Weakly-Supervised Relation Classification for Information Extraction

Zhu Zhang
Department of Electrical Engineering and Computer Science
and School of Information
University of Michigan
Ann Arbor, MI 48109
zhuzhang@umich.edu

ABSTRACT

This paper approaches the relation classification problem in information extraction framework with bootstrapping on top of Support Vector Machines. A new bootstrapping algorithm is proposed and empirically evaluated on the ACE corpus. We show that the supervised SVM classifier using various lexical and syntactic features can achieve promising classification accuracy. More importantly, the proposed *BootProject* algorithm based on random feature projection can significantly reduce the need for labeled training data with only limited sacrifice of performance.

Categories and Subject Descriptors

H.3 [**INFORMATION STORAGE AND RETRIEVAL**]: Miscellaneous; I.2.7 [**ARTIFICIAL INTELLIGENCE**]: Natural Language Processing—*Language parsing and understanding*; I.2.6 [**ARTIFICIAL INTELLIGENCE**]: Learning—*Concept learning, Induction*

General Terms

Algorithms, Experimentation

Keywords

information extraction, relation, classification, SVM, bootstrapping, co-training

1. INTRODUCTION

With the dramatic increase in the amount of textual information available in digital archives and the World Wide Web, there has been growing interest in techniques for automatically extracting information from text. Specifically, Information Extraction (IE) systems identify relevant information (usually of pre-defined types) from text documents in a certain domain, and put them in a structured format. Once extracted, the information can be used for purposes such as database population and text indexing. While significant progress has been made in IE research, stimulated in particular by the DARPA Message Understanding Conferences (MUC), it is generally agreed that main barriers exist to wider use of IE technologies due to the difficulties in adapting systems to new applications and domains. It is challenging as well to keep track of dynamic information sources (e.g., web pages).

To address these challenges, there has been a recent trend shift in the research community from knowledge-based approaches to machine learning techniques. While supervised learning is usually preferred and applicable, it is not always easy to acquire large amount of labeled training data. Therefore researchers have been investigating the so-called weakly-supervised learning algorithms, which aim at minimizing the need for labeled data while still achieving comparable or even better performance.

According to the scope of the LDC ACE program [1], current research in the information extraction has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). This paper focuses on the second subproblem, RDC. For example, we want to determine whether a person is at a location, based on the evidence given in the context. In particular, the goal of the study is to automatically classify relations between entities using machine learning techniques, especially weakly-supervised learning algorithms. We present a generalized co-training procedure on top of Support Vector Machine (SVM) classifiers and empirically evaluate it on the ACE corpus.

2. RELATED WORK

2.1 Relation Extraction and Classification

Within the realm of information extraction, there are two representative systems that use weakly-supervised learning for extracting relations.

Snowball [3] is a bootstrapping-based system that requires only a handful of training examples of tuples of interest. These examples are used to generate extraction patterns, which in turn result in new tuples being extracted from the document collection. At each iteration of the extraction

[1]http://www.ldc.upenn.edu/Projects/ACE/

process, Snowball evaluates the quality of these patterns and tuples without human intervention, and keeps only the most reliable ones for the next iteration. A scalable evaluation methodology is also developed for the task.

DIPRE (Dual Iterative Pattern Relation Expansion) [7] is another technique that exploits the duality between sets of patterns and relations to grow the target relation starting from a small sample. The technique was used to extract `(author, title)` pairs from the World Wide Web.

2.2 Support Vector Machines

Support vector machines (SVM) are supervised learning techniques motivated by statistical learning theory [15]. An SVM is a maximum-margin hyperplane that lies in a mapped feature space and splits the data points as cleanly as possible, while maximizing the distance to the nearest cleanly split points.

SVM has been widely applied and has met significant success in many computer science areas that involve classifying patterns from large amount of data. For example, [13] explores the use of SVMs for learning text classifiers from examples. Joachim analyzes the particular properties of learning with text data and identifies why SVMs are appropriate for this task. SVMs are also applied to other problems such as chunking [14].

