Modeling Commonality among Related Classes in Relation Extraction

ZHOU GuoDong Su Jian ZHANG Min
Institute for Infocomm research
21 Heng Mui Keng Terrace, Singapore 119613
Email: {zhougd, sujian, mzhang}@i2r.a-star.edu.sg

Abstract

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem in relation extraction by modeling the commonality among related classes. For each class in the hierarchy either predefined manually or automatically clustered, a linear discriminative function is determined in a top-down way using a perceptron algorithm with the lower-level weight vector derived from the upper-level weight vector. As the upper-level class normally has much more positive training examples than the lower-level class, the corresponding linear discriminative function can be determined more reliably. The upper-level discriminative function then can effectively guide the discriminative function learning in the lower-level, which otherwise might suffer from limited training data. Evaluation on the ACE RDC 2003 corpus shows that the hierarchical strategy much improves the performance by 5.6 and 5.1 in F-measure on least- and medium- frequent relations respectively. It also shows that our system outperforms the previously best-reported system by 2.7 in F-measure on the 24 subtypes using the same feature set.

1 Introduction

With the dramatic increase in the amount of textual information available in digital archives and the WWW, there has been growing interest in techniques for automatically extracting information from text. Information Extraction (IE) is such a technology that IE systems are expected to identify relevant information (usually of pre-defined types) from text documents in a certain domain and put them in a structured format.

According to the scope of the NIST Automatic Content Extraction (ACE) program, current research in IE has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and Characterization (EDC). This paper will focus on the ACE RDC task, which detects and classifies various semantic relations between two entities. For example, we want to determine whether a person is at a location, based on the evidence in the context. Extraction of semantic relationships between entities can be very useful for applications such as question answering, e.g. to answer the query "Who is the president of the United States?".

One major challenge in relation extraction is due to the data sparseness problem (Zhou et al 2005). As the largest annotated corpus in relation extraction, the ACE RDC 2003 corpus shows that different subtypes/types of relations are much unevenly distributed and a few relation subtypes, such as the subtype "Founder" under the type "ROLE", suffers from a small amount of annotated data. Further experimentation in this paper (please see Figure 2) shows that most relation subtypes suffer from the lack of the training data and fail to achieve steady performance given the current corpus size. Given the relative large size of this corpus, it will be time-consuming and very expensive to further expand the corpus with a reasonable gain in performance. Even if we can somehow expend the corpus and achieve steady performance on major relation subtypes, it will be still far beyond practice for those minor subtypes given the much unevenly distribution among different relation subtypes. While various machine learning approaches, such as generative modeling (Miller et al 2000), maximum entropy (Kambhatla 2004) and support vector machines (Zhao et al 2005; Zhou et al 2005), have been applied in the relation extraction task, no explicit learning strategy is proposed to deal with the inherent data sparseness problem caused by the much uneven distribution among different relations.

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem by modeling the commonality among related classes. Through organizing various classes hierarchically, a linear discriminative function is deter-

mined for each class in a top-down way using a perceptron algorithm with the lower-level weight vector derived from the upper-level weight vector. Evaluation on the ACE RDC 2003 corpus shows that the hierarchical strategy achieves much better performance than the flat strategy on least- and medium-frequent relations. It also shows that our system based on the hierarchical strategy outperforms the previously best-reported system.

The rest of this paper is organized as follows. Section 2 presents related work. Section 3 describes the hierarchical learning strategy using the perceptron algorithm. Finally, we present experimentation in Section 4 and conclude this paper in Section 5.

2 Related Work

The relation extraction task was formulated at MUC-7(1998). With the increasing popularity of ACE, this task is starting to attract more and more researchers within the natural language processing and machine learning communities. Typical works include Miller et al (2000), Zelenko et al (2003), Culotta et al (2004), Hasegawa et al (2004), Zhang et al (2005), Roth et al (2002), Kambhatla (2004), Zhao et al (2005) and Zhou et al (2005).

Miller et al (2000) augmented syntactic full parse trees with semantic information corresponding to entities and relations, and built generative models to integrate various tasks such as POS tagging, named entity recognition, template element extraction and relation extraction. The problem is that such integration may impose big challenges such as the need of a large annotated corpus. To overcome the data sparseness problem, generative models typically applied some smoothing techniques to integrate different scales of contexts in parameter estimation, e.g. the back-off approach in Miller et al (2000).

