a prepositional phrase is a noun-phrase. Examples of prepositional phrases include:

- in a week,
- with a hammer,
- under the table,
- for my family,
- near the pandas.

2.1.7 Lexico-syntactic Context

Syntactic information in language refers to grammatical properties, for example whether a word is a noun or a verb, as described above. It can also refer to whether a noun is used as the subject of a sentence, or the direct object, and whether a verb is in the past or present tense, or in the active or passive voice.

Lexical information refers to the properties of words themselves, both in isolation, and how they are used. Lexical information may be among the information listed for a word in a dictionary. For example the word "dogs" has the property that it refers to the species canine, and that it commonly occurs with "bark".

A lexico-syntactic context is a combination of syntactic and lexical information. An example of a lexico-syntactic context is "ate$_{activeVerb}X_{<dobj>}$". Here we have specified the context of something "X", we know that X is the direct object of an active verb (the syntactic part of the context) and that the active verb it is the direct object of is "ate" (the lexical part of the context).

2.2 Sundance and Autoslog

We use text which has syntactic categories assigned by Sundance. Sundance is a robust heuristic-based shallow parser produced at the University of Utah by Ellen Riloff and her students. Sundance identifies noun-phrases, verb phrases and prepositional phrases within sentences. Autoslog (Riloff, 1993) uses the syntactic categories identified by Sundance to recognize lexico-syntactic patterns. For example, Autoslog can identify patterns of the form "X read a book", or "the book was read by X", and may also use semantic constraints on the noun-phrase filling the slot "X". We will describe how we use Autoslog to preprocess sentences in text documents to give us lexico-syntactic contexts in Chapter 3 Section 3.2.2. Another freely available parser one could use for this kind of task is the Link Grammar Parser (Sleator & Temperley, 1993).

2.3 Supervision in Machine Learning

One way of characterizing approaches to machine learning is by the degree of *supervision* required by the approach. In *supervised learning*, the algorithm is provided with a label for every example, and uses this information to learn a mapping from examples to labels. In *unsupervised learning*, no labels are provided at all. Instead, the algorithm sorts the data into related clusters, based on measures of proximity on the example features. In *semi-supervised learning*, either some examples are provided without labels, or only approximate labels are provided. The algorithm iteratively uses those labels and the data to learn approximate models, which are used to relabel and relearn better models. In the following subsections we describe each of these paradigms of supervision, then explain *weak* labeling, the method of providing supervision for semi-supervised learning that we use in this dissertation.

We will assume that algorithms are provided with a set of *examples* $\{x_1 \ldots x_n\}$ drawn from the universe of possible examples $\mathcal{X}$ according to some distribution $P(\mathcal{X})$. Each example may be associated with a label from the universe of possible labels $\mathcal{Y}$, giving us pairs $< x, y >$. Our goal is to learn a function

$$f : \mathcal{X} \to \mathcal{Y}$$

by using the information provided in the set of training examples. When the set of labels $\mathcal{Y}$ is discrete and finite we call it the set of *target classes*.

Each example x_i has one or more properties, which we will call *features*. These features describe the properties of the examples, and can be used in learning as predictors of the target class or label. We will describe the features particular to our data and representation in Section 3.2.

While in general we assume that the labels $y_{l_1}..y_{l_n}$ represent an accurate representation of the function we wish to learn, in some cases they may be *noisy*, ie the label we see (y_{l_i}) may not be the true label (y_{t_i}) that is part of the function we wish to learn. This may be because of imperfect sensors reading the label, errors by human labelers, or errors introduced in the labeling processes in other ways.

2.3.1 Supervised learning

Supervised learning is the process of learning a mapping from examples to labels, by inferring models from sets of *labeled* examples provided as input. Examples are drawn from a distribution over the set of possible examples $\mathcal{X}$, and their labels are from the set $\mathcal{Y}$. A labeled example is the pair $< x, y >$. The training set is then a set of n examples from the space $\mathcal{X} \times \mathcal{Y}$, which we will denote $\{< x_1, y_1 >, \ldots < x_n, y_n >\}$.

A good overview of supervised learning can be found in (Mitchell, 1997).

2.3.2 Unsupervised Learning using Clustering

One form of unsupervised learning, also known as clustering, is the task of discovering groups underlying examples. In this case we have examples from the universe $\mathcal{X}$ which we wish to assign to a finite set of discrete groups. We do not provide an explicit description of which groups to use, but there is an implicit assumption that there exist coherent groups within the data, which will correspond well to potentially useful categories. The typical approach is to use a clustering algorithm and distance metric which is used to compare examples. The output of clustering is a grouping of examples into sets, which we call *clusters*. We may not know in advance how many clusters are inherent in the data. The way we characterize the examples, and the distance metrics we provide over those features, can have a strong impact on the actual clusters discovered (Kamvar et al., 2002). A good introduction to methods for clustering for natural language tasks can be found in (Manning & Schutze, 1999).

2.3.3 Semi-supervised Learning

Semi-supervised learning is based on the assumption that we can learn classifiers using a small number of labeled examples, together with unlabeled data. That is, we have one set of examples with labels:

$$\mathcal{L} = \{< x_{l_1}, y_{l_1} > \ldots < x_{l_n}, y_{l_n} >\}$$

and another set of examples without labels:

$$\mathcal{U} = \{x_{u_1} \ldots x_{u_m}\}.$$

We can use the examples with labels $\mathcal{L}$ to learn something about the target function $f : \mathcal{X} \rightarrow \mathcal{Y}$. Those examples without labels can be used to learn the relationships between examples, either by helping us understand the distribution of examples $P(\mathcal{X})$ or the relationships between features of the examples. This technique has been used successfully in document classification with naive Bayes and EM (Nigam et al., 1998), and support vector machines (Joachims, 2001). It has also proved effective in learning named entity classifiers (Collins & Singer, 1999).

Weak Labeling

In machine learning, labels are typically assigned to examples explicitly, either because they are measured along with example features, or because a human has examined the example and assigned the label. In the text classification domain, a document which is a conference paper may have the label of the conference proceedings it appeared in, for example *Proceedings of the International Conference of Machine Learning*. Alternatively, a human may inspect the paper and assign a label corresponding to the subject matter of the paper, for example "machine learning". In *weak labeling* labels are derived from a separate source and assigned to examples automatically. In the text classification example, weak labels may be assigned by automatically labeling documents as belonging to the class "machine learning" if they contain both words "machine" and "learning". Note that this may leave many examples unlabeled, which belong to this class, or assign examples to classes they do not belong to. This can lead to a larger proportion of noisy or incorrect labels. Thus we can summarize weak labeling as providing a learning algorithm with data which is (1) noisily labeled (2) only partially labeled and (3) possibly a small number of labeled examples.

Supervised machine learning examples often assume that there are regularities between the features and labels, which allow generalization for learning. In semi-supervised learning, we often assume that there are regularities between the features in examples, which provide sufficient information to overcome the fact that we have labels for only some of the instances. In the weak learning case, with very few, very noisy labels, the structure inherent in the data will need to supply information not only for missing labels, but also for incorrect labels. Weak labeling can be extremely useful if it permits us to supply a training signal without inspecting the data at all. For example, some research used a small set of words as a source of labels (Jones et al., 1999). Documents or phrases containing those words were labeled as positive, without any human inspection. Other work in weak labeling has been done for learning hidden-Markov models for information extraction from document headers, by using lists of names (Seymore et al., 1999). Bockhorst and Craven (2002) learn gene terminators using weakly labeled examples. Liu et al. (2002) perform partially supervised text classification by assuming all unlabeled examples are negative, then inserting known positive examples into the negative class as "spies" to help identify true positive examples in the unlabeled data, and learn appropriate thresholds for distinguishing between the classes.

In this dissertation, we used weakly labeled examples which are pairs of noun-phrases combined with their lexico-syntactic contexts. We will describe exactly how we perform this weak labeling in Chapter 3 Section 3.3. We then perform semi-supervised learning using those examples, and the remaining unlabeled examples.

2.4 The Information Extraction Problem

In this section we describe the information extraction problem we are addressing. This problem is common to all chapters of the dissertation, with details of training information and datasets varying from chapter to chapter.

The information extraction problem we address is that of identifying noun-phrases of a particular semantic class, in their contexts in sentences. This can be a part of a larger information extraction system, composed of many phases. We will now describe each of those phases, then return to define the narrower scope we address in this dissertation.

Information extraction can be divided into a number of subtasks, as described in

(Cardie, 1997) and (Appelt & Israel, 1999). These subtasks are illustrated in Figure 2.1 and consist of (i) identifying classes of semantically related words and phrases (semantic class finding) (ii) labeling those classes with a name that is meaningful to the user (semantic class labeling) (iii) semantic relation labeling (labeling relationships between individual semantic entities) and (iv) anaphora resolution (determining when an object referred to in one way at one point of the text is the same one referred to elsewhere).

2.4.1 Semantic Class Finding

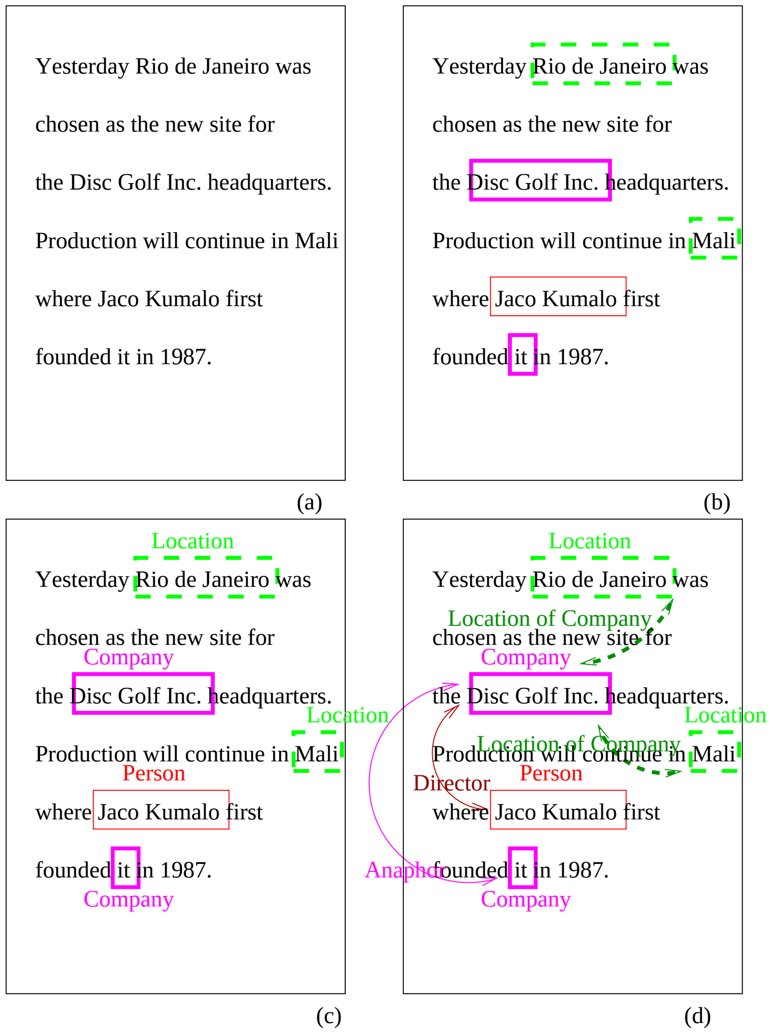
The information extraction task relies on the assumption that interesting target semantic classes have been identified for the task. In the MUC terrorism domain (Proceedings, 1991), the semantic classes `victim`, `perpetrator`, `weapon`, etc, were identified as being of interest. In the MUC management succession domain (MUC-6 Proceedings, 1996), the semantic classes `company, takeover-event,` etc were identified as being of interest. The classes can be chosen either a priori, or by inspection of the target corpus.

2.4.2 Dictionary Construction

For a given semantic class, it can be useful to have a dictionary of terms which are likely to belong to that class, perhaps with an associated probability of class membership. This dictionary can then be used as input to other algorithms, by identifying potential class members in a document whenever dictionary items are seen. Thelen and Riloff (2002) construct dictionaries by extrapolating from the contexts of initial sample class members, and assuming that a noun-phrase will be only of a single type in a given context.

2.4.3 Semantic Class Labeling

Given a region of text, semantic class labeling involves assigning a label or meaning, possibly from a predetermined set, to that region of text. This can be a text classification problem (Freitag, 1998) when we view the instances to be labeled in their contexts in texts (i.e. as tokens, or noun-phrases-in-context (NPIC)) . It can also be viewed as a cluster labeling problem, if we cluster NPICs in the texts, then seek to



Yesterday Rio de Janeiro was
chosen as the new site for
the Disc Golf Inc. headquarters.
Production will continue in Mali
where Jaco Kumalo first
founded it in 1987.
(a)
(b)
Location
Company
Person
(c)
Location of Company
Director
Anaphora
(d)

Figure 2.1: A document marked up with the results of information extraction subtasks: (a) Unlabeled document (b) after pro-active Semantic Class Finding (shown here as finding box boundaries and textures) (c) after Semantic Class Labeling (shown here as labeling boxes “Location”, “Company” etc), (d) after finding Semantic Relations (generating arrow labels : “Location of Company”), and Anaphora Resolution (arrow labels for identity)

assign labels to those clusters. A sub-problem of semantic class labeling is referred to in (Grishman., 1995) as name recognition, and would involve recognizing the names "Rio de Janeiro", "Disc Golf Inc", "Jaco Kumalo" and "Mali" as people, company and place names, from the sentences in Figure 2.1.

2.4.4 Semantic Relation Finding

In semantic relation finding we assign labels to pairs or groups of instances of semantic classes, or NPICs. Output from the sentences in Figure 2.1 may include `Director-of-Company(Jaco-Kumalo-NPIC-1, Disc-Golf-Inc-NPIC-2)` and `Location-of-Company(Mali-NPIC-3,Disc-Golf-Inc-NPIC-2 )`. This has also been described as template filling, and much of the literature in information extraction, including that by Califf (Califf & Mooney, 1997) and Huffman (Huffman, 1996), assumes that fields have been assigned to semantic types before this operation is performed. It identifies the relationships between the objects of interest. Huffman's system (1996) allows the training corpus to be built up interactively by the user by presenting pairs of NPICs to the user for labeling. Soderland (Soderland, 1999) assumes that we have a training corpus marked up with semantic classes, and some of the desired relationships. The work described there examines the use of active learning, but does not permit the simultaneous acquisition of the semantic classes themselves.

2.4.5 Anaphora and Coreference Resolution

An anaphor is a noun-phrase which does not refer to a real-world entity by itself, but refers to some earlier reference in the text to that real-world entity. For example, a character introduced in a text as "a tall woman" may later be anaphorically referred to as "the woman" or "she". Pronominal anaphora resolution involves replacing each instance of a pronoun with the full name with which it is identified elsewhere in the text. In Figure 2.1 this means identifying "it" in the second sentence as referring to "Disc Golf Inc." We can view this a special case of semantic relation finding. The relation we wish to find is the identity relation. Previous work of interest includes that by Dagan et al. (Dagan et al., 1995) which stresses the importance of syntactic features over lexical features for resolving pronominal anaphora. That paper suggests that lexical information derived from corpus statistics can contribute to resolving ambiguities remaining after syntactic resolution has contributed all possible information.

More generally coreference resolution refers to identifying when two mentions refer to the same entity, for example that “Dr Mitchell”, “Tom Mitchell” and “he” may all be referring to the same entity (Gooi & Allan, 2004) (Bean & Riloff, 2004)

2.5 Automating IE

Recent attempts to improve the development time for information extraction systems have focused on piece-wise application of machine learning algorithms to the sub-components, generally by taking a corpus marked-up with the target output, and applying a variety of traditional machine learning algorithms. Muslea (Muslea, 1999) surveys recent work on automating the learning of extraction patterns for both free text and structured documents and concludes that the trend is towards the use of a combination of syntactic, semantic and delimiter-based approaches, and that systems which learn multi-slot rules are preferable. Glickman and Jones (Glickman & Jones, 1999) survey the relevant literature for the entire information extraction problem, and conclude that most of the parts of the basic information extraction problems have been addressed with machine learning approaches. They suggest that the missing step is the integration of these pieces into a complete system in which the learning of each part informs the whole, with a corresponding reduction in the amounts of training data required.

2.5.1 Clustering for Semantic Class Labeling

The general goal of clustering techniques is to group like things together. In Euclidean-like spaces this is accomplished by defining (1) cluster centroids as points in the feature space, and (2) a distance metric over the feature space. An optimization of an objective function is then performed, for example to minimize intra-cluster distances (or distances of cluster elements from the closest centroids) while maximizing inter-cluster distances. For modeling data distributions which are parameterized as a mixture of Gaussian distributions, clustering is performed to find means of component distributions. Cluster radiuses (maximum intra-cluster distances) are then proportional to the component variances. In general, clustering is performed in an unsupervised way, with the feature space and objective function chosen a priori without reference to the target function over the data, generally to be a function which it is hoped will correlate well with the intended use of the clusters generated. A frequent problem, expressed in (Bensaid et al., 1996) is that the objective function does not correlate well enough with the task at hand.

Collocational regularities have been exploited to produce classes for language modeling and word-sense disambiguation, with the classes produced by clustering the data without external labels (Lee, 1997), (Rooth et al., 1999). This kind of distributional clustering has also been used for feature reduction for text classifiers (Hofmann, 1999), (Baker & McCallum, 1998). Agglomerative clustering was performed by Brown et al (Brown et al., 1992). Similarity metrics for clustering intended for language models are based on trigram or bigram statistics, and are generally performed by conducting bottom-up clustering, where two clusters are merged if their merge causes the smallest loss in between-class mutual information. A word w_i is *similar* to another w_j if w_i can be used to predict the probability of unseen word pairs involving w_j (Dagan et al., 1998). Other similarity measures used include KL divergence and Jensen-Shannon divergence.

The problem of unsupervised discovery of word-classes has been referred to both as "clustering" in (Lee, 1997), and as factor analysis (Hofmann, 1999). Clustering has traditionally been the term used when word-classes have hard boundaries, and a word can be found in a single class. Factor analysis assigns words to soft classes; a particular word is modeled as a set of weights for a mixture distribution. Concrete examples of semantic classes that are reliably produced by completely unsupervised clustering include weekdays, names and words referring to people, as well as the syntactic classes of definite and indefinite articles. By seeding with target examples, Riloff and Jones (1999) are able to cluster groups of Locations, Weapons and Titles for use in information extraction with reasonable accuracy.

2.5.2 Training Data Bottleneck in Automating IE

Most of the work in unsupervised clustering has focused on classes which are useful for language modeling and word-sense disambiguation, not information extraction. The rest of the work mentioned in section 2.4 requires large amounts of labeled training data, or many human-hours in hand-writing rules. For rapid deployment of an information extraction system in a new domain, we would like to have to provide a system with only a few examples of our target concepts, and have it automatically infer which other examples are also relevant, and learn rules over this expanded set.

Collins and Singer (Collins & Singer, 1999) have done some work on reducing training data requirements for the named entity classification task (semantic class labeling). They compared a variety of boot-strapping algorithms for named entity classification. Their algorithms use a rich feature set, including not only string-identity and context-identity, but context type (PP versus appositive), capitalization, contains_word, non_alphabetic_chars. They direct their approach at only a sub-set of the examples a full information extraction system would need to label, labeling only those NPs containing something already identified as a proper noun in either an appositive or PP, and perform a four-way classification (person /location / organization /noise). They achieved very good results by their conservative approach of adding a few new examples to the bootstrapping algorithm at each step. AUTOSLOG-TS (Riloff, 1996a) automates learning semantic relations by requiring the user to be involved only at the filtering stage, once the algorithm has identified relations likely to be of interest.

2.6 Scope of the Problem We Address

We focus on simultaneously addressing the problems of *semantic lexicon construction* and *semantic class labeling*. That is, our goal is to find words and phrases which are potential instances of the target class, and correctly classify whether they are used in this way in a particular context in text. In particular, we attempt to learn probability distributions over dictionary entries, then use those to assign probabilities over possible instances in new documents .The reason we need to address semantic class labeling in context is that dictionary entries may be ambiguous, for example “leader” is ambiguous between `people` and `organizations`. When we see the context it occurs in, we attempt to assign it to the correct class.

2.6.1 Target Classes

We will work with the target classes `people`, `locations` and `organizations`. Our goal will be to correctly identify instances of these in text documents. While a simple system which identifies all instances of dictionary entries as positive may appear tempting, in practice there is ambiguity over terms; for example “Arizona” is both the name of a US state, and a company, while “Paris” is both the name of a city and a person.

2.6.2 Inputs

The inputs we work with are small lists of words. We will describe these in more detail in Chapter 3. We also work with text documents which have been parsed into noun-phrases and their lexico-syntactic contexts. More detail on this is also given in Chapter 3. We work almost exclusively with small lists of words in this dissertation, though it would be reasonable also to work with long lists, if we have them at our disposal.

2.6.3 Outputs

The outputs of our learning are probability distributions over noun-phrases and contexts. We will evaluate these by using them to identify the target classes in new documents. More details about how this extraction is performed will be given in Chapter 3.

2.7 Related Work

We have included descriptions of related work in this chapter as we described the information extraction problem. We also give related work in later chapters as we cover topics in more detail. In this section we include a summary of the most salient related work, related work not covered in other sections, and a guide to the sections where other related work is discussed.

2.7.1 Information Extraction

We introduced some related work on learning to cluster similar word-types in this chapter in Section 2.4, learning specific word classes for information extraction in this Chapter in Section 2.5.2 as well as in Chapter 3, section 3.5.4.

Also relevant is work by Kou et al (Kou et al., 2005), who use a large dictionary converted to an HMM for high precision extraction of dictionary entries and variants on those entries. This is an application in which a pre-existing dictionary is required, in contrast with our approach, which learns both a dictionary and contexts from a small amount of additional data. While we require a shallow parser to pre-process our data, Sarawagi and Cohen (Sarawagi & Cohen, 2004) allow the incorporation of segmenting the text with the identification of named entities.

2.7.2 Active Learning

We describe most related work on active learning in Chapter 4 Section 4.2. In addition, Raghavan et al (Raghavan et al., 2005) show that users are able to efficiently label features independent of their context in documents, and that this contributes to greater efficiency in learning for document classification. This approach is similar to our single-feature-set labeling that we will describe in Chapter 4 Section 4.4.

2.7.3 Graph Structure and Semi-supervised Learning

We introduced some related work on semi-supervised learning in this chapter in Section 2.3.3. We describe other related work on semi-supervised learning in Chapter 3, sections 3.4.5 and 3.4.2.

We describe most related work on graph structure in Chapter 5. At the intersection of theory on graph structure and semi-supervised learning, Balcan et al (Balcan et al., 2004) show that cotraining effectiveness can be related to graph properties of the underlying labeled and unlabeled data. In particular, we can expect cotraining-like algorithms to perform well if the data has the *expander* property. That is, they will perform well if the probability mass of examples for which the algorithm is confident for only one of the two feature sets, is greater than the probability mass of example for which the algorithm is confident for both.

Blum et al (Blum et al., 2004) give an algorithm for using randomized min-cuts for semi-supervised learning, and describe a way of constructing the graph of the data which is amenable to this. They suggest that the data should be connected in one large component, or most of the data should be present in the largest component. They also compare joining nodes with edges based on a function of their similarity. This contrasts with our simple approach of joining all nodes, based on a single cooccurrence. We will show in Chapter 4 that active learning can compensate for cases that do not satisfy connectivity conditions, by selecting examples from components not covered by labeled examples. Using a threshold criterion for adding edges would be an interesting extension to our work.

Joachims (Joachims, 2003) also models data for semi-supervised learning, used a fixed number of nearest neighbors for each node as the edges in the graph. This way of setting up the graph means that the graph may not have power-law properties.

Agichtein's dissertation (Agichtein, 2004) is related in spirit to this work. The task Agichtein addresses is of identifying relations, such as locationOfCompany(L,C) or directoryOfCompany(C,P). The `Snowball` system is used to bootstrap from a small number of labeled examples. The `QTExtract` system constructs queries to identify new relations to be used for training. Agichtein evaluates the influence of the graph-structure on his learning task, by showing that the data has power-law structure, and measuring the *reachability* of the target examples within that graph.

2.8 Chapter Conclusions

In this Chapter we have given an overview of some background and terminology, from both linguistics and machine learning. We briefly defined the information extraction task we will be tackling in this thesis, and gave some pointers to where related work can be found in the thesis.

In the next chapter we will give more concrete details about the information extraction task, including the data and training information we use, and our evaluation methods. We will describe experiments on performing information extraction, which we will improve on in later chapters with active learning (Chapter 4).

Chapter 3

Comparison of Bootstrapping Algorithms for Information Extraction

Our goal is to minimize the effort required to train information extraction systems. In Chapter 2 we described the information extraction task of learning to identify locations, organizations and people in context, and the desired outputs of training. In this chapter we will show that we can train an information extraction system using a very small amount of training data in the form of example words from the target class. We will describe the properties of the data, and examine three algorithms which exploit a separation of the multiple feature sets present in the data: metabootstrapping, cotraining, and coEM. We will also examine EM, which does not make use of separation of the feature sets. We will concern ourselves with identifying the three semantic classes `locations`, `people`, and `organizations` in sentences in text documents. We will describe a set of assumptions about the properties of the data and the task, that make this task possible with very little training data. In addition we investigate the effects of different seeds on bootstrapping performance, the effects of unlabeled training set size, and the use of frequency information and stopwords.

3.1 Introduction

In this chapter we will show that we can train an information extraction system using a very small amount of training data in the form of example words from the target class (*seed words* which will be described in more detail in Section 3.3). In order to do this we will have to describe data representations and algorithms, as well as assumptions made by the algorithms.

The questions we will attempt to answer in this chapter are:

1. How can we represent data so we can use it to learn to extract instances of semantic classes?
2. What algorithms can we use for bootstrapping?
3. How much does bootstrapping contribute, over using the seeds alone for bootstrapping?
4. Does it matter which seeds we choose?
5. Should we correct any errors introduced by using seeds?
6. Can we learn all classes equally well with the same representation?
7. How does corpus size affect learning?
8. What assumptions do the algorithms make about the data representation, and how well are those assumptions satisfied?

In Table 3.1 we summarize the dimensions we will explore in this chapter. We will give more detail about each of these in the Sections below.

3.2 Data and Representation

Recall from Chapter 2 that we wish to learn to extract `locations`, `people` and `organizations` in context from text documents. In this section we describe the kinds of text documents we use in our experiments, and how we represent the target classes and their contexts in those web pages.

Dimension	Instantiations
Data representation	Output of a shallow parser
Training corpus	Company web pages, and TREC web collection
Target classes	`people`, `locations` and `organizations`
Algorithm used	Metabootstrapping, cotraining, coEM, and EM
Number of seeds used	10, 20 and 253
Seed selection method	Thoughtful, random from list
Treatment of errors in labeling	Uncorrected, corrected
Corpus size	Moderate, large

Table 3.1: Summary of issues we will consider in this chapter.

Note that we address only extraction of information from the sentences in text documents. Extraction from tables and semi-structured layout require different techniques, which are well described elsewhere (Cohen et al., 2002; Hurst, 2000; Knoblock et al., 2000).

3.2.1 Data Sources

The experiments in this chapter use data from two sources: company web pages and web pages from the TREC WT10g collection. Details about each are given below.

Company Web Pages

The company web pages come from crawling seven top-level economic sectors from a hierarchy published by Marketguide `http://www.marketguide.com`. The seven sectors are shown in Table 3.2. The web sites were crawled in 1998. This data is a subset of the company data described in (McCallum et al., 1998). It consists of 4392 corporate web pages of which 4160 were used for training and 232 were set aside as a test set. We refer to this data as the *7sector* data. It can be obtained from `http://www.cs.cmu.edu/afs/cs/project/theo-20/www/data/bootstrappingIE/`.

TREC WT10g Web Collection

The TREC WT10g Web Collection (Bailey et al., 2003) is a set of web pages collected for the TREC information retrieval evaluation. Information about obtaining this

basic materials sector
energy sector
financial sector
healthcare sector
technology sector
transportation sector
utilities sector

Table 3.2: Seven top-level economic sectors from Marketguide's hierarchy. Pages collected from these sectors were used as the *7sector* data-set.

WT10g name	num docs
wtx051	15,724
wtx051-052	30,552
wtx051-053	48,921
wtx051-056	99,324

Table 3.3: TREC WT10g data subsets used in our experiments and the number of documents with data in each subset, after parsing.

data can be found at `http://www.ted.cmis.csiro.au/TRECWeb/access_to_data.html`. The data consists of 10 gigabytes of html and text web pages, crawled in 1997. We identify a number of subsets, shown in Table 3.3, in order to have larger and larger collections of documents, ranging from 15,000 documents to just under 100,000 documents.

3.2.2 Data Preprocessing

All HTML was stripped from the pages. For the representation we used for our task, which will be described in Section 3.2.3, we need to be able to identify sentences and sentence fragments. In web pages, this can include noun-phrases in headings and lists that are not attached to any other sentences, as well as sentences in headings that do not include punctuation. In order to separate sentences and sentence fragments when punctuation is missing, we added periods heuristically. The algorithm for adding periods is due to Mike Thelen of the University of Utah and is given in Table 3.4.

Periods are added only at the end of lines.
If the line ends with punctuation, no period is added.
If the final word in the line is any of:
{"the", "a", "an", "to", "of", "by", "at", "on", "in", "click",
"is", "and", "or", "but", "not"} no period is added.
If the next line is blank, add a period.
If there are five or fewer words in the line, add a period.

Table 3.4: Heuristic Algorithm for inserting periods in web pages for subsequent parsing. The HTML is stripped from these pages before the algorithm is applied.

After adding periods to identify sentence boundaries, we parsed the sentences using Sundance (Riloff & Phillips, 2004), a parser from the University of Utah, then ran Autoslog (Riloff, 1996b) over the sentences. This results in a corpus of noun-phrases, each paired with local context in the form of a lexico-syntactic pattern. We described lexico-syntactic patterns in Section 2.1.7. These patterns come from a small set of possible patterns, given in Table 3.5. The version of Sundance and Autoslog that we used is version 3.0. We ran Autoslog in an exhaustive fashion, so all possible lexico-syntactic patterns occurring with any noun-phrase were identified, that is, the flag `-u 1` was employed. For extraction using those contexts, we used the domain "terrorism", and the shepherd output format was selected, *ie* `-p shepherd`.

3.2.3 Task Representation

An instance $x \in \mathcal{X}$ is a noun-phrase and a lexico-syntactic context, produced by Autoslog. The lexico-syntactic contexts were introduced in Section 2.1.7 and the way they are obtained from web pages was described in Section 3.2.2 and Table 3.5. We will refer to these lexico-syntactic contexts as *contexts* for the remainder of this thesis. Thus an example x_i is made up of the pair $< n_i, c_i >$, where $n_i \in \mathcal{N}$ refers to the noun-phrase in an instance x_i, and $c_i \in \mathcal{C}$ refers to the context. The instance space is thus $\mathcal{N} \times \mathcal{C}$, and we will be aiming to learn functions $f : \mathcal{N} \times \mathcal{C} \rightarrow \mathcal{Y}$, where $\mathcal{Y}$ refers to the set of possible class labels for instances. Table 3.6 summarizes this notation.

$\mathcal{N}$ and $\mathcal{C}$ can both take on a very wide set of values, but many of these values will not appear in the training corpus. Our training corpus is a sample from the set of possible values $\mathcal{N} \times \mathcal{C}$, sampled according to the underlying probability distribution

Pattern	Example
<subj> passive_verb	<> was murdered
<subj> active_verb	< > bombed
<subj> verb infinitive	< > attempted to kill
<subj> auxtobe noun	<> was victim
<subj> auxtohave noun	<> has talent
active_verb <dobj>	bombed < >
infinitive <dobj>	to kill <>
verb infinitive <dobj>	threatened to attack < >
noun auxtobe <dobj>	fatality was <>
noun auxtohave <dobj>	farmers have < >
noun pp <obj-prep>	bomb against < >
active_verb pp <obj-prep>	killed with < >
passive_verb pp <obj-prep>	was aimed at < >
<subj> active_verb dobj	< > declare dividend
infinitive pp <obj-prep>	to expand in < >

Table 3.5: All lexico-syntactic contexts used. Each of these is instantiated with words, as shown in the examples. For noun-phrases, the head of the noun is instantiated as part of the noun phrase; the context can match noun phrases with different determiners or adjectives as part of the noun phrase. For the extracted portion (eg <subj>) the entire noun phrase is extracted.

Symbol	Meaning
$\mathcal{X}$	the set of possible instances
X	a sample of instances
X_{train}	a sample of instances used for training
X_{test}	a sample of instances used for testing
k	the size of the training set X_{train}, in number of instances
x_i	the i^{th} example in sample of instances X
y_i	the label associated with x_i
z_i	the pair $< x_i, y_i >$
$\mathcal{N}$	the set of possible noun phrases
N	a bag of noun phrases; the projection of X onto $\mathcal{N}$
n_i	the i^{th} noun phrase in a sample X
$\{n : n \ \in X\}$	the set of unique noun phrases in X
m_n	$\|\{n : n \ \in X\}\|$ the number of unique noun phrases in X
$\mathcal{C}$	the set of possible contexts
C	a bag of contexts, the projection of X onto $\mathcal{C}$
c_i	the i^{th} context in a sample X
$\{c : c \ \in X\}$	the set of unique contexts in X
m_c	$\|\{c : c \ \in X\}\|$ the number of unique contexts in X

Table 3.6: Notation for referring to instances.

corpus name	num docs	num unlabeled examples k	m_n	m_c
7sector	4,160	228,574	75,737	23,278
wtx051	15,724	1,829,020	442,700	302,565
wtx051-052	30,552	3,555,045	800,192	504,048
wtx051-053	48,921	5,313,224	1,168,248	689,776
wtx051-056	99,324	10,588,096	2,190,965	1,151,849

Table 3.7: Descriptive overview statistics for corpora used in this thesis: the number of documents, number of <noun-phrase,context> pair instances (k), number of unique noun phrases (m_n) and contexts (m_c) in each set.

$P_{\mathcal{N},\mathcal{C}}$. For language, the probability distribution generating words and phrases is Zipf, which means that many possible values will be seen only rarely, and may not occur at all in a sample. We will refer to a sample from $P_{\mathcal{N},\mathcal{C}}$ as $X = x_1..x_k$. We will refer to the sample of noun phrases occurring in X as $N = n_1..n_k$ and the sample of contexts as $C = c_1..c_k$.

Because there is a chance that some features may not be seen in the training set, we are interested in the distribution of the noun-phrase and context features in the training and test sets. Specifically we are interested in the size of the training set, in number of instances k as well as the size of the noun phrase and context sets appearing in the training set: m_n and m_c.

Table 3.7 shows these values along with the number of documents for the *7sector* corpus and the TREC corpora used as training data in this thesis.

In Sections 3.5.3 and 3.7.6 we discuss overlap between the training and test corpora in terms of values taken on by the noun-phrase and context attributes.

The learning task we address is learning a function from noun-phrases and contexts to a binary classification indicating whether this instance of the noun phrase belongs to a particular semantic category:

$$f_{Class}(n_i, c_j) = y$$

where $y \in \{0, 1\}$.

In practice, the meaning of a noun-phrase context pair may partially depend on the larger context, and so a person who is labeling test examples may label the same example $< n_i, c_j >$ positive or negative depending on where it occurs. We discuss this with measurements of inter-rater agreement in Section 3.5.2. Thus we wish to map from the noun-phrase, context and target class to a probability of being in that class. The function we are trying to learn can be written as

$$f_{Class} : \mathcal{N} \times \mathcal{C} \rightarrow [0..1]$$

where the value of f_{Class} is $P(Class|n_i, c_i)$.

Viewed as a statistical estimation problem, we wish to estimate

$$P(Class|n_i, c_i)$$

using $\hat{P}(Class|n_i, c_i) = \hat{f}_{Class}(n_i, c_i)$ for arbitrary pairs of noun-phrases and contexts.

3.3 Weak Initial Labeling with Seeds

Since we are interested in minimizing the burden on a user for training an extraction system, we would like the method of providing initial information to be as easy as possible. We employ weak labeling, introduced in Section 2.3.3, in a form that does not necessarily require the user to inspect the data at all. The specific method we propose consists of asking the user to provide a small set of nouns or noun phrases which may be representative of the target class, which we will call *seeds*. In many of the experiments that are described in this chapter, 10 initial words are provided by the user. In the following sections we describe the types of seeds used in our experiments, how those seeds are used to automatically label an unlabeled corpus of data, and ways of addressing possible ambiguity in the initial seeds set.

3.3.1 Seeds

In order to characterize the target class, we ask the user for a small set of words which may occur in positive examples of it, as part of noun phrases, *ie* as examples of $n \subset \mathcal{N}$. For example, when the user wants to train the system to recognize the class `locations`, they may provide a list of 10 country names. These are positive examples from the feature set $\mathcal{N}$. The majority of experiments described in this thesis were run with the the seeds shown in Table 3.8. The `locations` seeds are the same as those used in (Riloff & Jones, 1999). In Section 3.7.5 we will also compare

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 3.8: Seeds used for weak labeling of data, for initialization of bootstrapping.

the effects of using different seeds for this task. The seeds for `organizations` and `people` were chosen by sorting noun-phrases in the training set by frequency, and asking the trainer to select the first ten matching the target class. Note that this method does not necessarily lead to the best choice of seeds, but is a simple method not requiring skill or experience. We could also ask the user to think up some words which may occur in the target class, avoiding the need to inspect the data at all. The seeds provided by the user are used to label the data, with one of two methods: (1) fixed initialization, via head-labeling and (2) active initialization. We describe these methods below.

3.3.2 Fixed Initialization

Fixed initialization uses a form of approximate or weak labeling we call *head-labeling*. All pairs in which the noun phrase head (often the right-most word, as described in Section 2.1.3) matches a seed word are considered to be positive training instances, regardless of the context in which they appeared. This approach was also used by Riloff and Jones (1999). This is frequently correct, but may also introduce some errors. For example, the word "Canada" as a seed correctly labeled the example "locations in Eastern Canada"[1] as a positive example of the class `locations`, but incorrectly labeled the example "Royal Bank of Canada used technology"[2] as a positive example. In both cases the word we identified automatically as the head-word was "Canada". In Section 3.5.1 we will discuss in detail the accuracy obtained by using the seed

[1]"Ultramar Diamond Shamrock has a strong network of approximately 4,400 locations in 10 Southwestern states and eastern Canada."

[2]"The Royal Bank of Canada has used smart card technology since 1985 for access control to the bank's on-line electronic business banking services".

words for weak labeling on our datasets, as well as providing more examples both correctly and incorrectly labeled using this approach.

3.3.3 Active Initialization

To address occasional errors introduced by ambiguity in the automatic labeling phase, we implemented a novel method of labeling training data that incorporates active learning. In *active initialization*, examples matching the seed words for a given task are interactively labeled by the trainer before beginning the learning process. The process is shown here as Algorithm 1.

Algorithm 1 Active Initialization

```
inputs: a set of seeds $S$ from feature set $\mathcal{N}$
for all $x_i =< n_i, c_i >$: matchesSeed($S, n_i$) do
    $\hat{f}_{Class}(x_i)$ = askUserToLabel($x_i$)
end for
```

Note that this contrasts with fixed initialization, in that additional examples are labeled by the user. It also contrasts with regular batch labeling, in which a random subset of examples, or all examples, are labeled by the user. The examples are selected for labeling by the algorithm, which asks for confirmation before labeling each as positive for bootstrapping.