Relevant to our research, [17] presents an application of kernel methods to extracting relations from unstructured natural language sources. The authors introduce kernels defined over shallow parse representations of text, and design efficient algorithms for computing the kernels. The devised kernels are used in conjunction with SVM and Voted Perceptron learning algorithms for the task of extracting `person-affiliation` and `organization-location` relations from text. The proposed methods are compared with feature-based learning algorithms, with promising results.

2.3 Bootstrapping

Supervised learning is generally preferable to unsupervised learning, due to its mathematical beauty and ease of evaluation. However, annotated data is usually expensive to obtain. Hence there has been growing interest in a family of learning techniques that aim at inducing classifiers by leveraging a small amount of labeled data and a large amount of unlabeled data. In particular, the so-called "bootstrapping" algorithm chooses the unlabeled instances with the highest probability of being correctly labeled and uses them to augment labeled training data.

One of the earliest papers on bootstrapping for natural language process problems, [16] presents an unsupervised learning algorithm for word sense disambiguation that, when trained on unannotated English text, rivals the performance of supervised techniques that require time-consuming hand annotations. The algorithm is based on two powerful constraints — that words tend to have one sense per discourse and one sense per collocation, exploited in an iterative bootstrapping procedure.

Another influential paper [5] also considers the problem of using a large unlabeled sample to boost performance of a learning algorithm when only a small set of labeled examples is available. In particular, the problem setting is motivated by the task of learning to classify web pages, in which the description of each example can be partitioned into two distinct views. For example, the description of a web page can be partitioned into the words occurring on that page, and the words occurring in the hyperlinks that point to that page. It is assumed that either view of the example would be sufficient for learning if there is enough labeled data, but the goal is to use both views together to allow inexpensive unlabeled data to augment a much smaller set of labeled examples. Specifically, the presence of two distinct views of each example suggests strategies in which two learning algorithms are trained separately on each view, and then each algorithm's predictions on new unlabeled examples are used to enlarge the training set of the other. This is the so-called "co-training" algorithm. A PAC-style framework for the general problem of learning from both labeled and unlabeled data is provided in the paper. Empirical results on real web-page data indicate that the use of unlabeled examples by the co-training algorithm can lead to significant improvement of hypotheses in practice.

Some more recent theoretical results are presented in [1], which refines the analysis of co-training, defines and evaluates a new co-training algorithm that has theoretical justification. More mathematic understanding of the Yarowsky algorithm [16] is developed in [2]. The paper analyzes it as optimizing an objective function. More specifically, a number of variants of the algorithm are shown to optimize either likelihood or a closely related objective function K.

Collins and Singer [11] discusses the use of unlabeled data for the problem of named entity classification, another subproblem of information extraction. Specifically, the requirements for supervision are reduced to just 7 simple "seed" rules. The approach gains leverage from natural redundancy in the data: for many named-entity instances both the spelling of the name and the context in which it appears are sufficient to determine its type. Two algorithms are presented. The first method uses a similar algorithm to that of [16], with modifications motivated by [5]. The second algorithm, *Co-Boosting*, extends ideas from boosting algorithms, designed for supervised learning tasks, to the framework suggested by [5].

In a perspective different from all the ones above, [12] presents a new "co-training" strategy for using unlabeled data to improve the performance of standard supervised learning algorithms. Unlike [5], it does not employ a split of the feature space, instead it leverages two different model classes. The only requirement the new co-training strategy places on each supervised learning algorithm is that its hypothesis partitions the example space into a set of equivalence classes (e.g. for a decision tree each leaf defines an equivalence class).

3. PROBLEM DEFINITION

The research problem of this paper is identification and characterization of relations between entities. To be more precise,

- We only deal with intra-sentence explicit relations in this study. In other words, the (two) EDT mentions of the entity arguments of a relation must occur within a common syntactic construction, in this case a sentence.