Zelenko et al (2003) proposed extracting relations by computing kernel functions between parse trees. Culotta et al (2004) extended this work to estimate kernel functions between augmented dependency trees and achieved F-measure of 45.8 on the 5 relation types in the ACE RDC 2003 corpus[1]. Hasegawa et al (2004) and Zhang et al (2005) adopted clustering algorithms in unsupervised relation extraction using tree kernels. To overcome the data sparseness problem, various scales of subtrees are applied in the tree kernel computation. Although tree kernel-based approaches are able to explore the huge implicit feature space without much feature engineering, further research work is necessary to make them effective and efficient.

Comparably, feature-based approaches achieved much success recently. Roth et al (2002) used the SNoW multi-class classifier to incorporate various features such as word, part-of-speech and semantic information from WordNet, and proposed a probabilistic reasoning approach to integrate named entity recognition and relation extraction. Kambhatla (2004) employed maximum entropy models with features derived from word, entity type, mention level, overlap, dependency tree, parse tree and achieved F-measure of 52.8 on the 24 relation subtypes in the ACE RDC 2003 corpus. Zhao et al (2005)[2] combined various kinds of knowledge from tokenization, sentence parsing and deep dependency analysis through support vector machines and achieved F-measure of 70.1 on the 7 relation types of the ACE RDC 2004 corpus[3]. Zhou et al (2005) further systematically explored diverse lexical, syntactic and semantic features through support vector machines and achieved F-measure of 68.1 and 55.5 on the 5 relation types and the 24 relation subtypes in the ACE RDC 2003 corpus respectively. To overcome the data sparseness problem, feature-based approaches normally incorporate various scales of context information into the feature vector extensively. These approaches then depend on adopted learning algorithms to weight and combine each feature effectively. For example, an exponential model and an equivalent linear model are applied in the maximum entropy models and feature-based support vector machines to combine each feature through the learned weight vector.

As discussed above, although various approaches have been employed in relation extraction, they only apply some implicit approaches to overcome the data sparseness problem. Until now, there are no explicit ways to resolve the data sparseness problem in relation extraction. Currently, all the current approaches apply the flat

[1] The ACE RDC 2003 corpus defines 5/24 relation types/subtypes between 4 entity types.

[2] Here, we classify this paper into feature-based approaches since the kernels used in Zhao et al (2005) can be easily represented by an explicit feature vector.

[3] The ACE RDC 2004 corpus defines 7/27 relation types/subtypes between 7 entity types.

learning strategy which equally treats training examples in different relations independently and ignore the commonality among different relations. This paper proposes a novel hierarchical learning strategy to resolve this problem by considering the relatedness among different relations and capturing the commonality among related relations. By doing so, the data sparseness problem can be well dealt with and much better performance can be achieved, especially for those relations with small or medium amounts of annotated examples.

3 Hierarchical Learning Strategy

Traditional classifier learning approaches apply the flat learning strategy. That is, they equally treat training examples in different classes independently and ignore the commonality among related classes. The flat strategy will not cause any problem when there are a large amount of training examples for each class, since, in this case, a classifier learning approach can always learn a nearly optimal discriminative function for each class against the remaining classes. However, such flat strategy may cause big problems when there is only a small amount of training examples for some of the classes. In this case, a classifier learning approach may fail to learn a reliable (or nearly optimal) discriminative function for a class with a small amount of training examples, and, as a result, may significantly affect the performance of the class or even the overall performance.

To overcome the inherent problems in the flat strategy, this paper proposes a hierarchical learning strategy which explores the inherent commonality among related classes through a class hierarchy. In this way, the training examples of related classes can help in learning a reliable discriminative function for a class with only a small amount of training examples. To reduce computation time and memory requirements, we will only consider linear classifiers and apply the simple and widely-used perceptron algorithm for this purpose with more options open for future research. In the following, we will first introduce the perceptron algorithm in linear classifier learning, followed by the hierarchical learning strategy using the perceptron algorithm. Finally, we will consider several ways in building the class hierarchy.