We might hypothesize that by actively labeling examples at the outset we can provide the learning algorithms with better initial examples and thus improve extraction performance. For reasonably frequent seed words, this requires significant numbers of examples to be labeled at the outset; 894 examples for `locations`, 3388 for `organizations`, and 5257 for `people`, for the seed words in Table 3.8 and the *7sector* training data.

Figure 3.1 summarizes the labeling process.

3.4 Algorithms

In this section we describe several algorithms for learning to extract semantic classes using noun-phrases and their contexts, given weak labeling of the type described in Section 2.3.3. These algorithms differ in the way they treat the two feature sets, and

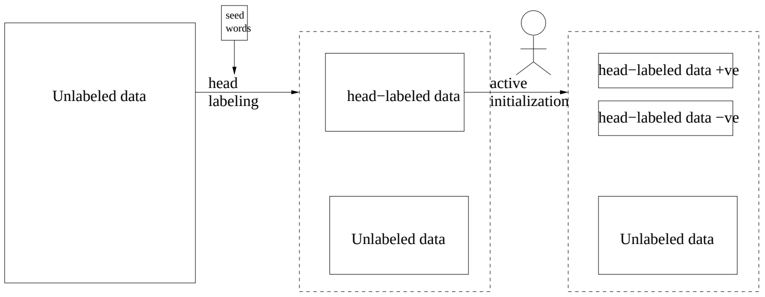


Figure 3.1: Automatically labeling data using seeds with head-labeling, then correcting these labels with active initialization.

how they use the unlabeled data. We referred to the groups of features as (1) the set of noun-phrases $\mathcal{N}$ and (2) the set of contexts $\mathcal{C}$. Algorithms which apply to two feature sets are applicable in other domains too.

The algorithms fall naturally into two classes. First are those that collapse the features into a single feature set $\mathcal{X} = \mathcal{N} \times \mathcal{C}$. These algorithms learn a model over $\mathcal{X}$, use it to label or relabel the unlabeled data, and then relearn the model. We will refer to this class of algorithms as *single-view bootstrapping* algorithms, and discuss them more fully in Section 3.4.2. The second set of algorithms uses the feature set split into $\mathcal{N}$ and $\mathcal{C}$ explicitly, by learning models over the two feature sets separately, and using those models separately to label or relabel the unlabeled data. This class of algorithms has been described as the class of *cotraining* algorithms, but we will refer to them as *two-view bootstrapping algorithms*, and we will discuss this class of algorithms in Section 3.4.3.

3.4.1 Algorithm Inputs

The algorithms all have the same inputs:

1. seed words S $\subset \mathcal{N}$, which are nouns likely to be in examples of the target class
2. unlabeled data $X = x_1..x_k$ sampled from $\mathcal{X} = \mathcal{N} \times \mathcal{C}$. Each example x_k is a pair consisting of exactly one feature whose value n_i is from the set $\mathcal{N}$ and one feature whose value c_j is from the set $\mathcal{C}$. $N(n_i, c_j)$ is the number of examples with the values n_i and the feature c_j.
3. A function matchesSeed(S,x_i) which specifies whether an example x_i matches one of the seeds in S.

3.4.2 Single-view Bootstrapping Algorithms

A single-view set bootstrapping algorithm uses a single model constructed over all features for learning. The general algorithm is given in Algorithm 2. One instantiation of this type of algorithm is *self-training* described by Nigam and Ghani (2000). The algorithm is trained on the initial labeled data, then assigns labels to the unlabeled examples for which it has the most confident predictions. It is then retrained using the initial examples, and those it has labeled itself. Nigam and Ghani found that self-training did not perform as well as *expectation-maximization (EM)* on their document classification task. In this thesis we use EM, and a full description of EM and its instantiation in our information extraction domain can be found below.

Algorithm 2 General procedure for single feature set bootstrapping algorithms with weakly labeled positive data. Algorithms vary in how they select and use automatically labeled examples.

```
initial training set $L_0 \leftarrow \{< x_i, \hat{y}_{i_0} >: x_i \subset X, \hat{y}_{i_0} = matchesSeed(S, x_i)\}$
repeat
  train $\hat{f}_t(n_i, c_j)$ on $L_{t-1}$
  $\forall x_i \in X$ use $\hat{f}_t(n_i, c_j)$ to update $\hat{y}_{i_t}$
  $L_t \leftarrow \{< x_i, \hat{y}_{i_{t+1}} >: satisfiesSelectionCriterion(x_i, \hat{y}_{i_{t+1}})$
until (change in models $< \epsilon$) $\vee$ (maxIterations reached)
```

EM

The Expectation Maximization (EM) algorithm is an iterative hill-climbing procedure for finding a parameterization of a probability density function (PDF) which locally maximizes the likelihood of the observed data. It is useful in situations in which the form of the probability density function is known, but the observed data does not provide the information we need to estimate the parameters. This is the case with calculating the parameters of mixture distributions, when class membership of observations is unknown. We call the unknown class membership labels of the observed data the *hidden* data. We then use the EM algorithm to iteratively estimate the values of the hidden data, and use these estimates to update estimates of the model parameters. In statistical terms, the observed data do not provide us with a means of calculating the *sufficient statistics* for the model, but we can iteratively improve our estimates of these sufficient statistics.

The overall description of the EM algorithm is as follows:

Initialization Initialize the parameters of the model to non-singular values.

E-step Calculate the expected values of the hidden data using the current model parameters.

M-step Update the model parameters using the observed data and the expectations of the hidden data calculated in the E-step.

Likelihood Test Calculate the likelihood of the observed and hidden data. If the likelihood has not converged, return to the E-step.

Probabilistic Model

In our setting we assume that the data is generated by a mixture model with two components. Component one, the positive component, generates a noun-phrase and a context according to a multinomial distribution for each. $M_{pos}(\mathcal{N})$ is a multinomial distribution over all noun-phrases in $\mathcal{N}$, given that an example is in the positive class. $M_{pos}(\mathcal{C})$ is a multinomial distribution over all contexts in $\mathcal{C}$, given that an example is in the positive class. We assume that the multinomials for the noun-phrases and contexts are conditionally independent, given the class label, and so the PDF for their joint distribution is just the product of the multinomials. In practice the noun-phrases and contexts are not completely conditionally independent. We will examine this assumption in more detail in Chapter 5.

We can therefore write the PDF for the first component as:

$$f_{pos}(\mathcal{N},\mathcal{C}) = M_{pos}(\mathcal{N})M_{pos}(\mathcal{C}) \tag{3.1}$$

Similarly,

$$f_{neg}(\mathcal{N},\mathcal{C}) = M_{neg}(\mathcal{N})M_{neg}(\mathcal{C}) \tag{3.2}$$

In addition we have the mixture parameter π which determines the probability of selecting a component to generate an observation. In Bayesian terms, this is the prior probability of the class. Class membership for each example is determined by the random variable Z, which follows a Bernoulli distribution determined by π. Our probability model of the complete data is then

$$f(\mathcal{N}, \mathcal{C}, Z) = \pi M_{pos}(\mathcal{N}) M_{pos}(\mathcal{C}) + (1 - \pi) M_{neg}(\mathcal{N}) M_{neg}(\mathcal{C}) \tag{3.3}$$

Note that while we can observe the noun-phrases $\mathcal{N}$ and contexts $\mathcal{C}$, we cannot observe the class memberships Z directly. These are our hidden data. The complete set of individual parameters that must be learned to fully specify our classifier is therefore:

$$n_{i_{pos}} \sim M_{pos}(\mathcal{N})$$

$$c_{j_{pos}} \sim M_{pos}(\mathcal{C})$$

$$n_{i_{neg}} \sim M_{neg}(\mathcal{N})$$

$$c_{j_{neg}} \sim M_{neg}(\mathcal{C})$$

for $n_i \in \mathcal{N}$, $c_j \in \mathcal{C}$

and

$$\pi \sim U(0, 1)$$

Initialization

In the initialization phase, we set the initial parameters of the model. We have partial information in the form of seed words, and make the assumption that any pair with a seed as the head of the noun-phrase is a positive example. We further assume that all unknown examples are negative.

$$\hat{P}_{init}(pos| < n_i, c_j >) = \begin{cases} 1 & \text{if } matchesSeed(n_i, S) \\ 0 & \text{otherwise} \end{cases} \tag{3.4}$$

In Section 3.8.1 we discuss three other ways of initializing $\hat{P}_{init}(pos| < n_i, c_j >)$ which were also used in the experiments.

We can then use this labeling to calculate the multinomial parameters and the class prior π. Recall that $N(n_i, c_j)$ is the number of examples with the values n_i and the feature c_j, while m_n is the number of unique noun-phrases in the sample X, and

m_c is the number of unique contexts, and k is the number of instances in the sample X. To avoid zero probabilities, we use Laplace smoothing:

$$\hat{P}_0(n_i|pos) = \frac{1 + \sum_j N(n_i, c_j, pos)}{\sum_{k,l} N(n_k, c_l, pos) + m_n}$$

$$\hat{P}_0(c_j|pos) = \frac{1 + \sum_i N(n_i, c_j, pos)}{\sum_{k,l} N(n_k, c_l, pos) + m_c}$$

$$\hat{P}_0(n_i|neg) = \frac{1 + \sum_j N(n_i, c_j, neg)}{\sum_{k,l} N(n_k, c_l, neg) + m_n}$$

$$\hat{P}_0(c_j|neg) = \frac{1 + \sum_i N(n_i, c_j, neg)}{\sum_{k,l} N(n_k, c_l, neg) + m_c}$$

$$\hat{P}_0(pos) = \frac{1 + N(pos)}{2 + k}$$

E-step

In the E-step, we calculate the expected labels of the data using the model parameters. Specifically, at step $t+1$ for each pair we update our estimate of the probability of that example being in the positive class, based on the parameters of the model we estimated at step t:

$$\hat{P}_{t+1}(pos|n_i, c_j) = \frac{\hat{P}_t(n_i, c_j|pos)\hat{P}_t(pos)}{\hat{P}_t(n_i, c_j)} \quad (3.5)$$

$$= \frac{\pi \hat{P}_t(n_i|pos)\hat{P}_t(c_j|pos)}{\pi \hat{P}_t(n_i|pos)\hat{P}_t(c_j|pos) + (1-\pi)\hat{P}_t(n_i|neg)\hat{P}_t(c_j|neg)} \quad (3.6)$$

These are our expectations for the hidden class labels Z. Note that for two instances $x_m =< n_i, c_j >$ and $x_r =< n_i, c_j >$, ie two distinct examples with the same features n_i and c_j, $z_m = z_r$.

M-step

In the **M-step** we estimate the parameters based on the labeling of the data. In particular, we estimate the class prior $\hat{\pi}$, as well as the parameters for each feature $\hat{P}(n_i|pos)$, $\hat{P}(n_i|neg)$, $\hat{P}(c_j|pos)$, and $\hat{P}(c_j|neg)$. We use Laplace smoothing, and find the expected number of occurrences of examples in each class by using the estimated class labels $\hat{P}(pos|n_i, c_j)$ in place of the hidden true labels Z. For examples in which the noun-phrase has a seed as its head, we use the assumed label positive rather than the expectation for that label. In other words, examples initially labeled positive remain unchanged throughout the EM process.

$$\hat{P}_t(n_i|pos) = \frac{1 + \sum_j E[N_{pos}(n_i, c_j)]}{\sum_{k,l} E[N_{pos}(n_k, c_l)] + m_n}$$

$$\hat{P}_t(c_j|pos) = \frac{1 + \sum_i E[N_{pos}(n_i, c_j)]}{\sum_{k,l} E[N_{pos}(n_k, c_l)] + m_c}$$

$$\hat{P}_t(n_i|neg) = \frac{1 + \sum_j E[N_{neg}(n_i, c_j)]}{\sum_{k,l} E[N_{neg}(n_k, c_l)] + m_n}$$

$$\hat{P}_t(c_i|neg) = \frac{1 + \sum_i E[N_{neg}(n_i, c_j)]}{\sum_{k,l} E[N_{neg}(n_k, c_l)] + m_c}$$

$$\hat{\pi}_t = \frac{1 + \sum_{i,j} E[N_{pos}(n_i, c_j)]}{|Sample| + 2}$$

where

$$E[N_{pos}(n_i, c_j)] = \hat{P}_t(pos|n_i, c_j)N(n_i, c_j)$$

with the $\hat{P}_t(pos|n_i, c_j)$ calculated in the E-step when not $matchesSeed(n_i, S)$, and with $\hat{P}_t(pos|n_i, c_j) = 1$ when $matchesSeed(n_i, S)$

Similarly,

$$\begin{aligned} E[N_{neg}(n_i, c_j)] &= \hat{P}_t(neg|n_i, c_j)N(n_i, c_j) \\ &= (1 - \hat{P}_t(pos|n_i, c_j))N(n_i, c_j) \end{aligned}$$

Termination Condition

Next we check the **termination condition** by calculating the log likelihood of the observed data given the model parameters.

$$\begin{aligned}\log(l) &= \sum_{i,j} \log[(\hat{\pi}_t \hat{P}_t(n_i, c_j|pos)) + (1-\hat{\pi}_t)\hat{P}_t(n_i, c_j|neg)] \\ &= \sum_{i,j} \log[(\hat{\pi}_t \hat{P}_t(n_i|pos)\hat{P}_t(c_j|pos)) + (1-\hat{\pi}_t)\hat{P}_t(n_i|neg)\hat{P}_t(c_j|neg)]\end{aligned}$$

If the log likelihood has not converged, we return to the E-step.

3.4.3 Two Feature Set Bootstrapping Algorithms

We have just finished discussing algorithms which infer new or updated labels for partially labeled examples, based on the current set of labels by considering examples in their entirety. In contrast, two feature set bootstrapping algorithms treat the two feature sets $\mathcal{N}$ and $\mathcal{C}$ separately. The algorithm learns a model for each: $\hat{f}_n(n_i)$ and $\hat{f}_c(c_j)$. These two models are used separately for labeling the unlabeled training data. Algorithm 3 provides the general algorithm for two-view bootstrapping algorithms. It differs from single feature set bootstrapping in that the two views are used in alternation to label the unlabeled data. This chapter describes experiments with three two-view bootstrapping algorithms: metabootstrapping, coEM, and cotraining, described below.

Algorithm 3 General procedure for two feature set bootstrapping algorithms with weakly labeled positive data. Algorithms vary in how they select and use automatically labeled examples. The results may differ depending on whether we start with $\hat{f}_{n_t}$ or $\hat{f}_{c_t}$.

```
initial training set $L_0 \leftarrow \{< x_i, \hat{y}_{i_0} >: x_i \subset X, \hat{y}_{i_0} = matchesSeed(S, x_i)\}$
repeat
    train $\hat{f}_{n_t}(n_i)$ on $L_{t-1}$
    $\forall x_i \in X$ use $\hat{f}_{n_t}(n_i)$ to update $\hat{y}_{i_t}$
    $L_t \leftarrow \{< x_i, \hat{y}_{i_t} >: satisfiesSelectionCriterion(x_i, \hat{y}_{i_t})\}$
    train $\hat{f}_{c_t}(c_i)$ on $L_t$
    $\forall x_i \in X$ use $\hat{f}_{c_t}(c_i)$ to update $\hat{y}_{i_{t+1}}$
    $L_{t+1} \leftarrow \{< x_i, \hat{y}_{i_t} >: satisfiesSelectionCriterion(x_i, \hat{y}_{i_t})\}$
until (change in models $< \epsilon$) $\vee$ (maxIterations reached)
```

3.4.4 Metabootstrapping

Metabootstrapping (Riloff & 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. There is no notion of negative examples or features, but only positive features and unlabeled features. The two feature sets, noun-phrases and contexts, are used asymmetrically. For a noun-phrase n_i the label $\hat{f}_n(n_i)$ does not change once it is assigned. For a context, however, the label $\hat{f}_c(c_j)$ can change at different steps in the algorithm.

Heuristics are used to score the noun-phrases n_i and contexts c_j at each iteration. For noun-phrases n_i the scores are based on co-occurrences with positive and unlabeled contexts c_j, using both frequency of co-occurrence, and diversity of co-occurring features. Similarly, for contexts the score depends on the cooccurring noun phrases and their scores. The highest scoring features which have not been labeled acquire a label $\hat{f}_n(n_i)$ or $\hat{f}_c(c_j)$.

Metabootstrapping treats the noun-phrases and their contexts asymmetrically, not only in the permanence of labeling, as described above, but also in the way noun phrases and contexts contribute to labeling each other. 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.

These asymmetries may derive from the empirical development of the algorithm. As we will see in Section 3.5.2, contexts are much more ambiguous than noun-phrases, when viewed in isolation. Thus an alternative algorithm which may be more effective would use the *reverse* asymmetry – permitting all contexts to be labeled positive when identified by a single noun-phrase, but requiring voting among contexts for labeling noun phrases.

Pseudocode for the metabootstrapping algorithm is shown in Figure 4.

Algorithm 4 Metabootstrapping algorithm.

```
initialize based on seeds S: $\hat{f}_{n_t}(n_i) == \begin{cases} 1 & \text{if } matchesSeed(n_i, S) \\ 0 & \text{otherwise} \end{cases}$
$S^t_{outer} \leftarrow S$
repeat
    $S^t_{inner} \leftarrow S^t_{outer}$
    initialize context list: $K = \{\}$
    repeat
        Score contexts:
            $$\hat{f}_{c_t}(c_j) = \frac{\sum_{n_i} \hat{f}_{n_t}(n_i) \times I(n_i, c_j)}{\sum_{n_k} I(n_k, c_j)} \times log \sum_{n_i} \hat{f}_t(n_i) \times I(n_i, c_j)$$
        where $I(n_i, c_j)$ is the indicator function which is 1 if $n_i$ and $c_j$ cooccur in the training set $X$.
        $c' = argmax_{\{c_i : c_i \neg\in K\}} \hat{f}_{c_t}(c_i)$
        $K \leftarrow K \cup c'$
        add all noun phrases cooccurring with context $c'$:
            $$\mathcal{S}^{t+1}_{inner} \leftarrow \mathcal{S}^t_{inner} \cup \{n_i : I(n_i, c') = 1\}$$
    until $|K| == 10 \vee \hat{f}_{c_t}(c') < 0.7$
    update $S_{outer}$:
        $$S^{t+1}_{outer} \leftarrow S^t_{outer} \cup argmax5_{n_i}(\sum(1 + 0.1(\hat{f}_{c_t}(c_j) \times I(n_i, c_j))$$
until maxIterations reached
```

3.4.5 Cotraining

Cotraining (Blum & Mitchell, 1998) is a bootstrapping algorithm that was originally developed for combining labeled and unlabeled data for text classification, and has been used successfully for named entity classification (Collins & Singer, 1999). 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. Unlike Collins and Singer (Collins & Singer, 1999), we treat noun phrases and contexts as atomic units, and do not match based on substrings or other properties, apart from the $matchesSeed(n_i, S)$ operator. This means that cotraining has access to the same view of the data as other algorithms described in this feature. Because cotraining requires negative examples, negative seeds are provided in the form of stopwords.

When used in our information extraction setting, the algorithm details are as given in Algorithm 5.

Note that cotraining assumes that we can accurately model the data by assigning noun-phrases and contexts to a single class. When we add an example, it is either entirely 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 3.5.2, many noun-phrases, and many more contexts, are inherently ambiguous. Cotraining may harm its performance through its hard (binary 0/1) non-probabilistic class assignment.

3.4.6 coEM

coEM is a hybrid algorithm, proposed by Nigam and Ghani (2000), combining features from both cotraining and Expectation-Maximization (EM). coEM is iterative, like EM, but uses the feature split present in the data, like cotraining. The separation into feature sets we use is that of noun-phrases $\mathcal{N}$ and contexts $\mathcal{C}$, as with our implementations of metabootstrapping and cotraining. coEM proceeds by initializing the noun-phrase classifier $\hat{f}_n(n_i) = \hat{P}(class|n_i)$ using the labeled data only. Then $\hat{f}_n(n_i)$ is used to probabilistically label all the unlabeled data. The context classifier $\hat{f}_c(c_j) = \hat{P}(class|c_j)$ is then trained using the original labeled data plus the unlabeled data with the labels provided by $\hat{f}_n$. Similarly, $\hat{f}_c$ then relabels the data for use by $\hat{f}_n$, and this process iterates until the classifiers converge. For final predictions over the test set, $\hat{f}_n$ and $\hat{f}_c$ predictions are combined by assuming independence, and assigning

Algorithm 5 Cotraining algorithm for our information extraction setting

```
initialize based on seeds $S_{pos}, S_{neg} : \hat{f}_{n_0}(n_i) = \begin{cases} 1 & \text{if } matchesSeed(n_i, S_{pos}) \\ 0 & \text{if } matchesSeed(n_i, S_{neg}) \end{cases}$
$K^1_{pos} = \{\}$
$K^1_{neg} = \{\}$
repeat
  Use $\hat{f}_{n_t}$ to score contexts, using labeled data only:
      $$\hat{f}_{c_t}(c_j) = \frac{\sum_{n_i \in S_{pos} \cup S_{neg}} \hat{f}_{n_t}(n_i) N(n_i, c_j)}{\sum_{n_i \in S_{pos} \cup S_{neg}} N(n_i, c_j)}$$
  select a single new positive context $c' = argmax_{\{c_i : c_i \neg \in K\}} \hat{f}_{c_t}(c_i)$
  $K^{t+1} \leftarrow K^t \cup c'$
  repeat
    select new negative context $c' = argmin_{\{c_i : c_i \neg \in K_{neg}\}} \hat{f}_{c_t}(c_i)$
    $K^{t+1}_{neg} \leftarrow K^t_{neg} \cup c'$
  until (10 new contexts added to $K_{neg} \wedge \hat{f}_{c_t}(c' > 0.7)$)
  Score contexts: $\hat{f}_{c_t}(c_j) = \begin{cases} 1 & \text{if } matchesSeed(n_i, K_{pos}) \\ 0 & \text{if } matchesSeed(n_i, K_{neg}) \end{cases}$
  Use $\hat{f}_{c_t}$ to score noun phrases, using labeled data only:
      $$\hat{f}_{n_t}(n_i) = \frac{\sum_{c_j \in K_{pos} \cup K_{neg}} \hat{f}_{c_t}(c_j) N(n_i, c_j)}{\sum_{c_j \in K_{pos} \cup K_{neg}} N(n_i, c_j)}$$
  select a single new positive noun phrase $n' = argmax_{\{n_i : n_i \neg \in S_pos\}} \hat{f}_{n_t}(n_i)$
  $S^{t+1}_{pos} \leftarrow S^t_{pos} \cup n'$
  repeat
    Select new negative noun-phrase $n' = argmin_{\{n_i : n_i \neg \in S_{neg}\}} \hat{f}_{n_t}(n_i)$
    $S^{t+1}_{neg} \leftarrow S^t_{neg} \cup n'$
  until 10 new negative noun phrases added
until (change in models $< \epsilon$) $\vee$ (maxIterations reached)
```

the test example probability proportional to $\hat{f}_n(n_i)\hat{f}_c(c_j)$.

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

Algorithm 6 CoEM algorithm for our information extraction setting

```
initialize based on seeds S: $\hat{f}_{n_0}(n_i) == \begin{cases} 1 & \text{if } matchesSeed(n_i, S) \\ 0 & \text{otherwise} \end{cases}$
repeat
    Use $\mathcal{N}$ to label $\mathcal{C}$:
    $\hat{f}_c(c_j) = \frac{\sum_{n_i} \hat{f}_n(n_i) * N(n_i, c_j)}{N(., c_j)}$
    Use $\mathcal{C}$ to label $\mathcal{N}$:
        $\hat{f}_n(n_i) = \begin{cases} 1 & \text{if } matchesSeed(n_i, S) \\ \frac{\sum_{c_j} \hat{f}_c(c_j) * N(n_i, c_j)}{N(n_i, .)} & \text{otherwise} \end{cases}$
until (500 iterations $\vee$ change $< \epsilon$)
```

The algorithm is as shown in Algorithm 6.

3.5 Assumptions and Biases

The bootstrapping algorithms described in Section 3.4 have a number of assumptions in common: (1) that initialization from seeds leads to labels which are accurate for the target class; (2) that seeds will be present in the data; (3) that distribution with similar phrase contexts correlates with semantic similarity; and (4) that noun-phrases and their contexts are redundant and unambiguous with respect to the semantic classes we are attempting to learn. These introduce a model selection bias which makes generalization possible, and which reduces the complexity of learning. Violation of these assumptions may affect the learning. This section assesses the validity of each of these assumptions by examining the data.

3.5.1 Initialization from Seeds Assumption

All the algorithms considered here use seed words as their source of information about the target class. An assumption made by all these algorithms is that seed words

Corpus	**Class**	**Seed-density (/100)**	**% Positive noun phrases**	**% Positive Instances**
7sector	locations	0.29	1.8	4
wtx-051-056		0.16		
7sector	organizations	1.49	6.4	12.8
wtx-051-056		0.25		
7sector	people	1.10	8.57	7.2
wtx-051-056		0.47		

Table 3.9: Density of seed words per 100 instances in fixed corpus of company web pages, as well as percent of training collection which is positive, based on sample of 500 (n, c) instance pairs.

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 (*7sector*), and one which we can consider to be drawn according to a uniform random distribution over documents on the world wide web (*wtx051-056*). The seeds we use for initializing bootstrapping all algorithms are shown in Table 3.8. The density of seed words in different corpora is shown in Table 3.9. Note that the `people` and `organizations` classes are much more prevalent in the company data we are working with than in random documents.

In addition, we can assess this assumption by looking at different candidate seeds for the same task. We took a list of 253 country names from a list of country domain names, and took subsets of size 10 and 20 from this list. The number of occurrences of the words in the list was quite variable, as shown in Table 3.10.

Accuracy of Head Labeling

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 3.11 shows the precision and estimated recall of head-labeling on the *7sector* training set. Tables 3.12, 3.13 and 3.14 show the precision for these classes,

Seed Set(s)	Num Seeds	Examples Matching (∗ Average over 10 sets)	σ Examples Matching (∗ Over 10 sets)
10-random	10	32.9∗	40.0∗
20-random	20	74.9∗	59.3∗
orig-10	10	894	
allcountries	253	1016	

Table 3.10: For the `locations` task, 10 random sets of 10 and 20 country names matched variable numbers of instances in the corporate web-page data. Shown here is the average number of instances matching, across the 10 sets, and the exact number of instances matching for the original 10 country names, and the entire list of 253 country names. The 10 country names used in basic experiments were very frequently occurring. Using all 253 country names from a list of country names did not match many more initial examples.

Class	**Examples Labeled**	**True Positives**	**Precision**	**Estimated Recall**
`locations`	894	865	98%	9.5%
`people`	3388	3207	95%	19.5%
`organizations`	5257	5247	99.8%	17.9%

Table 3.11: Precision and recall of labeling examples automatically using seed-heads. Recall was estimated based percentage positive examples in sample of 500 instances from the training corpus.

broken down by seed-word. For `people`, some seed words were generally unambiguous, with the exception of a few instances. An example of this is “customers”, which was unambiguous except in phrases 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 with active initialization in Section 3.7.4.

head-word	correct / labeled	precision
clients	328/334	0.98
customer	239/244	0.98
customers	1096/1124	0.98
director	74/76	0.97
individuals	139/139	1.00
leader	38/158	0.24
people	418/434	0.96
shareholders	379/379	1.00
subscriber	269/270	1.00
users	227/230	0.99
Overall	3207/3388	0.95

Table 3.12: Precision of labeling examples automatically using seed-heads for `people`.

head-word	correct / labeled	precision
arco	29/29	1.00
companies	896/898	1.00
company	3323/3329	1.00
dataram	42/42	1.00
inc.	844/844	1.00
praxair	13/13	1.00
puretec	73/75	0.97
xerox	27/27	1.00
Overall	5247/5257	1.00

Table 3.13: Precision of labeling examples automatically using seed-heads for `organizations`.

head-word	correct / labeled	precision
australia	36/38	0.95
canada	150/165	0.91
china	39/39	1.00
england	15/17	0.88
france	37/39	0.95
germany	43/43	1.00
japan	87/88	0.99
mexico	91/98	0.93
switzerland	13/13	1.00
united_states	354/354	1.00
Overall	865/894	0.97

Table 3.14: Precision of labeling examples automatically using seed-heads for `locations`.

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 3.15) we see that 78 noun-phrases were ambiguous between two classes, and 4 were ambiguous between three classes ("group", "they" and "them" were ambiguous between `organization`, `people` and no class, while "facility" was ambiguous between `location`, `organization` and no class). This means that these 82 noun-phrases (2% of the 4575 unique noun-phrases occurring in the test set, and 963 instances of the 8081 test instances, ie 12%) 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 3.16) we see even more ambiguity. 36% of contexts are ambiguous between two or more classes. This suggests that best results may be obtained with an algorithm which requires stronger evidence of class membership from contexts than from noun-phrases. An algorithm of this type was proposed by Thelen and Riloff (Thelen & Riloff, 2002) which adds words to the lexicon only when multiple contexts classify it as positive. Arguably this is similar to the scoring imposed by coEM. Exploration of this issue in greater depth is outside the scope of this thesis, but it warrants further examination.

Ambiguity	Class(es)	Number of NPs
Unambiguous Only Belonging to One Class	none	3677
	loc	114
	org	452
	person	191
Belonging to Two Classes	loc, none	6
	org, none	30
	person, none	23
	loc, org	6
	org, person	13
Belonging to Three Classes	loc, org, none	1
	org, person, none	3

Table 3.15: Distribution of NPs in the test set

Ambiguity	Class(es)	Number of Contexts
Unambiguous Only Belonging to One Class	none	1068
	loc	25
	org	98
	person	59
Belonging to Two Classes	loc, none	51
	org, none	271
	person, none	206
	loc, org	5
	org, person	50
Belonging to Three Classes	loc, org, none	18
	org, person, none	83
Belonging to Four Classes	loc, org, person, none	6

Table 3.16: Distribution of Contexts in the test set

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

Table 3.17: Conditions for inter-rate evaluation - All stands for NP, context and the entire sentence in which the NP-context pair appeared

Inter-rater agreement

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*)
- 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 3.17.

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.

		Noun phrase	Context	NP-Context Pair	
				Both	Either
% ambiguous	vocabulary	3%	41%	0.3%	57%
in test set	instances	18%	54%	0.8%	57%
% of test set	vocabulary	41%	91%	53%	97%
seen in training data	instances	58%	94%	56%	97%

Table 3.18: Ambiguity of nounphrases alone, contexts alone, and noun-phrase context pairs in the test set, along with training set coverage.

Seen Versus Inherent Ambiguity

We saw in Tables 3.15 and 3.16 that a number of phrases and contexts are ambiguous in our test set. This ambiguity may be both under- and overstated by these measures. The combination of noun phrase and context together may greatly reduce the ambiguity of examples in our test set. However, our test set may contain only one possible label for an example which is inherently more ambiguous. In addition, for many examples in our test set, we will not have seen both noun phrase and context during training. Table 3.18 shows ambiguity for noun phrases, context and their combination in the test set, along with coverage of test set vocabulary from the *7sector* training set. We see that less than 1% of examples are ambiguous when we have access to both noun-phrase and context. However, only for 56% of test instances have we seen both in the training set.

3.5.3 Relevance of Training Data Assumption

We assume that noun-phrases and contexts in the test set will be well-modeled based on information learned from the training set. In fact we find that, within the domain of company web-pages, of the 4516 unique noun-phrases in the test set, only 1836, or 41% of them, occurred in the training set. However, 91% of the 1961 unique contexts from the test set also appear in the training set. This is for training and test data drawn from the same distribution. For random data, we expect these discrepancies to be much greater. We will see in Section 3.7 that having train and test sets sampled from the same distribution helps bootstrapping more than adding extra documents for the `organizations` class.

We saw in Section 3.5.2 that noun-phrases are much less ambiguous than contexts. Now we see that noun-phrases are less well-modeled by the training data, we become aware of an asymmetry between the two feature sets. Both the phrase and the context will play a role in determining the correct classification in the test set, and the way the algorithm bias handles the asymmetry may be of great importance in its effectiveness.

3.5.4 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, that is, that distribution with similar phrase contexts correlates with semantic similarity. It has been shown (Dagan et al., 1998) 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 3.5.2. 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. We will see in Section 3.8.3 that different classes are very sensitive to the use of stopwords, in differing ways.

Meta-bootstrapping deals with this problem by specifically forbidding a list of 35 stop words (mainly pronouns) from being added to the dictionaries. We will also examine how the algorithm performs when stopwords are permitted in the model, in Section 3.7.3. In addition, metabootstrapping's heuristic that a context 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.

he, her, herself, here, him, himself, i, it, me, mine, ours, she, that, they, them
themselves, their, there, these, this, those, us, what, when, where, which, who
whom, we, you, percent, all, many, most, some

Table 3.19: Stopwords used in experiments.

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.

Stopwords

The stopwords shown in Table 3.19 are words frequently removed in natural language tasks, as they are frequent but may contribute less to the meaning of a section of text. When we are bootstrapping from seeds, we may wish to avoid learning stopwords as part of our model, as they may appear in many contexts. The word "it" for example, may appear in some contexts related to our task. At the same time, adding it to our model may cause the model learned to be much noisier in labeling unlabeled examples. As we discussed in Section 3.5.4, the word "they" is particularly ambiguous with respect to the tasks `people`, `organizations` and `locations`.

Frequency Information

A frequently occurring phrase may dominate a model, merely by being frequent. Metabootstrapping collapses repeated occurrences of a noun-phrase context pair $< n_i, c_i >$ to a single example, while coEM, cotraining and EM respect the frequency information in the training data. In Section 3.8.2 we will see the effects of models using and ignoring frequency information. There is an analogy between the use of term frequency versus binary term weight in models for document classification (McCallum & Nigam, 1998a), and filtering (Lewis, 1998). For document classification and filtering, document length is an issue, since a single occurrence of a word in a long document may suggest less about the topic than a word which occurs multiple times, or a word which occurs a single time in a short document.

In the bootstrapping information extraction case, the examples are of fixed size (a single noun-phrase and a single context, as described in Section 3.2). Table 3.20

context	occurrences	seeds	distinct nps	distinct seeds	$p_{mult}(pos)$	$p_{bin}(pos)$
located in	75	8	62	5	0.11	0.08
markets in	45	5	30	3	0.11	0.10
developed in	39	2	34	2	0.05	0.06

Table 3.20: Examples of probabilities of class membership based on multinomial and binomial occurrences. When we use binomial occurrence counts to calculate probabilities, removing frequency information, higher initial probabilities of class membership are assigned to contexts with a greater diversity of seeds.

shows examples of the different probabilities of class membership under these two representations, for contexts after initial head labeling. We see that "located in" and "markets in" have higher probabilities of class membership, when we consider all occurrences, compared to considering only binary occurrence counts. "developed in", however, has a higher probability of class membership when we consider only binary occurrence counts.

When deciding how representative of a class a given noun-phrase n_i is, we consider all examples it occurs in. If we ignore the frequency of each example, a high probability of class membership suggests that n_i is indicative of the target class, in the majority of the *contexts* it occurs in. If we also use the frequency information for each example, a high probability of class membership suggests that n_i is indicative of the target class, in the majority of the *examples* it occurs in. A high-frequency negative context may dwarf the influence of a plurality of positive contexts. However, for predictive accuracy, a model which takes into account the frequency of examples could be expected to work better.

3.5.5 Summary of Assumptions and Biases

As we saw in this section, the training corpus affects the density of seeds present in the data, and head labeling is relatively accurate. On the other hand, we saw that in the data the noun phrase and context feature sets are not redundant with respect to the task, with greater ambiguity with respect to target class in the contexts than in the noun phrases. In addition, as we will examine in Section 3.7.6, many of the noun phrases and contexts in the test set may not appear in the training set at all. Thus we anticipate that the best-performing algorithms will be those which are robust to these violations of the theoretical assumptions.

3.6 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.

3.6.1 Extraction on the Test Corpus

There are several ways of using the models produced by bootstrapping to extract from the test corpus. These are described below.

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. However, as we saw in Section 3.5.2, using only noun-phrases gives us information about only 58% of test instances for the *7sectors* task, since the training set vocabulary does not completely cover the test set.

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. We saw that 94% of test examples are covered by a context from the training set, though many of these are ambiguous.

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.

4. Bootstrap on the test corpus using both models. Here we use the models to initialize on the test corpus, then run bootstrapping. Surprisingly, preliminary experiments suggested that this approach does not work very well. We examine a related approach with transduction in Section 3.7.7.

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.

3.6.2 Evaluation Metrics: Precision, Recall and Breakeven

To evaluate, we assign scores to each of the test examples. We then sort the examples by score, and calculate precision and recall. Precision is defined with respect to a threshold value i. For all possible thresholds we calculate the number of examples we classify correctly as belonging to the class (*true positives* TP), the number of examples we incorrectly classify as belonging to the class (*false positive* FP), and the number of examples we incorrectly classify as negative (*false negative* FN). Precision is then given by:

$$Precision = \frac{TP_i}{TP_i + FP_i}$$

and Recall is given by:

$$Recall = \frac{TP_i}{TP_i + FN_i}$$

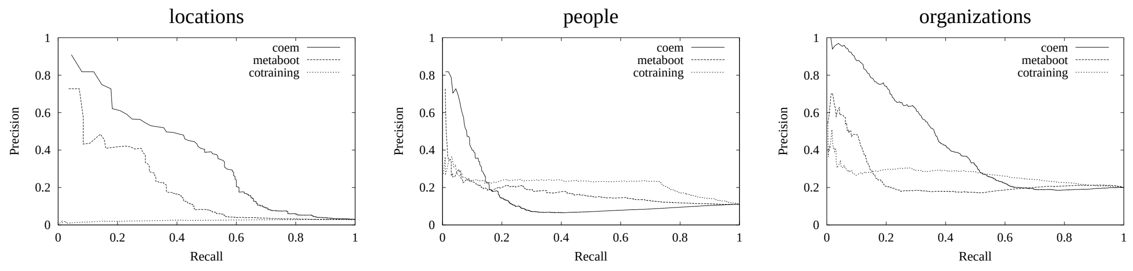


Figure 3.2: Comparison of bootstrapping using coEM, meta-bootstrapping and cotraining, for the classes `locations`, `people` and `organizations`.

Intuitively, precision tells us what proportion of the ones we chose were correct, and recall tells us what proportion of all positive examples we found.

When we want to summarize the results of an experiment with a single number, we use the *breakeven* score. The breakeven score is the value of precision and recall when they are equal on the curve. A higher breakeven score is better.

3.7 Results Comparing Two-view Bootstrapping Algorithms

In this section we give results for two-view algorithms for bootstrapping semantic classes. We will see results for the single-view algorithm EM in Section 3.8.

Figure 3.2 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.

3.7.1 Bootstrapping Improves Over Using Seeds Alone

Figure 3.3 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.

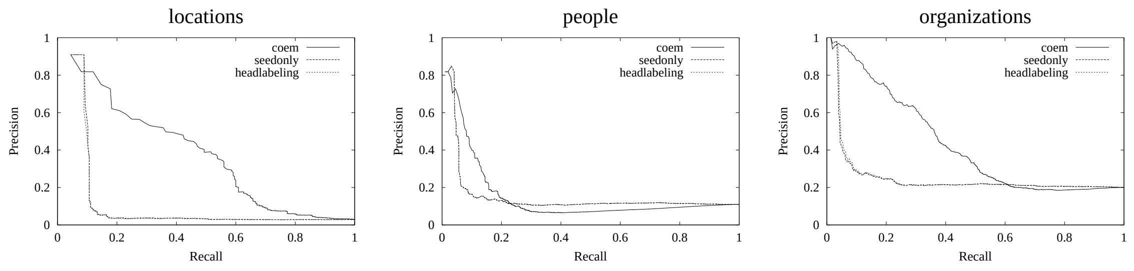


Figure 3.3: Comparison of the effects of using seeds alone, noun-phrases with seeds as heads (head-labeling) and models learned by bootstrapping with coEM to extract on the unseen test set. Seeds and head-labeling lead to good precision, but poor recall. Bootstrapping using coEM improves recall without loss of precision.