- We don't actually "detect" relations. Rather, the goal is to classify the type of relation between two entities given that they are known to be related.
- It is also assumed that EDT already takes place beforehand, hence all entity-related information is available at the time of relation classification.

Therefore, the problem is cast into a standard classification framework, which can be formalized as follows:

$$(c_{pr}, e_1, c_m, e_2, c_{pt}) \rightarrow r$$

where e_1 and e_2 represent the entities, and c_{pr}, c_m, and c_{pt} represent the pre-, mid-, and post-context respectively. A relation label r is assigned to the five-tuple.

We use the five high-level relations defined in ACE RDC Annotation Guidelines V3.6 as the target taxonomy of the classification task, which are:

ROLE affiliation between people and organizations, facilities, and GPEs (Geo-Political Entities). This includes employment, office holder, ownership, founder, member, and nationality relationships, etc.

PART part-whole relationships between organizations, facilities and GPEs.

AT location of a Person, Organization, GPE, or Facility entity. For example, a person is at a Location, GPE or Facility if the context indicates that the person was, is or will be there. An Organization is in a Location/GPE if it has a branch there.

NEAR indicates that an entity is explicitly near a location, but not actually in that location or part of that location.

SOC personal or professional relationships between people, such as relative, associate, etc.

4. DATA

We use the ACE corpus for our task. Specifically, ACE-2 version 1.0 is used, which contains 519 files from sources including broadcast, newswire, and newspaper. A total of 5,260 relations are tagged (a very small number of additional relations are dropped out due to parsing failure).

A break-down of the data by different section is given in Table 1. No "test" data is contained in this release of ACE corpus. (When we say "test" data in the remaining part of the paper, it refers to the "devtest" portion.)

A break-down of the data by different relation type is given in Table 2.

The data treatment process involves the following steps:

1. Parse the ACE data in XML format; extract and index entities and relations.

Section	Training	Devtest
bnews	1511	361
npaper	1347	291
nwire	1380	371
Total	4238	1022

Table 1: Number of relations: break-down by section

Section	Training	Devtest
ROLE	1964	472
PART	549	123
AT	1249	328
NEAR	78	31
SOC	398	68
Total	4238	1022

Table 2: Number of relations: break-down by relation type

2. Segment the text into sentences using the sentence segmenter provided by the DUC competition [2].
3. Parse the sentences using Charniak parser [10].
4. Convert the parse trees into chunklink format using chunklink.pl [8].
5. Extract and compute features from the chunklink format.

5. WEAKLY-SUPERVISED RELATION CLASSIFICATION

A weakly-supervised learning procedure usually contains two components: a underlying supervised learner and a bootstrapping algorithm on top of it. We discuss both of them in this section, respectively.

5.1 Supervised Learning: Algorithm and Features

The underlying supervised learner is a support vector classifier, which classifies the unlabeled data based on the model trained on the labeled data currently available.

The reason why we choose SVM for this purpose is that it represents state of the art in machine learning research, and there are good implementations of the algorithm available. In particular, we use LIBSVM [9], an integrated software for support vector classification, regression, and distribution estimation (one-class SVM). It supports multi-class classification, and provides probability estimates as well.

In terms of the specific SVM model, the RBF (Radial Basis Function) kernel is used. According to [9], it is a reasonable choice because it can handle non-linear relation between class labels and attributes, and it subsumes linear kernel as a special case. Further more, it has fewer hyperparameters than polynomial kernel.

[2]http://duc.nist.gov/past_duc/duc2003/software/

We extract and compute the following lexical and syntactic features for the classification task:

Lexical features Surface tokens of the two entities and three context windows.

Shallow-syntactic features Part-Of-Speech tags corresponding to all tokens in the two entities and three context windows.