3.1 Perceptron Algorithm

This section first deals with binary classification using linear classifiers. Assume an instance space $X = R^n$ and a binary label space $Y = \{-1,+1\}$. With any weight vector $w \in R^n$ and a given instance $x \in R^n$, we associate a linear classifier h_w with a linear discriminative function [4] $f(x) = w \cdot x$ by $h_w(x) = sign(w \cdot x)$, where $sign(w \cdot x) = -1$ if $w \cdot x < 0$ and $sign(w \cdot x) = +1$ otherwise. Here, the margin of w at (x_t, y_t) is defined as $y_t w \cdot x_t$. Then if the margin is possitive, we have a correct prediction with $h_w(x) = y_t$, and if the margin is negative, we have an error with $h_w(x) \neq y_t$. Therefore, given a sequence of training examples $(x_t, y_t) \in X \times Y, t = 1,2...,T$, linear classifier learning attemps to find a weight vector w that achieves a positive margin on as many examples as possible.

Input: the initial weight vector w, the training example sequence $(x_t, y_t) \in X \times Y, t = 1,2...,T$ and the number of the maximal iterations N (e.g. 10 in this paper) of the training sequence[5]

Output: the weight vector w for the linear discriminative function $f = w \cdot x$

BEGIN

$w_1 = w$

REPEAT for t=1,2,…,T*N

1. Receive the instance $x_t \in R^n$
2. Compute the output $o_t = w_t \cdot x_t$
3. Give the prediction $\hat{y}_t = sign(o_t)$
4. Receive the desired label $y_t \in \{-1,+1\}$
5. Update the hypothesis according to
$$w_{t+1} = w_t + \delta_t y_t x_t \quad (1)$$
where $\delta_t = 0$ if the margin of w_t at the given example (x_t, y_t) $y_t w_t \cdot x_t > 0$ and $\delta_t = 1$ otherwise

END REPEAT

Return $w = \sum_{i=N-4}^{N} w_{T*i+1} / 5$

END BEGIN

---

[4] $(w \cdot x)$ denotes the dot product of the weight vector $w \in R^n$ and a given instance $x \in R^n$.

[5] The training example sequence is feed N times for better performance. Moreover, this number can control the maximal affect a training example can pose. This is similar to the regulation parameter C in SVM, which affects the trade-off between complexity and proportion of non-separable examples. As a result, it can be used to control over-fitting and robustness.

Figure 1: the perceptron algorithm

The well-known perceptron algorithm, as shown in Figure 1, belongs to online learning of linear classifiers, where the learning algorithm represents its t-th hyposthesis by a weight vector $w_t \in R^n$. At trial t, an online algorithm receives an instance $x_t \in R^n$, makes its prediction $\hat{y}_t = sign(w_t \cdot x_t)$ and receives the desired label $y_t \in \{-1,+1\}$. What distinguishes different online algorithms is how they update w_t into w_{t+1} based on the example (x_t, y_t) received at trial t. In particular, the perceptron algorithm updates the hypothesis by adding a scalar multiple of the instance, as shown in Equation 1 of Figure 1, when there is an error. Normally, the tradictional perceptron algprithm initializes the hypothesis as the zero vector $w_1 = 0$. This is usually the most natural choice, lacking any other preference.

Smoothing

In order to further improve the performance, we iteratively feed the training examples for a possible better discriminative function. In this paper, we have set the maximal iteration number to 10 for both efficiency and stable performance and the final weight vector in the discriminative function is averaged over those of the discriminative functions in the last few iterations (e.g. 5 in this paper).

Bagging

One more problem with any online classifier learning algorithm, including the perceptron algorithm, is that the learned discriminative function somewhat depends on the feeding order of the training examples. In order to eliminate such dependence and further improve the performance, an ensemble technique, called bagging (Breiman 1996), is applied in this paper. In bagging, the bootstrap technique is first used to build M (e.g. 10 in this paper) replicate sample sets by randomly re-sampling with replacement from the given training set repeatedly. Then, each training sample set is used to train a certain discriminative function. Finally, the final weight vector in the discriminative function is averaged over those of the M discriminative functions in the ensemble.

Multi-Class Classification

Basically, the perceptron algorithm is only for binary classification. Therefore, we must extend the perceptron algorithms to multi-class classification, such as the ACE RDC task. For efficiency, we apply the *one vs. others* strategy, which builds K classifiers so as to separate one class from all others. However, the outputs for the perceptron algorithms of different classes may be not directly comparable since any positive scalar multiple of the weight vector will not affect the actual prediction of a perceptron algorithm. For comparability, we map the perceptron algorithm output into the probability by using an additional sigmoid model:

$$p(y = 1 \mid f) = \frac{1}{1 + \exp(Af + B)} \quad (2)$$

where $f = w \cdot x$ is the output of a perceptron algorithm and the coefficients A & B are to be trained using the model trust alorithm as described in Platt (1999). The final decision of an instance in multi-class classification is determined by the class which has the maximal probability from the corresponding perceptron algorithm.