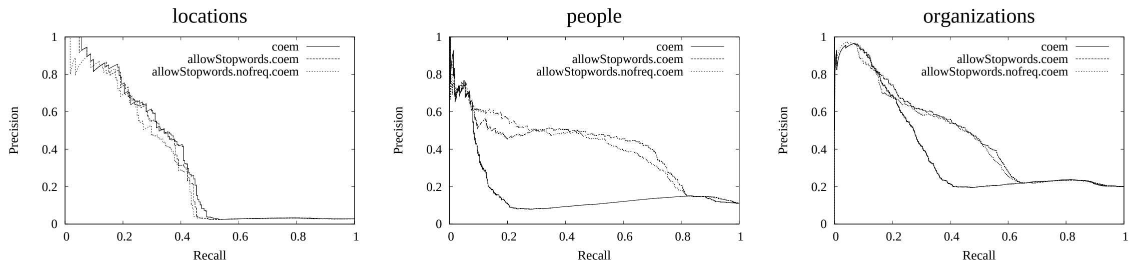


Figure 3.4: Comparison of allowing stopwords in the model, and ignoring frequency information, against forbidding stopwords and using frequency information for coEM. Best results are obtained by allowing stopwords to be used in the model, while frequency information does not appear to affect results greatly.

3.7.2 Using Stopwords is Important for coEM

We see in Figure 3.4 that allowing stopwords in the model improves results greatly for the `people` and `organizations` classes, without having a deleterious effect on the `locations` class. “We” is a very good indicator for organizations, and “he” and “she” are very good indicators for people. For subsequent results we allow stopwords and frequency information for coEM, except where noted.

3.7.3 Metabootstrapping Benefits from Stopwords in Increased Precision

When we allow stopwords in the model for metabootstrapping, gains are achieved at the high-precision end of the precision recall curve, but there is no improvement in recall. Overall the algorithm still does not perform as well as coEM. Results are

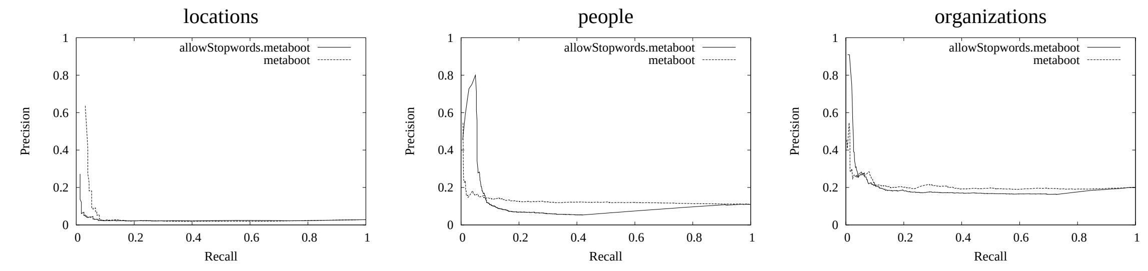


Figure 3.5: Comparison of allowing stopwords in the model, for metabootstrapping. While allowing stopwords gives some improvement, overall the results are still not as good as for coEM.

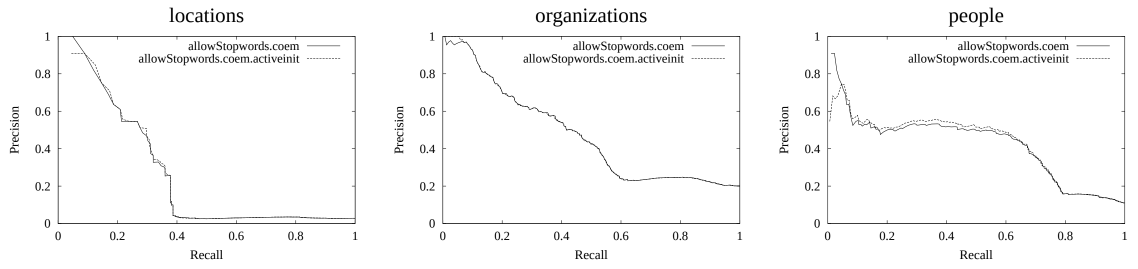


Figure 3.6: Comparison of the effects of hand-labeling all examples matching the seed-words before commencing bootstrapping (active initialization), against bootstrapping assuming all are correct (coEM). A small gain is obtained by labeling all data input.

shown in Figure 3.5.

3.7.4 Small Gains from Correcting Labels for coEM

Figure 3.6 shows that only a small gain is obtained by hand-labeling all 669 unique examples matching the `location` seeds before commencing bootstrapping, all 3406 examples matching the `organization` class, and all 2521 examples matching the `people` class before commencing bootstrapping. We see a slight improvement in precision at low recall for `organizations`, very slight improvements in precision for `locations`, and for the `people` class precision is hurt in the low-recall range, recovering a little at higher recall.

This shows that head-labeling is an effective way of initializing, and correcting errors introduced does not substantially improve results. We will examine whether more intelligent ways of selecting additional examples to label could provide more leverage in Chapter 4.

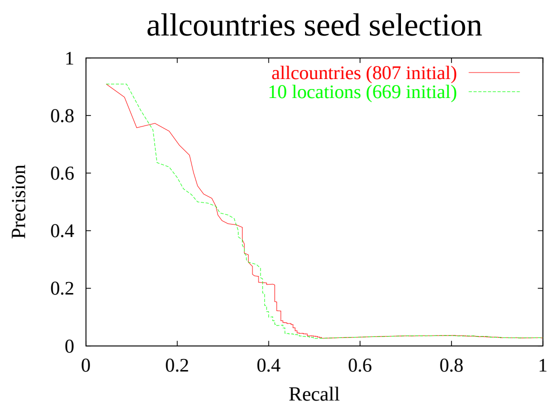
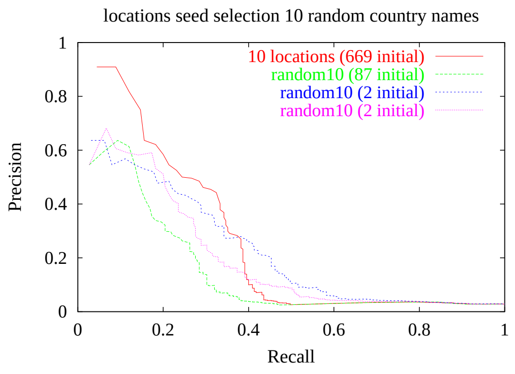


Figure 3.7: The 10 seeds which were chosen for baseline experiments are very effective. Using all 253 country names does not substantially improve results. The 10 original seeds cover a large number of the instances of country names in the *7sector* training corpus.

3.7.5 Initial Seed Choice Influential on Results

Figure 3.7 shows the effects of seed choice on bootstrapping accuracy for the `locations` task with coEM. When we initialize with all 253 country names, we obtain best results, though the 10 country names shown in Table 3.8 are nearly as effective, as they are very frequent in the training set, accounting for most instances of country names.

By producing random subsets of the country names, we obtain seed sets which have varying numbers of examples in the training set. We saw in Table 3.10 that these seed sets vary greatly in the number of initial examples they cover in the *7sector* training set. Figures 3.8 show the results of using 10 and 20 random seeds. The results suggest that we do best with frequent seed words. If our seed words are fixed, we may do well by using a set of unlabeled documents which contain many examples of those seed words. One way of doing this may be by automatically constructing a training corpus by retrieving documents containing seeds, then automatically constructing search queries to retrieve similar documents. Ghani et al. showed this is effective for identifying documents in a specific language (Ghani et al., 2003); it may also be effective for finding documents matching a target domain.



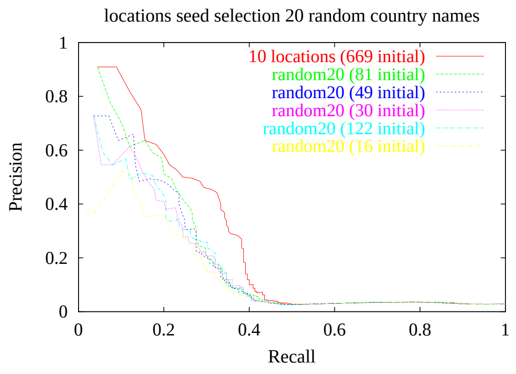


Figure 3.8: A seed set corresponding to more instances of seeds in the training data generally produces better results for coEM.

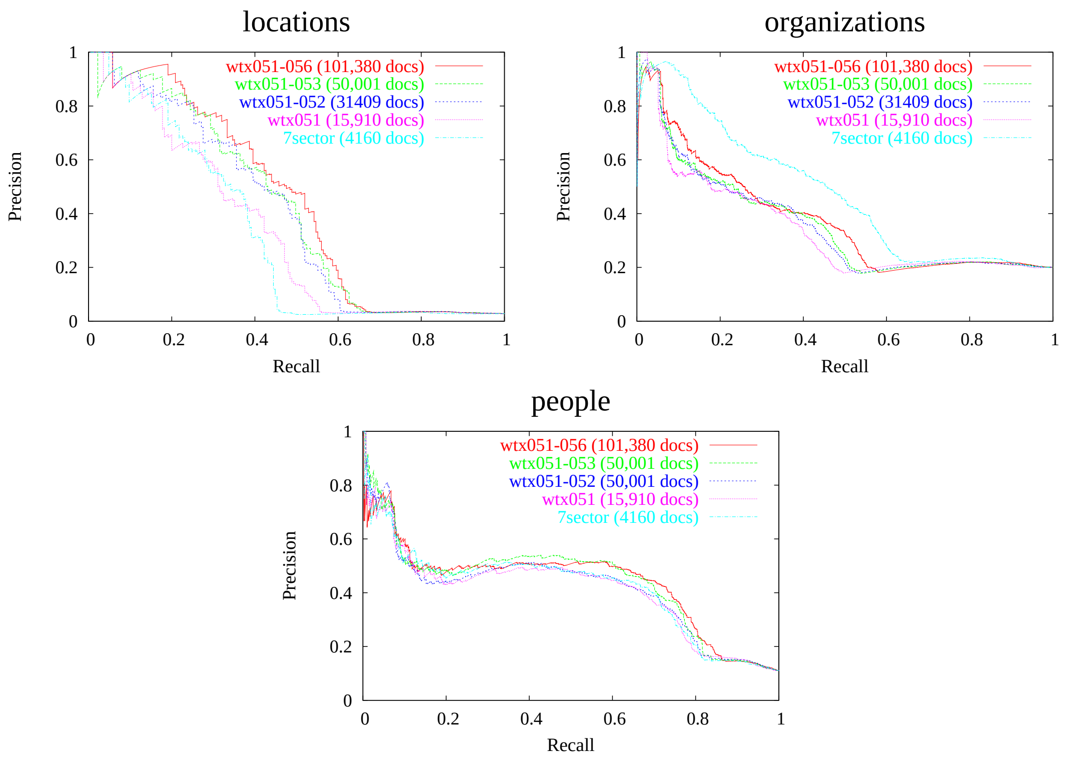


Figure 3.9: Effects of corpus-size for coEM. A larger training set is significantly better for `locations`. However, for `organizations`, a training set which matches the distribution of the test set appears to be a better fit. For the `people` class, a training set which matches the distribution of the test set appears to be a slightly better fit at the low-recall end of the scale.

3.7.6 Training Set Size and Distribution Affects Results

It is frequently the case in machine learning that increasing the training set size increases the accuracy of the learned models. Since we are performing semi-supervised learning, changes in training set size may have a number of effects. Firstly, a larger training set may contain more instances of the seeds we choose, and thus be provided with more initial training information. Secondly, a larger training set may contain a larger vocabulary, and thus semi-supervised learning may learn a model which is applicable to more of the test instances. Note that in machine learning we also expect algorithms to perform best when the training and test data are drawn from the same distribution. Using data from a different distribution may, in our task, have vocabulary and cooccurrence statistics that vary greatly from our test set. Since our original *7sector* training set is a fixed collection, we can increase the size above it only by using data from a different distribution.

As we can see in Figure 3.9, a larger training set is significantly better for `locations`. However, for `organizations`, a training set which matches the distribution of the test set appears to be a better fit. However, larger and larger mismatched training sets get closer to the accuracy achieved by the matched test set. For the `people` class, a training set which matches the distribution of the test set appears to be a slightly better fit at the low-recall end of the scale.

A reasonable question is why different classes benefit differently from increasing corpus size. We may conjecture several explanations for the difference in effect of corpus size on learned model accuracy. First, we may conjecture that the number of seeds matching the training corpus differs across classes. However, we see in Table 3.21 that the number of seeds matching is always as least as large for `people` and `organizations` as it is for `locations`, so this is not a reasonable explanation.

A second potential explanation is the vocabulary size of the target class in the test set. A target class which is represented by a wider vocabulary may benefit more from increased training set size than a target class which is represented by a narrow vocabulary. In Table 3.22 we see the vocabulary size of each class in the test set. We see that both `organizations` and `people` have much larger positive test set vocabulary sizes than `locations`. Thus test set vocabulary does not explain why `locations` benefits more from increased training set size from a different domain.

A third possible explanation is the vocabulary intersection between the training

WT10g name	num docs	num unlabeled examples	num instances matching seeds
7sector	4,160	228,574	894 (locations) 5,257 (organizations) 3,388 (people)
wtx051	15,910	1,829,020	5,240 (locations) 7,105 (organizations) 12,609 (people)
wtx051-052	31,409	3,555,045	10,498 (locations) 15,360 (organizations) 24,737 (people)
wtx051-053	50,001	5,313,224	16,383 (locations) 23,746 (organizations) 37,885 (people)
wtx051-056	101,380	10,588,096	28,233 (locations) 47,319 (organization) 81,975 (people)

Table 3.21: TREC WT10g data subsets used in our experiments, the number of documents in each subset, and the number of examples extracted from each subset. We see that the number of examples matching seeds increases with corpus size.

Class	Positive Test Set Instances	Unique Positive Test Set NPs	Unique Positive Test Set Contexts
`locations`	183	127	115
`organizations`	1099	505	533
`people`	827	230	415

Table 3.22: Number of positive examples, and size of vocabulary of positive examples in test set for classes `locations`, `organizations` and `people`.

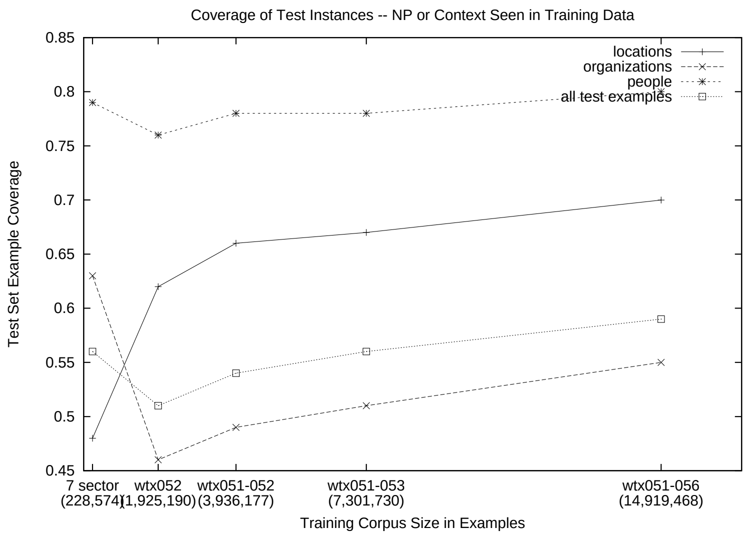


Figure 3.10: Coverage of Positive Test Examples by Some Noun-phrase or Context in the Training Set.

and test sets. If the test set contains vocabulary that is not in the training set, an algorithm may have trouble learning to model a class. We might expect higher intersection of train and test set vocabulary either when the training and test set are drawn from the same distribution, or when the training set is large. We learned in Section 3.5.3 that a greater proportion of the test contexts have been seen in the `7sector` training data, than for the NPs. This was independent of the target class. Figure 3.10 shows the training set instance overlap with positive examples in the test set for all three classes and several training corpora. We see that for `locations`, increasing corpus size corresponds to increasing coverage of positive test examples by some noun-phrase or context in the training set. For all examples, and for `organizations`, the smaller corpora from the TREC wtx data cover less positive test examples. For `organizations`, increasing TREC corpus size approaches the coverage achieved by the *7sector* training corpus, which comes from the same distribution. For the `people` class, size of corpus does not have a big effect on coverage of positive test examples. These effects match the effects of increasing corpus size on extraction accuracy. Thus we can expect to learn well when the test set has much of its vocabulary covered by the training set. This also suggests that transductive learning may be very effective. We will look at the effects of transductive learning in Section 3.7.7.

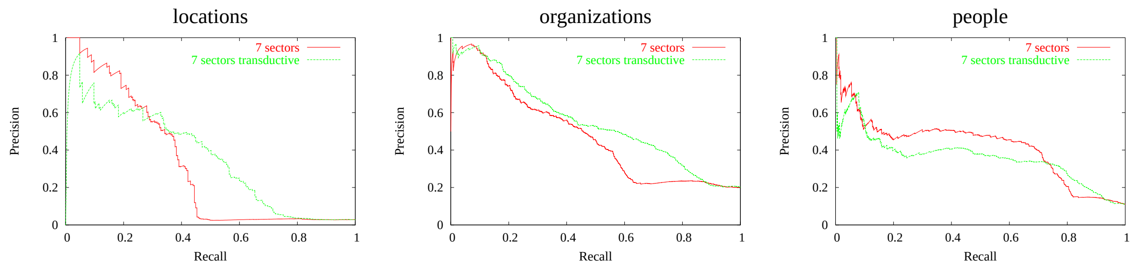


Figure 3.11: Transductive learning versus standard learning for coEM, for the classes `locations`, `organizations`, and `people`.

3.7.7 Transductive Learning Sensitive to Errors in Head-Labeling

Transductive learning is learning using the test data as part of the training data. Joachims (Joachims, 2001) showed effective results for transductive learning for text classification. To implement transductive learning, we combined training and test corpora into a single unlabeled training set, performed head-labeling, trained using coEM, then extracted on the test corpus in the usual way. Recall that the training corpus for the *7sector* dataset consists of 228,574 noun-phrase context pairs, while the test set consists of just 8081 examples. Thus adding the test set does not significantly increase the amount of unlabeled data available for training, though it does give the algorithms the possibility of learning to model the data distribution more accurately for the test set. Figure 3.11 shows the results of transductive learning with the *7sector* dataset. The `organizations` class benefits greatly from transductive learning, because many test examples are covered by NPs or contexts in the training corpus. Thus adding the test examples helps model the class. For `locations` improvements in precision are made at the higher recall levels, with some loss of precision at low recall. The `people` class seems to be hurt by transductive training.

We may wonder why precision is harmed for the locations class. We find that some of the damage is caused by head-labeling. For example, examples containing the company Altos Hornos de Mexico are labeled positive during transductive learning, which hurts precision. In addition, the context “Morgan on” (a misparsing from “J. P. Morgan on”) is learned with high confidence in the transductive case, as it twice occurs with “Mexico”. In the original training set “J. P. Morgan on” occurs only once, and not with a place name. Thus it appears that transductive learning is overfitting to the test set. A solution to this may be to perform head-labeling on the training set only, then learn over both train and test sets together, or to train on the training set, and then run further iterations of bootstrapping on the test set. In the case of the people class, we find the same issues, with known problematic head-labeling resulting in positive labels for "corporate clients" and "industry leader".

We may need to correct the examples labeled by seeds, active initialization, described in 3.7.4, to enable us to make use of the full power of transductive learning. We leave these experiments for future work.

3.8 Results Comparing CoEM and EM

As we saw in Section 3.7, coEM is the most successful of the two-view algorithms described in this thesis. Now we wish to compare coEM to EM, which collapses the two feature sets into a single model. In addition, we will compare the effects of varying parameters to EM, including initialization, use of stopwords, and use of frequency information.

3.8.1 Initialization Conditions

We used four different initialization conditions for EM. Three of these are are different ways of using the unlabeled examples as a source of initial information about the negative class, while the fourth uses the output of coEM for initialization. For *zero initialization*, all unlabeled examples were initialized with $\hat{P}_{init}(pos| < n_i, c_j >) = 0$ as shown in the EM algorithm initialization in Equation 3.4 in Section 3.4.2. For *small initialization* unlabeled examples were initialized with score 0.1. In *random initialization* unlabeled examples were initialized with a random score between 0 and 1.0. Figure 3.12 shows that this initialization affects only the *unlabeled* examples, and not the head-labeled examples, which were initialized using the seeds.

As we will see in Figure 3.17, initializing with zero and small scores are both slightly better than initializing with random scores. This may suggest that the strategy used by each of coEM, metabootstrapping, and cotraining, of treating unlabeled examples as having score 0 initially may be reasonable. Initializing with the output of coEM's models *coeminit* gives the best results. Tables 3.23, 3.24 and 3.25 show results sorted by final breakeven score, for various initialization conditions. These tables also show the number of iterations EM ran before convergence or reaching 500 iterations, as well as the final log likelihood of the training data under the model.

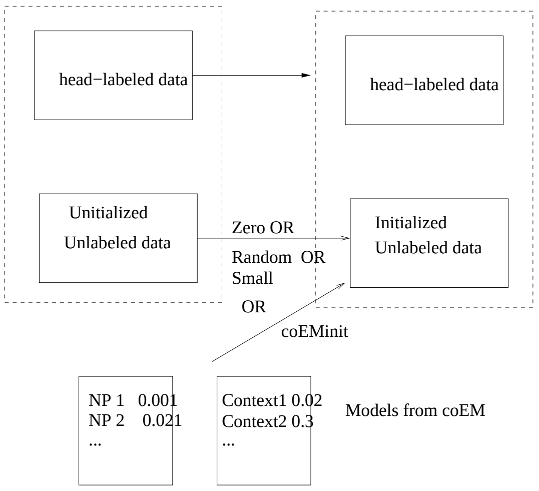


Figure 3.12: Initialization of the unlabeled examples with 0, small values, random values or the output of coEM does not affect the head-labeled examples, which were labeled using the seeds.

We will examine the relationship between log likelihood of the model and breakeven scores in Section 3.8.3.

3.8.2 Frequency Information

In our description of EM in Section 3.4.2 probability estimates were calculated for examples using their counts $\mathrm{N}(n_i, c_j)$. If we collapse repeated occurrences of the same pair of features into a single count, we reduce the impact of very frequent occurrences. Since EM is seeking to maximize the likelihood of the data under the model, examples with very frequent occurrences have a strong effect on the model. However, if our target class does not contain these frequent instances (for example, if our target class does not contain pronouns or dates) then we do not wish frequent instances to have a strong effect on the model learned.

Experiments using this counting method are designated *nofreq*. Figure 3.13 shows that for the `organizations` class it is best to use frequency information, while for the `locations` and `people` classes no major difference is discernible for EM. All remaining experiments with EM use frequency information in the model.

final breakeven	# iter	final ll	init	freq	stopwords
0.116122	17	-3.55614e+06	coeminit	freq	stopwords
0.111612	18	-3.55628e+06	coeminit	freq	nostopwords
0.10823	10	-3.55657e+06	coeminit	nofreq	stopwords
0.105975	500	-3.55844e+06	zero1init	freq	stopwords
0.105975	500	-3.55844e+06	small1init	freq	stopwords
0.105975	500	-3.55777e+06	random1init	freq	stopwords
0.105975	10	-3.55683e+06	coeminit	nofreq	nostopwords
0.104848	500	-3.55861e+06	zero1init	freq	nostopwords
0.104848	500	-3.55861e+06	small1init	freq	nostopwords
0.104848	500	-3.55789e+06	random1init	freq	nostopwords
0.100338	8	-3.55835e+06	random1init	nofreq	nostopwords
0.0980834	9	-3.55827e+06	random1init	nofreq	stopwords
0.0980834	351	-3.56215e+06	zero1init	nofreq	stopwords
0.0980834	269	-3.56225e+06	zero1init	nofreq	nostopwords
0.096956	6	-3.56135e+06	small1init	nofreq	nostopwords
0.096956	6	-3.56126e+06	small1init	nofreq	stopwords

Table 3.23: For `people` with EM, best results are obtained by initializing from coEM, using frequency information, and allowing stopwords in the model.

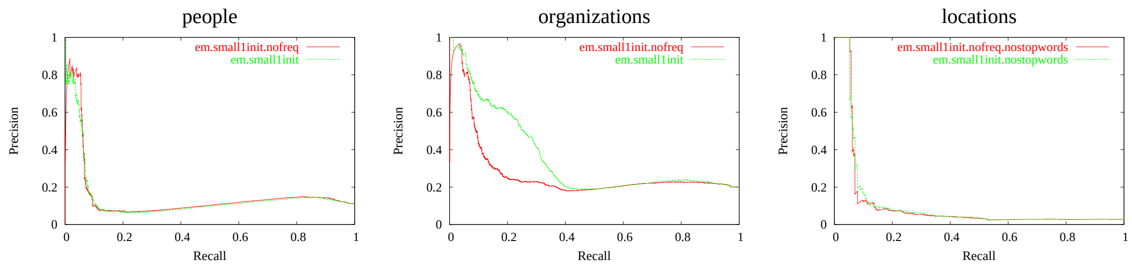


Figure 3.13: Results for EM with and without frequency information.

final breakeven	# iter	final ll	init	freq	stopwords
0.155556	16	-3.5594e+06	random1init	freq	stopwords
0.155556	15	-3.55954e+06	random1init	freq	nostopwords
0.128889	500	-3.5628e+06	coeminit	freq	nostopwords
0.128889	500	-3.56246e+06	coeminit	freq	stopwords
0.124444	93	-3.56857e+06	zero1init	freq	stopwords
0.124444	91	-3.56858e+06	zero1init	freq	nostopwords
0.12	8	-3.55751e+06	random1init	nofreq	nostopwords
0.12	8	-3.55736e+06	random1init	nofreq	stopwords
0.12	500	-3.56489e+06	small1init	freq	stopwords
0.12	14	-3.56541e+06	small1init	freq	nostopwords
0.115556	5	-3.5599e+06	coeminit	nofreq	stopwords
0.115556	5	-3.55999e+06	coeminit	nofreq	nostopwords
0.115556	5	-3.55991e+06	small1init	nofreq	nostopwords
0.115556	5	-3.55986e+06	small1init	nofreq	stopwords
0.111111	48	-3.5608e+06	zero1init	nofreq	stopwords
0.111111	48	-3.5608e+06	zero1init	nofreq	nostopwords

Table 3.24: For `locations` with EM, best results are obtained by using random initialization and frequency information, and allowing stopwords in the model.

final breakeven	# iter	final ll	init	freq	stopwords
0.488243	23	-3.54627e+06	coeminit	freq	stopwords
0.375	15	-3.55062e+06	coeminit	nofreq	stopwords
0.331064	18	-3.54967e+06	coeminit	freq	nostopwords
0.328589	500	-3.5529e+06	small1init	freq	stopwords
0.328589	500	-3.55291e+06	zero1init	freq	stopwords
0.327351	500	-3.55309e+06	zero1init	freq	nostopwords
0.327351	500	-3.55309e+06	small1init	freq	nostopwords
0.326733	500	-3.55251e+06	random1init	freq	stopwords
0.326114	500	-3.55274e+06	random1init	freq	nostopwords
0.322401	14	-3.552e+06	coeminit	nofreq	nostopwords
0.245668	9	-3.5581e+06	random1init	nofreq	stopwords
0.243193	9	-3.55828e+06	random1init	nofreq	nostopwords
0.237005	500	-3.56039e+06	zero1init	nofreq	nostopwords
0.237005	500	-3.56039e+06	small1init	nofreq	nostopwords
0.237005	500	-3.56026e+06	zero1init	nofreq	stopwords
0.237005	500	-3.56026e+06	small1init	nofreq	stopwords
0.165842	6	-3.56584e+06	small1init	freq	stopwords

Table 3.25: For `organizations` with EM, best results are obtained by initializing from coEM, using frequency information, and allowing stopwords in the model.

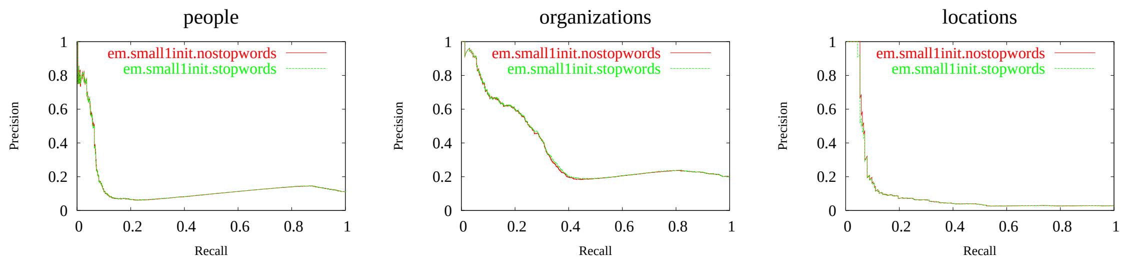


Figure 3.14: Results for EM with and without stopwords.

3.8.3 Effect of Stopwords on EM

Unlike with coEM, our experiments with EM show little difference between allowing and disallowing stopwords in the model. Figure 3.14 shows this using the *small* initialization condition. We also saw in Tables 3.23, 3.24 and 3.25 that for most initialization conditions, results are very slightly better allowing stopwords in the model. All subsequent experiments with EM allow stopwords in the model.

Likelihood of Data Under EM

EM will work well for a class for which accuracy correlates well with log likelihood of the training data under the model. Figure 3.15 show the breakeven score on the test set plotted against the log likelihood of the training data under the model, through training iterations of EM. We see that an increase in log likelihood correlates with an increase of the breakeven score for the `organizations` class, but less so for the `people` and `locations` classes. For the `locations` class, the breakeven point drops slightly with increased log likelihood of the training data under the model. These results explain why EM is most successful for the `organizations` class.

3.8.4 Corpus Size for EM

We see in Figure 3.16 that using a larger corpus helps a little with EM for the `people` class. For the `locations` class, however, it is not clear if adding more documents helps when using EM. This contrasts with coEM, which benefited greatly from added documents for `locations`, as we saw in Figure 3.9.

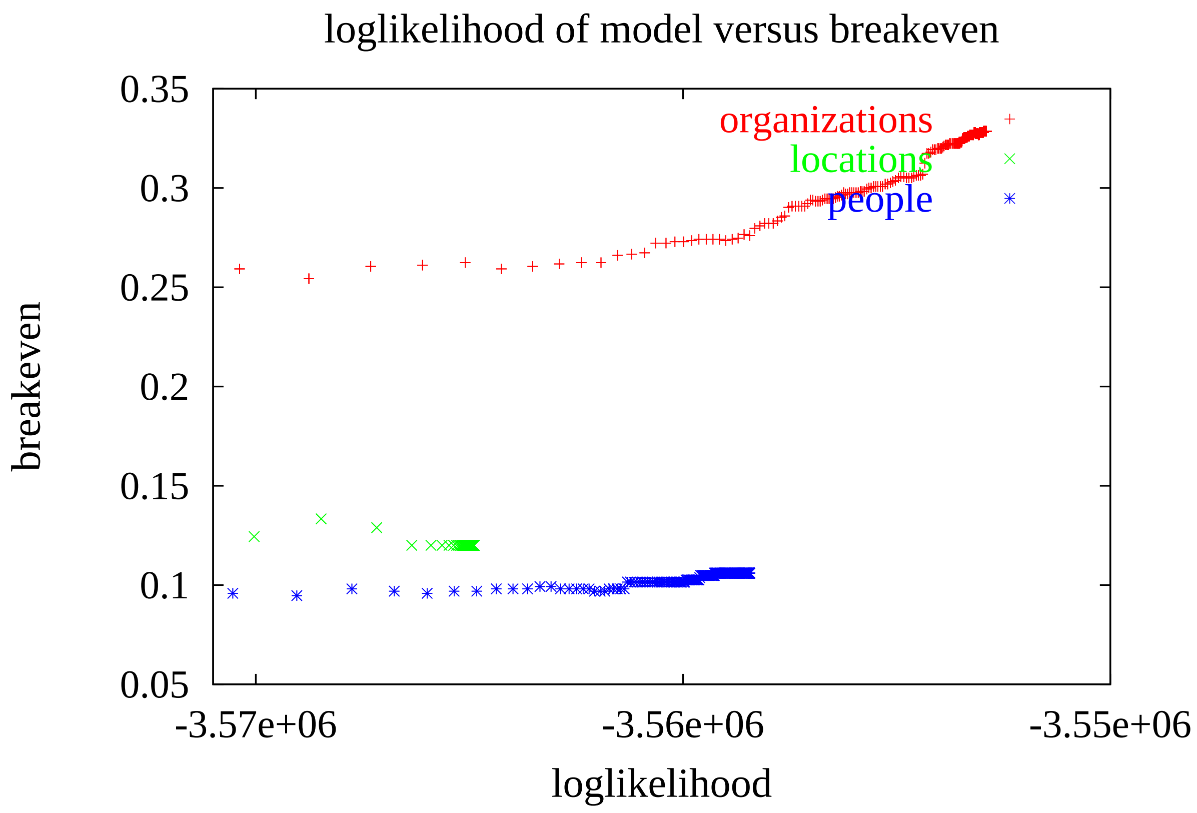


Figure 3.15: Log likelihood plotted against breakeven point. We see that log-likelihood is predictive of accuracy for the `organizations` class, but less so for the `people` and `locations` classes.

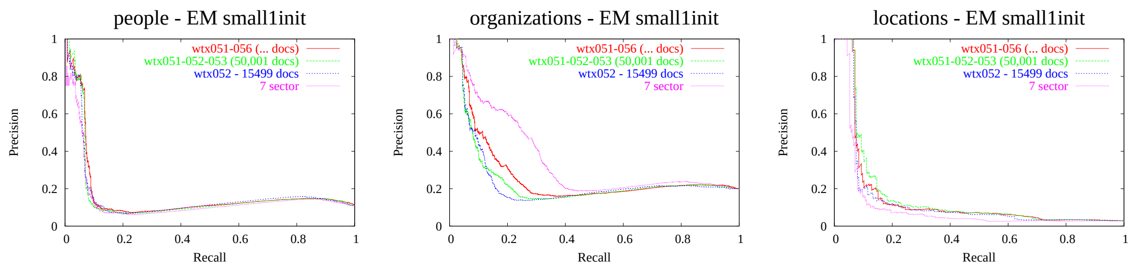


Figure 3.16: Results for EM with large corpora.

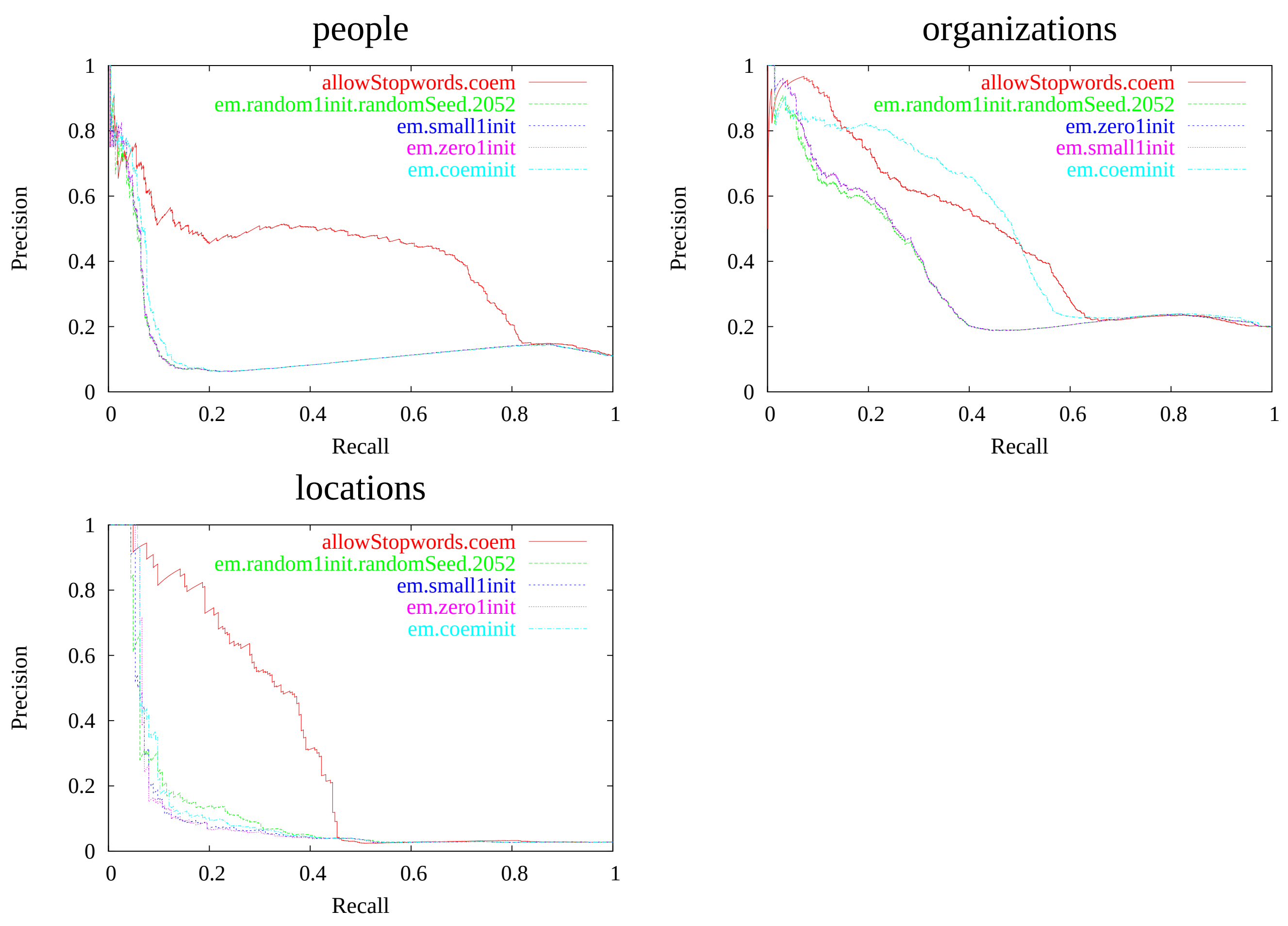


Figure 3.17: Comparison of EM, against baseline coEM algorithm. All initialization conditions are shown for EM. Stopwords were permitted in the model, and frequency information used for both EM and coEM.

3.8.5 EM versus CoEM

In Figure 3.17 coEM outperforms EM for the `locations` and `people` tasks. Stopwords were permitted in the model, and frequency information was used, for both EM and coEM. coEM outperforms EM for all initialization conditions for the `locations` and `people` tasks. For `organizations` task however, when we initialize EM with the output of coEM, the breakeven point is somewhat better than for EM with random, zero or small initialization.

3.9 Chapter Conclusions

We will now go over the questions we raised at the start of this chapter, in the light of knowledge provided by our experimental results.

How can we represent data so we can use it to learn to extract instances of semantic classes?