Deep-syntactic features These are a set of features extracted from the chunklink representation:

- *Chunk tags* of the two entities and three context windows. This information is not explicitly present in the treebank format. For example, the "O" tag means that the current word is outside of any chunk; the "I-XP" tag means that this word is inside an XP chunk; the "B-XP" by default means that the word is at the beginning of an XP chunk.
- *Grammatical function tags* of the two entities and three context windows. The last word in each chunk is its head, and the function of the head is the function of the whole chunk. For example, "NP-SBJ" means an NP chunk as the subject of the sentence. The other words in a chunk that are not the head have "NOFUNC" as their function.
- *IOB-chains* of the heads of the two entities, each of which is a lexicalized path, specifically a concatenation of the syntactic categories of all the constituents on the path from the root node to this leaf node of the tree.

Other features Miscellaneous information including:

- An *ordering flag* that indicates the relative position of the two entity arguments of a relation.
- *Types* of the two entities, such as "PERSON" or "GPE".

The context windows are defined as the following:

- Mid-context: everything between the two entities.
- Pre- and post- context: up to two words before or after the corresponding entity.

5.2 Bootstrapping

Before we propose the generalized co-training algorithm based on random feature projection, for comparison purposes, we present two baseline bootstrapping procedures.

5.2.1 *Self Bootstrapping (*SelfBoot*): a Naive Baseline*

We first define a self bootstrapping strategy (Algorithm 1), a naive baseline algorithm which keeps augmenting the labeled data set with model straightforwardly trained from previously available labeled data.

Algorithm 1 requires a confidence measure of the prediction. Suppose P is the estimated probability distribution

```
Algorithm 1 Self bootstrapping
Require: labeled seed set $L$
Require: unlabeled data set $U$
Require: batch size $S$
  repeat
    Train a single classifier on $L$
    Run the classifier on $U$
    Find (at most) $S$ instances in $U$ that the classifier has
    the highest prediction confidence
    Add them into $L$
  until No data points available
```

the classifier assigns to the class labels on a data point, then the confidence score of the example is defined by the entropy of the label probability distribution (the lower the entropy, the higher the confidence):

$$H = -\sum_{i}^{C} p_i \log p_i \tag{1}$$

where C is the total number of classes, and p_i is the probability of current example having class label r_i.

5.2.2 *Bagging-based Bootstrapping (*BootBagging*): a More Competitive Baseline*

For comparison purposes, it may make sense to use a standard co-training algorithm as baseline. However, it is non-trivial to build the two "views" that satisfy the independence assumption in this case, because no specific domain knowledge is available to split the features. Inappropriate views may weaken the argument. Therefore we decide to use another popular bootstrapping algorithm based on the idea of bagging [6], which is in spirit similar to those presented in [4] and [18], and has been shown to be useful.

In this setting, first, bootstrap data sets are generated from the small set of labeled examples using bagging, and a committee of classifiers are trained on them. Next, each committee member attempts to label additional data points. If the agreement among committee members reaches certain threshold, the examples get the dominant label. The whole process then iterates (Algorithm 2).

```
Algorithm 2 Bootstrapping using bagging
Require: labeled seed set $L$
Require: unlabeled data set $U$
Require: batch size $S$
Require: number of bags $B$
  repeat
    Generate $B$ bootstrap datasets from $L$ using bagging
    Train a committee of $B$ classifiers on the bootstrap data
    sets respectively
    Run the committee on $U$
    Find (at most) $S$ instances in $U$ with the highest agree-
    ment among committee members and assign the most
    dominant label
    Add them into $L$
  until No data points available or no reasonable agreement
  can be reached
```

In this case, we still use the notion of entropy to measure the agreement among the committee members (the lower the entropy, the higher the agreement):

$$H = -\sum_{i}^{C} \frac{\|r_i\|}{B} \log \frac{\|r_i\|}{B} \quad (2)$$

where C is again the total number of classes, B is the total number of bags, i.e., the size of the committee, and $\|r_i\|$ is the total number of committee members that assign label r_i to the current example.