3.2 Hierarchical Learning Strategy using the Perceptron Algorithm

Assume we have a class hierarchy for a task, e.g. the one in the ACE RDC 2003 corpus as shown in Table 1 of Section 4.1. The hierarchical learning strategy explores the inherent commonality among related classes in a top-down way. For each class in the hierarchy, a linear discriminative function is determined in a top-down way with the lower-level weight vector derived from the upper-level weight vector iteratively. This is done by initializing the weight vector in training the linear discriminative function for the lower-level class as that of the upper-level class. That is, the lower-level discriminative function has the preference toward the discriminative function of its upper-level class. For an example, let's look at the training of the "Located" relation subtype in the class hierarchy as shown in Table 1:

1) Train the weight vector of the linear discriminative function for the "YES" relation vs. the "NON" relation with the weight vector initialized as the zero vector.
2) Train the weight vector of the linear discriminative function for the "AT" relation type vs. all the remaining relation types (including the "NON" relation) with the weight vector initialized as the weight vector

of the linear discriminative function for the "YES" relation vs. the "NON" relation.

3) Train the weight vector of the linear discriminative function for the "Located" relation subtype vs. all the remaining relation subtypes under all the relation types (including the "NON" relation) with the weight vector initialized as the weight vector of the linear discriminative function for the "AT" relation type vs. all the remaining relation types.
4) Return the above trained weight vector as the discriminatie function for the "Located" relation subtype.

In this way, the training examples in different classes are not treated independently any more and the commonality among related classes can be captured via the hierarchical learning strategy. The intuition behind the hierarchical strategy is that the upper-level class normally has much more positive training examples than the lower-level class so that the corresponding linear discriminative function can be determined more reliably. In this way, the training examples of related classes can help in learning a reliable discriminative function for a class with only a small amount of training examples in a top-down way and thus alleviate its data sparseness problem.

3.3 Building the Class Hierarchy

We have just described the hierarchical learning strategy using a given class hierarchy. Normally, a rough class hierarchy can be given manually according to human intuition, such as the one in the ACE RDC 2003 corpus. In order to explore more commonality among sibling classes, we make use of binary hierarchical clustering for sibling classes of both lowest and all levels. This can be done by first using the flat learning strategy to learn the discriminative functions for individual classes and then iteratively combining the two most related classes using the cosine similarity function between their weight vectors in a bottom-up way. The intuition is that related classes should have similar hyper-planes to separate from remaining classes and thus have similar weight vectors.

- **Lowest-level hybrid**: Binary hierarchical clustering is only done at the lowest level while keeping the upper-level class hierarchy. That is, only sibling classes at the lowest level are hierarchically clustered.
- **All-level hybrid**: Binary hierarchical clustering is done at all levels in a bottom-up way. That is, sibling classes at the lowest level are hierarchically clustered first and then sibling classes at the upper-level. In this way, the binary class hierarchy can be built iteratively in a bottom-up way.

4 Experimentation

This paper uses the ACE RDC 2003 corpus provided by LDC to train and evaluate the hierarchical learning strategy. Same as Zhou et al (2005), we only model explicit relations and explicitly model the argument order of the two mentions involved.

4.1 Experimental Setting

Type	Subtype	Freq	Bin Type
AT	Based-In	347	Medium
	Located	2126	Large
	Residence	308	Medium
NEAR	Relative-Location	201	Medium
PART	Part-Of	947	Large
	Subsidiary	355	Medium
	Other	6	Small
ROLE	Affiliate-Partner	204	Medium
	Citizen-Of	328	Medium
	Client	144	Small
	Founder	26	Small
	General-Staff	1331	Large
	Management	1242	Large
	Member	1091	Large
	Owner	232	Medium
	Other	158	Small
SOCIAL	Associate	91	Small
	Grandparent	12	Small
	Other-Personal	85	Small
	Other-Professional	339	Medium
	Other-Relative	78	Small
	Parent	127	Small
	Sibling	18	Small
	Spouse	77	Small

Table 1: Statistics of relation types and subtypes in the training data of the ACE RDC 2003 corpus (Note: According to occurrence frequency, all the subtypes are divided into three bins: large/ middle/ small, with 400 as the low threshold for the large bin and 200 as the up threshold for the small bin).