We found that representing the data as pairs of (1) noun-phrases and (2) their local syntactic contexts allowed us to learn to extract instances of semantic classes, using a small number of initial examples. We noted in Section 3.5.2 that when 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. We could use other representations, for example n-grams, as has been used in other work (Sarawagi & Cohen, 2004). We could also augment the representation by incorporating whether examples had capital letters or punctuation (Collins & Singer, 1999). We won't experiment with alternative data representations in this thesis, but it is worth noting that the amount of context and information we use could be augmented, with the possibility of improved results.

What algorithms can we use for bootstrapping?

We found that in general coEM outperformed EM, except when coEM's output was used for initialization, and increases in the breakeven point for the class correlated with increased log likelihood of the training data under the model, as was the case for the `organizations` class. This suggests that EM could be effective for the `organizations` class, given good initialization conditions.

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.

How much does bootstrapping contribute, over using the seeds alone for bootstrapping?

We can see from the comparison of gains from bootstrapping over using the seeds or head-labeling (see Figure 3.3), that classes for which we have ambiguous seeds words, such as our `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.

Does it matter which seeds we choose?

For the seed-words and datasets we used, we saw in Figure 3.8 that seed density in the training corpus appears to affect the accuracy of the results. We should select seeds, or corpora, such that many occurrences of seeds are in the training corpus.

Should we correct any errors introduced by using seeds?

For `locations` and `people` we saw in Figure 3.6 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 transductive learning, however, we should be sure to correct the examples labeled by seeds.

Can we learn all classes equally well with the same representation?

We found that the `people` and `organizations` classes were sensitive to the presence or absence of stopwords. This may be due to pronouns which are correlated with the class. For example "we" may be a good predictor of `organizations`, while "he" or "she" may be a good predictor of `people`. We will examine this question in more detail in Chapter 5. Since the `locations` class was not adversely affected by allowing stopwords in the model, we can use the same data representation for all three classes.

We also saw in Section 3.7.6 Figure 3.10 that the classes varied greatly in how much of their test vocabulary was covered by the training corpus. We can expect to learn better when the test set has much of its vocabulary covered by the training set.

How does corpus size affect learning?

We saw in Sections 3.7.5 and 3.7.6 that both number of seeds in the training corpus and corpus size affect results. We may wonder if there is a way of increasing the number of seeds in the training corpus without greatly increasing the corpus size, for greater computational efficiency. One way may be to label more examples. We will address this question in Chapter 4 by examining the most efficient ways to choose examples for labeling, using active learning. A different approach, motivated by related work in acquiring documents in a target language (Ghani et al., 2003), would involve automatically acquiring training data that is likely to contain many instances of the seeds and the target class. This may be a promising direction for future work.

What assumptions do the algorithms make about the data representation, and how well are those assumptions satisfied?

We have assumed for extraction with the models described in this chapter that noun-phrases and contexts are conditionally independent given their class labels, though we do not expect this to hold completely. It would be interesting to measure the actual level of independence, and examine whether gains can be made by dropping this assumption. We will perform this measurement in Chapter 5. As we discussed in the related work in Chapter 2 Section 2.7.3, Balcan et al (2004) have showed that conditional independance given the target class may not be necessary for bootstrapping.

The algorithms also assumed that seeds will be present in the data, which we saw depends on the choice of seeds. There is an implicit assumption we have not addressed until now, that the bootstrapping algorithms we employ are able to modify the labels of all examples, by modifying labels based on cooccurrence, *ie* that following cooccurrence links will allow us to reach all relevant examples. We will examine this assumption more deeply in Chapter 5.

3.10 Chapter Conclusions

In this chapter we explored a number of variations on initialization for algorithms for bootstrapping information extraction. There are more combinations of these which could prove interesting to pursue. These include extending algorithm initialization to permit the user to specify verbs and prepositional phrases, as well as nouns. It would also be interesting to try using negative seeds for coEM and EM, and active initialization for EM.

In the next chapter we will examine how we could improve on these results by incorporating active learning. We will look at active learning algorithms customized to the feature-set split, which may lead to more efficient learning.

Chapter 4

Active Learning for Semi-supervised IE

We saw in Chapter 3 that coEM is effective for bootstrapping the learning of semantic classes from a small collection of seed-words. This requires only minutes of user training time. However, if the user has more time for labeling, it may be useful for the user to spend more time labeling if this leads to increased accuracy. In this chapter we show that judicious selection of examples for labeling can lead to greatly increased accuracy, without greatly increasing the burden on the user. In particular, we show that using the feature set redundancy allows selection of examples for labeling which are much more effective than examples chosen randomly, or without use of feature set redundancy. In addition, these results show that active learning can compensate for a bad choice of initial seeds and that the labeling effort is better spent *during* the active learning process rather than at the beginning.

4.1 Introduction

Active learning seeks to make efficient use of a trainer's time by having the learner intelligently select examples to label based on the anticipated value of that label to the learner. The bootstrapping approaches considered in Chapter 3 make use of the fact that each example is described by two distinct sets of features, either of which is sufficient to approximate the function; that is, they fit the cotraining problem setting. We discuss a range of active learning algorithms and show that using feature set disagreement to select examples for active learning leads to improvements in extraction performance regardless of the choice of initial seeds.

Active labeling is performed in an interleaved manner with bootstrapping. As we described in Chapter 3, we initialize the data with seeds using head-labeling, and then use both head-labeled and unlabeled examples as data for bootstrapping, and as candidates for active labeling. Figure 4.1 shows a schematic of this process.

The questions we will attempt to answer in this chapter are:

1. Is it more effective to correct head-labeled examples, or actively label new examples?
2. What is the most effective way to select new examples to actively label?
3. Is it more effective to actively label just noun-phrases, or to actively label noun-phrase context pairs, in active learning?
4. If we have a larger corpus to choose examples from, does this lead to greater accuracy for the same amount of user labeling effort?
5. How much does increasing the number of actively labeled examples improve accuracy? Do improvements tail off, or do we continue to get increased accuracy as we increase the number of labeled examples?
6. Does active learning make bootstrapping algorithms more robust to the set of seeds chosen for head-labeling?
7. Does active learning make bootstrapping algorithms more robust to the way unlabeled examples are initialized?

Table 4.1 summarizes the dimensions we will experiment with in this chapter. Some of these dimensions, such as training set size, have already been addressed in Chapter 3 but they may interplay in different ways when we add active learning. We will give more details about each in the sections below.

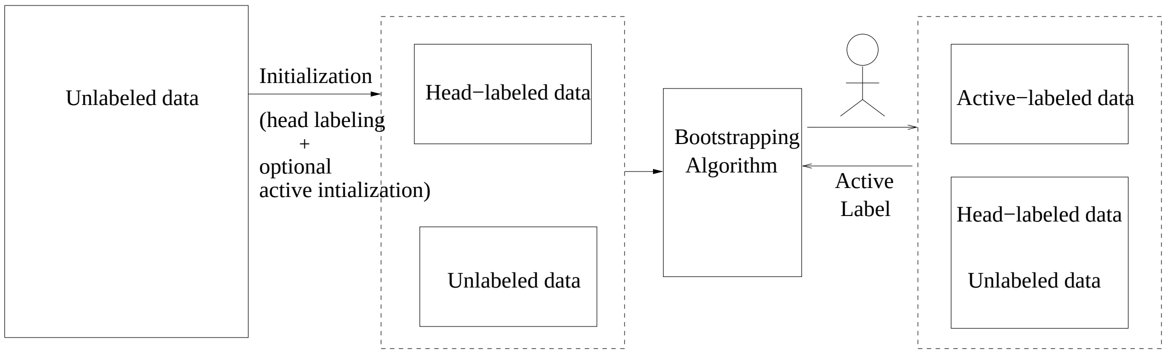


Figure 4.1: Automatically labeling data using seeds with head-labeling, optionally correcting these labels with active initialization, then performing active learning interleaved with bootstrapping. Both unlabeled and head-labeled examples are candidates for active labeling. Active initialized examples are treated as active-labeled examples, and are not candidates for relabeling.

Dimension	Instantiations
Training set size	230,000 - 10 million examples
Training set distribution	same as test set; different from test set
What user labels	whole example or just noun-phrase
Number of examples labeled	0, 5, 25, 100, 500, 2500
Algorithm used to select examples	uniform random, density, feature set disagreement, context disagreement
Initialization conditions	unlabeled examples 0, small or random; choice of seedwords
Properties of the model	frequency, stopwords
Algorithm used for bootstrapping	coEM or EM

Table 4.1: Summary of dimensions we can vary when performing active learning for bootstrapping semantic classes of noun-phrases.

4.2 Related Work

Using a pool of unlabeled examples and prompting the user to actively label examples that have high anticipated value reduces the number of examples required for tasks such as text classification (Lewis & Gale, 1994) and parsing and information extraction (Thompson et al., 1999; Soderland, 1999). Bootstrapping algorithms for similar learning problems fall into the cotraining setting (Blum & Mitchell, 1998; Collins & Singer, 1999; Muslea et al., 2000), *ie.* they have the property that each example can be described by multiple feature sets, any of which are independently sufficient to approximate the function.

The cotraining problem structure lends itself to a variety of active learning algorithms. In *naïve co-testing* (Muslea et al., 2000) the two classifiers are trained on available labeled data, then run over the unlabeled data. A *contention set* of examples is then created, consisting of all unlabeled examples on which the classifiers disagree. Examples from this contention set are selected at random, a label is requested from the trainer, both classifiers are retrained, and the process repeats.

While this co-testing algorithm was shown to be quite effective, it represents just one possible approach to active learning in the co-training setting. It is based on training the two classifiers $\hat{g_1}$ and $\hat{g_2}$ using labeled examples only, whereas earlier work (Collins & Singer, 1999; Blum & Mitchell, 1998; Riloff & Jones, 1999) has shown that unlabeled data can bootstrap much more accurate classifiers. Instead of selecting new examples uniformly at random from the contention set, one might rank the examples in the contention set according to some criterion reflecting the value of obtaining their label. In this chapter, we propose and experiment with active learning algorithms that use unlabeled data for training $\hat{g_1}$ and $\hat{g_2}$, in addition to using $\hat{g_1}$ and $\hat{g_2}$ to determine which unlabeled example to present to the trainer. We also consider a variety of strategies for selecting the best example from the contention set.

4.3 Training Set Size and Distribution

We saw in Chapter 3 Figure 3.9 that a larger training set is more effective for `locations`, even when the domain of train and test are mismatched. We also expect active learning to increase accuracy for a bootstrapping algorithm (we will see this in Section 4.10.2, for example). Given these two background empirical facts, we wish to find out whether a greater win is obtained by increasing the training set size, that is, increasing the size pool of unlabeled examples available for bootstrapping and selecting examples from, or by increasing the number of examples labeled.

4.4 Whole Example Labeling Versus Single Feature Set Labeling

Typically in active learning a labeler examines a whole example in order to assign a label, using what we call the *standard labeling* paradigm. In the context of our information extraction task, standard labeling would ask the user to label a pair consisting of a noun-phrase and its context. However, it may be as easy to actively label noun phrases independent of context, and since each noun phrase may occur in many contexts, this may lead to greater economy in labeling. For example, "Italy" occurs with "centers in < >", "operations in < >", "introduced in < >", "partners in < >", and "offices in < >", so labeling "Italy" provides us with information about all of these contexts. We will call the approach of labeling noun-phrases and applying the labels to the whole example *single feature set labeling*.

4.5 Number of Examples Labeled With Active Learning

Since labeling more examples provides us with more information about the target function, we expect increases in the number of examples labeled with active learning to improve overall performance. However, different example selection methods may be more effective at finding informative examples for labeling, and thus increase performance at different rates. We may also find that even labeling very few examples provides a great deal of benefit, making active learning an attractive addition to bootstrapping, even if the user has very little time for labeling. We will compare no active learning (zero examples labeled) against tiny numbers of examples labeled (5 examples labeled) up to 2500 examples labeled (around 1% of the examples in the 7 sector dataset).

4.6 Active Learning Selection Methods

Uniform Random Selection: This baseline method selects examples according to a uniform distribution. Each noun-phrase, context pair that occurs at least once in the training set is selected with equal probability. Example frequency is ignored. This method is applicable to both standard labeling, and single feature set labeling. With single feature set labeling, each noun-phrase is selected with equal probability.

Density Selection: The most frequent unlabeled example is selected for labeling at each step. This method is applicable to both standard labeling, and single feature set labeling. With standard labeling, the most frequent unlabeled noun-phrase context pair is selected. With single feature set labeling, the most frequent unlabeled noun-phrase is selected.

Feature Set Disagreement: Since we learn two distinct classifiers that apply to the same instance, one way to select instances where a human trainer can provide useful information is to identify instances where these two classifiers disagree. This approach can be viewed as a form of *query-by-committee* (QBC), (Freund et al., 1997; Liere & Tadepalli, 1997; Muslea et al., 2000) or *uncertainty sampling* (Lewis & Gale, 1994; Thompson et al., 1999), where the committee consists of models that use different feature sets and is similar to that used by (McCallum & Nigam, 1998b). Our selection criterion is based on Kullback-Leibler (KL) divergence. It gives each example a density-weighted KL score, by multiplying $KL(\hat{P}_{g_1}(+|x), \hat{P}_{g_2}(+|x))$ by the frequency of the example.

Let the set of predictors be $g_1..g_n$ then the mean of the scores they assign an example is

$$\hat{P}_{mean}(class|x) = \frac{\sum_i \hat{P}_{g_i}(class|x)}{n} \tag{4.1}$$

$$\hat{P}_{mean}(\neg class|x) = \frac{\sum_i \hat{P}_{g_i}(\neg class|x)}{n} \tag{4.2}$$

Then the score assigned by KL is given by

$$KL(x) = \frac{1}{n}\sum_i \hat{P}_{g_i}(class|x) log \frac{\hat{P}_{g_i}(class|x)}{\hat{P}_{mean}(class|x)} + \frac{1}{n}\sum_i \hat{P}_{g_i}(\neg class|x) log \frac{\hat{P}_{g_i}(\neg class|x)}{\hat{P}_{mean}(\neg class|x)} \tag{4.3}$$

Our overall score is then given by

$$KL(x) \times freq(x) \tag{4.4}$$

Examples are selected deterministically, with the highest ranked unlabeled example taken each time. We use a total of two committee members, with one committee member $\hat{g}_1$ to represent the probability assigned by the noun-phrase, and a second committee member $\hat{g}_2$ to represent the probability assigned by the context. A possible extension motivated by earlier work on query-by-committee would have several committee members for the noun-phrase, and several for the context. These committee members would differ from each other either by being sampled from a probability distribution, or by being derived from different initialization conditions. This method is applicable only to the standard labeling paradigm. A variant applicable to single feature set labeling, *context disagreement*, is described below.

Context Disagreement: We noted earlier that since each noun phrase may occur in many contexts, this may lead to greater economy in labeling. For example, "Italy" occurs with "centers in < >", "operations in < >", "introduced in < >", "partners in < >", and "offices in < >", so labeling "Italy" provides us with information about all of these contexts.

In addition, we may find that while some contexts of a noun-phrase suggest it is positive, others suggest it should be classified as negative. We can take probabilities assigned by the different contexts as votes by committee members about the label for the noun-phrase. Disagreement among these committee members may suggest that labeling the noun-phrase would be informative. Selecting the noun-phrase with the most *context disagreement* may provide us with the most informative labeling. This can be thought of as query-by-committee (QBC) with the committee consisting of different cooccurrences of elements of one feature set with elements of the other feature set. We quantify context disagreement using density weighted KL divergence to the mean, as in feature set disagreement, but all the contexts of the noun-phrase are used as input to the KL divergence measure.

The classifiers g_i simply use the scores assigned by the model to the probability of class membership given the context c, for all contexts that the noun-phrase n occurs in:

$$KL(n) = \frac{1}{n}\sum_i \hat{P}_{g_i}(class|c)log\frac{\hat{P}_{g_i}(class|c)}{\hat{P}_{mean}(class|c)} + \frac{1}{n}\sum_i \hat{P}_{g_i}(\neg class|c)log\frac{\hat{P}_{g_i}(\neg class|c)}{\hat{P}_{mean}(\neg class|c)} \quad (4.5)$$

We use the frequency of the noun-phrase to density-weight the KL divergence. The score is then given by

$$KL(n)freq(n) \quad (4.6)$$

The user then labeled noun-phrases, in *single-feature set labeling.*

4.7 Initialization Conditions

As we saw in Chapter 3 Section 3.8.1 we can initialize the unlabeled examples for bootstrapping with EM in several different ways; we can initialize unlabeled examples with a score of zero, a small constant, a random value, or with the scores assigned by coEM. We saw in Chapter 3 that the way we initialize unlabeled examples affects the performance of EM for our bootstrapping task. When we have more examples labeled with active learning, we expect the influence of the initialization of unlabeled examples to have less impact on the overall performance. We will examine to what extent active learning reduces the impact of initialization conditions for unlabeled examples.

We also saw in Chapter 3 Section 3.5.1 that the choice of seeds for initial labeling of positive examples with head-labeling can lead to quite different numbers of examples labeled, and also to different overall algorithm effectiveness (Section 3.7.5). We will examine in this chapter whether active learning can make bootstrapping more robust to the initial choice of seeds.

4.8 Properties of the Model

We saw in Chapter 3 Section 3.7.2 that including or disallowing stopwords in the model has a substantial effect on bootstrapping effectiveness, as does whether we factor example frequency into our model. We will examine how these properties of the model come into play when we add active learning to our bootstrapping algorithms.

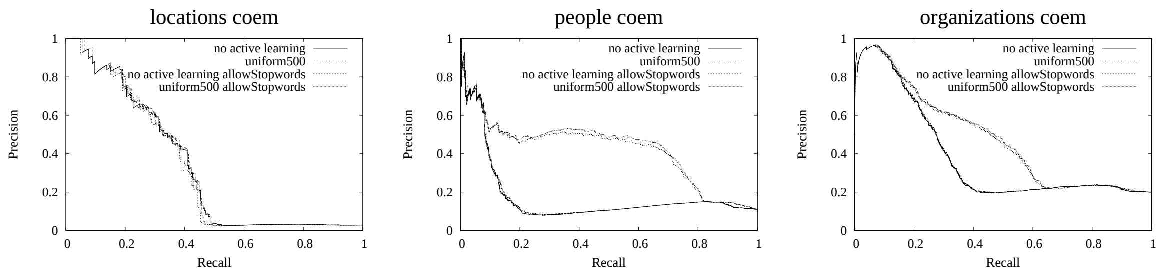


Figure 4.2: We compare a baseline of no active learning to adding 500 labeled examples, examples selected uniformly at random, for *locations*, *people* and *organizations*, interleaved with bootstrapping with coEM. We also compare models which allow or disallow stopwords.

4.9 Bootstrapping Algorithms

4.9.1 Choice of Bootstrapping Algorithm

In this chapter we examine active learning for coEM, and EM. We chose coEM as it performed reasonably well for bootstrapping with no human annotated data at all (Chapter 3), and EM because it often performs well for semi-supervised tasks (see eg. (Nigam & Ghani, 2000)). Although in our experiments in Chapter 3 EM performed poorly for bootstrapping with only head-labeled data, it may perform better when we add a little more labeled data with active learning.

4.9.2 Combining Bootstrapping with Active Learning

We interleave active learning with the bootstrapping algorithm. We use coEM or EM to bootstrap for one iteration, and then actively label five examples per iteration, until the target number of examples have been labeled with active learning. Then, we continue running the bootstrapping algorithm till convergence or till 500 iterations total. We sort the test instances according to the score assigned by the extraction method and calculate precision-recall values.

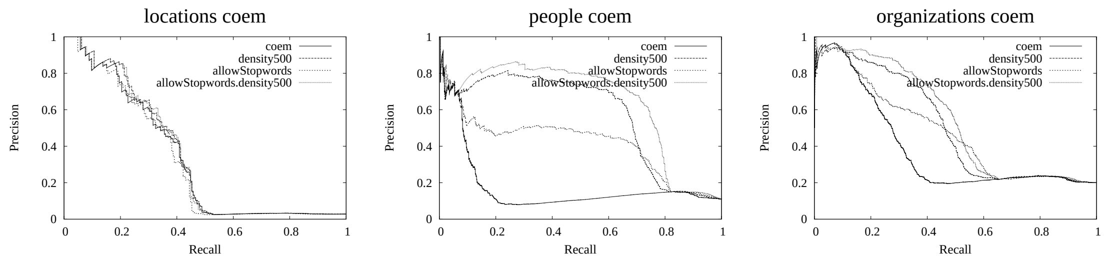


Figure 4.3: Adding 500 labeled examples, examples selected according to their density, for *locations*, *people* and *organizations*.

4.10 Results

4.10.1 Uniformly Selected Labeled Examples of Little Utility

Allowing stopwords to influence the model has a much greater effect for the people class, than adding 500 labeled examples selected uniformly at random. Adding the 500 uniformly selected examples to the model allowing stopwords does not provide any additional leverage, as can be seen in Figure 4.2. This is supported by research in the literature, which shows that a model whose assumptions about the distribution of data are not greatly divergent from the empirical data distribution benefits more from unlabeled data (Cozman & Cohen, 2002). The amounts of data we are labeling here are small compared to the size of the overall unlabeled data set, which contains 228,574 examples, and the size of the test set, which contains 8,081 examples. We will examine the effect of labeling more examples in Section 4.10.6.

4.10.2 Example Selection Based on Density

As we see in Figure 4.3 selecting examples according to their density provides an improvement over the baseline. This shows that selecting examples according to a *non-uniform* distribution can help learn, particularly for the *people* class. Even when we do not allow stopwords to influence the model, examples selected in this way aid greatly in learning the people class. Stopwords such as pronouns tend to be the most frequent examples.

Since the selection criterion is based on example frequency and not the model or the target class, the same 500 examples are chosen for labeling for all experiments on the same training set. When we examine the 500 examples selected for labeling for

noun-phrase	number of unique examples	total examples labeled	most frequent context (frequency)
you	69	2460	you LIKE (166)
we	42	1435	we BELIEVE (123)
us	14	354	CONTACT us (90)
share	12	44	share INCREASED (15)
company	11	258	company OPERATES (44)
this	10	247	this PRESS (38)
it	10	200	it PROVIDES (31)
1997	9	364	ENDED 1997 (114)
1996	9	373	ENDED 1996 (114)
trademarks	7	181	REGISTERED trademarks (65)

Table 4.2: Noun-phrases selected for labeling when most frequent *examples* (noun-phrase context pairs) are selected for labeling. Frequencies given here are number of *different* contexts these noun-phrases occurred in, though selection was based on number of *occurrences* of those contexts.

density-based active learning, we find that only 211 distinct noun-phrases appeared among them. Table 4.2 shows the noun-phrases that occurred in the most examples selected for labeling.

Our labeler labeled 69 different <noun-phrase, context> examples which contained the noun-phrase “you”, and 11 different examples which contained the noun-phrase “company”. However, as we learned in Chapter 3, noun-phrases alone are relatively unambiguous with respect to the classes we are attempting to learn. Thus we may be able to economize on labeling time by avoiding redundancy in the noun-phrases labeled. One way of doing this is to ask the labeler to actively label noun-phrases alone, using *single-feature set labeling.*

We describe experiments and results examining this approach in the next section.

4.10.3 Single Feature Set Labeling

When we ask our labeler to actively label only noun-phrases, the labeled noun-phrases are assigned probability 0 or 1 of class membership, depending on the label.

In Figure 4.4 we find that labeling frequently occurring NPs is effective for `people`,

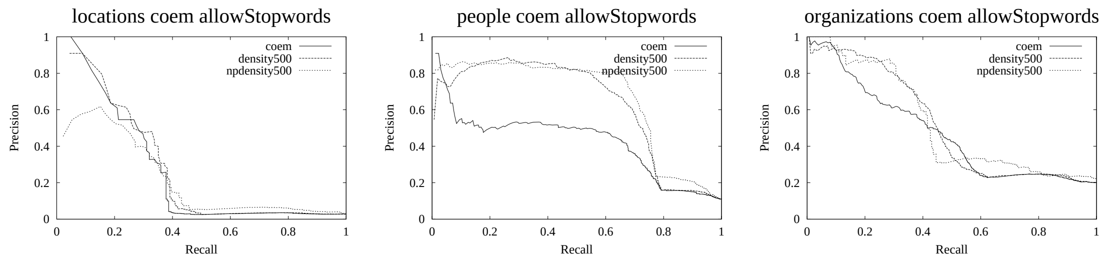


Figure 4.4: Adding 500 labeled noun-phrases, selected according to their density, for *locations*, *people* and *organizations*.

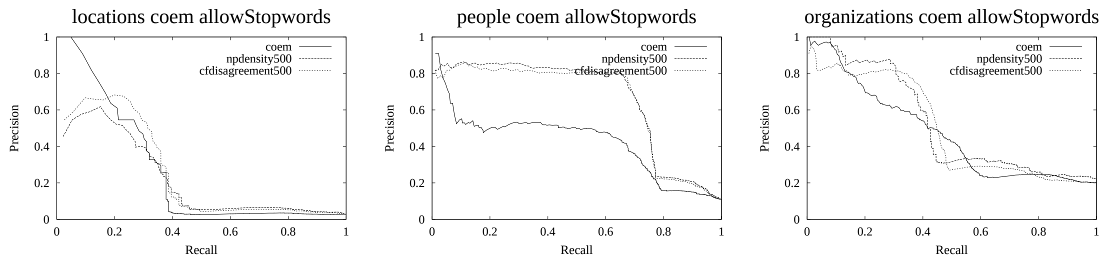


Figure 4.5: Adding 500 labeled noun-phrases, selected according to context disagreement, for *locations*, *people* and *organizations*.

whereas for `companies` we are best off labeling entire examples. For the *locations* class, labeling NPs in isolation harms precision. This is surprising – most of the examples are not in the location class, only 24 of the 500 examples labeled were labeled as positive. However, we find that ambiguity in the noun-phrases leads to *incorrect labeling* by the oracle. For example, "capital" was labeled as positive, however in our test set, "capital" refers to the company. For `organizations` we find similar effects. Noun-phrases which do not obviously refer to companies when seen in isolation, including "energy solutions" and "vcs technologies" actually are company names in our dataset. Using capital letters as part of our model could aid in mitigating this problem, but noun-phrases occurring in titles or at the start of sentences would remain ambiguous.

Context Disagreement

We see in Figure 4.5 the results of labeling 500 examples, selected according to context disagreement. For the `people` class, context disagreement works about as well as NP density for selecting examples to actively label. However, for the `locations` class, context disagreement is more effective. When we examine the examples chosen for labeling, we find that "capital" is no longer selected for labeling.

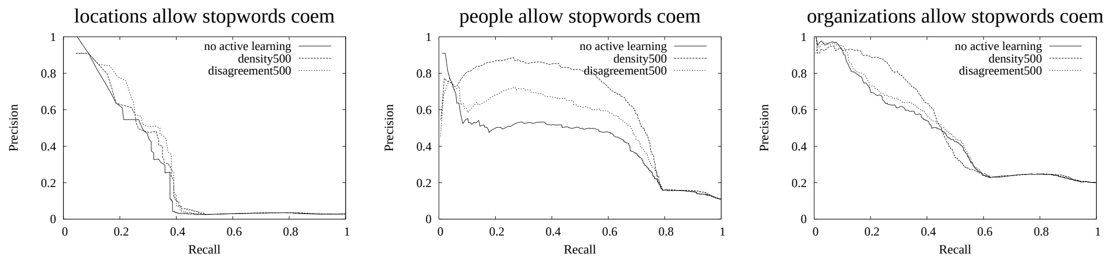


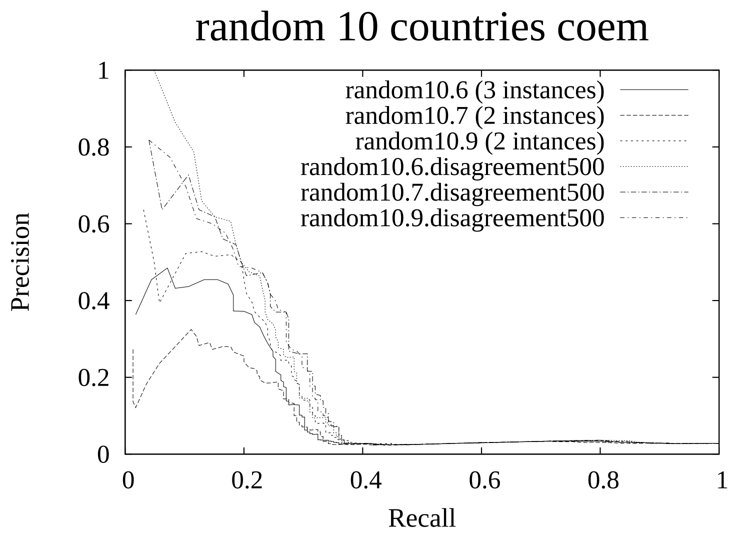
Figure 4.6: Adding 500 labeled examples, selected according to their density or feature-set disagreement, for *locations*, *people* and *organizations*.

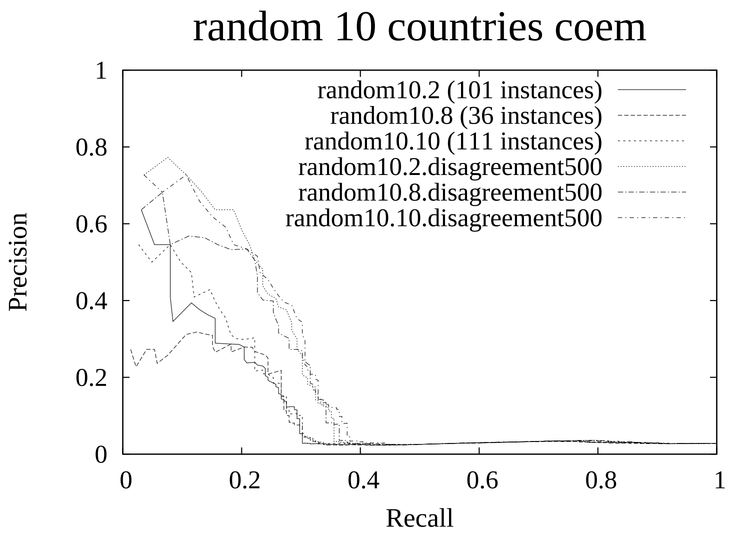
4.10.4 Disagreement and Density Active Learning

In Figure 4.6 we see the results of labeling examples selected according to the disagreement between the two feature sets. We see that for locations this is slightly more effective than density based selection, whereas for both `organizations` and `people` density based selection provides the most leverage. This matches the intuition we gained in Chapter 3, that good features for the `people` and `organizations` classes are the most frequently occurring ones, and that stopwords (which are frequent) are important in the model.

4.10.5 Active Learning Compensates for Infrequent Seeds

Figure 4.7 shows that using feature set disagreement for active learning can compensate for very infrequent seeds. Recall from Chapter 3 Section 3.3.2 that we use the seeds to actively label as positive all examples for which the head of the noun-phrase matches the seed, in a process we call *head-labeling*, while all remaining examples are unlabeled. Infrequently occurring seed words will therefore lead to fewer examples labeled at the outset with head-labeling. In the figure on the left, different sets of 10 randomly chosen country names led to only 2 and 3 examples labeled at the outset using head-labeling. In the center figure, the randomly chosen seeds were commonly occurring in the training data, occurring a total of 111, 101 and 36 times. With 500 examples chosen for active learning, the difference between the initial seed sets is virtually eliminated. We see in the graph on the right that starting with Riloff and Jones' 10 frequent country names, or a nearly complete list of 253 country names gives virtually the same results, after 500 examples have been labeled using active learning.





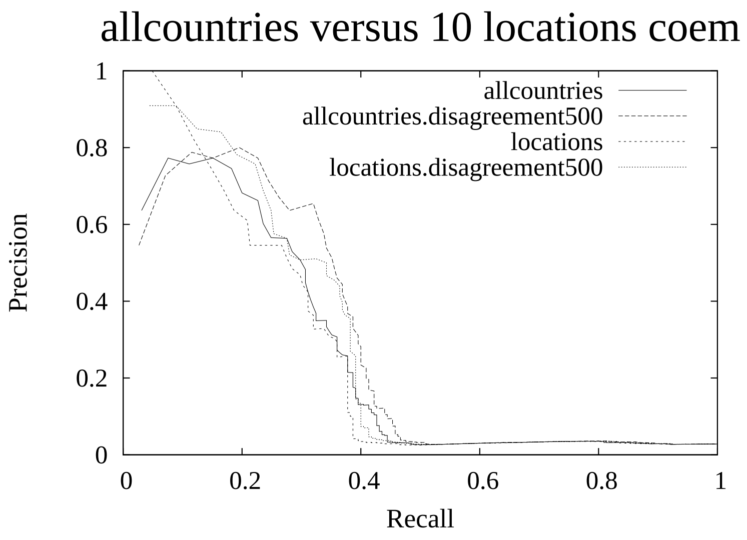
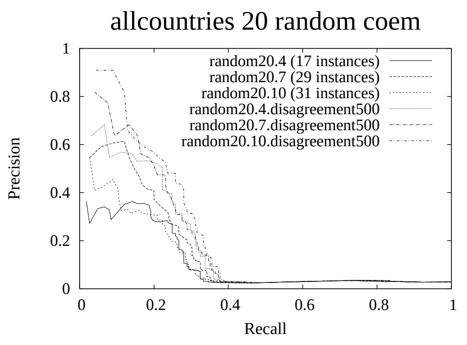


Figure 4.7: Active Learning Compensates for Infrequent Seeds. Left, 10 randomly chosen locations as seeds are infrequent in the training corpus. Center, the 10 randomly chosen seeds are relatively frequent in the training corpus. Right, 10 very frequent seeds and a near-complete list of country names used as seeds. In all cases active learning produces comparable results, after 500 examples have been labeled using active learning.



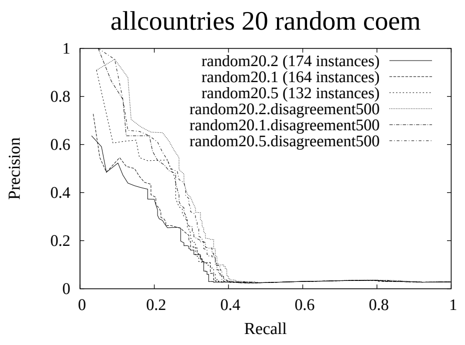


Figure 4.8: Active Learning Compensates for Infrequent Seeds. Even with 20 random seeds, active learning can produce considerable improvements. On the left, infrequently occurring seeds, on the right frequently occurring seeds.

Figure 4.8 shows that even when we double the number of randomly selected seeds to 20, active learning can still provide significant improvement in the results. This suggests that active learning makes bootstrapping robust to a poor initial choice of seeds.

4.10.6 Number of Examples Labeled

In Figures 4.9 and 4.10 we see that labeling more examples improves results for `locations` using disagreement labeling, whereas from 0 to 2500 examples labeled the change is not great for density-based labeling. When we contrast the effect of labeling more examples for `people` and `organizations` we see that both density and disagreement labeling are effective for the *people* class, while for `organizations` disagreement-based labeling shows the most effect for labeling more examples. Furthermore, when we look at learning curves showing the breakeven score against the number of iterations, it is clear that the bulk of the benefit is obtained during the first few examples labeled. Figure 4.11 shows the breakeven point at each iteration for both density and disagreement based active learning. Recall that we label 5 examples at each iteration. We see that the largest gains are obtained in the first few iterations, however continued labeling contributes to improvements too.

However, we see in Figure 4.12 that a larger corpus is more helpful for the `location` class than having up to 2500 examples labeled.

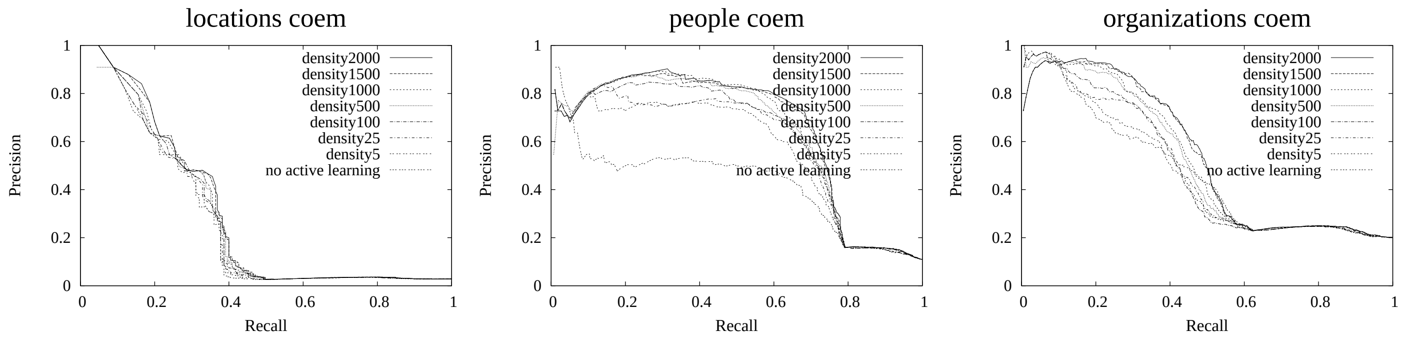


Figure 4.9: Labeling More Examples Improves Results. For `people` with density based selection, labeling just 5 examples greatly improves results.

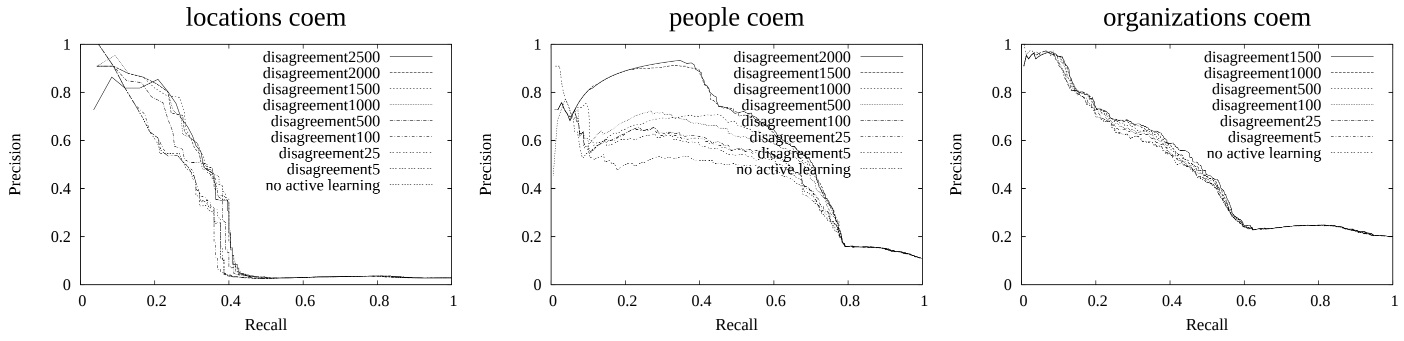


Figure 4.10: Labeling More Examples Improves Results For Disagreement-based active learning, for `people` and `locations` while for `organizations` little improvement can be seen.