5.2.3 *Bootstrapping Using Random Feature Projection (*BootProject*)*

The original co-training algorithm [5] assumes two "views" that

- are both sufficient for classification, and
- are conditionally independent given the label

While co-training is a very influential and useful algorithm, it is not always easy to build the two "views" that satisfy its two assumptions. When domain knowledge is not available, and when the feature space is large, it is often computationally infeasible to construct the two views automatically.

On the other hand, Abney [1] argues that the conditional independence assumption is too strong to be useful, and that weaker independence assumption suffices.

Inspired by the above considerations, we propose the following bootstrapping procedure based on random feature projection (Algorithm 3). The basic idea is to generalize the co-training algorithm and relax its two assumptions, in fact to discard both assumptions by only exploiting the potential redundancy in the feature space. Instead of explicitly "splitting" the feature space, we generate multiple overlapping "views" by random projection from the original feature space. Specifically, in each projection, the features are randomly selected with probability p, and therefore the eventual projected feature space has $p * F$ features on average, where F is the size of the original feature space. Classifiers trained in the projected spaces are then asked to vote on the unlabeled data points.

In this case, we use the same agreement measure as in formula (2). The only difference is that the committee members now work in different feature subspaces.

Algorithm 3 Bootstrap using random feature projection

```
Require: labeled seed set $L$
Require: unlabeled data set $U$
Require: batch size $S$
Require: number of projections $P$
Require: feature sampling probability $p$
  repeat
    for $i = 1$ to $P$ do
      Generate projected feature space $F_i$, by randomly selecting features with probability $p$
      Project both $L$ and $U$ onto $F_i$, thus generate $L_i$ and $U_i$
      Train classifier $C_i$ on $L_i$
      Run $C_i$ on $U_i$
    end for
    Find (at most) $S$ instances in $U$ with the highest agreement among the $P$ classifiers and assign the most dominant label
    Add them into $L$
  until No data points available or no reasonable agreement can be reached
```

6. EXPERIMENTS AND RESULTS

The purpose of the experiments is to explore the effectiveness of weakly-supervised learning compared to strictly supervised learning, and in particular, the performance of the *BootProject* algorithm compared to the two baseline algorithms.

6.1 Strictly Supervised Learning

The baseline classification strategy for the relation classification problem is to assign the most frequent label ("ROLE") to all examples, which gives an accuracy of 44.6% on the test data.

The "ceiling" classification performance is achieved by training the SVM on all available training data, which gives an accuracy of 79.6%. The performance is optimized by performing grid search in the SVM parameter space [9]. The class-specific performance measures for this classifier are given in Table 3. (It is not surprising that the performance reported on relation type "NEAR" is low, because it occurs rarely in both the training and test data.)

Relation type	Precision	Recall	F-measure
ROLE	85.7%	82.4%	84.0%
PART	61.8%	72.4%	66.7%
AT	82.7%	82.9%	82.8%
NEAR	20.0%	3.2%	5.6%
SOC	68.9%	91.2%	78.5%

Table 3: SVM performance on individual relation types (model trained on all $4,328$ training examples)

Correspondingly, the "floor" classification performance is achieved by training the SVM on 100 instances randomly selected from the training data (this set is also used as the seed set L in the bootstrapping experiments), which gives an accuracy of 62.3%. The class-specific performance measures for this classifier are given in Table 4.

As we can see, there is a significant gap between the "floor" and the "ceiling" performance. It would be nice if we could approximate the "ceiling" performance when we only have a small set of labeled data available, which is exactly the purpose of this study.

6.2 Bootstrapping Experiments

With the bootstrapping experiments, we start from the seed set L, and use the remaining data points in the training data

Relation type	Precision	Recall	F-measure
ROLE	71.1%	69.3%	70.2%
PART	87.5%	11.4%	20.1%
AT	55.9%	79.9%	65.7%
NEAR	*undef*	0	*undef*
SOC	44.2%	50.0%	46.9%

Table 4: SVM performance on individual relation types (model trained on 100 seed examples)

as U, to keep augmenting the labeled set. The batch size S is 100.