The training data consists of 674 documents (~300k words) with 9683 relation examples while the held-out testing data consists of 97 documents (~50k words) with 1386 relation examples. All the experiments are done five times on the 24 relation

subtypes in the ACE corpus except otherwise specified with the final performance averaged using the same re-sampling with replacement strategy as the one in the bagging technique. Table 1 lists various types and subtypes of relations for the ACE RDC 2003 corpus, along with their occurrence frequency in the training data. It shows that this corpus suffers from a small amount of annotated data for a few subtypes such as the subtype "Founder" under the type "ROLE".

For comparison, we also adopt the same feature set as Zhou et al (2005): word, entity type, mention level, overlap, base phrase chunking, dependency tree, parse tree and semantic information.

4.2 Experimental Results

Table 2 shows the performance of the hierarchical learning strategy using the existing class hierarchy in the given ACE corpus and its comparison with the flat learning strategy, using the perceptron algorithm. It shows that the pure hierarchical strategy outperforms the pure flat strategy by 1.5 in F-measure. It also shows that further smoothing and bagging improve the performance of the hierarchical and flat strategies by 0.6 and 0.9 in F-measure respectively. As a result, the final hierarchical strategy achieves F-measure of 57.8 and outperforms the final flat strategy by 1.8 in F-measure.

Strategies	P	R	F
Flat	58.2	52.8	55.4
Flat+Smoothing	58.9	53.1	55.9
Flat+Bagging	59.0	53.1	55.9
Flat+Both	**59.1**	**53.2**	**56.0**
Hierarchical	61.9	52.6	56.9
Hierarchical+Smoothing	62.7	53.1	57.5
Hierarchical+Bagging	62.9	53.1	57.6
Hierarchical+Both	**63.0**	**53.4**	**57.8**

Table 2: Performance of the hierarchical learning strategy using the existing class hierarchy and its comparison with the flat learning strategy

Class Hierarchies	P	R	F
Existing	63.0	53.4	57.8
Entirely Automatic	63.4	53.1	57.8
Lowest-level Hybrid	63.6	53.5	58.1
All-level Hybrid	**63.6**	**53.6**	**58.2**

Table 3: Performance of the hierarchical learning strategy using different class hierarchies

Table 3 compares the performance of the hierarchical learning strategy using different class hierarchies. It shows that, the lowest-level hybrid approach, which only automatically updates the existing class hierarchy at the lowest level, improves the performance by 0.3 in F-measure while further updating the class hierarchy at upper levels in the all-level hybrid approach only has very slight effect. This is largely due to the fact that the major data sparseness problem occurs at the lowest level, i.e. the relation subtype level in the ACE corpus. As a result, the final hierarchical strategy using the class hierarchy built with the all-level hybrid approach achieves F-measure of 58.2 in F-measure, which outperforms the final flat strategy by 2.2 in F-measure. For comparison, we also experiment using entirely automatically built class hierarchy, which performs comparably with using only the existing class hierarchy.

With the major goal of resolving the data sparseness problem for the classes with a small amount of training examples, Table 4 compares the best-performed hierarchical and flat learning strategies on the relation subtypes of different training data sizes. Here, we divide various relation subtypes into three bins: large/middle/small, according to their available training data sizes. For the ACE RDC 2003 corpus, we use 400 as the lower threshold for the large bin[6] and 200 as the upper threshold for the small bin[7]. As a result, the large/medium/small bin includes 5/8/11 relation subtypes, respectively. Please see Table 1 for details. Table 4 shows that the hierarchical strategy outperforms the flat strategy by 1.0/5.1/5.6 in F-measure on the large/middle/small bin respectively. This indicates that the hierarchical strategy performs much better than the flat strategy for those classes with a small or medium amount of annotated examples although the hierarchical strategy only performs slightly better by 1.0 and 2.2 in F-measure than the flat strategy on those classes with a large size of annotated corpus and on all classes as a whole respectively. This suggests that the proposed hierarchical strategy can well deal

[6] The reason to choose this threshold is that no relation subtype in the ACE RC 2003 corpus has training examples in between 400 and 900.

[7] A few minor relation subtypes only have very few examples in the testing set. The reason to choose this threshold is to guarantee a reasonable number of testing examples in the small bin. For the ACE RC 2003 corpus, using 200 as the upper threshold will fill the small bin with about 100 testing examples while using 100 will include too few testing examples for reasonable performance evaluation.

with the data sparseness problem in the ACE RDC 2003 corpus.