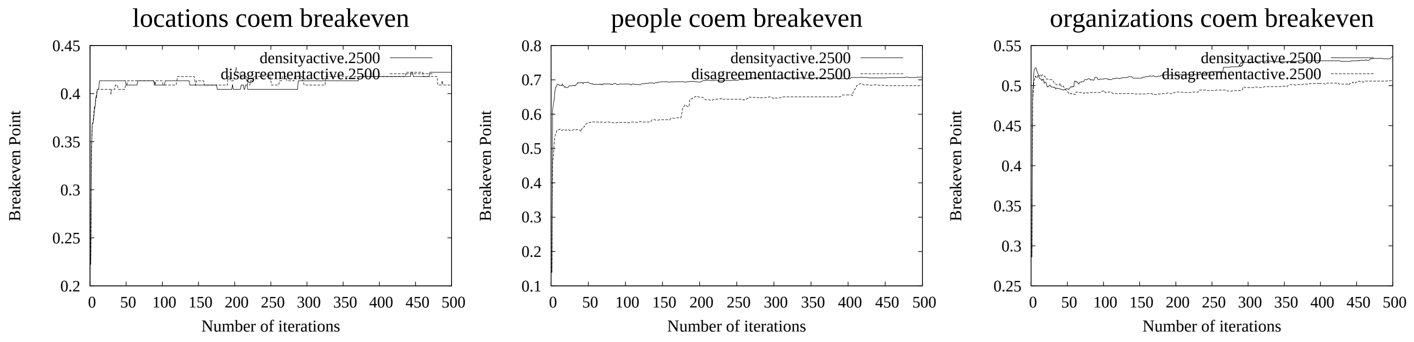


Figure 4.11: Breakeven point for each iteration. We see that most of the gains from active learning come in the first 50 iterations (ie the first 250 examples labeled) but labeling more continues to improve results. For the `organizations` class there is a dip from iterations 8 through 50. This may be due to the ambiguity in examples containing “we” which is labeled positive for `organizations` but may lead the model to incorrectly identify `people` as `organizations` until more examples have been labeled.

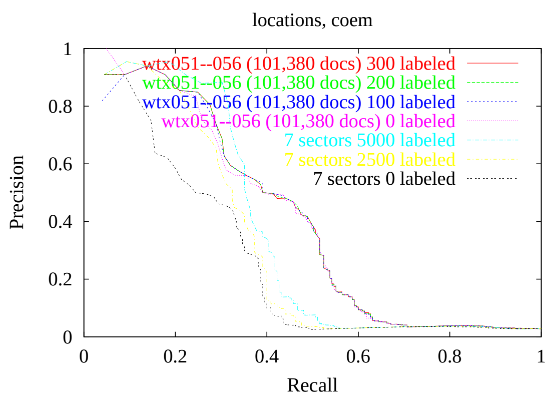


Figure 4.12: More data more important than labels

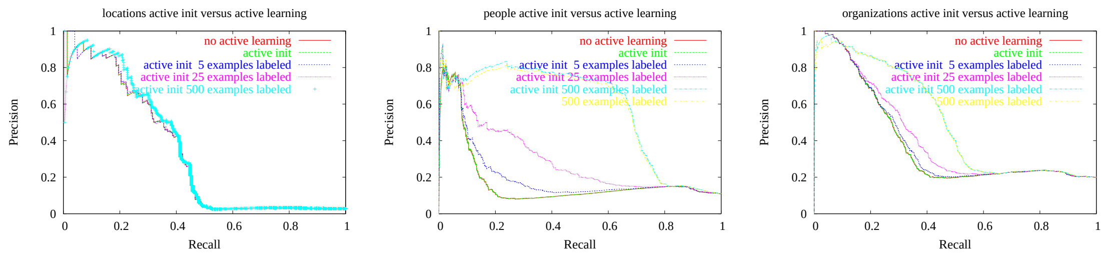


Figure 4.13: Labeling with active learning has more impact than active initialization. Combining active initialization with active learning provides modest incremental gains for the `people` class.

4.10.7 Active Learning More Useful than Active Initialization for CoEM

We found that labeling pairs of noun-phrases and contexts which matched seeds at the outset did not perform significantly better than using active learning by itself. When our active learning method is provided a set of initial instances that are "clean" and unambiguous, the extraction performance does not improve. This suggests that the active learning methods are robust to ambiguous/noisy training data and can recover from poor initial seeds. This is shown in Figure 4.13. We also find that the active learning method (with 500 examples labeled for `locations`) performed better than using bootstrapping with coEM with active initialization (with 693 examples labeled). This is an important result since if we have a fixed amount of time to actively label instances, active learning can be a more effective use of this time than labeling the instances at the outset.

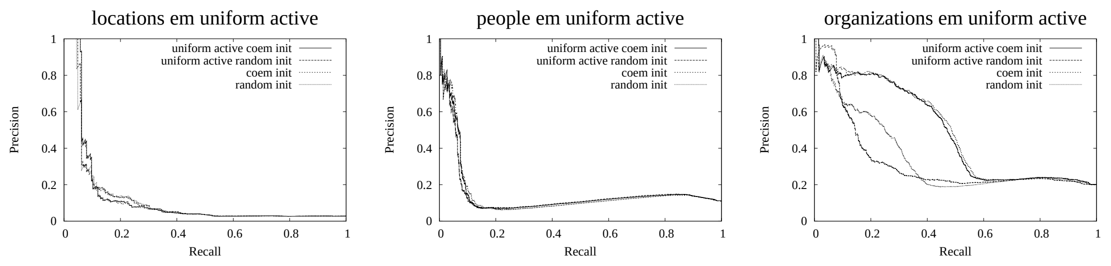


Figure 4.14: Adding 500 labeled examples, examples selected uniformly at random, for *locations*, *people* and *organizations*. As with coEM, EM does not benefit greatly from examples selected uniformly at random, both for random initialization, and initialization from coEM.

4.10.8 Active Learning with EM

Recall from Chapter 3 Sections 3.4.6 and 3.4.2 that coEM uses a split in the feature set to label with noun-phrases and contexts in alternation, while EM uses both noun-phrases and contexts together to label in each iteration. When we had only examples labeled automatically from seeds with *head-labeling* (see Section 3.3.2), coEM provided more accurate results. One reason may have been that EM attempts to maximize the likelihood of the data given the model, but some of our target classes may be better represented by models which give a lower likelihood to the data.

If we have the opportunity to provide more labels to the algorithm through active learning, these will change the distribution of the data over which EM attempts to maximize the likelihood. In particular, examples given definitive labels by a human labeler in active learning have a known class label, which will affect the overall data likelihood. Thus we may expect the performance of EM to improve with active learning.

Uniform Labeling Less Influential than Initialization Condition for EM

As with coEM, we find that EM does not benefit greatly from adding 500 labeled examples, with examples selected uniformly at random, as shown in in Figure 4.14. We see in particular, that for `organizations` results are markedly better with initialization from coEM than with random initialization, and that this difference holds regardless of whether we label 500 examples selected uniformly at random.

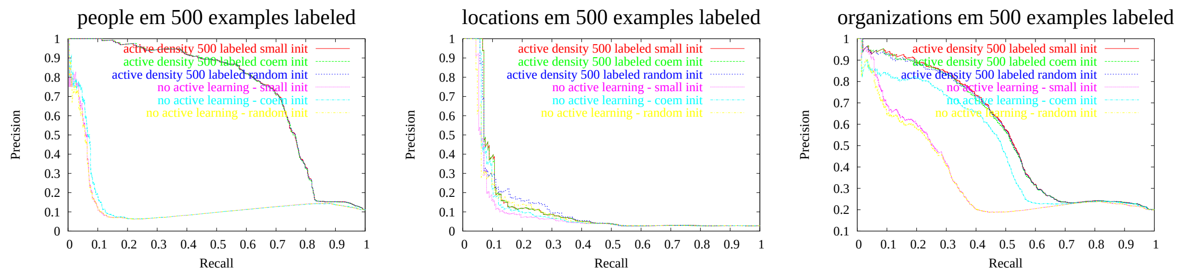


Figure 4.15: For EM, when we select examples for labeling according to density we see very large improvements in results for the `people` class and the `organizations` class. This transcends the initialization condition, unlike with uniform active labeling.

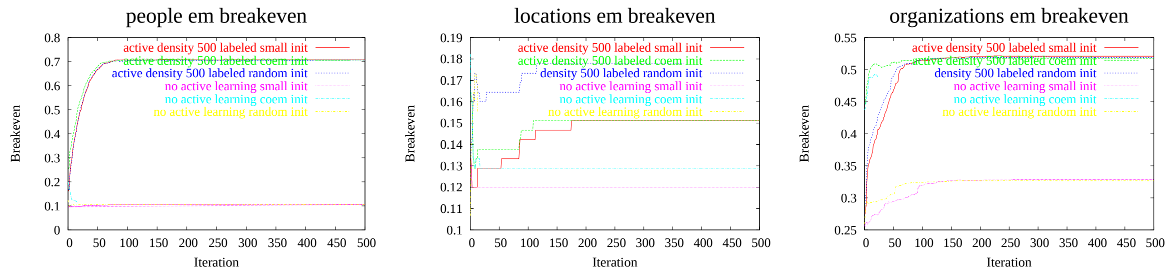


Figure 4.16: Breakeven versus iteration, for density based active learning.

Density Based Example Selection Effective Regardless of Initialization

When we select examples for labeling according to density we see substantial improvements in results for the `people` class and the `organizations` class, as shown in Figure 4.15. This mirrors the results we saw for coEM, where the most substantial gains with density-based example selection were realized in the `people` class. Results shown here are for `small`, `random` and `coem` initialization for examples not matching seeds. Interestingly, the gains from labeling 500 examples selected according to density transcend the differences due to initialization condition. When we look at the breakeven graphs in Figure 4.16 we see that the early iterations of labeling have a substantial impact on the breakeven score.

Single Feature Set Labeling Effective for EM

We saw that two forms of single feature set labeling were reasonably successful for coEM: NP density based selection, and context disagreement based selection. We see in Figure 4.17 that both are reasonably successful for EM, regardless of the initialization condition. For `people` and `organizations` both appear to be equally successful.

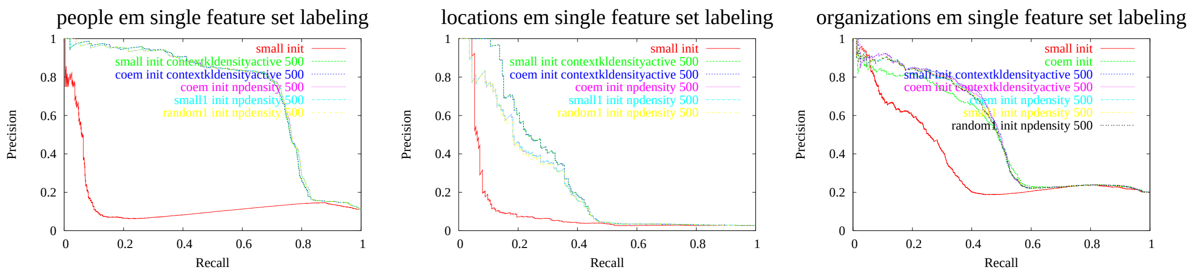


Figure 4.17: Single feature set labeling is reasonably successful for EM. It provided great benefit for the `organizations` and `people` classes

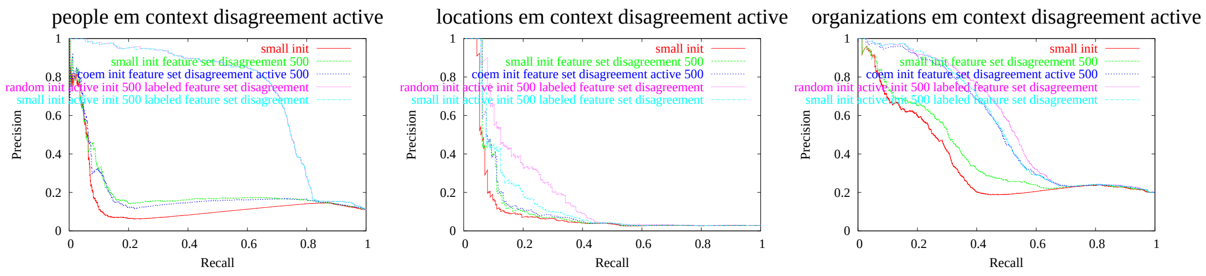


Figure 4.18: When we select examples for labeling according to disagreement between the noun-phrase and the context, we see the largest improvements for the `organizations` class, because half of the examples selected were not among the original head-labeled examples. Eliminating this manual relabeling of automatically labeled examples by using active initialization, we see improvements in all classes.

For `locations` context disagreement performs somewhat better.

Feature Set Disagreement Selects Seeds; Improves with Active Initialization

When we select examples for labeling according to disagreement among contexts (Figure 4.18) we see the biggest improvements in the `organizations` class. However, when we observe which examples are selected for labeling, we see that many of these examples contain the original seedwords. We can eliminate this effect by combining active initialization with active learning. We then see substantial gains in accuracy with active learning for all classes. This suggests that we should focus active learning effort on examples not already labeled with head-labeling.

Table 4.3 shows that for the `people` and `locations` classes, feature-set disagreement mostly selected examples which had seeds as heads. Active initialization elim-

Class	Num examples chosen with seed heads
locations	482
organizations	286
people	441

Table 4.3: For the `people` and `locations` classes, feature-set disagreement mostly selected examples which had seeds as heads. We can eliminate this effect by performing active initialization, or by excluding examples with seeds as heads from the pool available for active learning.

inates this effect. However, we have seen that active initialization is in general not important for improving results. We would expect to get similar improvements from excluding examples with seeds as heads from active learning.

EM Robust to Choice of Initial Seeds With Single Feature Set Labeling

We wish to examine the effect of different seed sets with EM, as we did with coEM. As a baseline we use the locations seed set, with 2500 examples labeled, using density based labeling. Since we are using density based example selection, the set of labeled examples will be the same for all seed sets. The difference will be between the initial conditions for EM. Since EM tends to find local maxima, we may expect that different initial conditions may have an important effect on the final model found.

Figure 4.19 shows that starting with random seed sets performs extremely poorly, even if we label 2500 examples using density based selection. Active initialization does not improve results. One reason for this may be that density based active learning does not select examples for labeling which are likely to be in the positive class, so for the random seed sets we may never have training data which permits us to model the positive class. We may expect that feature-set disagreement would perform better, as it selects examples for which the feature sets disagree on class membership, and may provide a better range of training examples. However, Figure 4.20 shows that starting with random seed sets performs extremely poorly, even if we label 2500 examples using feature-set disagreement based selection.

Surprisingly, however, context disagreement based labeling greatly improves the results. This may be because context disagreement works so well for the locations task, by providing labels for as many different noun-phrases as possible. Recall from Chapter 3 that the `locations` task has a large vocabulary which is only partially

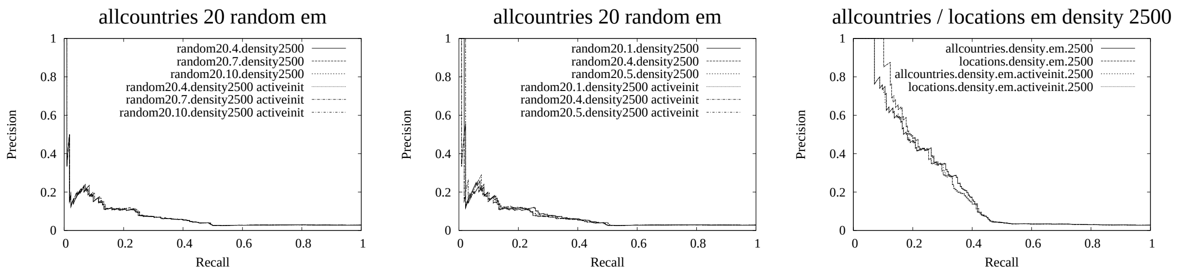


Figure 4.19: Using different initial seed sets and labeling examples according to frequency. Starting with random country names performs extremely poorly, when contrasted with starting with all country names.

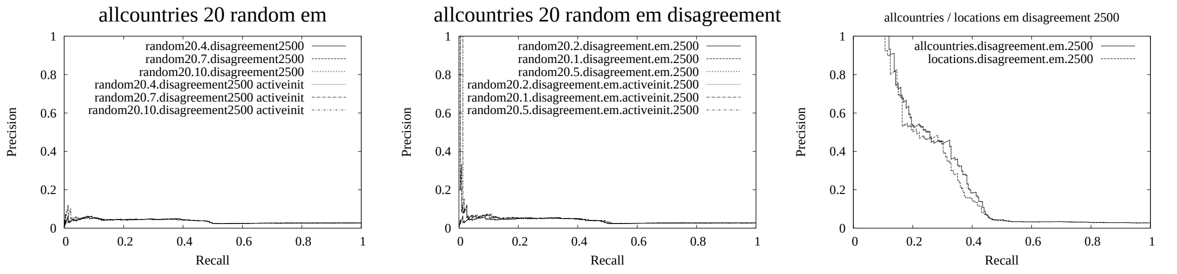


Figure 4.20: Using different initial seed sets of random country names does not perform well with feature-set disagreement, even with active initialization and 2500 examples labeled, when compared with using all country names.

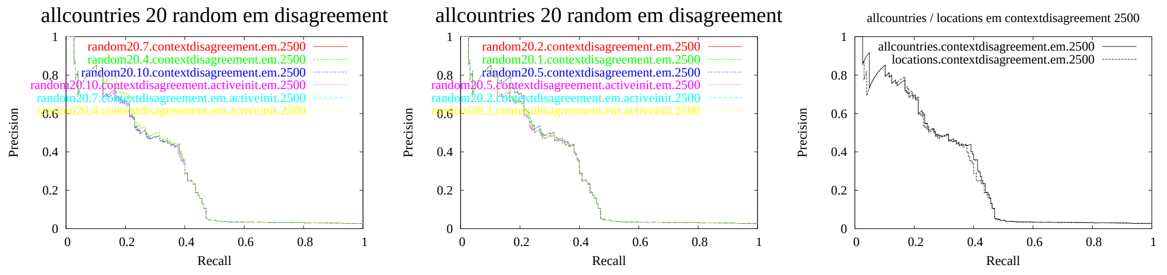


Figure 4.21: Using different initial seed sets of random country names is reasonably effective with EM, when we label 2500 examples using context-disagreement.

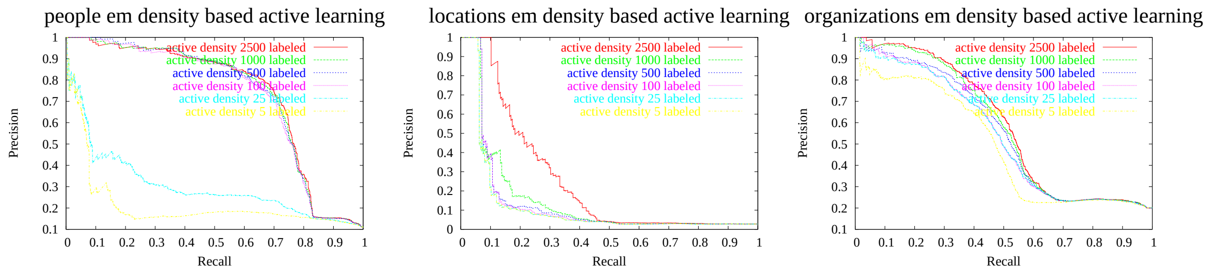


Figure 4.22: For all classes, labeling more examples greatly improves the results for EM. In particular, for the `people` class, labeling 100 or more examples provides the biggest gains in model effectiveness. With both `organizations` and `locations`, adding more examples continues to improve the effectiveness of the model.

covered by the training set.

Labeling More Examples Improves Results for EM

Figure 4.22 shows that for all classes, labeling more examples greatly improves the results for EM. In particular, for the `people` class, labeling 100 or more examples provides the biggest gains in model effectiveness. With both `organizations` and `location`, adding more examples continues to improve the effectiveness of the model. We see in Figure 4.23 that when we label 2500 examples, disagreement based labeling is somewhat better for both the `locations` and `people` classes. When we examine the breakeven curves (Figure 4.24) we see that for `locations`, with active initialization, disagreement based selection dominates density based selection, but by 500 iterations (2500 examples labeled) results using the two methods are quite similar. For the `people` class we are seeing the opposite effect. In both cases, however, once we label 2500 (just over 10% of the 23,000 unlabeled examples available for labeling) the exact method we are using for labeled becomes of less importance.

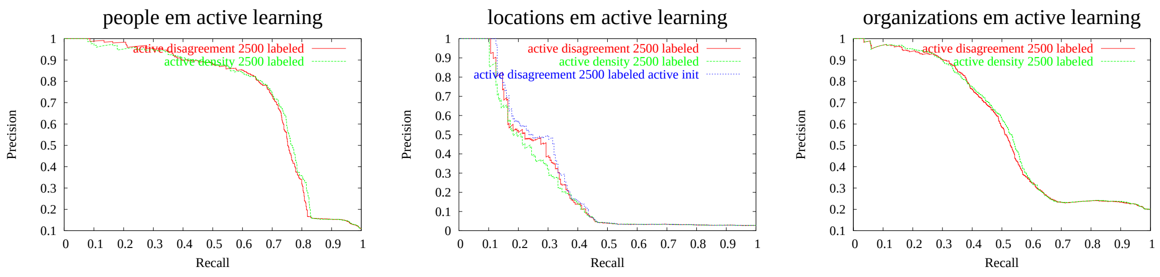


Figure 4.23: With 2500 examples labeled, the `people` and `locations` classes perform slightly better with disagreement based labeling.

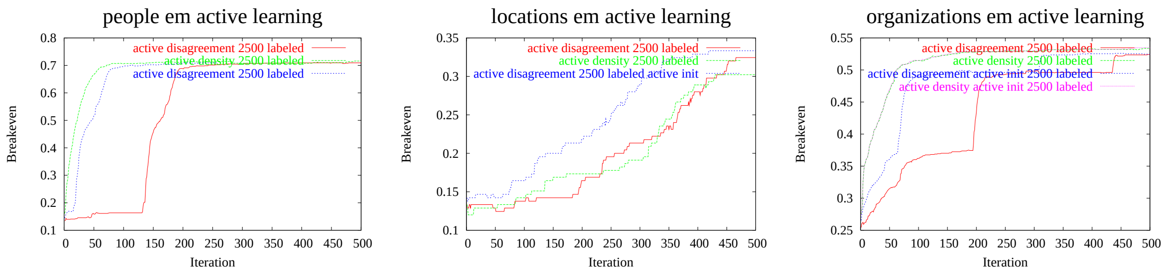


Figure 4.24: Breakeven score versus iteration number, for EM, with 2500 examples labeled (5 per iteration). The `people` and `locations` class perform slightly better with disagreement based labeling. Organizations is always better with density based labeling.

Corpus Size and Mismatch

We see in Figure 4.25 that labeling just 100 examples greatly improves results for the `people` and `organizations` class on the TREC wtx corpus which comes from a different distribution than the test set. This suggests that active learning may help compensate for a training set which comes from a different distribution from the test set.

4.11 Chapter Conclusion

Recall the questions we raised in Section 4.1. We will now go over the answers provided by the experimental results in this chapter.

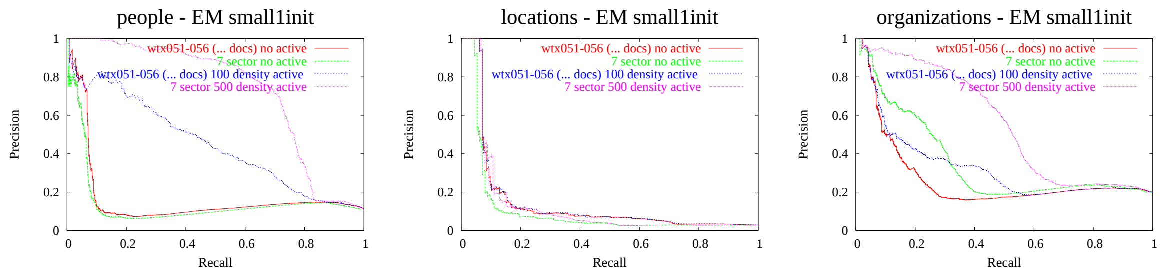


Figure 4.25: When we use a large corpus from a different distribution, and label examples using density-based selection, we can recover from the losses in accuracy due to train-test corpus mismatch.

Is it more effective to correct head-labeled examples, or actively label new examples?

Given that we have automatically labeled some examples at the outset with head-labeling, it is more effective to label different examples during the active learning phrase. Recall that we saw in Chapter 3 that correcting the labels automatically assigned during head-labeling does not greatly improve bootstrapping results. In this chapter we see that more benefit is derived from labeling examples selected with active learning than from correcting head-labeled examples with active initialization. An important point, however, is that we reap benefit from active learning primarily when it selects *different* examples from those labeled with head-labeling. For this reason, active learning algorithms which tend to select head-labeled examples perform better when coupled with active initialization, as redundancy in the labeling is removed. A more efficient and equally effective strategy would be to force the active learning algorithm to select examples for labeling, which were not part of the initial head-labeling set.

What is the most effective way to select new examples to actively label?

We found that if we are labeling few examples, density based selection is most effective for `organizations` and `people`, while for `locations` feature set disagreement was most effective. However, when we label 2,500 examples, the gap in accuracy between the methods is narrowed, so it may be more effective overall to label examples with feature set disagreement.

Is it more effective to actively label just noun-phrases, or to actively label noun-phrase context pairs, in active learning?

We saw in Section 4.10.3 that for coEM, single-feature set labeling, or asking users to label noun-phrases out of context, occasionally leads to the incorrect label. For coEM this lead to poor performance of bootstrapping, while for EM, this was not as much of a problem, as we saw in Section 4.10.8. In particular, we found that single feature set labeling with context disagreement was very effective for the locations class, even with infrequent seeds. Thus it appears that the effectiveness of single feature set labeling depends on the algorithm used for bootstrapping, and that for classes with a diverse vocabulary, single feature set labeling is effective with EM.

If we have a larger corpus to choose examples from, does this lead to greater accuracy for the same amount of user labeling effort?

We saw in Figure 4.12 that for coEM on the `locations` class, using a larger corpus from a different distribution was more effective than labeling many examples on a smaller corpus from the same distribution as the test set. For `people` and `organizations` however, changing the distribution harmed accuracy as we saw in Figure 4.25 and while labeling examples with active learning from the larger corpus improved accuracy over no active learning, performance was not as good as when we performed active learning on a smaller corpus from the same distribution as the test set.

How much does increasing the number of actively labeled examples improve accuracy? Do improvements tail off, or do we continue to get increased accuracy as we increase the number of labeled examples?

Across both coEM and EM, for all active learning methods, we saw increased accuracy as we increased the number of examples labeled with active learning. The increases in accuracy continued up to 2,500 examples, the maximum number of examples we labeled in these experiments. The largest improvements were generally in the first 5 - 100 examples labeled. This suggests that it is worth performing active learning, even if we have only a tiny amount of user time at our disposal. If our goal is to maximize accuracy, however, we should label as many examples as possible.

Does active learning make bootstrapping algorithms more robust to the set of seeds chosen for head-labeling?

Active learning makes bootstrapping examples robust to the choice of seeds: for EM we found that context disagreement based selection eliminated much of the difference between performance with different seed sets, while with coEM we found that feature set disagreement eliminated much of the difference in performance. We can conclude that if we use active learning, we need not be too concerned with the quality of the initial seeds used for initialization with head-labeling.

Does active learning make bootstrapping algorithms more robust to the way unlabeled examples are initialized?

For EM, we have a number of choices about how to initialize unlabeled examples. In Chapter 3 we found that these choices had a significant effect on bootstrapping accuracy. In this Chapter, however, we found that most active learning methods eliminated these differences.

Summary

We showed that employing the redundancy in feature sets and designing algorithms that exploit this redundancy enables the combination of bootstrapping and active learning to be effective for training information extractors. We compared different metrics for selecting examples to actively label and found that using the disagreement between classifiers built with the two feature sets worked well. Manually correcting initial examples that were mislabeled due to ambiguous seeds is not as effective as providing the active learning algorithm with an arbitrary set of seeds and labeling examples during the learning process. However, some algorithms may select examples which have already been labeled in the initial phase, reducing the effectiveness of labeling. Context-disagreement used the single feature set labeling setting, and did not perform as well as methods using standard labeling for coEM, but was very effective for EM, when we label up to 2500 examples. Using only a single feature set for labeling may allow inaccuracies to creep into the labeled set, if any of the examples are ambiguous with respect to that feature set. In addition, disagreement between members of a single feature set may reflect inherent ambiguity in the example, and not uncertainty in the learner. We presented an active learning setting that is able to make use of the multiple view feature sets and provided experimental evidence of its effectiveness. Although the results shown here are specific to the information extraction setting, our approach and framework are likely to be useful in designing active learning algorithms for settings where a natural, redundant division of features exists.

Chapter 5

Analysis

In Chapters 3 and 4 we saw empirical results of bootstrapping for learning to extract information in the form of semantic classes specified through the use of seed words. In this chapter we perform a deeper analysis of some of these results. We measure properties of the noun-phrase context connectivity graph, and show that it exhibits small-world and power-law graph structure rather than random graph structure. We show that pronouns and certain very common nouns form the hubs of the connectivity graph, which explains the importance of stopwords and frequent seeds in our models. We measure the mutual information between noun-phrases and contexts in each class, in order to test our conditional independence hypothesis. We also perform Spearman rank correlation tests over multiple experiments, to find the correlation between algorithm breakeven point and features including the number of contexts labeled by multiple seeds, and the percent of examples labeled positive during active learning. These can help us pinpoint the important properties of active learning and bootstrapping algorithms for information extraction. Comparing these across classes also highlights the different desiderata for active learning algorithms for classes with sparse feature sets and extremely small priors.

In this chapter we analyze the results of experiments from Chapters 3 and 4. To do this, we first summarize the results and describe some tools we will use for analysis, in Section 5.1. In Section 5.2 we analyze our data for small-world and power-law properties. In Section 5.3 and show that some graph-theoretic properties are predictive of algorithm performance. This kind of analysis is novel, and sets the stage for research into selecting appropriate algorithms and labeling techniques based on examining properties of the unlabeled data.

Many machine learning algorithms assume feature set independence. A common trait is to acknowledge that feature set independence does not hold, and show that the algorithm is effective anyway, perhaps because exactness in the generative model is not essential for classification (see eg. (Domingos & Pazzani, 1997)). In Section 5.4 we quantify feature-set dependence by measuring mutual information between features, and suggest that differences in conditional dependence across classes may translate into different algorithmic performance across classes. This novel analysis opens up a spectrum of algorithm performance prediction based on labeled training data, in the absence of labeled test data.

These analyses add up to some concrete suggestions for algorithm design, informed by properties of the data, which we explore in Section 5.5. Finally we give some basic advice about seed selection and assessing data and tasks in Section 5.6.

5.1 Analyzing Results of Experiments

In Chapter 3 we saw the results of experiments with algorithms for bootstrapping semantic classes, initializing with small sets of seed example words. In Chapter 4, we saw the results of experiments adding a variety of active learning algorithms on top of the basic bootstrapping algorithms. In this chapter we will step back and examine the overall lessons learned through these experiments, by examining the trends and exploring several possible explanatory factors.

In Table 5.1 we provide an overview of the experimental conditions we explored in Chapters 3 and 4, along with the tentative conclusions we drew from those. In the following sections we will explore the issues of stopword and example frequency, by examining our data in graph-theoretic terms. We will also tease apart the effect of adding more examples, and more *positive* examples, by examining their effect across different active learning methods. Finally we will examine the degree to which our data fails to exhibit the class conditional independence which we have assumed, by measuring the mutual information between noun-phrases and contexts within classes.

Bootstrapping		
Experimental condition	Summary of Results	Section
Bootstrapping algorithm	Metabootstrapping: poor; cotraining: poor	3.7
	EM: mixed; coEM: okay	3.8.5
Seed word labels corrected	Unimportant	3.7.4
Train and test from same distribution	`people`: important, `organizations`: important	
	`locations`: not important	3.7
Allow stopwords (pronouns) in the model	Important for `people`, `organizations`	3.7.2
Use example frequency	`organizations`: important with EM	3.8.2
Seedword frequency in training set	High frequency important	3.7.5
Training set size	`locations`: large training set best	3.7.6
Transduction	`organizations`: helpful; `people`: deleterious	3.7.7
Bootstrapping with Active Learning		
Experimental condition	Summary of Results	Section
Train and test from same distribution	Less important with active learning	4.10.8
What user labels	whole example labeling: effective	4.10.3, 4.10.8
	Noun-phrase labeling: effective	4.10.3, 4.10.8
Number of examples labeled	Improvements with just 5 labeled examples; increasing	
	number labeled gives continuing improvements	4.10.6
Algorithm used to select examples	• uniform random: poor	4.10.1
	• density: effective	4.10.2, 4.10.8
	• feature set disagreement: effective	4.10.4
	• context disagreement: effective	4.10.3
Initialization conditions	EM: large effect	4.10.8
Seedword frequency in training set	Less important with active learning	4.10.5, 4.10.8
Seed word labels corrected	Unimportant	4.10.7, 4.10.8
Allow stopwords in the model	More important than labeling 500 random instances	4.10.1
Algorithm used for bootstrapping	coEM: most effective across conditions	4.10.2
	EM: sensitive to class and active-learning algorithm	4.10.8

Table 5.1: Summary of experiments in Chapters 3 and 4, along with the general trends we saw.

5.1.1 Breakeven Score as Summary of Result

In Section 3.6.2 we defined our evaluation metrics of *precision*, *recall* and *breakeven*, and for most experiments in this thesis we showed results by plotting the entire precision-recall curve. Now we are looking at the results of many experiments simultaneously, we would like to summarize the results of each experiment with just one number. While using a single number does not capture all the nuances of the experimental result, it captures much of the performance, and allows us to compare across experiments.

There are several single number evaluation scores we can use to compare experiments. We will describe several candidates, and summarize the pros and cons in Table 5.1.1.

Accuracy

Accuracy is a measure commonly used in machine learning. We generally assume that we make a prediction for every example in the test set, so the accuracy is then the percentage of test examples for which we make a correction prediction.

$$Ac = \frac{correctPredictions}{TotalPredictions} \tag{5.1}$$

A draw-back of accuracy as a measure of efficacy is that when one of the classes is very sparse (for example, our `locations` class, which constitutes about $225/8081 = 2.8\%$ of the test instances), a naive classifier which classifies every example as negative can appear to perform well (say with accuracy above 95%), while being wrong about all examples in the class of most interest.

F-measure

The F-measure (Van Rijsbergen, 1979) combines precision and recall with a parameter α which quantifies the importance the user attaches to each. The general form of the F-measure is given by the weighted harmonic mean of precision and recall:

$$F = [\frac{\alpha}{P} + \frac{1-\alpha}{R}]^{-1} = \frac{PR}{(1-\alpha)P + \alpha R}, 0 \leq \alpha \leq 1 \tag{5.2}$$

Accuracy	Insensitive to performance on sparse class
F-measure	Score is dominated by the lower of precision and recall
Breakeven	F score at $\alpha = 0.5$ where precision and recall are equal

.

Table 5.2: Comparison of Single-number summary statistics of Experimental Results

where P is precision and R is recall (defined in Section 3.6.2 in Chapter 3). When $\alpha = 0.5$, we weight precision and recall as equally important, and obtain

$$F = \frac{2PR}{P+R}, \alpha = 0.5 \tag{5.3}$$

When we have a set of pairs of precision and recall scores, we combine them by first calculating average precision and average recall. To calculate average precision, we take recall values at intervals of 0.1 (interpolating where necessary) and average precision at these points. Average recall is calculated similarly, by averaging over fixed precision points.

Breakeven Point

When we have a precision recall curve, we find the breakeven point by finding the point where precision and recall are equal, interpolating where necessary. Note that the breakeven score is a specific value of the F-measure at $\alpha = 0.5$, when P and R are equal. The breakeven is always less than the optimal F-measure score for a system. We chose to work with the breakeven score because of the simplicity of computation.

5.1.2 Spearman Rank Correlation Test

To understand the extent to which a variety of experimental conditions predict performance, we can consolidate results from Chapters 3 and 4, and see if general trends emerge. By focusing on different properties of experiments, such as the number of examples labeled by seeds, or the number of examples labeled during active learning, and aggregating over multiple experiments, we can measure the degree to which each property affects the results. While each individual experimental result is affected by the combination of conditions, over many different experiments we can see general trends.

To measure these trends, we can perform the Spearman rank correlation test over the results of the experiments, in combination with a candidate predictive property of the experiments. If we have a ranking function R and a ranking function S, the formula for the Spearman correlation test is:

$$r_s = \frac{\sum_i (R_i - \overline{R_i})(S_i - \overline{S_i})}{\sqrt{\sum_i (R_i - \overline{R_i})^2}\sqrt{\sum_i (S_i - \overline{S_i})^2}} \tag{5.4}$$

where R_i is the rank of point i according to R, and S_i is the rank of point i according to S.

The Spearman rank correlation test is a non-parametric test, *ie* it does not make assumptions about the form of the relationship between two variables. For example, in this chapter we will use the Spearman rank correlation test to test whether the *rank* of algorithm performance is predicted by the *rank* of the number of examples labeled. This means that we can detect a positive relationship between algorithm performance and number of examples labeled with active learning, without making any assumptions about the form of that relationship (for example, without assuming the relationship is linear). The Spearman rank correlation test is related to the Pearson correlation test, but uses the rank of the value rather than the value itself. For our example of measuring how the number of examples labeled with active learning predicts performance, we will order the number of examples labeled, such that each experiment has a rank in that ordering, and order the results, such that each experiment has a position in the ordering of results. Then for all n experiments, the i^{th} experiment gives the pair of $< breakevenScore_i, numExamplesActivelyLabeled_i >$. For each experiment i we find both the rank by breakevenScore: $R_i = rank(breakeven_i)$ and the rank by number of examples labeled with active learning: $S_i = rank(examplesLabeled_i)$. Ties are assigned their average rank. The formula for the Spearman rank correlation test is then found by using ranks in the Pearson linear correlation formula (and is given in Equation 5.4 (Press et al., 1992)). A Spearman correlation score r_s close to 1.0 shows a positive correlation in the ranks. A Spearman correlation score close to -1.0 shows a negative correlation, while scores close to 0 show little correlation.

r_s^2 gives the percentage of variability in rank ordering that is explained by the predictor variable. For example, for $r_s = 0.6$, 36% of variability in the rank ordering of datapoints using S is explained by R.

To see whether a measured value r_s is significantly different from 0 (the null hypothesis) we measure the variable

$$t = \frac{r_s}{(1 - {r_s}^2/(n-2)} \tag{5.5}$$

which is distributed according to a t-distribution with 2 degrees of freedom, where n is the number of points used in the Spearman correlation test. We can then use standard t-tables to obtain a significance score. A significance score near 0 shows that our measurement of the correlation is statistically significant. Typically we would like to see significance scores below 0.05 to have confidence in the correlation score.

To test whether the difference between two correlation coefficients r_{s_1} and r_{s_2} is significant, we must use another test: the r to Z transformation:

$$Z_f = \frac{1}{2}\ln(\frac{1+r}{1-r}) \tag{5.6}$$

Assuming we have transformed r_s to Z_f, and transformed r_{null} (the correlation coefficient for our null hypothesis) to Z_{null}, we can then compute a test statistic:

$$Z = \frac{Z_f - Z_{null}}{\sqrt{\frac{1}{N-3}}} \tag{5.7}$$

If $|Z| > 1.96$ then r_s differs from r_{null} with $\alpha = 0.05$.

We will use these significance tests when we compare different properties of the data, to find whether some properties are more predictive of algorithm performance than others.

5.2 Small World Nature of Noun-phrase Context Cooccurrence Graph

We will now analyze our experimental results by focusing our attention on the first of three different aspects of the data and the labeled examples: the small world nature of the noun-phrase context cooccurrence graph. In this section we describe the data in terms of the cooccurrence graph, and test whether it has small-world and power-law properties (to be defined below). In Section 5.3 we analyze the effects of graph properties on algorithm performance.