For the bagging-based bootstrapping, we always generate 10 bootstrap data sets and train a committee of 10 classifiers correspondingly; for the projection-based bootstrapping, we always produce 10 random projections and train 10 classifiers accordingly.

6.2.1 Different Agreement Threshold

We first fix the feature sampling probability p to 0.5 and experiment with different agreement threshold. The reported experimental results on the ACE data are summarized in Figure 1 (next page).

We observe the following patterns from the experimental results:

- When the agreement threshold is relatively high, both *BootProject* and *BootBagging* strategy significantly outperform the naive *SelfBoot* strategy, not only in terms of better classification accuracy, but also in terms of early stoppage (The *SelfBoot* strategy, as it is defined in Algorithm 1, doesn't stop until there is no more unlabeled data available.)
- When the agreement threshold is relatively low, the performance of the *BootBagging* strategy starts to degrade. In fact, it performs even worse than the *SelfBoot* strategy when the agreement threshold is 7 or 8 out of 10.
- The *BootProject* algorithm consistently outperforms *BootBagging*, not only in terms of accuracy and early stoppage, but also in terms of robustness to relatively low agreement threshold.
- One thing particularly encouraging is that the *BootProject* strategy can achieve accuracy around 67% with about 200 data points (for example, it achieves an accuracy of 67.03% with 207 data points when the agreement threshold is 9 out of 10), while a model trained on 200 randomly selected data points can only achieve accuracy of 62.3%.

6.2.2 Different Feature Sampling Probability

Another interesting parameter to investigate is the feature sampling probability p in the *BootProject* algorithm. We present results (Figure 2) obtained with different p, when the agreement threshold is set to 9 out of 10.

A nice property to notice is that, except for the extreme case of very sparse projection ($p = 0.1$), the stopping point of the algorithm is close to where the optimal performance (which is significantly higher than the "floor" performance) can be achieved. This is very useful in practice, because otherwise it would be very tricky to use validation to search the optima when the performance fluctuates in a relatively small range.

We also notice that when $p = 0.3$, i.e., when the projection is relatively sparse but with reasonable overlap between views, the bootstrapping process seems to work the best, which achieves the classification accuracy of 67.3%. The corresponding class-specific performance measures are given in Table 5. As we can see, the performance is significantly boosted from the "floor" case in Table 4.

Relation type	Precision	Recall	F-measure
ROLE	78.5%	69.7%	73.8%
PART	65.6%	34.1%	44.9%
AT	61.0%	84.8%	70.9%
NEAR	*undef*	0	*undef*
SOC	47.0%	57.4%	51.7%

Table 5: Boostrapped SVM performance on individual relation types (feature sampling probability: 0.3; agreement threshold: 9 out of 10)

6.3 Discussion

The proposed *BootProject* algorithm works well, as has been shown in the experimental results above. But why does it work?

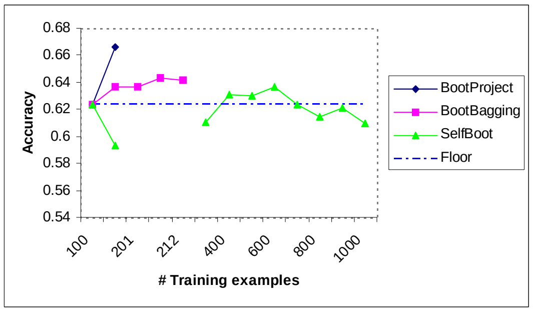
One possible explanation is that, given the relatively large size of the feature space, there is considerable amount of redundancy in it. What *BootProject* does is in fact multiple rounds of random feature selection. If there is enough redundant information in the feature space and if the feature sampling probability p is appropriate, the multiple overlapping "views" can approximately capture the essence of the optimal classifier by agreement.