Bin Type(cosine similarity)	Large Bin (0.98)			Middle Bin (0.92)			Small Bin (0.81)		
	P	R	F	P	R	F	P	R	F
Flat Strategy	62.3	61.9	62.1	60.8	38.7	47.3	33.0	21.7	26.2
Hierarchical Strategy	66.4	60.2	63.1	67.6	42.7	52.4	40.2	26.3	31.8

Table 4: Comparison of the hierarchical and flat learning strategies on the relation subtypes of different training data sizes. Notes: the figures in the parentheses indicate the cosine similarities between the weight vectors of the linear discriminative functions learned using the two strategies.

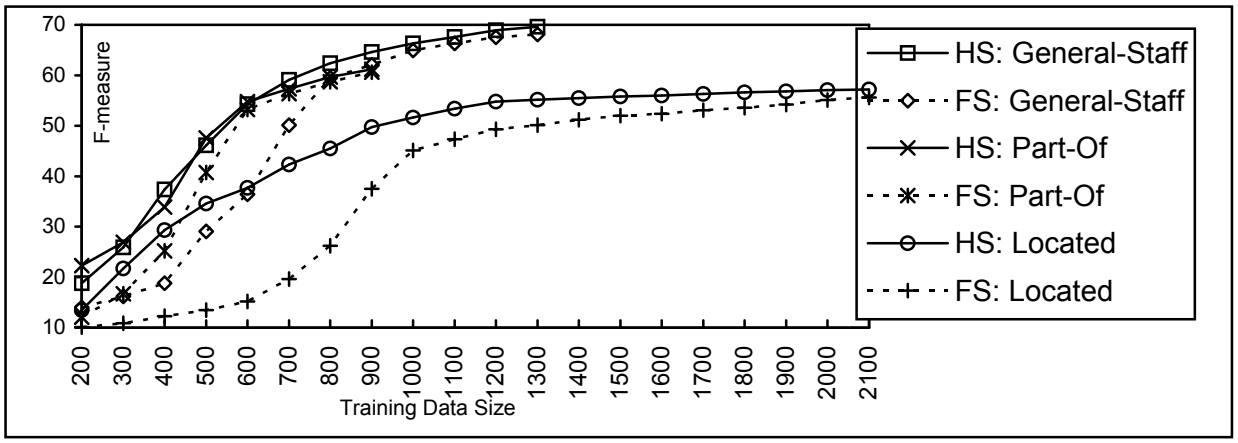

Figure 2: Adaptability of the hierarchical strategy and its comparison with the flat strategy for some major relation subtypes (Note: FS for the flat strategy and HS for the hierarchical strategy)

System	Performance		
	P	R	F
Our: Perceptron Algorithm + Hierarchical Strategy	63.6	53.6	58.2
Zhou et al (2005): SVM + Flat Strategy	63.1	49.5	55.5
Kambhatla (2004): Maximum Entropy + Flat Strategy	63.5	45.2	52.8

Table 5: Comparison of our system with other best-reported systems

An interesting question is about the similarity between the linear discriminative functions learned using the hierarchical and flat learning strategies. Table 4 compares the cosine similarities between the weight vectors of the linear discriminative functions using the two strategies for different bins, weighted by the training data sizes of different relation subtypes. It shows that the linear discriminative functions learned using the two strategies are very similar (with the cosine similarity 0.98) for the relation subtypes belonging to the large bin while the linear discriminative functions learned using the two strategies are not for the relation subtypes belonging to the medium/small bin with the cosine similarity 0.92/0.81 respectively. This means that the use of the hierarchical strategy over the flat strategy only has very slight change on the linear discriminative functions for those classes with a large amount of annotated examples while its effect on those with a small amount of annotated examples is obvious. This contributes to and explains (the degree of) the performance difference between the two strategies on the different training data sizes as shown in Table 4.

Due to the difficulty of building a large annotated corpus, another interesting question is about the adaptability of the hierarchical learning strategy and its comparison with the flat learning strategy. Figure 2 shows the effect of different training data sizes for some major relation subtypes while keeping all the training examples of remaining relation subtypes. It shows that the hierarchical strategy performs much better than the flat strategy when only a small amount of training examples is available. It also shows that the hierarchical strategy can achieve stable performance much faster than the flat strategy. Finally, it shows that the ACE RDC 2003 task suffers from the lack of training examples. Among the three major relation sub-

types, only the subtype "Located" achieves steady performance.