Graph Property	Brief description	Hypothesis
Small-world	Short path-lengths	All nodes in component reachable in few steps probabilistic labeling best
Power-law	One large component, many small components	Distribution of seeds over components affects learning
Power-law	Skewed distribution of node degrees	Node degree of labeled examples affects learning

Table 5.3: High-level graph properties of the data we will examine, as well as conjectures that we will test about their effect on algorithm performance

We can view our data consisting of pairs $< n, c >$ of noun-phrases and contexts as a graph, if we represent each noun-phrase and each context as a node, and each pair as an edge in the graph. We will give more detail about this mapping in the sections below. The bootstrapping algorithms we explored in Chapter 3 exploited cooccurrence information to propagate evidence of class membership. Thus examining the graph structure of the data may provide insight into the expected effectiveness of algorithms on the data. In addition, the extent to which nodes are connected into many or few components will affect the likely performance of algorithms, since cooccurrence information will provide evidence only among nodes in the same component. The presence of seeds in different components may provide insight into the performance of bootstrapping algorithms with a given seed set. Any tendency of active learning to pick out examples in different components may explain how active learning contributes to bootstrapping over data initially labeled with seeds. We will examine the consequences of this structure later in this section.

In Table 5.3 we see a brief summary of the graph properties we think may affect learning performance. In the following sections we will define each of these properties, then, in Sectionsec:correlations:performance where appropriate and possible using the experimental data from Chapters 3 and 4, test whether the hypothesized effect on learning performance holds by examining related graph properties.

5.2.1 Small-world Graphs of Data

We are accustomed to thinking of examples in machine learning as vectors of features, where an example x_i is made up of n features: $x_i =< x_{i_1}..x_{i_n} >$, and may also be accompanied by a label y_i. In this section we describe how we can view a set of examples $X = \{x_1..x_m\}$ as a graph.

We describe two representations of examples: (i) examples with splittable feature sets, which we represent as bipartite graphs and (ii) the more general representation of examples as nodes and edges in a graph.

f_1	f_2	label
australia	flew to $< x >$	?
australia	headquartered in $< x >$	+
australia	$< x >$ broadened	?
china	flew to $< x >$	?
france	headquartered in $< x >$	?
thailand	$< x >$ broadened	+
thailand	gulf of $< x >$	?
director	$< x >$ of multinational company	?
leader	$< x >$ in its industry	?

Table 5.4: Training examples in feature vector format. Each example has two features, f_1 (the noun phrase) and f_2 (its context). Some examples are labeled positive, while all other examples are unlabeled.

We will consider a unique instantiation of a feature or features to be a node in the graph, and cooccurrence among features or feature sets to be an edge in the graph.

Bipartite Graphs over Examples Represented with Two Feature Sets

Table 5.4 shows training examples for the semantic labeling task which we described in Chapter 3 Section 3.2.3. Each example has two features, f_1 (the noun phrase) and f_2 (the context). Some examples are labeled positive, while all other examples are unlabeled.

In Figure 5.1 we see a bipartite graph representing the same instances. Each instance is represented by an edge joining two nodes in the graph. For example, the instance "flew to china" is represented by the two nodes "flew to $< x >$" and "china", with an edge joining them. In this graph, the node "australia" is connected to three other nodes, so it has degree 3. This also represents the fact that "australia" occurred in 3 unique examples, with three unique different contexts: `flew to` $< x >$, `headquartered in` $< x >$, and $< x >$ `broadened`.

In supervised machine learning, viewed from this graphical perspective, we are given a set of nodes and edges, with a label provided for each of the edges in a training set. We use these labels to learn to predict labels on edges in a held-out test set graph.

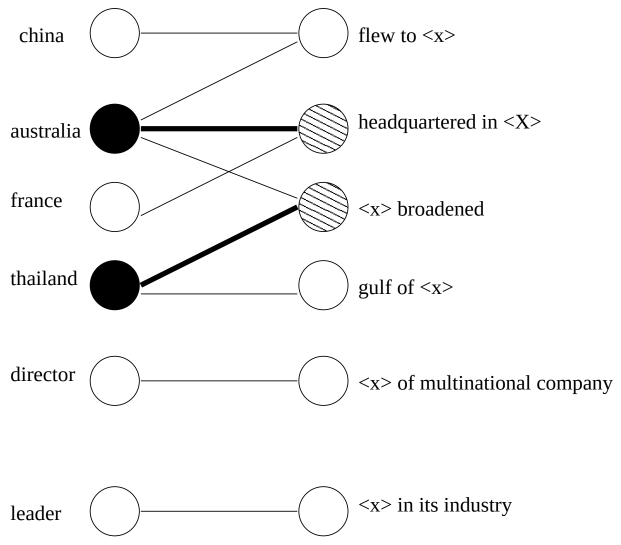


Figure 5.1: Each instance represents an edge joining two nodes in the graph. For example, the instance "flew to China" is represented by the two nodes "flew to $< x >$" and "china", with an edge joining them.

For this work we are concerned with semi-supervised learning. Our training set is a collection of documents which we parse into noun-phrases and contexts, which we then identify as nodes in a graph, with cooccurrences forming the edges in the graph. We are given a small number of labeled examples, that is, a small number of labeled edges, and we use an algorithm to infer labels for other edges based on the partially labeled initial set. Our held-out test set is a set of distinct web pages parsed in the same manner to form a graph of cooccurring noun-phrases and contexts, and with edges assigned class labels. The set of nodes and edges in the test-set graph may not be identical to those in the training set. There may be both nodes and edges that were not seen in training. We measured the degree of overlap between train and test set in chapter 3 Section 3.5.3 and found that 41% of labeled test noun-phrases, and 91% of labeled test contexts had been seen in the training set. We then use the learned model to infer labels on the held-out test set.

In general this graph is an incomplete sample from the underlying distribution, as discussed in Section 5.2.2.

Our example in Figure 5.1 has two feature sets, each of which contain exactly one feature. More generally, this bipartite graph representation can have multiple features in each feature set. Blum and Mitchell (Blum & Mitchell, 1998) described their data in graph-theoretic terms, by splitting the feature set into two redundant sets. For an example $x_i =< x_{i_1}..x_{i_k}, x_{i_{k+1}}..x_{i_n} >$, we view the instantiations of the features $< x_{i_1}..x_{i_k} >$ to be one node in the graph, and the example's second set of features $< x_{i_{k+1}}..x_{i_n} >$ to be a second node. Nigam and Ghani (Nigam & Ghani, 2000) formed similar bipartite graphs with their data by dividing their feature set randomly into two sets, and showed improvements on semi-supervised learning by using this feature set split.

Unipartite Graphs over all Example Features

More generally, if we consider each unique feature instantiation as a node in the graph, then an edge between two nodes represents cooccurrence between the two feature instantiations, and an example consists of the set of nodes which are its feature values, and the fully connected graph over them.

Unipartite Graphs over Single Feature Set By Projections

When working with a bipartite graph, we can also project it onto a unipartite graph, by considering the two feature sets as the set of nodes $n_i \in N$ and $c_k \in C$, then projecting as follows:

for all $n_i, n_j \in N$ create an edge e_{ij} from n_i to n_j if there exists a context in the bipartite graph c_k with e_{ik} and e_{jk}. Remove all context nodes c_k.

The projection is analogous for creating a unipartite graph for the context nodes c_k.

5.2.2 Graph Samples from an Underlying Distribution

We are working with a set of data collected for running experiments. The size of the dataset is limited by constraints such as disk space and algorithm run-time. We specifically sample web-pages, and each web-page contributes multiple <noun-phrase, context> edges to the graph. It is likely that if we sampled more web-pages, we would wind up with a graph with more nodes and edges. We can think of an underlying true distribution, and our dataset as a sample from that distribution. Figure 5.2 shows illustratively how a sample might differ from the underlying distribution, by containing fewer nodes and edges, though the general structure is similar.

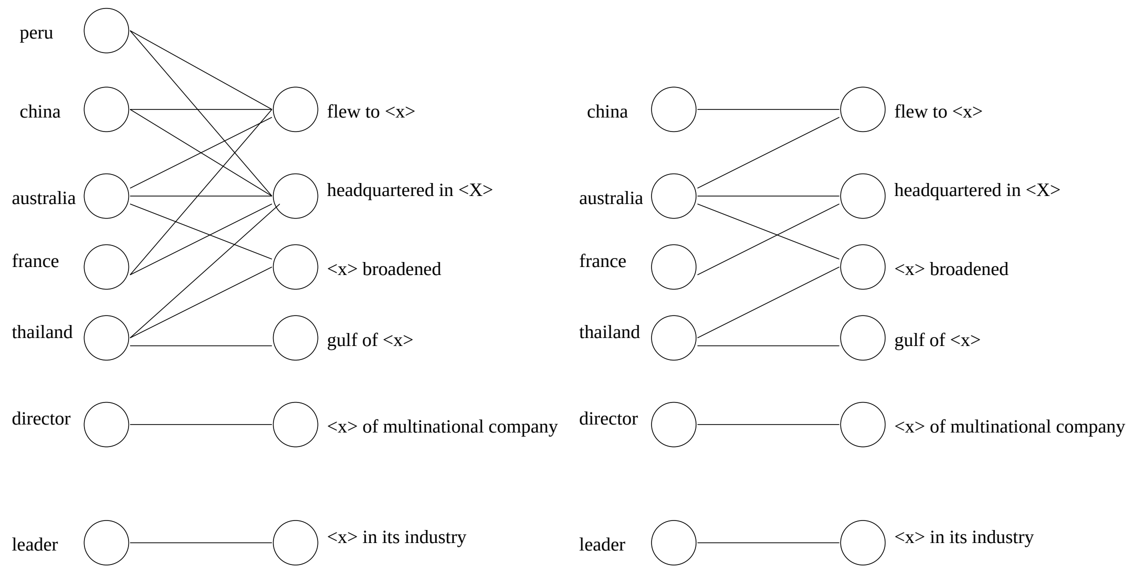


Figure 5.2: The whole graph is as shown at left. If we take a sample from the graph, we may wind up with a graph as shown at right. Some edges and some nodes are missing, though the general structure is similar.

The number of samples from a random graph affects whether the sample contains the full connectivity information (Karger, 1994). Our graph may not be random, so we will measure the effects of this sampling by checking graph properties for different size graphs.

5.2.3 Small World and Power-law Graph Properties

Recent work on naturally occurring graphs and networks has identified interesting properties of *small-world* and *power-law* (or *scale-free*) graphs (Albert & Barabási, 2002). Small-world graphs are graphs in which most nodes are within a few steps from most other nodes. Power-law graphs are those in which the degree distribution of nodes can be observed to obey a power law.

When looking for small-world characteristics, we can characterize graphs according to their behavior with respect to two properties:

- **clustering coefficient** C: the extent to which nodes tend to form fully connected cliques or mostly connected cliques

- **characteristic path length L**: the shortest path between a pair of nodes, averaged over all pairs of nodes

When looking for graphs with the power-law property, we measure

- **power-law coefficient** α: how well the node degrees fit a power law.

In this section we describe these and other graph properties in more detail, and highlight how we can expect algorithms to be affected when the data they are run on exhibit these properties.

Small World Properties: clustering coefficient and path length

To see whether our data exhibits the small-world property, we will examine the clustering coefficient (Newman et al., 2002). Intuitively, the clustering coefficient measures how densely connected the graph is, by measuring, for each node, how many of its neighbors are also neighbors of one another. For a node v_i which has a set of neighbors $n(v_i)$, we calculate this percentage of connected neighbors:

$$c(v_i) = \frac{\sum_{j \in n(v_i)} \sum_{k \in n(v_i)} I_{jk}}{n(v_i)(n(v_i) - 1)} \tag{5.8}$$

where $I_{jk} = 1$ if $j \neq k$ and $j \in n(k)$, 0 otherwise.

Then the formula for the clustering coefficient of the graph is given by averaging over all nodes:

$$C = \frac{\sum_{v_i \in V} c(v_i)}{|V|} \tag{5.9}$$

where V is the set of nodes in the graph. Note that there is an alternative definition of clustering coefficient, which weights each edge equally. Rather than averaging for each node, than averaging over nodes, using the alternative definition we perform the average over the edges. This gives greater weight to higher degree nodes. We will stick to the formulation in equation 5.9.

By definition, the clustering coefficient is always 0 for a bipartite graph, so we will perform a projection of the bipartite graph onto a unipartite graph, by joining any pair of noun-phrases with a common context, then calculating the clustering coefficient over the noun-phrases. We will perform the analogous projection to obtain the clustering coefficient for contexts.

For a random graph, the clustering coefficient is given by

$$C_{random} = \frac{2 \times (|E|)}{|V| \times (|V| - 1)} \tag{5.10}$$

where $|V|$ is the number of nodes in the graph.

The characteristic path length L is the average of the distances of all-pairs-shortest-paths over the entire graph. Note that for a disconnected graph, that is, one with multiple components, the path length for a pair of nodes in different components is infinite. The characteristic path length for a graph with multiple components will therefore also be infinite. We will therefore calculate the characteristic path length only over the largest component in the graph.

The characteristic path length for a random graph is given by:

$$L_{random} = \frac{\ln(|V|)}{\bar{k}} \tag{5.11}$$

where $|V|$ is the number of nodes in the graph, and $\bar{k}$ is the average node-degree in the graph.

Watts and Strogatz relate the characteristic path length L and the clustering coefficient C (Watts & Strogatz, 1998) of a graph. They define a small-world graph to be one on which $L \geq L_{random}$ (the characteristic path length of a random graph of the same size) but $C \gg C_{random}$ (the clustering coefficient of a random graph of the same size).

Small-world graphs are those which lie in between, at the one extreme, *regular graphs*, which require a large number of steps to move between arbitrary pairs of nodes, but are locally highly clustered, and at the other extreme *random graphs*, which require only a small number of steps to move between arbitrary pairs of nodes.

It has been shown that word cooccurrence graphs and synonymy relationships do not exhibit random graph structure, but rather a small-world structure, with most nodes reachable from most other nodes within two to three steps (Ferrer i Cancho & Solé, 2001; Sigman & Cecchi, 2002). If our noun-phrase context pairs exhibit this kind of structure, and pronouns and very common nouns form the hubs of the connectivity graph, it may explain the importance of stopwords and frequent seeds in our models. Recall from Chapter 3 Section 3.7.2 that that if we did not permit stopwords (including pronouns) in the model, it had a deleterious effect on learning the `people` and `organizations` classes. Understanding the effects of removing stopwords on the connectivity of the graph may help explain this.

Steyvers and Tenenbaum (2005) studied the graph structure of semantic relationships from WordNet (Miller et al., 1990), constructed with great care by psycholinguists, as well as word associations from people prompted to respond spontaneously to word cues (Nelson et al., 1999). They found that the graphs exhibited both small-world and power-law structure, and hypothesized a model of language acquisition, in which newly learned concepts are attached to well-known concepts with greater likelihood.

Average Node Degree

The average node degree is a simple measure of the local connectivity within a graph. If on average, nodes are connected to many other edges, then a bootstrapping algorithm labeling one node will affect many more nodes, on average, after few steps. If the average node degree is small, more bootstrapping steps may be required to impact many nodes in the graph.

Power-law Distribution of Node Degree

In power-law graphs the distribution of node degrees follows a power law, where the probability p_k of a node having k neighbors is given by

$$p_k \sim ck^{-\alpha} \tag{5.12}$$

where c is a constant, and α is the power-law coefficient, which reflects how extreme the disparity of node-degree in the graph is.

This means that most nodes are connected to few other nodes, while a few nodes are connected to a large number of other nodes. If our data has this property, then we might expect the degree of the nodes representing our labeled examples to be of some importance in predicting algorithm effectiveness on a data set. It could affect the propagation both of accuracies and inaccuracies in the model. For example, in the cotraining setting, if we correctly label a high-degree node, we obtain correct labels for many different adjacent nodes. Many different examples sharing one half part of the split feature set can be labeled correctly via this node. Conversely, an incorrect label on a high-degree node can propagate the error to many other examples.

The smaller the exponent, the more extreme the difference in node degree between high-degree nodes and low-degree nodes. In a graph of word-cooccurrence data (Ferrer i Cancho & Solé, 2001; Sigman & Cecchi, 2002), the coefficient of the power law was 3 if measurement was restricted to the largest connected component of the graph, and 1.8 otherwise. Ferreri Cancho and Solé explain this in terms of a core vocabulary, found in the main connected component, and specialized vocabulary which falls into the other components. From a node-degree perspective, we can see that considering the components outside the many connected component adds many low degree nodes to the computation.

A graph divided into underlying groups or communities may explain degree correlations (degree of adjacent nodes positive correlated) and clustering. Newman and Park (2003) show that the value of α from the power law is also predictive of the clustering coefficient. In particular, for $\alpha < \frac{7}{3}$, we expect to see large values of the clustering coefficient C, as C increases with increasing system size. Strogatz (2001) suggests that if $1 < \alpha < 3.47$, the nodes in the graph form a large, but not fully connected component, whereas if $\alpha < 1$ there are so many high-degree hubs that the network forms one connected component. Steyvers and Tenenbaum (2005) mention that α typically lies between 2 and 4 for systems like the WWW and metabolic networks. For the three semantic networks: WordNet, Roget's thesaurus and associative networks, α was between 3.01 and 3.19. This was calculated by calculating node-degree separately for each of words and classes, only on the largest connected component. For the graph consisting of header files included in C files, α ranged between 1.9 and 2.9 (de Moura et al., 2003). For graphs linking users when they access the same data resource (Iamnitchi et al., 2004), the graphs are small-world, but the degree distribution does not necessarily follow a power-law.

In their analysis of cotraining, Blum and Mitchell assume random graphs with connected sub-components (Blum & Mitchell, 1998), which are disjoint from one another. Instead as we will see, the data has power-law distribution. Pastor-Sartorus and Vespignani (Pastor-Sartoras & Vespignani, 2001) showed that as the infectiousness of a disease increases, scale-free or power-law graphs show more gradual rates of infection than do random graphs. We can think of properties of the labeling component of semi-supervised learning algorithms as analogous to infectiousness. A labeling algorithm which requires more information before labeling can be viewed as less infectious.

Connected Components

In chapter 3 we discussed several bootstrapping algorithms that use cooccurrence of noun-phrases and contexts to propagate label information from a few labeled examples to the entire unlabeled set. In Section 3.5.1 we drew attention to one of the assumptions underlying this approach, that the seeds chosen by the user will be present in the data. We measured variable density of seeds according to the seed set chosen.

We would now like to draw attention to another critical assumption underlying these algorithms that has remained implicit till now. Each of the algorithms we described made use of cooccurrence of noun-phrases and contexts to propagate labels. Since we hope to learn about a phrase from its cooccurrences, and our algorithms transmit information about likelihood of class membership through cooccurrence links, we are dependent on the existence of links between portions of the graphs which have labels on the edges, and portions of the graphs which have no labels on the edges. In Figure 5.1, the portion of the graph which contains "<leader, $< x >$ in its industry>" is a separate component. We have no labeled edges in this component.

If some of the data we need to learn about is in a disconnected component from the region labeled by the initial seeds or examples labeled during active learning, no algorithm using connectivity information will be able to learn the true labels of these disconnected examples. Thus the connectivity of the cooccurrence graph is key to the success of any bootstrapping algorithm. Agichtein et al. (2003) measured *reachability* of data for query-based sampling strategies, taking into account the power-law distribution of their data, with a large connected component and many smaller components, and quantified the learnability of two different tasks over several corpora. We will examine not only reachability, but examine the correspondence between the distribution of labeled data over connected graph components and algorithm performance.

Graph Connectivity and Initialization Conditions

We can propagate label information only through edges on the graph. In particular, we cannot propagate label information from one component of the graph to another disconnected component. In Figure 5.1, we cannot use labels from other portions of the graph to learn to label the edge in the disconnected component which contains

"<leader, $< x >$ in its industry>". Thus our set of initial examples and their distribution over components in the graph will be key in our how effective the semi-supervised learning algorithm can be.

Graph Connectivity and Active Learning

The connectivity of the graph may also explain the importance of active learning for algorithm effectiveness. Active learning may compensate for the lack of component coverage in initial examples, by selecting examples for labeling which lie in different components of the graph.

5.2.4 Measuring Graph Properties of Noun-phrase Context Data

In this section we measure the graph-theoretic properties of our data, and where appropriate show whether the graph properties of labeled examples are predictive of algorithm performance.

Average Node Degree

The mean degree of noun-phrases is 2.32 in the `7sector` training data, while the mean degree of contexts is 7.56. This indicates that class information about a noun-phrase can be propagated to just over two different contexts on average, while class information about a context can be propagated to over seven different noun-phrases. Labeling a noun-phrase in isolation will affect fewer nodes than labeling a context in isolation. We also can expect a variety sources of information about the label of a context, since on average seven different noun-phrases will be connected to it, providing more information than the two contexts connected on average to a randomly selected noun-phrase. This suggests that a bootstrapping algorithm such as metabootstrapping, which labels all noun-phrases cooccurring with a given context, will propagate information, or noise, quickly throughout the graph.

We also examined average node degree $\bar{k}$ by class in the `7sector` test set, for both noun-phrases and contexts. We see in Table 5.2.4 that nodes in our target classes tend to have higher average degree than the overall average node degree in the test set. This could be explained by the fact that pronouns are included in these classes.

Class	$\bar{k}_{np}$	$\bar{k}_{context}$
locations	1.6	5.1
organizations	2.7	4.6
people	3.3	3.6
whole test set	1.6	3.6

Table 5.5: Average node degree $\bar{k}$ by class in the `7sector` test set, for both noun-phrases and contexts. We see that nodes in our target classes tend to have higher average degree than the overall average node degree in the test set. This could be explained by the fact that pronouns are included in these classes. The presence of "he" and "we" may well explain the high average node degree for noun-phrases in the `organizations` and `people` classes. The higher average node degree for contexts in the locations class may be explained by the fact that there are more varied ways of referring to locations than to people and organizations.

The presence of "he" and "we" may well explain the high average node degree for noun-phrases in the `organizations` and `people` classes. The higher average node degree for contexts in the locations class may be explained by the fact that there are more varied ways of referring to locations than to people and organizations.

Power-law distribution of Node Degree

When we look at just the noun-phrases in our corpus, we see in Figure 5.3 that the distribution of the number of contexts they are linked to follows a power law. In Table 5.6 we see the noun-phrases with the highest degree, which are connected to the most different contexts, ie these are the *hubs* of the graph. Note that some of these examples, such as "company" and "customers" are common nouns which are members of our target classes. Others are pronouns which would also be members of our target classes; "he" for example is a member of the `people` class.

Figure 5.3 also shows the distribution of outdegrees for contexts. Table 5.6 also shows the contexts with the highest degree. This list contains a mixture of very ambiguous contexts, like "including", which could occur with almost any noun-phrase, and quite unambiguous ones, like "said" which would occur primarily with people and occasionally with organizations.

We can find the coefficient of the power law, by fitting a line to the log-log graph. We have the probability p_k of a node having k neighbors given by the formula

Noun-phrase	Outdegree
you	1656
we	1479
it	1173
company	1043
this	635
all	520
they	500
information	448
us	367
any	339
products	332
i	319
site	314
one	311
1996	282
he	269
customers	269
these	263
them	263
time	234

Context	Outdegree
<x> including	683
including <x>	612
<x> provides	565
provides <x>	565
provide <x>	390
<x> include	389
include <x>	375
<x> provide	364
one of <x>	354
<x> made	345
<x> offers	338
offers <x>	320
<x> said	287
<x> used	283
includes <x>	279
to provide <x>	266
use <x>	263
like <x>	260
variety of <x>	252
<x> includes	250

Table 5.6: The twenty noun-phrases and contexts with the highest out-degree. The out-degree is the number of different contexts that the noun-phrase cooccurs with. The noun-phrase list contains a mixture of pronouns, anaphora and common nouns. The context list contains a mixture of very ambiguous contexts, like “including”, which could occur with almost any noun-phrase, and quite unambiguous ones, like “said” which would occur primarily with people or organizations.

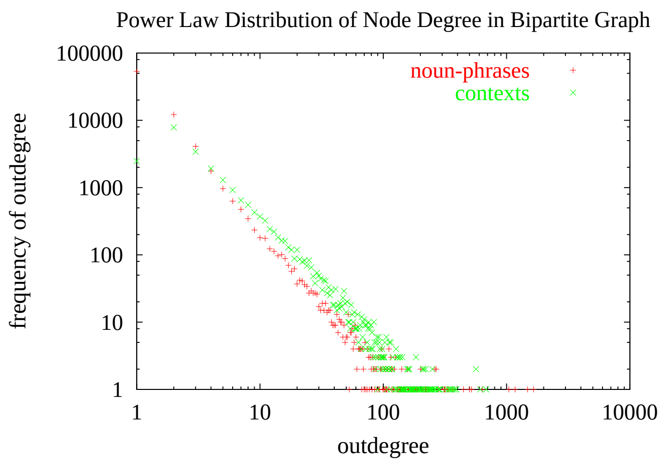


Figure 5.3: Some noun-phrases occur with many different contexts, while most occur with few; the distribution of links follows a power-law, suggesting a small-world graph structure.

$$\begin{aligned} p_k &= ck^{-\alpha} && (5.13)\\ \log(p_k) &= \log(ck^{-\alpha}) && (5.14)\\ \log(p_k) &= \log(c) - \alpha \log(k) && (5.15) \end{aligned}$$

where c is a constant that accounts for the intercept. Then $-\alpha$ is the slope of a line we fit to the data points, when plotted on log-log axes.

For contexts, the coefficient of the power law α is 1.95, *ie* we can express the formula for the number of noun-phrases for each context as

$$p_k \sim k^{-1.95} \qquad (5.16)$$

while for noun-phrases, the constant α in the power law is 2.24, *ie*

$$p_k \sim k^{-2.24} \qquad (5.17)$$

Figure 5.4 shows the lines we fit to the graph of node degrees in the noun-phrase context graph. Since fitting power-laws graphically is difficult due to the sparsity in counts of the high-frequency elements (Goldstein et al., 2004), these lines were fit by manually adjusting the cut-off of node degree to be considered in fitting the line. All following coefficients were made without such adjustment, and as such may have significant bias. Nevertheless, we can still make comparisons between power-law

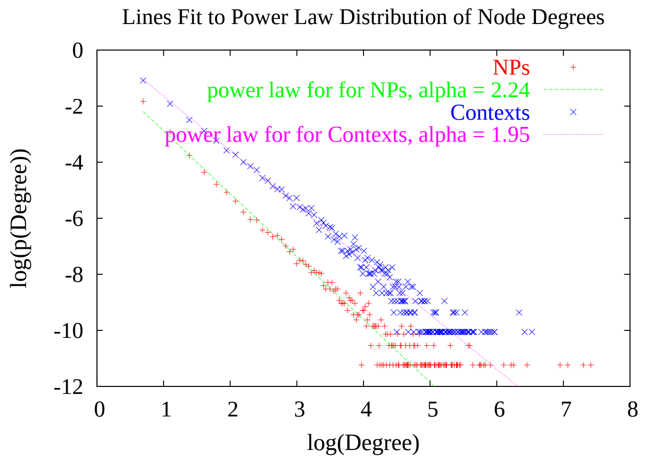


Figure 5.4: When we fit a line to the log-log plot, we find the power law parameter α is 2.24 and 1.95 for noun-phrases and contexts respectively.

coefficients calculated with the same kind of bias. Thus we will still be able to examine questions about the effect of sample size, and whether we consider only nodes in the same component, though the coefficients we measure may not be reliable.

Is our measured value of α the power law coefficient related to our sample size? As we discussed in Section 5.2.2, the exact set of nodes and edges in our graph may differ depending on the sample size. To test the effects of sample size for our data, we calculated the power law parameter α for samples of different sizes from the `7sector` data, as well as the TREC wtx051-053 data. In Figure 5.5 we see that the coefficient converges for both datasets[1]. For contexts, it converges to around 1.5 for both corpora, while for noun-phrases it converges to just under 1.6 for the `7sector` data, and just under 1.4 for the TREC data. This may suggest that the properties of context distribution are independent of corpus type or origin, though the properties of noun-phrase distribution do depend on the corpus type.

Finally, note that we measured our power-law coefficients over the entire graph, while other researchers have restricted their attention to the largest component. The nodes in the smaller components may tend to have lower degree, so measuring degree on the largest component only may lead to a smaller power law parameter α. To be able to compare our measurements of α with those of other researchers, we also calculated them on the largest component. Table 5.2.4 shows these numbers. We see

[1]though larger and larger subsamples of the same graph are less and less independent of one another.

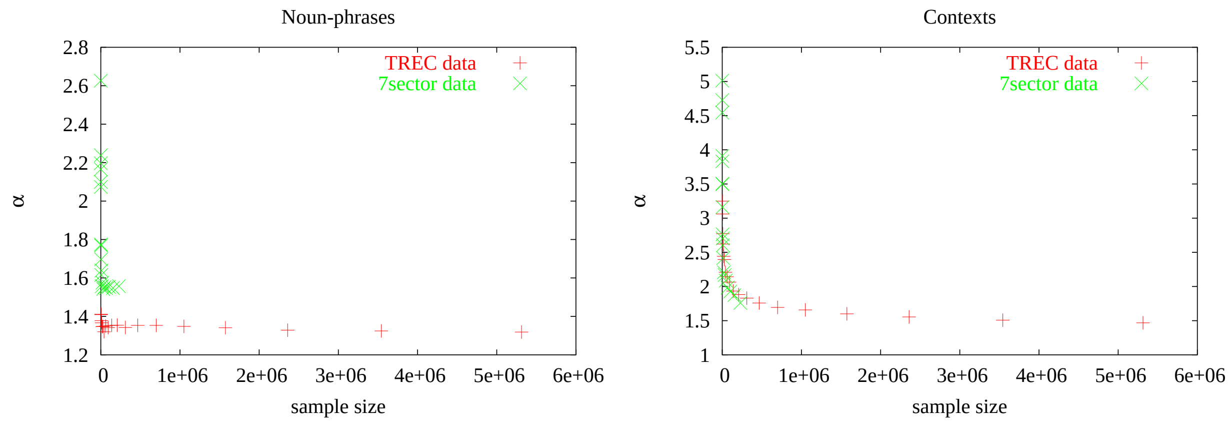


Figure 5.5: Asymptotic values for the power law coefficient α, calculated over two corpora. The coefficient converges for both datasets. For contexts, it converges to around 1.5 for both corpora, while for noun-phrases it converges to just under 1.6 for the `7sector` data, and just under 1.4 for the TREC data. This may suggest that the properties of context distribution are independent of corpus type or origin, though the properties of noun-phrase distribution do depend on the corpus type.

	noun-phrase degrees	context degrees	all node degrees
$\alpha_{wholeGraph}$	1.56	1.76	1.77
$\alpha_{largestComponent}$	1.55	1.75	1.77

Table 5.7: Power-law coefficients of node degree, over noun-phrases connected to contexts, over the whole graph and the largest component. We see that measuring the power law coefficient over the whole graph does not give a very different result to measuring it over the largest component.

that the coefficients do not vary greatly. This contrasts with the results obtained by Ferreri Cancho and Solé (2001; 2002) on a graph of word-cooccurrence data (Ferrer i Cancho & Solé, 2001; Sigman & Cecchi, 2002), and suggests that our data has more extreme power-law properties: the degrees of the most frequent nodes are much higher than those of the next highest-degree nodes. This may be explained by the different types of underlying data: our data includes an edge if a pair of phrases has any cooccurrence, while Ferreri Cancho and Solé include edges only when words cooccur with probability greater than chance.

Recall that when we have a graph with the power-law property, this means that most nodes are connected to few other nodes, while a few nodes are connected to a large number of other nodes. Since our data has this property, then we might expect the degree of the nodes representing our labeled examples to be of some importance in predicting algorithm effectiveness on a data set. It could affect the propagation both of accuracies and inaccuracies in the model. For example, in the cotraining setting, if we correctly label a high-degree node, we obtain correct labels for many different adjacent nodes. Many different examples sharing one half part of the split feature set can be labeled correctly via this node. Conversely, an incorrect label on a high-degree node can propagate the error to many other examples.

This simple observation suggests a hybrid approach to semi-supervised learning that has not been proposed previously, in which greater confidence is required for labeling high-degree nodes than low-degree nodes.

This power-law structure may also explain the poor performance of metabootstrapping and cotraining on our data. Both metabootstrapping and cotraining label examples positive or negative, unlike the probabilistic labeling of cotraining and coEM.

In addition, the power-law distribution of node degree is a likely explanatory factor for the different behavior of bootstrapping across classes. `people` and `organizations` both contain high-degree nodes (pronouns) as members of the class, leading to more volatile behavior depending on whether these nodes are labeled as positive or negative.

We will examine the effect of labeling examples with high node degree later in this section.

Clustering Coefficient

On the `7sector` training data, we find that the clustering coefficient is 0.83 for noun-phrases and 0.67 for contexts. This means that noun-phrases exhibit more "transitive" behavior, that is, noun-phrases tend to share common neighbors. When we restrict attention to the largest component, the clustering coefficients are 0.86 and 0.74 for noun-phrases and contexts respectively. In Table 5.9 we see these results along with the results for characteristic path length. We see that paths are short, and similar to the path-length in a random graph, while the clustering coefficient is much higher than for a random graph. This means our graph has small-world properties.

	noun-phrases	contexts
whole graph	0.83	0.67
largest component	0.86 (0.0018)	0.74 (0.025)

Table 5.8: Clustering coefficients C for unipartite graphs of noun-phrases and contexts, for both the entire graph, and for the largest component. C_{rand}, the clustering coefficient for a random graph of the same size, is shown in parentheses for the largest component.

	$\|V\|$	$\bar{k}$	L_{rand}	L	C	C_{rand}
noun-phrases	71,090	62	2.7	2.7	0.86	0.0018
contexts	21,039	265	1.78	2.54	0.74	0.025
bipartite	92,129	1.86	18	5.4	-	-

Table 5.9: Characteristic path length L, or average average path length within the graph, and clustering coefficient C, for the largest component in the graph. We see that paths are short, and similar to the path-length in a random graph, while the clustering coefficient is much higher than for a random graph. This means our graph has small-world properties.

Characteristic Path Length

We measured the characteristic path length over the unipartite graph on noun-phrases in the largest component, as well as the unipartite graph over contexts in the largest component. For contexts the characteristic path length was 2.5 , that is, the average number of context-steps between a pair of contexts is 2.5. For the noun-phrase unipartite graph, the characteristic path length was 2.7. Over the bipartite graph, the characteristic path length was 5.4. This means that an algorithm which alternates taking suggestions from noun-phrases and contexts can relabel all nodes in the largest component within 6 steps. Table 5.9 shows these values for the largest component of our graph, for both the bipartite graph, and both unipartite projections. We see that paths are short, and similar to the path-length in a random graph, while the clustering coefficient is much higher than for a random graph. This means our graph has small-world properties. Labeling a node may affect any other node in the graph in just a few steps, and a node may acquire labels from its tightly coupled neighborhood. This mean we need bootstrapping algorithms which are accurate or cautious with each label assignment, as errors will be easily propagated.

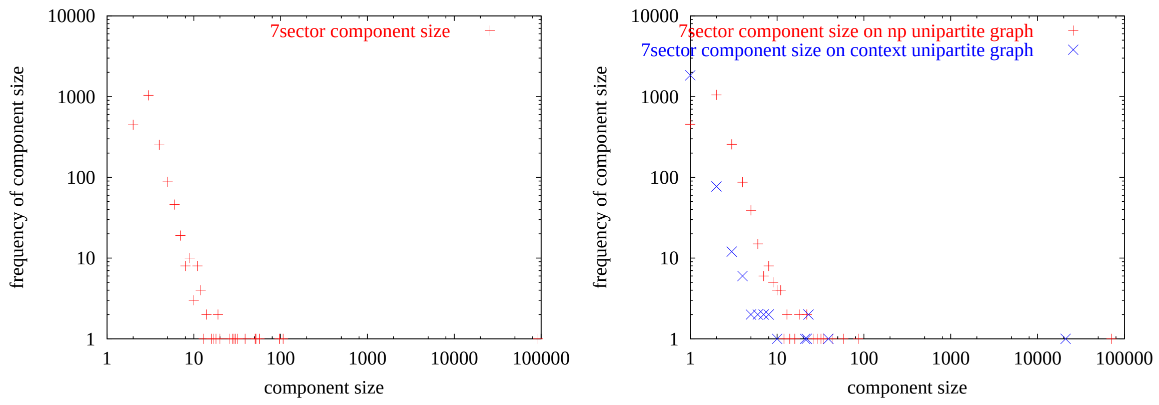


Figure 5.6: The 7 sector data has many components with 2 or 3 nodes. When we project down to unipartite graphs, we see that this distribution holds for both noun=phrases and contexts.

Connected Components

The connectivity of the cooccurrence graph is key to the success of any bootstrapping algorithm. Since we hope to learn about a phrase from its cooccurrences, and our algorithms transmit information about likelihood of class membership through cooccurrence links, we are dependent on the existence of links between labeled and unlabeled portions of the graph.

Measuring the connectivity of the 7sector-train corpus, we find 1945 separate connected components. 92129 nodes of 99014 are in the largest component, *ie* 93% of all nodes are connected. However, this leaves 7% of nodes which are not part of the large connected component. The second largest component contains only 107 nodes, with most components containing less than 10 nodes. We can see this graphically in Figure 5.6 shown on a log-log scale. The 7 sector data has many components with 2 or 3 nodes. When we project down to unipartite graphs, we see that the distribution of nodes in components is similar for both noun-phrases and contexts, with most in the largest component, and many smaller components containing few nodes. And viewed as unipartite graphs of noun-phrases or contexts, the majority of components contain a single isolated node. Labeling any node in a singleton component like this cannot affect any of the other nodes in the graph.

5.3 Predicting Performance with Graph Properties

In the previous section we showed that our graph shows small-world and power-law properties. We now examine the interaction between these properties, our training data, and algorithm performance.

Recall from Section 3.3 that we initialize our bootstrapping algorithms with a small set of seed words, which are examples of the target class. We could conjecture that seed set frequency will be more important if a graph consists of many unconnected components, and the seeds occur in the largest connected component, or many different components. We observed that frequently occurring examples are important to algorithm effectiveness. Thinking of our data now in small-world graph-theoretic terms, we can see that frequently occurring terms are more likely to be hubs, and are more likely to be connected to many other examples.

We find that for the basic seed sets for the classes `people, locations` and `organizations`, given in Table 3.8 in Chapter 3, all 10 seeds can be found in the main connected component of the graph. Thus difference between the performance of algorithms over these tasks cannot be explained by differing presence in the main connected component.

For the random sets of country names, described in Section 3.5.1, more varied distribution can be found in the training set. We showed in Chapter 3 Table 3.10 that different sets of seeds have quite a large variance in the number of examples they label in the training set, both in absolute numbers of examples, and in the number of unique examples. We note that the number of unique examples labeled by the seeds is exactly the sum of node degrees of the noun-phrases labeled, since each unique example labeled corresponds to one edge from a noun-phrase to a context. We will first consider how predictive this is of algorithm performance, then contrast it with the number of seeds found in the large connected component.