It is reasonable to speculate that the *BootProject* algorithm won't work well when the number of features is very small, because random sampling in the feature space becomes statistically invalid. This is inescapable for any algorithms based on random sampling. The *BootBagging* algorithm sees the same caveat when the labeled data set L is small, such that bootstrap sets generated from it are not guaranteed to capture its statistical properties any more.

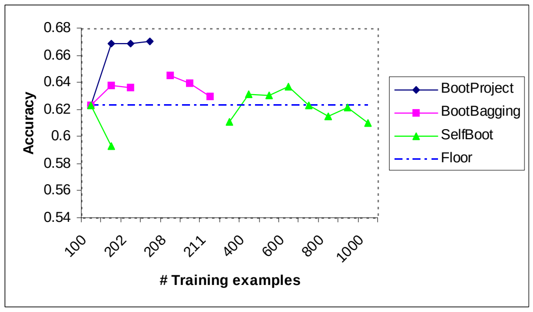
The same issue can be either a convenience or a challenge to the original co-training algorithm. A small feature space would make it easier to validate its two assumptions and to automatically construct the two "views" if they do exist. But if the features are strongly correlated to each other, the co-training algorithm loses its ground of applicability.

7. CONCLUSIONS AND FUTURE WORK

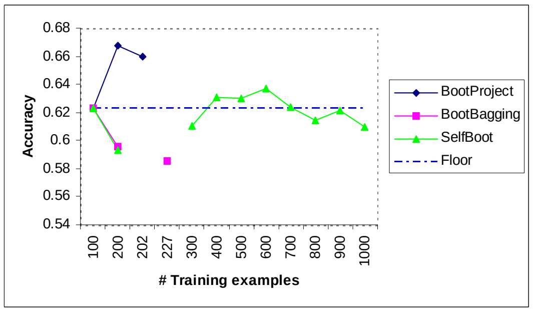
This paper approaches the problem of relation classification with bootstrapping on top of SVM. On one hand, the SVM



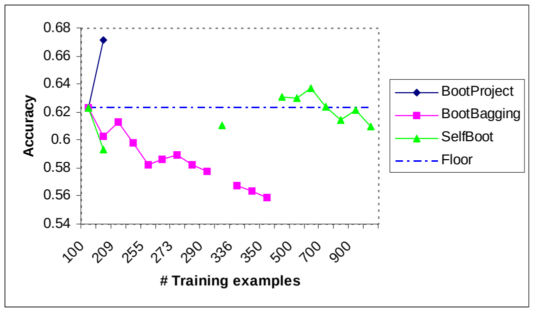
(a) Agreement threshold: 10 out of 10



(b) Agreement threshold: 9 out of 10



(c) Agreement threshold: 8 out of 10



(d) Agreement threshold: 7 out of 10

Figure 1: Comparison of different bootstrapping strategies (seed labeled set size: 100**; batch size:** 100**; feature sampling probability for BootProject:** 0.5)

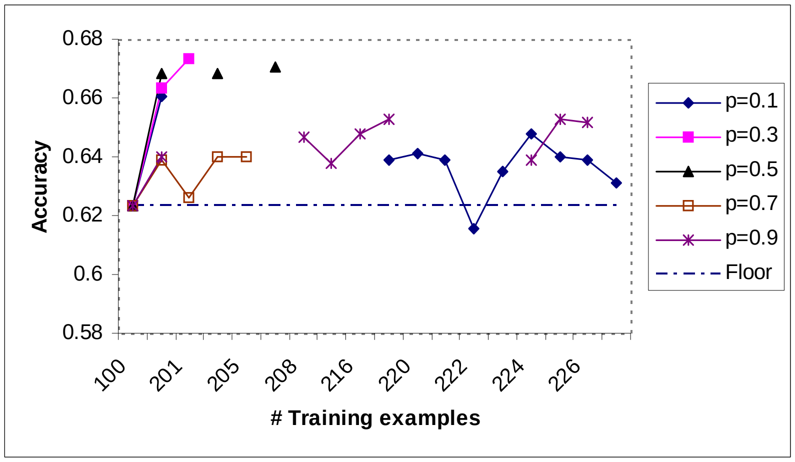


Figure 2: Comparison of different projection probabilities (seed labeled set size: 100**; batch size:** 100**; agreement threshold: 9 out of 10)**

classifier by itself achieves promising results; on the other hand, the proposed *BootProject* algorithm is shown to significantly reduce the need for labeled training data with only limited sacrifice of performance.