Finally, we also compare our system with the previously best-reported systems, such as Kambhatla (2004) and Zhou et al (2005). Table 5 shows that our system outperforms the previously best-reported system by 2.7 in F-measure, largely due to the gain in recall. It indicates that, although support vector machines and maximum entropy models have much more solid support in theory and always perform better than the simple perceptron algorithm in most (if not all) applications, the hierarchical learning strategy using the perceptron algorithm can easily overcome the difference and outperforms the flat learning strategy using the overwhelming support vector machines and maximum entropy models in relation extraction, at least on the ACE RDC 2003 corpus.

5 Conclusion

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem in relation extraction by modeling the commonality among related classes. For each class in a class hierarchy, a linear discriminative function is determined in a top-down way using the perceptron algorithm with the lower-level weight vector derived from the upper-level weight vector. In this way, the upper-level discriminative function can effectively guide the lower-level discriminative function learning. Evaluation on the ACE RDC 2003 corpus shows that the hierarchical strategy performs much better than the flat strategy in resolving the critical data sparseness problem in relation extraction.

In the future work, we will explore the hierarchical learning strategy using other machine learning approaches besides online classifier learning approaches such as the simple perceptron algorithm applied in this paper. Moreover, just as indicated in Figure 2, most relation subtypes in the ACE RDC 2003 corpus (arguably the largest annotated corpus in relation extraction) suffer from the lack of training examples. Therefore, a critical research in relation extraction is how to rely on semi-supervised learning approaches (e.g. bootstrap) to alleviate its dependency on a large amount of annotated training examples and achieve better and steadier performance.

References

Breiman L. (1996) Bagging Predictors. *Machine Learning*, 24(2): 123-140.

Collins M. (1999). Head-driven statistical models for natural language parsing. *Ph.D. Dissertation*, University of Pennsylvania.

Culotta A. and Sorensen J. (2004). Dependency tree kernels for relation extraction. *ACL'2004*. 423-429. 21-26 July 2004. Barcelona, Spain.

Hasegawa T., Sekine S. and Grishman R. (2004). Discovering relations among named entities from large corpora. *ACL'2004*. 415-422. 21-26 July 2004. Barcelona, Spain.

Miller G.A. (1990). WordNet: An online lexical database. *International Journal of Lexicography*. 3(4):235-312.

Miller S., Fox H., Ramshaw L. and Weischedel R. (2000). A novel use of statistical parsing to extract information from text. *ANLP'2000*. 226-233. 29 April - 4 May 2000, Seattle, USA

MUC-7. (1998). *Proceedings of the 7th Message Understanding Conference (MUC-7)*. Morgan Kaufmann, San Mateo, CA.

Kambhatla N. (2004). Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. *ACL'2004(Poster)*. 178-181. 21-26 July 2004. Barcelona, Spain.

Platt J. 1999. Probabilistic Outputs for Support Vector Machines and Comparisions to regularized Likelihood Methods. In *Advances in Large Margin Classifiers*. Edited by Smola .J., Bartlett P., Scholkopf B. and Schuurmans D. MIT Press.

Roth D. and Yih W.T. (2002). Probabilistic reasoning for entities and relation recognition. *CoLING'2002*. 835-841.26-30 Aug 2002. Taiwan.

Zelenko D., Aone C. and Richardella. (2003). Kernel methods for relation extraction. *Journal of Machine Learning Research*. 3(Feb):1083-1106.

Zhang M., Su J., Wang D.M., Zhou G.D. and Tan C.L. (2005). Discovering Relations from a Large Raw Corpus Using Tree Similarity-based Clustering, *IJCNLP'2005, Lecture Notes in Computer Science (LNCS 3651)*. 378-389. 11-16 Oct 2005. Jeju Island, South Korea.

Zhao S.B. and Grisman R. 2005. Extracting relations with integrated information using kernel methods. ACL'2005: 419-426. Univ of Michgan-Ann Arbor, USA, 25-30 June 2005.

Zhou G.D., Su J. Zhang J. and Zhang M. (2005). Exploring various knowledge in relation extraction. *ACL'2005*. 427-434. 25-30 June, Ann Arbor, Michgan, USA.