Figure 5.7 shows a scatter plot of the total node degree of seeds, against the final breakeven score, for multiple experiments with random subsets of country names as seeds. On the left is a scatter plot of the node degree of seeds, against breakeven score. On the right, we have converted each to ranks, to allow us to find correlations that may be non-linear. We see that the node degree of seeds is correlated with algorithm performance, both linearly (left, Pearson correlation of 0.772) and in ranks

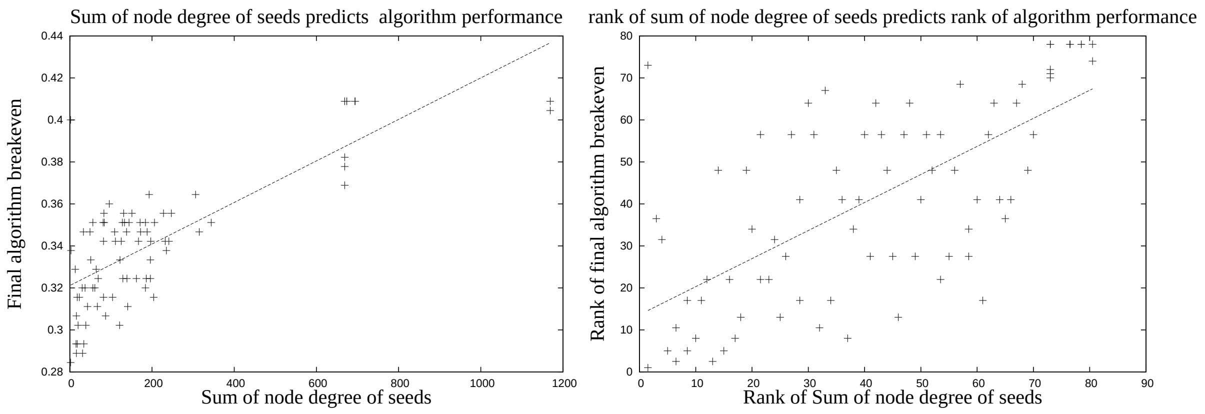


Figure 5.7: Total node degree of examples labeled with seeds is correlated with algorithm performance, both linearly (left, Pearson correlation of 0.772) and in ranks (right Spearman correlation of 0.670). Both correlations scores are significant at the 0.01 level.

(right Spearman correlation of 0.670). All correlations scores are significant at the 0.01 level. In this case, the linear correlation is stronger than the rank correlation, but we will continue to use rank correlations with all our predictive variables, to allow for non-linear but monotonic correlations.

In Table 5.10 we see a list of features which are predictive of algorithm performance, along with their Spearman correlation scores. All correlations are significant at the 0.01 level. We will now go through each, explaining why each may be relevant.

5.3.1 Number of unique seeds head-matching some NP in graph

When we choose a set of seeds for initial labeling, we make the assumption that those seeds will appear in the data. If we choose the seeds without inspecting the data, however, we may find that some of the seeds are not present at all. In addition, we may have better results, the more matching seeds are present in the graph. We found in fact, a weak but statistically significant positive correlation between the number of unique seeds matching noun-phrases heads in the graph. The Spearman correlation coefficient was 0.295.

Feature	r_s
Number of unique seeds head-matching some NP in graph	**0.295**
Number of unique seeds exact-matching some NP in the graph	**0.302**
Number of unique seeds head-matching NPs in the largest component	**0.295**
Number of unique examples labeled (sum node degree)	**0.670**
Total examples labeled	**0.678**
Number of components containing at least one seed	**0.541**
Number of unique seeds-examples or positive example in the largest component	**0.669**
Number of unique contexts covered by seeds	**0.657**
Number of unique contexts covered by more than one seed	**0.716**

Table 5.10: Features predictive of algorithm performance, along with their Spearman correlation coefficients.

5.3.2 Number of unique seeds exact-matching some NP in the graph

If a seed matches a noun-phrase exactly, that is, there is no modifier, it may be more unambiguously correct as an initial example. We found in fact that the number of unique seeds exact matching noun-phrases was statistically significantly correlated with breakeven score, with a Spearman correlation of 0.302. This is slightly higher than for unique seeds head-matching some noun-phrase in graph (though the difference is not statistically significantly different).

5.3.3 Number of unique seeds head-matching NPs in the largest component

Since most examples are in the largest component, as we saw in Section 5.2.4 we may expect that the distribution of seeds in the largest component may be predictive of algorithm performance. While there is a positive correlation of 0.295, this correlation is no stronger than the number of seeds found in the total graph.

5.3.4 Number of Unique Examples labeled - Sum of Seed Node Degrees

As we have discussed already, the number of unique examples labeled by seeds, that is, the total unique $< n, c >$ pairs labeled with seeds, is correlated with performance, with a correlation score of 0.670.

5.3.5 Total examples Labeled By Seeds

In machine learning, it is customary to measure the relationship between examples labeled and algorithm performance. The total examples labeled is slightly higher than the number of unique examples, since some examples occur more than once. It is not obvious whether this feature would be more predictive of algorithm performance than unique examples labeled, since it does not strictly speaking introduce new information about example types, though it does introduce more information about the distribution. We find in fact that the number of examples labeled by seeds correlated with algorithm performance with a score of 0.678. This is higher than the score for the number of unique examples labeled, though not statistically significantly different. They are strongly correlated with one another, with a Spearman correlation score of 0.997.

5.3.6 Number of components containing at least one seed

Knowing that our data falls into multiple components, albeit of disparate sizes, we might expect the number of components containing seeds to be predictive of algorithm performance. Recall, however, from Section 5.2.4 that only 7% of examples were found outside the largest component, so using seeds to label examples in the largest component may be sufficient. On the other hand, bootstrapping can only propagate labels to other examples in the same component. Figure 5.8 shows the breakeven score against the number of components covered by seeds in the `locations` class, once again using a variety of sets of country names as seeds. The experiments with seeds in four components appear to have better results than the experiments with seeds in only 1 or 2 components.

The correlation is 0.541, larger than the correlation with the number of seeds in the graph, which had a correlation of 0.295. This difference is statistically significant,

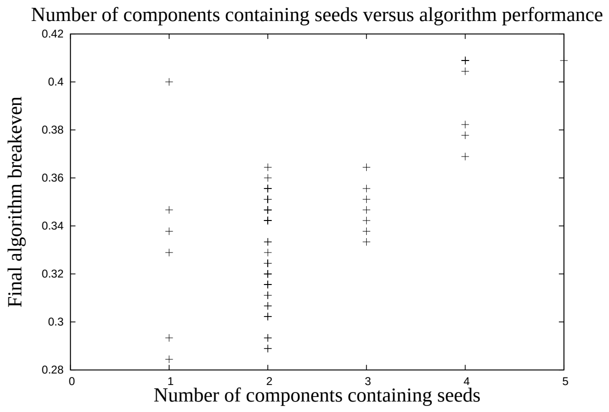


Figure 5.8: We find that the number of components covered by seeds (Spearman rank correlation of 0.541) is more predictive than the total number of seeds in the graph (Spearman rank correlation of 0.295).

which means it may be important to choose seeds spanning multiple components of the graph.

5.3.7 Number of unique seed-labeled examples in the largest component

Now we examine the number of seeds contained in the largest component for the random country seeds sets. Since the bulk of our data is in the largest component, it is important that we have some seeds providing initial labels for members of the largest component. We would clearly expect that having having a single seed in the largest component would be better than having none. In addition, if some seeds are better than others (for examples, by being less ambiguous, or occurring with more informative contexts in our training data), then having more seeds in the largest component increases our chance of having a good seed in the largest component. Finally, if seeds are noisy or ambiguous, having multiple seeds provides different types of noise, with an amplification of the useful signal we can learn from. Seeds in disjoint components cannot interact in labeling contexts, while multiple seeds in the same component, particularly the largest component, are likely to interact in ways which may counteract the effects of ambiguity or noise. When we look at the number of unique examples labeled by seeds in the largest component, we find a strong positive correlation of 0.669. This may reflect the fact that we are doing a good job of providing initial examples which will then propagate to the majority of

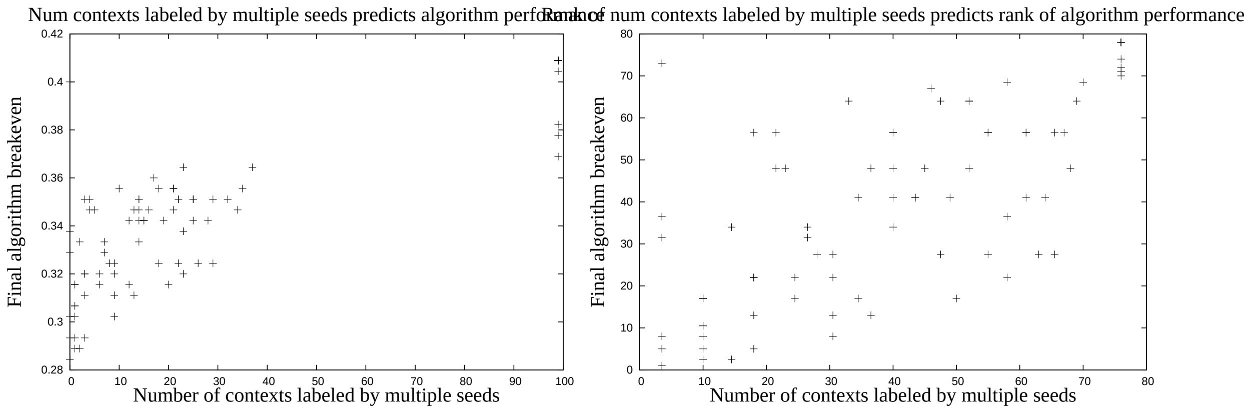


Figure 5.9: The number of contexts labeled by more than one seed is strongly predictive of algorithm performance both linearly (left, Pearson correlation of 0.80) and in rank (right, Spearman correlation of 0.72)

examples, due to short path-lengths and the power-law distribution of node degrees.

5.3.8 Unique Contexts Covered By Seeds

Each example labeled by a seed implicitly assigns a label to the associated context. The number of unique contexts labeled by seeds has a correlation of 0.657, while the number of unique contexts covered by more than one seed 0.716. Looking back at Table 5.10, we see that the number of contexts labeled by multiple seeds is the strongest predictor of algorithm performance (Spearman correlation of 0.72). To understand this better, we show scatter plots of raw counts and scores, and their ranked equivalents in Figure 5.9.

In Table 5.11 we show example contexts that were extracted by multiple seeds in the original locations seed set (given in Table 3.8 in Chapter 3), as well as contexts extracted by only one of these seeds. We see that those selected by more than one seed appear to be more unambiguously indicative of the target class `locations`.

5.3.9 Combinations of Predictors with Multivariate Regression

Next we would like to know whether some combination of the features shown in Table 5.10 is more predictive than any of the features individually. To do this, we performed

Context	Num Seeds Selected By
OPERATIONS:IN	10
LOCATIONS:IN	9
<X> COMMENTS	8
<X> UPDATED	7
OFFICES:IN	6
OPERATES:IN	6
HEADQUARTERED:IN	6
FACILITIES:IN	5
CUSTOMERS:IN	5
OWNED:IN	1
ORIGINATED:IN	1
GROWN:IN <X>	1
FOUND:IN <X>	1
FILED:IN <X>	1
DUE:IN <X>	1
TARGETING $< X >$	1
COVERING <X>	1

Table 5.11: The number of contexts labeled by more than one seed was most predictive of algorithm performance. We see from a sample of contexts, that those selected by more than one seed appear to be more unambiguously indicative of the target class `locations`.

Feature	Coefficient
Number of unique seeds head-matching NPs in the largest component	-0.267
Total examples labeled	2.059
Number of unique seed-labeled-examples in the largest component	-2.370
Number of unique contexts covered by more than one seed	1.210

Table 5.12: Using the interactions between multiple predictor variables, we obtain a correlation of 0.78 with the rank of algorithm breakeven, higher than the correlation of 0.72 that we obtained from the best predictor (unique contexts covered by more than one seed) in isolation.

multiple linear regression over the ranks of the features. This gives us the Spearman correlation between linear combinations of the ranks of those features and the rank of algorithm breakeven. We performed backwards regression, in which features are greedily removed until removing another feature changes the correlation score by more than a small amount. The final set of feature in the model is shown, along with their normalized coefficients, in Table 5.3.9. Using the interactions between multiple predictor variables, we obtain a correlation of 0.78 with the rank of algorithm breakeven, higher than the correlation of 0.72 that we obtained from the best predictor (unique contexts covered by more than one seed) in isolation. While the coefficients here show how the line was fit against the ranks of these variables, interactions between these and the other variables mean that we cannot draw precise conclusions from these coefficients. Nevertheless, the strong correlation with breakeven scores means that we could use a combination of these features to design better seed selection algorithms.

.

5.3.10 Cross-class Comparison of Node Degree as Predictor

Overall we find that the total node degree of examples labeled is more predictive of algorithm performance than the number of components we find seeds in, or the number of seeds we find in the largest component. These comparisons have all been over the locations class. We will now measure the predictiveness of node degree of algorithm performance, across the three classes `locations`, `people` and `organizations`. We see in Figure 5.10 that node degree is predictive of performance, even across classes. The number of data-points here is too few to calculate Spearman rank correlation meaningfully, but by visual inspection we can see that the correlation seems to hold

Feature	r_s
Number of unique seeds head-matching some NP in graph	**0.282**
Number of unique seeds exact-matching some NP in the graph	**0.285**
Number of unique seeds head-matching NPs in the largest component	**0.282**
Number of unique examples labeled (sum node degree)	**0.630**
Total examples labeled	**0.628**
Number of components containing at least one example	**0.501**
Number of components containing at least one seed or positive example	**0.529**
Number of unique seed-examples or positive example in the largest component	**0.624**
Number of unique contexts covered by seeds	**0.551**
Number of unique contexts covered by more than one seed	**0.581**
Number of examples labeled during active learning	**0.434**
Number of positive examples labeled during active learning	**0.460**
Percent positive examples labeled during active learning	**0.167**
Number of nonseed examples labeled during active learning	**0.434**
Number of nonseed examples labeled positive during active learning	**0.460**
Percent of nonseed examples labeled positive during active learning	**0.167**

Table 5.13: Features predictive of algorithm performance, along with their Spearman correlation coefficients, when we consider experiments conducted with active learning, over 255 combinations of random sets of `locations` seeds, number of example labeled with active learning, and active learning method. All correlations are significant are the 0.01 level.

in these cases.

5.3.11 Graph Features and Active Learning

Next we would like to check how these features correlated with algorithm performance when we perform active learning. Some of the seed dependence may be reduced, as the examples selected during active learning compensate for poor initial seeds. Table 5.13 shows that some of the seed-based features are stronger than the active-learning based features. We see that while the correlation with number of contexts covered by more than one seed is reduced, from a correlation of (0.72) without active learning, to 0.58 with active learning, it still seems to be an important factor.

We now perform multivariate regression with the experiments using active learning. In Table 5.14 we see the variables selected for inclusion in the model, along with their coefficients. The correlation of this model with algorithm performance is 0.73,

Feature	Coefficient
Number of unique seeds head-matching NPs in the largest component	-5.475
Number of unique examples labeled	-1.747
Total examples labeled	2.400
Number of unique contexts covered by seeds	-2.756
Number of unique contexts covered by more than one seed	6.230
Number of positive examples labeled during active learning	3.514

Table 5.14: A combination of features for predicting algorithm performance for bootstrapping with active learning, over random seeds sets for `locations`. The correlation of this model with algorithm performance is 0.73, greater than the correlation of any individual feature in isolation.

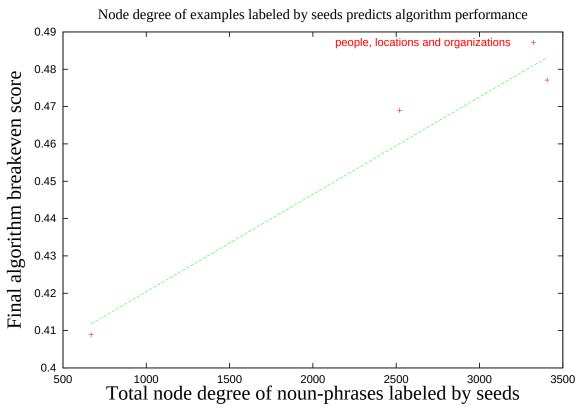
greater than the correlation of any individual feature in isolation. We see that the number of contexts selected by more than one seed continues to be important, as does the number of *positive* examples labeled during active learning.

5.4 Feature Set Independence

We will now switch gears from examining the relationship between graph properties and algorithm performance, to looking at the way the two feature sets are interrelated, in an information-theoretic sense. Specifically we will examine the assumption that the two features noun-phrases $\mathcal{N}$ and contexts $\mathcal{C}$ were conditionally independent, given the target class. This kind of independence assumption is common in machine learning, and is generally stated but not examined. Exceptions are Nigam and Ghani (2000) and Muslea (2000) who performed empirical evaluations by deliberately manipulating the level of independence between the features). Jensen and Neville (2002) examine the effects of feature dependence in relational learning. We will examine the independence assumption in greater detail, by measuring mutual information between feature sets, exploring the dependence of this on sample size, and drawing some tentative conclusions about its importance in our task.

We assumed that the two features noun-phrases $\mathcal{N}$ and contexts $\mathcal{C}$ were conditionally independent, given the target class. This means that we assumed:

$$\forall n \in \mathcal{N}\ \forall c \in \mathcal{C}\ \forall\ class \in Classes\ P(n, c|class) = P(n|class)P(c|class) \tag{5.18}$$



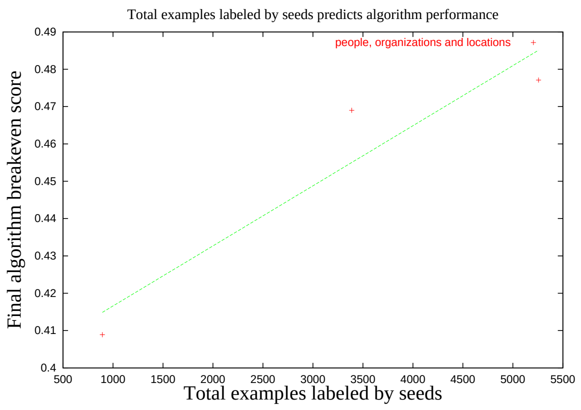


Figure 5.10: Node degree of examples labeled with seeds versus breakeven, across classes (left) and total number of examples labeled with seeds versus breakeven (right). We find that even across classes, the total node degree of examples labeled with seeds is predictive of algorithm performance.

In Table 5.15 we see some examples of noun-phrases and contexts, and their joint and marginal probabilities given the target class `locations`. We see that empirically, conditional independence given the target class does not hold, since $P(n|class) \times P(c|class) \neq P(n,c|class)$. In this section we measure how far from the ideal the distributions are, by measuring the mutual information between the feature sets, given the target class.

We used the assumption of class-conditional feature independence in two ways: (1) to assign the probability of class membership to an example (as described in Section 3.4.6 in Chapter 3) and (2) as the basis for multiview algorithms (it underlies the coEM algorithm (Nigam & Ghani, 2000), as well as the probabilistic model for EM described in Section 3.4.2). We will now examine this assumption more closely, as a way of understanding the performance of the algorithms we explored in Chapter 3, as well as their performance with active learning, which we explored in Chapter 4.

We can explore the degree of dependence between NPs and contexts by calculating the mutual information between them. The mutual information between two discrete probability distributions X and Y is defined to be

$$I(X;Y) = \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \tag{5.19}$$

Note that if X and Y are independent, then $P(X|Y) = P(X)$. Since $P(x|y) = \frac{P(x,y)}{P(y)}$, when X and Y are independent $\frac{P(x,y)}{P(x)P(y)} = 1$, so $\log_2 \frac{P(x,y)}{P(x)P(y)} = 0$, so $I(X;Y) =$

noun phrase	context	class	$P(n\|class)$	$P(c\|class)$	$P(n\|class)\times$ $P(c\|class)$	$P(n,c\|class)$
united states	plants in <x>	1	0.0383	0.0109	0.0004	0.0109
united states	one in <x>	1	0.0383	0.0164	0.0006	0.0164
united states	organizations in <x>	1	0.0383	0.0055	0.0002	0.0055
united states	<x> experienced	0	0.0001	0.0003	0.0000	0.0001
united states	<x> is called	1	0.0383	0.0055	0.0002	0.0055
company	<x> markets	0	0.0066	0.0014	0.0000	0.0003
company	<x> mines	0	0.0066	0.0003	0.0000	0.0001
company	<x> offers	0	0.0066	0.0033	0.0000	0.0004
company	<x> operates	0	0.0066	0.0008	0.0000	0.0001
company	<x> owns	0	0.0066	0.0008	0.0000	0.0001
company	<x> produced	0	0.0066	0.0003	0.0000	0.0001
company	<x> produces	0	0.0066	0.0005	0.0000	0.0003
you	allows <x>	0	0.0346	0.0019	0.0001	0.0001
you	did <x>	0	0.0346	0.0009	0.0000	0.0005
you	do <x>	0	0.0346	0.0014	0.0000	0.0006
you	enjoy <x>	0	0.0346	0.0003	0.0000	0.0001

Table 5.15: Some examples of noun-phrases and contexts, and their joint and marginal probabilities given the target class `locations`. We see that empirically, conditional independence given the target class does not hold, since $P(n|class) \times P(c|class) \neq P(n,c|class)$. In this section we measure how far from the ideal the distributions are, by measuring the mutual information between the feature sets, given the target class. "United States" is used as an `organization` rather than a `location` in the example "United States experienced".

0. Thus the mutual information between two variables is bounded below by 0, when the two variables are completely independent. As the dependence between the variables increases, the mutual information increases. To find an upper bound for I(X;Y) we observe first that it can be rewritten in terms of entropies:

$$\begin{aligned}
I(X;Y) &= \sum_x \sum_y P(x,y) \log_2 \frac{P(x,y)}{P(x)P(y)} \\
&= \sum_x \sum_y P(x,y) [\log_2 \frac{P(x|y)}{P(x)}] \\
&= \sum_x \sum_y P(x,y) [\log_2 P(x|y) - \log_2 P(x)] \\
&= \sum_x \sum_y P(x,y) \log_2 P(x|y) - \sum_x \sum_y P(x,y) \log_2 P(x) \\
&= -H(X|Y) - \sum_x \log_2 P(x) \sum_y P(x,y) \\
&= -H(X|Y) - \sum_x \log_2 P(x) P(x) \\
&= H(X) - H(X|Y)
\end{aligned}$$

Similarly

$$I(X;Y) = H(Y) - H(Y|X)$$

Now $H(X|Y) \geq 0$ and $H(Y|X) \geq 0$ by the definition of entropy, so $I(X;Y) \leq H(X)$ and $I(X;Y) \leq H(Y)$ so we have

$$I(X;Y) \leq \min(H(X), H(Y)) \tag{5.20}$$

So the mutual information between X and Y is bounded above by the minimum of the entropy of the two random variables X and Y.

Ignoring the class variable for the time being, we can measure the mutual information between noun-phrases and contexts over corpora of different sizes. Table 5.16 gives the mutual information between the noun-phrases and contexts for several different training corpora. Note that the mutual information does not grow strictly with the size of the corpus.

dataset	size in examples	I(N;C)
7sector-test	8,081	9.05
7sector-train	228,574	9.90
wtx052	1,726,025	10.56
wtx051-052	3,555,045	10.52
wtx051-056	10,588,096	10.26

Table 5.16: Mutual Information between noun phrases and contexts

Now consider the joint distribution of noun-phrases and contexts $P(\mathcal{N}, \mathcal{C})$. We can consider any corpus of noun-phrase context pairs to be a sample from this underlying distribution. So when we calculate I(N;C) over a given corpus, we are estimating $I(\mathcal{N}; \mathcal{C})$ based on a finite sample. If $|\mathcal{N}|$ and $|\mathcal{C}|$ are large and our sample is small, we may underestimate $I(\mathcal{N}; \mathcal{C})$ when we use the sample to make the estimate, since there may be instances of $< n, c >$ pairs that we never see in our sample.

We can view the cooccurrences of $< n, c >$ pairs as links in a bipartite graph, as described in Section 5.2.1; our measurement of $I(\mathcal{N}; \mathcal{C})$ as another measurement of the connectivity structure in that graph. Karger (1994) showed that we can place a bound on the number of examples which we must see sampled from a graph, to find a spanning tree with high probability. However, Karger assumed a random graph, whereas we showed in Section 5.2 that our data exhibits small-world graph properties. We will estimate bounds on the number of examples we must see to reliably estimate mutual information empirically, by measuring changes in $I(\mathcal{N}; \mathcal{C})$ against changes in corpus size.

In order to understand how sample size affects the measured value of mutual information, we took random samples of instances from the test data (which contains 8081 instances), with 30 samples at each sample size, from 50 instances up to 8000 instances. We then calculated the mutual information between noun-phrases and contexts for each sample, and calculated the mean and standard deviation for mutual information at the given sample size. We see in Figure 5.11 that estimates of the mutual information between noun-phrases and contexts are dependent on the sample size, for samples of less than 2000 instances. Note that our classes of interest are all represented by less then 2000 instances in the test set (228 for `locations`, 888 for `people` and 1632 for `organizations`), so we expect the mutual information we measure over these samples will be smaller than for larger samples. Nevertheless, since the standard deviations for all sample sizes are small, we can expect that our

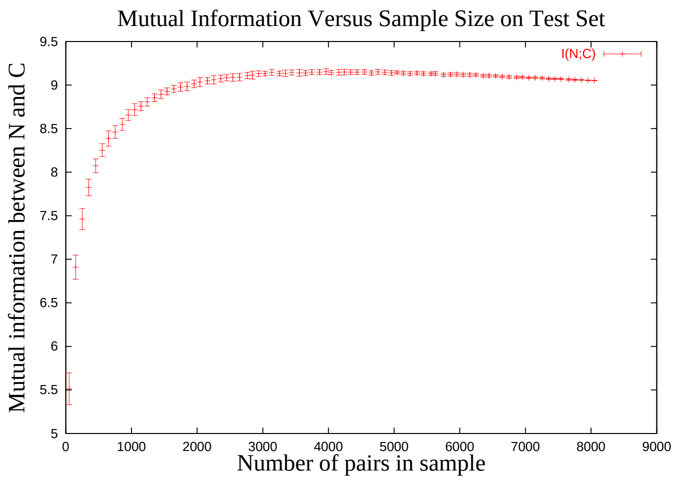


Figure 5.11: Mutual information between noun-phrases and contexts over random subsets of the test set. For each sample size, 30 random subsets were sampled; error bars of one standard deviation are shown. We see that as the sample size increases to around 2000 pairs, the mutual information is close to converging. However, at every sample size, the error bars are relatively small.

estimates of the mutual information between noun-phrases and contexts for each class will be fairly reliable.

The conditional mutual information is conditioned on the class label and is defined as in equation 5.21 (Cover & Thomas, 1991):

$$I(X;Y|Class) = \sum_x \sum_y \sum_{c \in Class} P(c)P(x,y|c) \log_2 \frac{P(x,y|c)}{P(x|c)P(y|c)} \tag{5.21}$$

We will compare the estimate for mutual information between the noun-phrases and contexts in a given class with a given size with upper bounds calculated from the entropy of the variables, and the mean value computed over a random sample of the same size. Recall from Equation 5.20 that the upper bound for $I(X;Y)$ is $\min(H(X),H(Y))$. For $I(X;Y|Class)$ we find an upper-bound analogously:

$$\begin{aligned} I(X;Y|Class) &= \sum_x \sum_y \sum_c P(c)P(x,y|c) \log_2 \frac{P(x,y|c)}{P(x|c)P(y|c)} \\ &= \sum_x \sum_y \sum_c P(c)P(x,y|c) \log_2 \frac{P(x|y,c)}{P(x|c)} \\ &= \sum_x \sum_y \sum_c P(c)P(x,y|c)[\log_2 P(x|y,c) - \log_2 P(x|c)] \end{aligned}$$

$$\begin{aligned}
&= \sum_x \sum_y \sum_c P(c)P(x,y|c)\log_2 P(x|y,c) - \sum_x \sum_y \sum_c P(c)P(x,y|c)\log_2 P(x|c) \\
&= \sum_x \sum_y \sum_c P(c)\frac{P(x,y,c)}{P(c)}\log_2 P(x|y,c) - \sum_c P(c)\sum_x \log_2 P(x|c)\sum_y P(x,y|c) \\
&= \sum_x \sum_y \sum_c P(x,y,c)\log_2 P(x|y,c) - \sum_c P(c)\sum_x \log_2 P(x|c)P(x|c) \\
&= -H(X|Y,Class) - \sum_c \sum_x P(c)P(x|c)\log_2 P(x|c) \\
&= -H(X|Y,Class) - \sum_c \sum_x P(x,c)\log_2 P(x|c) \\
&= -H(X|Y,Class) + H(X|Class) \\
&= H(X|Class) - H(X|Y,Class)
\end{aligned}$$

and since by the definition of entropy, $H(X|Y,Class) \geq 0$, $I(X;Y|Class) \leq H(X|Class)$. Similarly, $I(X;Y|Class) \leq H(Y|Class)$, so we have

$$I(X;Y|Class) \leq \min(H(X|Class), H(Y|Class)) \tag{5.22}$$

We now have lower and upper bounds on the mutual information between noun-phrases and contexts given the class. Let us now consider what values at these extrema would mean for bootstrapping approaches to learning semantic classes.

At the lower bound, when the two variables are completely independent given the target class, the mutual information between two variables is 0. An example graph structure for this state of affairs is shown in Figure 5.4a. There are two disconnected components in the graph, one for each class. Within each component, the bipartite graph is fully connected. Clearly this graph structure would be ideal for a bootstrapping algorithm, since assigning labels to an edge (example) based on the label of the nodes will lead to a correct assignment. In addition, given the existence of one correct label in the component, all other examples are reachable within two steps. This argument is interesting when we consider the small-world properties of our data, described in Section 5.2.3. Since our data observes small-world properties, and therefore has small average path-length, the propagation of labels will be rapid. However, another graph which also has small path lengths and zero mutual information between noun-phrases and contexts is shown in Figure 5.4b. Here the labels are shown on edges (bold for positive). In this graph, no bootstrapping algorithm can learn a good model of the classes. Here the feature sets are not sufficient. Thus low

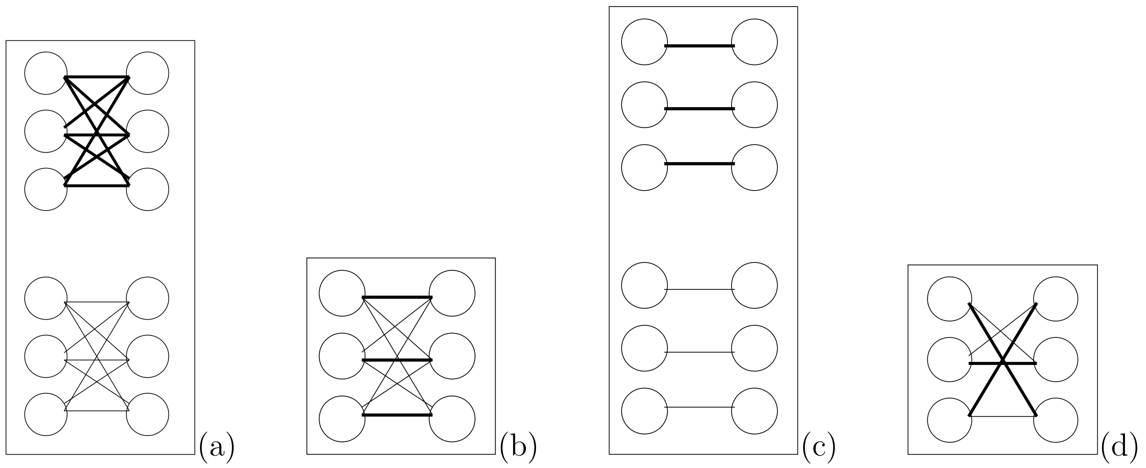


Figure 5.12: Example graphs which attain (a,b) the lower bound, 0, on the mutual information between noun-phrases and contexts given the class, and (c,d) the upper bound min(H(X),H(Y)). Positive examples are shown as bold edges.

mutual information of feature sets given the target class is not sufficient to expect good learning performance from bootstrapping algorithms.

As an example of a graph attaining the upper bound on mutual information given the class, we see the "step-ladder" graph configuration in Figure 5.4c. Here knowing the noun-phrase uniquely determines the context. Bootstrapping on this configuration would require a seed example for every component, *ie* as many labels are there are examples. Again in Figure 5.4d, the mutual information attains its maximum, but the lack of feature-set redundancy means that learning would be difficult.

Let us now consider how we can relate mutual information given the target class back to the small-world and power-law graph properties of our data that we saw in Section 5.2. Short path lengths, as discussed above, may not be predictive of high mutual information, since the labeling of edges with class information may increase the mutual information. We know that our graph has a large clustering coefficient, meaning that node neighborhoods tend to be tightly connected. This would suggest higher mutual information, but again the edge labeling may affect this. The power-law distribution shows that our graph has great degree disparity. The low-degree nodes would contribute to higher mutual information.

We notice in Table 5.17 that the mutual information between noun phrases and contexts given the positive class is lower than the mutual information given the negative class. Based on our observation that smaller samples lead to lower estimates of

task	$I(N;C\|pos)$ (max)	$I(N;C\|neg)$ (max)	$I(N;C\|Class)$ (max)
locations	0.14 (0.15)	8.81 (10.23)	8.95 (10.4)
organizations	0.94 (1.09)	7.86 (8.99)	8.80 (10.16)
people	0.43 (0.50)	8.40 (9.38)	8.84 (10.22)

Table 5.17: Mutual information between noun phrases and contexts, given class labels of test examples. Shown in parentheses are the maximum possible value for mutual information, which is $\min(H(N|Class), H(C|Class)$. The minimum possible value is 0.

mutual information, as shown in Figure 5.11, we might assume that this is completely explained by the fact that the positive class has a small prior and is based on a small sample. We can compare to a uniform random subsample of the data, of the same size as the positive class, and see that the mutual information for the random sample is higher. Thus the assumption of statistical independence given the positive class label is somewhat true for these classes. For the negative class, the mutual information is *higher* for the negative class than for a similar sized random subsample of the data. This suggests that for the negative class the assumption of independence given the class label does *not* hold. Our negative class in reality consists of several subclasses (specifically the negative class for `organizations` for example, contains the classes `people` and `locations`, as well as other classes we have not examined here). Thus we would not expect independence given the class label, for the negative class.

Overall, the mutual information between NPs and contexts given the class label is lowest for the `people` class, then `organizations`, then `locations`. Thus when class-conditional statistical independence is important, we would expect the `people` class to outperform `organizations`, which would outperform `locations`.

We observe this ordering among the classes when we label many examples; 2500 examples actively labeled for both coEM and EM in Chapter 4. This suggests that once we have 2500 examples actively labeled, our models are reliable enough for levels of statistical dependence between noun-phrases and contexts to make a difference in importance. It also suggests that violations of statistical independence are not the most important factor in predicting algorithm importance, since this ordering emerges only when many examples are labeled with active learning.

5.5 Algorithm Desiderata for Small Worlds

On the basis of the arguments and empirical results presented in this chapter, we suggest the following for semi-supervised learning on data exhibiting small-world properties:

- algorithm sensitivity to node degree: a hybrid approach, in which greater confidence is required for labeling high-degree nodes than low-degree nodes
- initial examples selected to have high degree
- initial examples selected to span many components of the graph
- examples selected for active learning chosen for high degree
- node degree used as part of feature-set selection criteria

5.6 Questions about Designing Bootstrapping Algorithms

5.6.1 How Can I Know if a Set of Seeds Will Lead to Successful Bootstrapping?

For a given set of seed-words $S = \{s_1..s_k\}$ and dataset $X = \{< np_1, context_1 > .. < np_n, context_n >\}$ we can measure how many of the seeds are heads of one of the noun-phrases.

No Seeds are Heads of Noun-phrases in the Dataset

If no seeds are heads of noun-phrases in the dataset, this combination of seeds and data cannot lead to learning anything about the target class. We have no training information to use for learning.

No Seeds are Heads of noun-phrases in the Largest Component

Knowing that our data exhibits small-world properties as described in Section 5.2, we know that the bulk of examples are in the largest component. If we have no seeds which are heads of noun-phrases in the largest component, we will only be able to learn properties of a tiny fraction of the data.

Few Seeds Are Heads of Noun-Phrases in the Dataset

We saw that performance improves when more seeds are heads of noun-phrases in the dataset, and in particular, when more are heads of noun-phrases in the largest component. Bootstrapping may not perform well if few seeds are found in the dataset.

Seeds are Ambiguous in the Dataset

If the seeds chosen are ambiguous, this may lead to poor bootstrapping performance. If we have two or more sets of seeds for distinct classes, we can measure their overlap in the contexts they select. High overlap may lead to poor performance. If we have only one seed set, using only seeds with intersecting sets of contexts they cooccur with may lead to improved performance, by eliminating the ambiguous seeds. We saw that when many contexts cooccur with two or more different seeds, we have improved performance.

Seeds Are In Basic-Level Category

The psycho-linguistic notion of *basic-level categories* (Rosch et al., 1976) captures the standard names we apply to objects (such as “dog”, and “chair”) and are the first categories learned by children. *Superordinate* categories (eg. “animal”, “furniture”) are generalizations of these categories, and are less easily visualized. *Subordinate* categories (eg. “dalmatian”) are more specific than is required for general conversation, and are learned later by children.

All the examples we worked with were super-ordinate categories. Our seeds were descriptions of the super-ordinate category, or category members of the basic level or subordinate level categories. We learned both basic level and subordinate caetgory members. Attempts to learn more specific or more general categories may require larger corpora, to cover more instances of the use of these phrases.

5.6.2 How Should I Select Seeds For Bootstrapping?

We saw that the node degree of noun-phrases labeled by heads was most predictive of algorithm performance. Thus in order to select seeds for an arbitrary new learning task, we should identify the heads of noun-phrases, sort by their node degree (the number of unique contexts they co-occur with) then examine this list on order of frequency. We should continue labeling seeds till we have seen about 5 seeds in our target class, and till we have seen at least one seed in the largest component.

5.6.3 How Can I Decide if Two Classes Represented By Seeds Are Confusable?

The ambiguity of seeds can be measured by calculating the intersection of the contexts they co-occur with. If there is a high degree of co-occurrence of these contexts, the seeds are ambiguous.

5.6.4 How Can I Know if I Have Enough Data

Is There Sufficient Data to Represent the Underlying Graph?

By measuring the connectivity structure we can measure whether there is sufficient data sampled from the underlying graph. If there is sufficient data, the bulk of the nodes should form one large connected component. If there is not one large connected component, there may not be sufficient data.

Is There Sufficient Data to Learn My Target Class?

If examining high frequency head-nouns does not lead to identifying candidate seeds, there is not enough data. One way to remedy this is to gather data specifically containing the seed terms. Several systems have been proposed to do this (Agichtein et al., 2003)(Ghani & Jones, 2002).

5.6.5 Should I Correct Examples Labeled By Seeds if I will be Performing Active Learning?

We learned in Chapter 3 that there is not a great deal of utility in correcting the examples labeled by seeds. Any extra labeling time would be better off spent labeling during active learning, as we saw in Chapter 4.

5.6.6 What Properties Should My Active Learning Algorithm Have?

An active learning algorithm for learning semantic classes from text should choose frequently occurring examples, as well as examples likely to be positive. For example, in *uncertainty sampling* we choose examples for which the algorithm is uncertain, *ie* examples near the decision boundary. In order to select positive examples, we should select examples slight further to the positive side of the decision boundary.