In the future, we are interested in pursuing the following directions and answering the following questions:

- More theoretical understanding needs to be gained regarding why and how the proposed *BootProject* algorithm actually works. If indeed the two assumptions of the original co-training algorithm can be dropped, are there any other assumptions to make? It would be interesting to measure the correlation between the bootstrapping performance and the quantified overlap between views and the number of views.
- In this study, we only used a randomly selected portion of the actual training data available as the seed labelled set. This may not be the best strategy. It is conceivable that if we anchor the seed data points more intelligently (e.g., using clustering or other techniques), so that the seed model is more accurate, we may be able to expect better bootstrapping performance.

8. ACKNOWLEDGMENTS

This material is based upon work supported by the National Science Foundation under Grant No. 0308024. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

9. REFERENCES

[1] S. Abney. Bootstrapping. In *Proceedings of ACL-02, 40th Annual Meeting of the Association for Computational Linguistics*, 2002.

[2] S. Abney. Understanding the yarowsky algorithm. *Computational Linguistics*, 30(3), 2004.

[3] E. Agichtein and L. Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM International Conference on Digital Libraries*, 2000.

[4] M. Banko and E. Brill. Scaling to very very large corpora for natural language disambiguation. In *Meeting of the Association for Computational Linguistics*, pages 26–33, 2001.

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

[6] L. Breiman. Bagging predictors. *Machine Learning*, 24(2):123–140, 1996.

[7] S. Brin. Extracting patterns and relations from the world wide web. In *WebDB Workshop at 6th International Conference on Extending Database Technology, EDBT'98*, 1998.

[8] S. Buchholz. *The chunklink script*, 2000. Software available at `http://ilk.uvt.nl/~sabine/chunklink/`.

[9] C.-C. Chang and C.-J. Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[10] E. Charniak. A maximum-entropy-inspired parser. Technical Report CS-99-12, Computer Scicence Department, Brown University, 1999.

[11] M. Collins and Y. Singer. Unsupervised models for named entity classification. In *EMNLP/VLC-99*, 1999.

[12] S. Goldman and Y. Zhou. Enhancing supervised learning with unlabeled data. In *Proc. 17th International Conf. on Machine Learning*, pages 327–334. Morgan Kaufmann, San Francisco, CA, 2000.

[13] T. Joachims. Text categorization with support vector machines: learning with many relevant features. In C. Nédellec and C. Rouveirol, editors, *Proceedings of ECML-98, 10th European Conference on Machine Learning*, number 1398, pages 137–142, Chemnitz, DE, 1998. Springer Verlag, Heidelberg, DE.

[14] T. Kudo and Y. Matsumoto. Chunking with support vector machines. In *Proceedings of NAACL*, pages 192–199, 2001.

[15] V. N. Vapnik. *The nature of statistical learning theory.* Springer-Verlag New York, Inc., 1995.

[16] D. Yarowsky. Unsupervised word sense disambiguation rivaling supervised methods. In *Meeting of the Association for Computational Linguistics*, pages 189–196, 1995.

[17] D. Zelenko, C. Aone, and A. Richardella. Kernel methods for relation extraction. *J. Mach. Learn. Res.*, 3:1083–1106, 2003.

[18] Z. Zhang and D. Radev. Combining labeled and unlabeled data for learning cross-document structural relationships. In *Proceedings of the The First International Joint Conference on Natural Language Processing (IJCNLP-04)*, Sanya, Hainan, P.R.China, March 2004.