5.7 Chapter Conclusions

In this chapter we explored some of our experimental results in more detail. We found that the total node degree of examples labeled by seeds during initialization was an important predictor of algorithm performance. This is explained by the small-world graph structure of the data, which we also established in this chapter. We found that our data strayed from conditional independence. In addition we found that the number of examples labeled *positive* was an important predictor of performance with active learning, particularly for `locations`, the class with the most skewed distribution.

We have laid out a set of properties of data sets which we should examine when applying semi-supervised learning, together with a description of their likely impact on learning performance. We have shown that a real world data set, which has been explored in the the context of semi-supervised learning, exhibits small-world graphs properties. We measured algorithm performance in terms of these graph-theoretic properties, showing that node degree of initial examples is predictive of algorithm performance, and that the distribution of labeled examples over components on the graph is also predictive of performance. We also suggested some ways in which algorithms and labeling might be customized around these properties, opening up a range of approaches to algorithm design and application that is sensitive to the underlying distribution of the data.

Chapter 6

The Held-out Task

We have been making suggestions for methods of bootstrapping semantic classes on the basis of three classes we have worked with, on two different datasets (`7sector` and TREC wtx10g). There is always a risk in this kind of work, of learning a great deal about the specifics of one's own dataset and tasks, and therefore producing a method which does not generalize well to new tasks, datasets or domains. In this chapter we describe a short experiment on applying to new domains the methods we have explored and recommended. This will also be an exercise in applying some of the recommendations and diagnostics for learnability that we have developed.

6.1 Task Selection

The experiments we have conducted have been on classes with largely open-ended vocabularies. These are classes for which obtaining an exhaustive list or dictionary of class members is unrealistic. For example, there may always be new companies or people we haven't heard of, as well as misspellings or novel ways of referring to locations or points of interest. They have also been classes which we can expect to find many instances of in our training and test corpora, which consist of company web pages.

Recall from Chapter 5 Section 5.3 that the number of examples labeled by seeds is highly predictive of performance. If we wish to perform a bootstrapping experiment with this data, we would be well advised to choose a class which is likely to have many examples in our training corpus. If we must learn a class which is not likely to occur often in our training data, we would be well advised to collect new training data. We can detect whether there are instances of the target class in our training corpus, by seeing if we can find seeds which extract many examples.

We expect that the `products` class will be reasonably well-represented, both on the general web pages in the `wtx051-053` training collection, and our `7sector` test set. In addition, the `products` class likely consists of a relatively open-ended vocabulary which may be difficult to write down or obtain, so could benefit well from a bootstrapping approach. Thus we choose to perform experiments on the `products` class.

In general we expect the bootstrapping approach to learning semantic classes to work well for classes which can be disambiguated using the immediate context. It is most useful for classes for which it is difficult to obtain an exhaustive list of class members, but may also have utility in cases where the class members can be exhaustively enumerated or described, but some may be ambiguous.

The class `dates / times` may be well-represented using regular expressions. We can list exhaustively the months and days of the months, as well as the likely years, and write down common ways they are composed together. However, there may be other expression we do not expect, such as “third quarter”. In general our representation here is not ideally suited to learning time and data expressions, since we do not have a general feature for representing digits. Nevertheless, we will also conduct a brief experiment with extraction of the `dates / times` class to see how the general framework is applicable here.

6.2 Seed Selection

We selected the semantic categories `products` and `dates / times`. We will need a small number of noun-phrases we think may be examples of our target class (that is, a small number of words which may be `products`, or `dates / times` as appropriate) to provide the training information. These are called our *seeds.* Recall from Chapter 3 that an assumption in the bootstrapping approach we are using is that the set of seed-words we choose will be present in the data. We found in Section 3.5.1 that when using random subsets of country names as our seed-words, there was a highly variable distribution of those seed-words in our training corpus. We found in Chapter 5 Section

5.3 that the number of examples labeled by seeds is highly predictive of performance. Thus we would like our seed-words to be relatively frequently occurring in our training data. There are two ways of ensuring this: (1) to collect the training documents to correspond to the seed examples, as suggested in previous work (Agichtein et al., 2003), (Ghani & Jones, 2002), or (2) to select the seed words using the training data. For this held-out experiment, we will select the seed words using the training data, though for the `products` class we will also conduct a baseline experiment asking people to supply seeds for the target class.

6.2.1 Choice of Training Data

We decided to use a fixed corpus and select seeds from this training corpus, rather than collect a corpus specifically for use with pre-selected seeds. One good reason for this is that a training corpus collected to contain seeds may have special properties (such as over-representation of seeds, and under-representation of other vocabulary) which are unlike general data we may need to test on, and thus our generalization power may be reduced.

There remains the question of which training corpus to use. Recall from Chapter 3 Section 3.7.6 that we compared the use of a smaller corpus (`7sector`) and a larger corpus (`wtx10g`) as training data. The smaller corpus was drawn from the same distribution as the test data (the test data was subsampled from the original `7sector` crawl of company websites, and the training corpus is a disjoint subsample from the same data). The `wtx10g` corpus is larger, but from a different, more general distribution (web pages, not selected specifically to be company web pages). We found that for the `locations` task, using the larger disjoint corpus was more effective, while for `people` and `organizations` the smaller corpus from the same distribution was more effective. This may be because of the overlap of `organization` and `people` names in training and test collections.

For realism, for the `products` class we will use the `wtx051-053` corpus (described in detail in Chapter 3, Section 3.7.6) for this held-out task. This is disjoint from the test set, and will show us how well our methods perform when applied to a brand-new test collection sampled separately from the training collection, which may be a quite realistic setting when a system is deployed.

For the `dates / times` class we will use the `7sector` training corpus. Since we

	Seeds	n
1-a	20GB iPod, Jetclean II, Tungsten T5, InFocus ScreenPlay 4805 DLP Projector, Sony PSP, Barbie Fairytopia, Crayola Construction Paper Crayons, Kodak Advantix 200 Speed Color Film, Timbuk2 Commute Messenger Bag, Sony MDR-V6 Stereo Headphones	0
1-b	mp3 player, Maytag dishwasher, Palm Pilot, home theater projector, PSP, Barbie, crayons, 35mm film, messenger bag, headphones	100
2-a*	Nestle, disposable razor, Toyota Prius, SUV, Armani Suit, Yemen Mocha Matari, 8" 2x4, cheddar cheese, HP Compaq nc6000, q-tips	5
2-b	Lipton Tea, 00 buckshot, Tomatoes, Loose-leaf paper, Nike shoes, Basil seeds, 2004 Toyota Camry SE, Laptop battery, Gummibears, M&Ms	83
3	Leather sofa, Electric violin, Chocolate cake, Mountain bike, Pair of glasses, K2 Rollerblades, Ipod, Dress shirt, Headphones, Webcam	20

Table 6.1: Five sets of seeds for the `products` class, chosen by introspection by three labelers. The set 2-a was chosen to be difficult. n indicates how many examples in the training corpus matched each seed set.

do not have `digit` features to allow us to generalize outside the vocabulary learning during training, a mismatched train-test corpus may be harmful.

6.2.2 Seed Selection By Introspection

We asked three employees of a web search company to supply lists of possible products. These lists are shown in Table 6.1. For each seed-set, n indicates the number of examples in the wtx-051-053 training corpus which which one of the seeds. Set 1-a was chosen to be very specific product names, and matching no examples in the training corpus at all. Set 1-b consisted of somewhat more general names for the same products as in set 1-a, and matched 100 examples in the training corpus. Set 2-a was chosen to be difficult, to contain ambiguous words. It matched only 5 examples in the training corpus. Set 2-b was chosen to be unambiguous, and matched 83 examples in the training corpus. Finally labeler 3 chose to list objects he had recently purchased or planned to purchase. This set of seeds matched 20 examples in the training corpus.

6.2.3 Select Seeds Using the Training Data

We have chosen to select our seeds using the `wtx051-053` training corpus. From Chapter 5 Section 5.3 we know that the frequency of these seeds will be predictive of algorithm performance, even when we use active learning. Specifically we know that when using active learning, algorithm performance is predicted by:

1. Number of unique seeds head-matching some NP in graph (Spearman correlation of 0.282)
2. Number of unique seeds exact-matching some NP in the graph (Spearman correlation of 0.285)
3. Number of unique seeds head-matching NPs in the largest component (Spearman correlation of 0.282)
4. Number of unique examples labeled (sum node degree) (Spearman correlation of 0.630)
5. Total examples labeled (Spearman correlation of 0.628)
6. Number of components containing at least one example (Spearman correlation of 0.501)
7. Number of components containing at least one seed or positive example (Spearman correlation of 0.529)
8. Number of unique seed-examples or positive example in the largest component (Spearman correlation of 0.624)
9. Number of unique contexts covered by seeds (Spearman correlation of 0.551)
10. Number of unique contexts covered by more than one seed (Spearman correlation of 0.581)

According to this list, the best criterion we could use for seed selection is to optimize 4, the number of unique examples labeled (though this criterion also includes examples labeled during active learning). We also found that item 5, the total number of examples labeled, is not statistically significantly different in predictive power.

Thus if we choose seeds to maximize the number of examples labeled, we should be providing a good starting point to bootstrapping.

Thus we sorted noun-phrases in the `wtx051-053` corpus by frequency, then labeled the members. For the task `products` we examined the 200 most frequent noun-phrases and labeled each as either unambiguously likely to correspond to an example of `products` and therefore a relevant seed, or not relevant. Of these two hundred most frequent noun-phrases, we found three seeds which we thought unambiguously likely to correspond to an example of `products`: "services", "software" and "products". Some numbers relating to the distribution of these seed words in the `wtx051-053` are given in Table 6.2. 20,331 examples total were labeled by these three seeds. This contrasts with the seeds chosen by introspection, which matched between 0 and 100 examples in the training corpus.

Looking at the top three contexts labeled by each of the three high-frequency seeds, we see that there is some overlap ("range of <x>" was covered by both "services" and "products"), as well as some variation – "services" selects "provides <x>", while "software" selects "<x> provides". It makes intuitive sense that a product is something that can provide utility to a customer, as well as something which is provided by a company. However, this suggests that these contexts will also be likely to cooccur with noun-phrases which represent vendors and customers. To distinguish these we may need even more context, for example by adding another disambiguating noun-phrase, as in, "the company provides <x>". Thus our contexts may not be as extensive in expressive power as we might like.

For the `dates / times` class we found the years 1993, 1994, 1995, 1996 and 1997 in the first 200 most frequent noun-phrases of the `7sector` corpus, and so used these as our seeds. These matched 1753 examples in the `7sector training corpus`. Contexts occurring frequently with these seeds include: "increased in <X>", "million in <X>" and "months of <X>".

6.2.4 No Active Initialization

Recall from Chapter 3 Section 3.7.4 that correcting the examples labeled through automatic labeling using the seed-words did not contribute substantially to learning performance. Therefore we will not correct the examples labeled by seeds.

Seed-word	nps	exs/ np-heads	u. np-heads	u. Cont's	ex. Cont's
services	2711	7236	2427	4333	provides <x>, offers $< x >$, range of <x>
software	2679	7100	2159	4581	use of <x>, use <x>, <x> provides
products	2113	6281	2267	3952	information on <x>, range of <x>, line of <x>

Table 6.2: Seeds for `products` task. Seeds were selected by examining the top 200 most frequent noun-phrases, and identifying the terms that seemed most unambiguously like products. Shown are the number of times the seed words occurred as whole noun-phrases ("nps"), the number of times the seed-words occurred as heads of noun-phrases ("np-heads"), the number of unique noun-phrases with the seed as head ("u. np-heads"), the number of unique contexts labeled by the seed ("u. contexts") and the three most common contexts labeled by the seed. 20,331 examples total were labeled by these three seeds.

6.3 Bootstrapping and Active Learning Algorithm

6.3.1 Bootstrapping Algorithm

We used 500 iterations of coEM (described in Chapter 3 Section 3.4.3), which we found to be robust across classes. We found in Chapter 3, Section 3.7.2 that allowing stopwords in the model improves results greatly for the `people` and `organizations` classes, without having a deleterious effect on the `locations` class. "We" is a very good indicator for organizations, and "he" and "she" are very good indicators for people. It is plausible that "it" may be a weak positive predictor for `products`, whereas "he", "she" and "we" will be negative predictors. Thus we allowed stopwords in the model.

6.3.2 Number of Examples Labeled With Active Learning

We found in Chapter 4, Section 4.10.6 that most of the benefit of active learning comes in the first few iterations. While we also saw statistically in Section 5.3.11 of Chapter 5 that the number of examples labeled is positively correlated with performance, we will label just 25 examples with active learning, to make this a real test of minimally supervised training for the the `products` class. For the `dates/ times` class, we labeled 500 examples, as the contexts for this class are relatively ambiguous (dates and times can occur in similar contexts to locations). In earlier experiments described in this thesis, we excluded numbers from being labeled with active learning. For the `dates / times` experiments, we removed this exclusion.

6.3.3 Active Learning Algorithm

In Section 5.3.11 in Chapter 5 we saw that the number of positive examples labeled is correlated with algorithm performance. Our active learning algorithm cannot have access to the true labels of examples when selecting them, though we could modify the algorithm to select examples it thinks are more likely to be positive. We will simplify matters here by using disagreement between the noun-phrases and the context to select examples (*feature-set disagreement*, which was described in Chapter 4 Section 4.6). This selects examples about which there is some uncertainty as to the label (and therefore the example could be positive). This method performed well in experiments on other classes, as we saw in Chapter 4 Section 4.10.8. In order to remove the redundancy in relabeling seeds that we saw in Section 4.10.8, we restricted the set of possible examples to be labeled to be those that did not contain seeds.

6.4 Evaluation

We used the model to extract on the `7sector` test set. We sorted extracted examples by confidence score, and labeled the top 200 examples according to whether or not they contained member of the target class (a `product` or `date/time`, depending on the task). We then calculated precision at 10, 50, 100 and 200. We also examined the dictionary of noun-phrases produced on the training, and similarly scored the first 200, to obtain precision at 10, 50, 100 and 200. For the `products` class, he test set contained many noun-phrases whose heads were the words “services”, “products” or “software”. These noun-phrases could be extracted using just the seed words, so for a fuller evaluation of the effects of bootstrapping and active learning, we performed an analogous evaluation in which we removed examples with “services”, “products” or “software” as the head of the noun-phrase, then labeled the top 200 of the remaining examples.

	1-a	1-b	2-a	2-b	3	freq-noseeds	freq
P@10	0.2	0.2	0	0.1	0.4	**0.4**	1.0
P@50	0.2	0.18	0.02	0.14	0.22	**0.56**	1.0
P@100	0.21	0.19	0.03	0.23	0.31	**0.55**	0.8
P@200	0.185	0.22	0.055	0.245	0.39	**0.535**	0.685

Table 6.3: Results of extracting test examples based on models obtained by bootstrapping the `products` class, using the seeds created by introspection (1-a - 3), and seeds chosen by frequency (freq). Set 1-a matched no seeds, and can be seen as a baseline reflecting the prior distribution of `products` terms in the test corpus. Set 2-a was chosen to be ambiguous, containing brand-names which may also be company names. This set performed much worse than the baseline. The frequency based seeds (freq) with 25 instances labeled with active learning perform best, even when we remove test examples which match a seed (freq-noseeds).

6.5 Results

6.5.1 Products

In Table 6.3 we see results of extracting test examples based on models obtained by bootstrapping the `products` class, using the seeds created by introspection (1-a - 3), and seeds chosen by frequency (freq). Set 1-a matched no seeds, and can be seen as a baseline reflecting the prior distribution of `products` terms in the test corpus. Set 2-a was chosen to be ambiguous, containing brand-names which may also be company names. This set performed much worse than the baseline. The frequency based seeds (freq) with 25 instances labeled with active learning perform best, even when we remove test examples which match a seed (freq-noseeds)

In Table 6.4 we show more detail of the results from the frequency selected `products` seeds. Observing the dictionary construction, we can see that including seeds in our evaluation we see 98% correct noun-phrases in the dictionary (though all but 4 were based on the original seeds). When we remove examples based on seeds, the precision of the dictionary ranges from 60% to 35% over the top 200 noun-phrases. This is within the range achieved by Basilisk (Thelen & Riloff, 2002) which learned the categories `building` (40 - 35% precision), `event` (61 - 57% precision), `human` (84-87% precision), `location` (84-87% precision), `time` (30-15% precision) and `weapon` (42 - 31% precision). Basilisk's results were obtained by bootstrapping multiple classes

	nps	nps (non-seed)	Examples	Examples (non-seed)
P@10	1.0	0.6	1.0	0.4
P@50	1.0	0.56	1.0	0.56
P@100	0.97	0.56	0.8	0.55
P@200	0.975	0.345	0.685	0.535

Table 6.4: Precision at different numbers of extracted test examples or training dictionary noun-phrases examined, for bootstrapping `products` with coEM with 3 initial examples (selected from 200 candidates) and 25 examples labeled with active learning using noun-phrase-context disagreement. Many of the test examples contained the seed-words, so we show also results on examples filtered to remove seeds ("non-seed").

simulataneously, which improved results over learning a single class at a time.

Now let us consider in more detail the examples extracted from the held-out test set, that is, the combined noun-phrase context pair evaluated for correctness. Among the 200 top-ranked extracted examples (extracted using both noun-phrases and contexts), a minority were based on the seed-set ("services" with 25 and "products" with 30, though 0 with "software"). Among these 70% were correct. When we remove the examples aided by the presence of seeds, we still have 40-55% correct extraction on held-out unseen data. These results are promising, as we did not use any extra constraints (such as bootstrapping multiple classes simultaneously and assuming one sense per discourse). Using such constraints should lead to even better results.

6.5.2 Dates / Times

In Table 6.5 we see the results of bootstrapping to learn to extract `dates / times` from company web pages, using the years 1993 through 1997 as seeds. We see that without needing to write out rules defining the form of a date, we are able to identify dates and times in the highest scoring extractions with around 60% precision. This exceeds the accuracy achieved by Basilisk (Thelen & Riloff, 2002).

Errors in extraction resulted from ambiguous contexts, such as "growth in <X>" which occurred with both "1970s" and "revenue"; and "founded in <X>" which occurred with "1937" "1982", "1886", "1987", "1907", "1832", and "london".

The kinds of `date / time` expressions we were able to extract included:

	noactive	noactive-noseeds	active500	active500-noseeds
P@10	1	0.4	0.9	0.6
P@50	0.8	0.5	0.88	0.62
P@100	0.68	0.48	0.74	0.56
P@200	0.54	0.385	0.52	0.34

Table 6.5: Results of extracting test examples based on models obtained by bootstrapping the `dates / times` class, using the seeds created by looking at frequent noun-phrases in the training corpus (seeds were the years 1993 - 1997).

- generic words for time-frames ("month", "year", "century")
- business time-frames (we learned all of "quarter", "first quarter", "second quarter", "third quarter" and "fourth quarter")
- approximate dates ("early 1994", "spring/summer 1996")
- four-digit years (we learned 19 four-digit years other than the initial seeds)
- relative time descriptions ("previous year", "year following", "following year")
- month names concatenated with years ("march 1998", "february 1997")
- month names in isolation (we learned every month name)
- days of the week in isolation (we learned every day of the week)
- very precise times ("jan 6 23:00:00 1996")
- other date formats ("6/30/96", "fy97")

This suggests that this approach is a good way of initializing a date extractor. In order to build a complete date extractor, it would be good to generalize some of these, for example, 4 digit years would be well-represented with a regular expression.

6.6 Chapter Conclusions

We have shown that we can use the methods we demonstrated in this dissertation to extract data from a held-out test set, for a new task not used during algorithm development. For the `products` task, with human-labeled input of (1) 200 noun-phrases examined to obtain three seed-words, and (2) 25 additional examples labeled with active learning, we obtain precision of 40-55% correct extraction on held-out unseen data. We used a disjoint training set from a different distribution than the test-set, and did not use the full set of information which could be used for information extraction, such as features incorporating the presence of digits and capital letters, and learning multiple classes simultaneously to. Adding this kind of information should lead to even stronger results.

For the `dates / times` tasks, our input was 5 4-digit years selected from the 200 most frequent non-phrases in the `7sector` training corpus, and 500 examples labeled with active learning. We learned days-of-the-week and months-of-the-year, as well as combinations of days and dates. These learned patterns could be used as is, for lower precision date and time extraction, or to initialize a hybrid extractor which would have rules added to identify all four-digit years.

These results are likely to generalize to other classes and data, since we were conservative in selection of training data, as well as the amount of data labeled.

Chapter 7

Conclusions and Future Work

We have examined in depth bootstrapping algorithms and active learning for learning to extract entities belonging to a particular semantic class. We now summarize the main conclusions we have reached as a result of this depth of study. In addition, for this depth we have sacrificed some breadth. In this chapter we also describe future work that would make interesting follow-ons to this work.

7.1 Conclusions

Out conclusions tie together results found in Chapter 3 on bootstrapping algorithms and their assumptions, in Chapter 4 on incorporating active learning into these bootstrapping algorithms, and in Chapter 5 on measuring the graph properties of the labeled and unlabeled data, and tying it back to learning performance.

7.1.1 Active Learning has More Impact then Correcting Initial Examples

We used *weak labeling*, in which labels are assigned to examples based on, for example, the presence of a single seed word, but without any manual examination of those labels. While it is natural to think that correcting those labels by manually inspecting them could be used to improve performance, we found that a better use of a labelers time is to label new examples with active learning. From this we can conclude that the choice of examples we label is key, for a fixed amount of labeling.

7.1.2 Highly Connected Noun-phrases are Important In Learning

We found experimentally in Chapter 3 that removing pronouns from the model used in bootstrapping could lead to reduced learning performance. We found statistically in Chapter 5 that we can predict learning performance, using the node degree of examples labeled during seeding and active learning, the number of contexts connected to more than one of those seeds, and the distribution of seeds over components.

We can conclude from these results that it is useful to take into account the graph structure connecting noun-phrases and contexts for bootstrapping information extractors. The pronouns form highly connected nodes in the graph, which are highly influential, even when only weakly predictive. Other nodes are useful depending on their connectivity within the graph too.

7.1.3 Overall Graph Structure is Important in Learning

We found that the distribution of labeled examples over components of the graph affected learning outcome. This suggests that we should take into account the graph structure, and in particular, the distribution of data into components, when selecting examples for labeling.

7.2 Future Work

7.2.1 Using Pre-existing Dictionaries

We have worked almost exclusively with small lists of words as input. It would also make sense to start with large dictionaries if we have them available. We looked at using a longer list of 253 country names in Chapter 3 and saw slight improvements, though the differences were lessened with active learning, as we saw in Chapter 4. Still for even larger pre-existing dictionaries there may be benefits and interactions that we have not considered here.

7.2.2 Applicability to a Range of Semantic Classes

In Chapter 5, we mentioned the psycho-linguistic notion of *basic-level categories* (Rosch et al., 1976) which captures the standard names we apply to objects (such as “dog”, and “chair”) and are the first categories learned by children. *Superordinate* categories (eg. “animal”, “furniture”) are generalizations of these categories, and are less easily visualized. *Subordinate* categories (eg. “dalmatian”) are more specific than is required for general conversation, and are learned later by children.

All the examples we worked with were super-ordinate categories. Our seeds were descriptions of the super-ordinate category, or category members of the basic level or subordinate level categories. We learned both basic level and subordinate caetgory members. Attempts to learn more specific or more general categories may require larger corpora, to cover more instances of the use of these phrases.

It would be useful to conduct experiments to see whether we can characterize the set of semantic classes learnable by this approach, either in terms of Rosch’s categories, or other categories of semantic classes.

7.2.3 Applicability Across Languages

We have looked at learning to identify semantic classes in English text. We may find that transferring this approach to other languages requires different treatment of words and phrases. For example, for languages with more complex morphology than English, the interaction between tokenization and extraction should be explored.

7.2.4 Alternative Data Representations and Sources

Another area for future work is examining the implications of the simplification we performed by having only two features. We noted in Chapter 3 Section 3.5.2 that when 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.

We should examine the effect that adding more features would have on our results and analyses. We also ignored capital letters and other syntactic clues which can be useful in identifying entities. It would be useful to generalize the representation from noun-phrases and their local lexico-syntactic contexts to sequences of words or ngrams. This would allow this approach to be applied to, for example, web search queries, or other text types with less grammatical structure.

7.2.5 Automatically Acquiring Relevant Training Data

We saw in Sections 3.7.5 and 3.7.6 that both number of seeds in the training corpus and corpus size affect results. We may wonder if there is a way of increasing the number of seeds in the training corpus without greatly increasing the corpus size, for greater computational efficiency. One way may be to label more examples. We addressed this question in Chapter 4 by examining the most efficient ways to choose examples for labeling, using active learning. A different approach, motivated by related work in acquiring documents in a target language (Ghani et al., 2003) and relevant semantic relationships (Agichtein et al., 2003), would involve automatically acquiring training data that is likely to contain many instances of the seeds and the target class. This may be a promising direction for future work.

7.2.6 Applicability Across Domains and Data Types

We explored learning from few examples within the domain of entity extraction in text documents. The methods we explored may be applicable to text classification at the document level, or extraction and classification with completely different data, such as image data or time series data.

7.2.7 Predicting and Improving Algorithm Performance Based on Data Structure

Finally, we showed how for one kind of machine learning task, we can find correlations between the structure of the labeled and unlabeled data and the learning performance. This kind of analysis may be usefully applied to other learning tasks.

Bibliography

Agichtein, E. (2004). *Extracting relations from large text collections.* Doctoral dissertation, Columbia University.

Agichtein, E., Ipeirotis, P., & Gravano, L. (2003). Modeling query-based access to text databases. *Proceedings of the Sixth Annual Workshop on the Web and Databases.*

Albert, R., & Barabási, A.-L. (2002). Statistical mechanics of complex networks. *Reviews of Modern Physics*, *74*. http://xxx.lanl.gov/abs/cond-mat/0106096.

Appelt, D. E., & Israel, D. J. (1999). Introduction to information extraction technology. A Tutorial Prepared for IJCAI-99.

Bailey, P., Craswell, N., & Hawking, D. (2003). Engineering a multi-purpose test collection for Web retrieval experiments. *Information Processing and Management*, *39*.

Baker, L. D., & McCallum, A. K. (1998). Distributional clustering of words for text classification. *Proceedings of the Twenty-first Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-98).*

Balcan, N., Bluem, A., & Yang, K. (2004). Co-training and expansion: Towards bridging theory and practice. *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems (NIPS-2004).*

Bean, D., & Riloff, E. (2004). Unsupervised learning of contextual role knowledge for coreference resolution. *Proceedings of the Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004).*

Bensaid, A. M., Hall, L. O., Bezdek, J. C., & Clarke, L. P. (1996). Partially supervised clustering for image segmentation. *Pattern Recognition*, *29*, 859–871.

Blum, A., Lafferty, J., Rwebangira, M. R., & Reddy, R. (2004). Semi-supervised learning using randomized mincuts. *Proceedings of The Twenty-First International Conference on Machine Learning (ICML-2004).*

Blum, A., & Mitchell, T. (1998). Combining labeled and unlabeled data with co-training. *Proceedings of the 11th Annual Conference on Computational Learning Theory (COLT-98).*

Bockhorst, J., & Craven, M. (2002). Exploiting relations among concepts to acquire weakly labeled training data. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002).*

Brown, P., Pietra, V. D., deSouza, P., Lai, J., & Mercer, R. (1992). Class-based n-gram models of natural language. *Comutational Linguistics*, *18*, 467–479.

Califf, M. E., & Mooney, R. J. (1997). Relational learning of pattern-match rules for information extraction. *Proceedings of the ACL Workshop on Natural Language Learning* (pp. 9–15).

Cardie, C. (1997). Empirical methods in information extraction. *AI magazine*, *18*, 65–79.

Cohen, W. W., Hurst, M., & Jensen, L. S. (2002). A flexible learning system for wrapping tables and lists in HTML documents. *Proceedings of the Eleventh International World Wide Web Conference (www-2002).*

Collins, M., & Singer, Y. (1999). Unsupervised models for named entity classification. *proceedings of EMNLP/VLC-99*.

Cover, T. M., & Thomas, J. A. (1991). *Elements of information theory.* New York: John Wiley and Sons.

Cozman, F., & Cohen, I. (2002). Unlabeled data can degrade classification performance of generative classifiers. *Fifteenth International Florida Artificial Intelligence Society Conference* (pp. 327–331).

Dagan, I., Justeson, J., Lappin, S., Leass, H., & Ribak, A. (1995). Syntax and lexical statistics in anaphora resolution. *Applied Artificial Intelligence*, *9*, 633–644.

Dagan, I., Lee, L., & Pereira, F. C. N. (1998). Similarity-based models of word cooccurrence probabilities. http://xxx.lanl.gov/abs/cs/9809110.

de Moura, A. P. S., Lai, Y.-C., & Motter, A. E. (2003). Signatures of small-world and scale-free properties in large computer programs. *Physics Review E*, *68*.

Domingos, P., & Pazzani, M. (1997). On the optimality of the simple Bayesian classifier under zero-one loss. *Machine Learning*, *29*, 103–130.

Ferrer i Cancho, R., & Solé, R. V. (2001). The small world of human language. *Proceedings of the Royal Society of London, B*, 2261–2265.

Freitag, D. (1998). Multistrategy learning for information extraction. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-1998).*

Freund, Y., Seung, H. S., Shamir, E., & Tishby, N. (1997). Selective sampling using the query by committee algorithm. *Machine Learning*, *28*, 133–168.

Ghani, R., & Jones, R. (2002). *Automatic training data collection for semi-supervised learning of information extraction systems* (Technical Report). Accenture Technology Labs. http://labs.accenture.com/techreports/ghani_jones_TR2002.pdf.

Ghani, R., Jones, R., & Mladenic, D. (2003). Building minority language corpora by learning to generate web search queries. *Knowledge and Information Systems.*

Glickman, O., & Jones, R. (1999). Examining machine learning for adaptable end-to-end information extraction systems. *AAAI 1999 Workshop on Machine Learning for Information Extraction.*

Goldstein, M. L., Morris, S. A., & Yen, G. G. (2004). Problems with fitting to the power-law distribution. http://arxiv.org/abs/cond-mat/0402322.

Gooi, C. H., & Allan, J. (2004). Cross-document coreference on a large scale corpus. *Proceedings of the Human Language Technology conference / North American chapter of the Association for Computational Linguistics annual meeting (HLT/NAACL-2004).*

Grishman., R. (1995.). The NYU system for MUC-6 or where's the syntax? *Proceedings of the Sixth Message Understanding Conference (MUC-6),*. Columbia, MD,. Morgan Kaufmann. 10.

Hofmann, T. (1999). Probabilistic latent semantic indexing. *Proceedings of the Twenty-Second Annual SIGIR Conference on Research and Development in Information Retrieval.*

Huffman, S. (1996). Learning information extraction patterns from examples. In S. Wermter, E. Riloff and G. Scheler (Eds.), *Connectionist, Statistical, and Symbolic Approaches to Learning for Natural Language Processing*, 246–260. Springer-Verlag, Berlin.

Hurst, M. (2000). *The interpretation of tables in texts.* Doctoral dissertation, University of Edinburgh.

Iamnitchi, A., Ripeanu, M., & Foster, I. (2004). Small-world file-sharing communities. *The 23rd Conference of the IEEE Communications Society (InfoCom 2004).* Hong Kong.

Jensen, D., & Neville, J. (2002). Linkage and autocorrelation cause feature selection bias in relational learning. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML2002)* (pp. 259–266).

Joachims, T. (2001). *Learning to classify text using support vector machines.* Doctoral dissertation, University of Dortmund.

Joachims, T. (2003). Transductive learning via spectral graph partitioning. *Proceedings of the International Conference on Machine Learning (ICML-2003).*

Jones, R., Nigam, K., McCallum, A., & Riloff, E. (1999). Bootstrapping for text learning tasks. *IJCAI-99 Workshop on Text Mining: Foundations, Techniques and Applications.*

Kamvar, S., Klein, D., & Manning, C. (2002). Interpreting and extending classical agglomerative clustering algorithms using a model-based approach. *Proceedings of the 19th International Conference on Machine Learning (ICML-2002).*

Karger, D. R. (1994). Random sampling in cut, flow, and network design problems. *Proceedings of the Twenty-Sixth Annual ACM Symposium on the Theory of Computing* (pp. 648–657).

Knoblock, C., Lerman, K., Minton, S., & Muslea, I. (2000). Accurately and reliably extracting data from the web:a machine learning approach. *IEEE Data Engineering Bulletin, 23*, 33–41.

Kou, Z., Cohen, W. W., & Murphy, R. F. (2005). High-recall protein entity recognition using a dictionary. *Proceedings of the 13th Annual International conference on Intelligent Systems for Molecular Biology.*

Lee, L. (1997). *Similarity-based approaches to natural language processing.* Doctoral dissertation, Harvard University.

Lewis, D. D. (1998). Naive (Bayes) at forty: The independence assumption in information retrieval. *Proceedings of the 10th European Conference on Machine Learning (ECML-98)* (pp. 4–15). Chemnitz, DE: Springer Verlag, Heidelberg, DE.

Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text classifiers. *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR-94)* (pp. 3–12).

Liere, R., & Tadepalli, P. (1997). Active learning with committees for text categorization. *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)* (pp. 591–596).

Liu, B., Lee, W. S., Yu, P. S., & Li, X. (2002). Partially supervised classification of text documents. *Proceedings of the Nineteenth International Conference on Machine Learning (ICML-2002).*

Manning, C., & Schutze, H. (1999). *Foundations of statistical natural language processing.* MIT Press.

McCallum, A., & Nigam, K. (1998a). A comparison of event models for naive Bayes text classification. *AAAI-98 Workshop on Learning for Text Categorization.* Tech. rep. WS-98-05, AAAI Press.

McCallum, A., Rosenfeld, R., Mitchell, T., & Ng, A. (1998). Improving text clasification by shrinkage in a hierarchy of classes. *Proceedings of the Fifteenth International Conference on Machine Learning (ICML-1998)* (pp. 359–367).

McCallum, A. K., & Nigam, K. (1998b). Employing EM in pool-based active learning for text classification. *Machine Learning: Proceedings of the Fifteenth International Conference (ICML 1998)* (pp. 350–358).

Miller, G. A., Beckwith, R., Fellbaum, C., Gross, D., & Miller, K. (1990). *Five papers on wordnet* (Technical Report CSL 43). Cognitive Science Laboratory, Princeton University.

Mitchell, T. M. (1997). *Machine learning.* New York: McGraw-Hill.

MUC-6 Proceedings (1996). *Proceedings of the sixth message understanding conference (muc-6).* San Francisco, CA: Morgan Kaufmann.

Muslea, I. (1999). Extraction patterns for information extraction tasks: A survey. *AAAI 1999 Workshop on Machine Learning for Information Extraction.*

Muslea, I., Minton, S., & Knoblock, C. A. (2000). Selective sampling with redundant views. *Proceedings of the Seventeenth National Conference on Artificial Intelligence and Twelfth Conference on Innovative Applications of Artificial Intelligence (AAAI/IAAI-2000).*

Nelson, D. L., McEvoy, C., & Scheiber, T. A. (1999). The University of South Florida word association norms. http://www.udf.edu/FreeAssociation/.

Newman, M. E. J., & Park, J. (2003). Why social networks are different from other types of networks. *Physics Review E.*

Newman, M. E. J., Watts, D. J., & Strogatz, S. H. (2002). Random graph models of social networks. *Proceedings of the National Academy of Sciences of the USA*, *99*, 2566–2572.

Nigam, K., & Ghani, R. (2000). Analyzing the effectiveness and applicability of co-training. *Ninth International Conference on Information and Knowledge Management (CIKM-2000)* (pp. 86–93).

Nigam, K., McCallum, A., Thrun, S., & Mitchell, T. (1998). Learning to classify text from labeled and unlabeled documents. *AAAI-98*.

Pastor-Sartoras, R., & Vespignani, A. (2001). Epidemic spreading in scale-free networks. *Physical Review Letters*, *86*, 3200–3203.

Press, W. H., Teukolsky, S. A., Vetterling, W. T., & Flannery, B. P. (1992). *Numerical recipes in C*. Cambridge University Press. second edition edition.

Proceedings, M.-. (1991). *Proceedings of the third message understanding conference (muc-3)*. San Mateo, CA: Morgan Kaufmann.

Radford, A. (1988). *Transformational grammar: A first course*. Cambridge University Press.

Raghavan, H., Madani, O., & Jones, R. (2005). Interactive feature selection. *Proceedings of the Nineteenth International Joint Conference on Artificial Intelligence (IJCAI-2005)*.

Riloff, E. (1993). Automatically constructing a dictionary for information extraction tasks. *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)* (pp. 811–816). AAAI Press/The MIT Press.

Riloff, E. (1996a). Automatically Generating Extraction Patterns from Untagged Text. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)* (pp. 1044–1049). The AAAI Press/MIT Press.

Riloff, E. (1996b). An empirical study of automated dictionary construction for information extraction in three domains. *Artificial Intelligence*, *85*, 101–134.

Riloff, E., & Jones, R. (1999). Learning Dictionaries for Information Extraction Using Multi-level Boot-strapping. *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)* (pp. 1044–1049). The AAAI Press/MIT Press.

Riloff, E., & Phillips, W. (2004). *An introduction to the sundance and autoslog systems* (Technical Report UUCS-04-015). University of Utah School of Computing.

Rooth, M., Riezler, S., Prescher, D., Carroll, G., & Beil, F. (1999). Inducing a semantically annotated lexicon via EM-based clustering. *Proceedings of the 37th Annual Meeting of the ACL*.

Rosch, E., Mervis, C. B., Gray, W. D., Johnson, D. M., & Boyes-Bream, P. (1976). Basic objects in natural categories. *Cognitive Psychology*, *8*, 382–439.

Sarawagi, S., & Cohen, W. W. (2004). Semi-Markov conditional random fields for information extraction. *Proceedings of the Eighteenth Annual Conference on Neural Information Processing Systems (NIPS-2004)*.

Seymore, K., McCallum, A., & Rosenfeld, R. (1999). Learning hidden Markov model structure for information extraction. *AAAI-99 Workshop on Machine Learning for Information Extraction*.

Sigman, M., & Cecchi, G. A. (2002). The global organization of the WordNet lexicon. *Proceedings of the National Academy of Sciences of the USA*, *99*, 1742–1747. http://www.pnas.org/cgi/reprint/99/3/1742.pdf.

Sleator, D. D., & Temperley, D. (1993). Parsing English with a link grammar. *Third International Workshop on Parsing Technologies*.

Soderland, S. (1999). Learning information extraction rules for semi-structured and free text. *Machine Learning*, *34*, 233–272.

Steyvers, M., & Tenenbaum, J. B. (2005). The large-scale structure of semantic networks: Statistical analyses and a model of semantic growth. *Cognitive Science*, *29*. http://arxiv.org/abs/cond-mat/0110012.

Strogatz, S. H. (2001). Exploring complex networks. *Nature*, *410*, 268 – 276.

Thelen, M., & Riloff, E. (2002). A bootstrapping method for learning semantic lexicons using extraction pattern contexts. *2002 Conference on Emprirical Methods in Natural Language Processing (EMNLP 2003)*.

Thompson, C. A., Califf, M. E., & Mooney, R. J. (1999). Active learning for natural language parsing and information extraction. *Proceedings of the 16th International Conference on Machine Learning (ICML-1999)*.

Van Rijsbergen, C. J. (1979). *Information retrieval, 2nd edition*. Dept. of Computer Science, University of Glasgow.

Watts, D. J., & Strogatz, S. H. (1998). Collective dynamics of 'small-world' networks. *Nature*, *393*, 440–442.