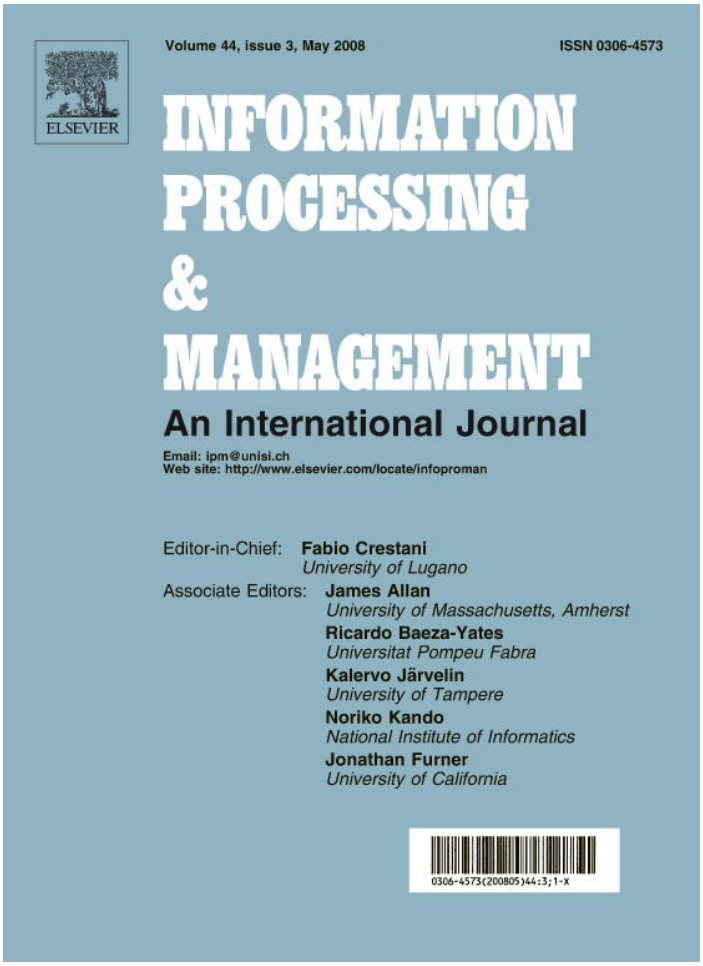

Available online at www.sciencedirect.com
ScienceDirect
Information Processing and Management 44 (2008) 1008–1021
INFORMATION PROCESSING & MANAGEMENT
www.elsevier.com/locate/infoproman

Hierarchical learning strategy in semantic relation extraction ☆

Zhou GuoDong [a,*], Zhang Min [b], Ji DongHong [c], Zhu QiaoMing [a]

[a] School of Computer Science and Technology, Soochow University, 215006, China
[b] Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore 119613, Singapore
[c] Center for Study of Language and Information, Wuhan University, 430072, China

Received 11 January 2007; received in revised form 6 June 2007; accepted 16 July 2007
Available online 29 August 2007

Abstract

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem in semantic relation extraction by modeling the commonality among related classes. For each class in the hierarchy either manually predefined or automatically clustered, a discriminative function is determined in a top-down way. As the upper-level class normally has much more positive training examples than the lower-level class, the corresponding discriminative function can be determined more reliably and guide the discriminative function learning in the lower-level one more effectively, which otherwise might suffer from limited training data. In this paper, two classifier learning approaches, i.e. the simple perceptron algorithm and the state-of-the-art Support Vector Machines, are applied using the hierarchical learning strategy. Moreover, several kinds of class hierarchies either manually predefined or automatically clustered are explored and compared. Evaluation on the ACE RDC 2003 and 2004 corpora shows that the hierarchical learning strategy much improves the performance on least- and medium-frequent relations.

Keywords: Semantic relation extraction; Data sparseness problem; Hierarchical learning strategy; Class hierarchy; Flat learning strategy

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 Information Extraction (IE), which identifies relevant information (usually of predefined types) from text documents in a certain domain and put them in a structured format.

According to the scope of the ACE program (ACE, 2000–2005), IE has three main objectives: Entity Detection and Tracking (EDT), Relation Detection and Characterization (RDC), and Event Detection and

☆ This paper is a much extended version of Zhou, Su, and Zhang (2006) at COLING-ACL'2006 and Zhou, Zhang, and Fu (2006) at AIRS'2006.

* Corresponding author. Tel.: +86 512 65241043; fax: +86 512 65243205.
E-mail addresses: gdzhou@suda.edu.cn (G. Zhou), mzhang@i2r.a-star.edu.sg (M. Zhang), dhji@whu.edu.cn (D. Ji), qmzhu@suda.edu.cn (Q. Zhu).

0306-4573/$ - see front matter
doi:10.1016/j.ipm.2007.07.007

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 (Hirschman & Gaizauskas, 2001), e.g. to answer the query "Who is the president of the United States?", and information retrieval (Strzalkowski & Carballo, 1998; Gao, Nie, Wu, & Cao, 2004), e.g. to expand the query "George W. Bush" with "the president of the United States" via his relationship with the country "the United States".

One major challenge in semantic relation extraction is due to the data sparseness problem (Zhou, Su, Zhang, & Zhang, 2005). As the largest annotated corpus in relation extraction, the ACE RDC 2003 corpus shows that different relation subtypes are much unevenly distributed and a few 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 Fig. 1) 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. While various machine learning approaches, such as generative modeling (Miller, Fox, Ramshaw, & Weischedel, 2000), maximum entropy (Kambhatla, 2004) and support vector machines (Zhao & Grisman, 2005; Zhou et al., 2005), have been applied in the relation extraction task, no explicit learning strategy is proposed to deal with the data sparseness problem.

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 according to their relatedness, a discriminative function for a given class is determined in a top-down way by capturing the commonality among related classes to guide the training of the given class. Here, two classifier learning approaches are applied: the simple perceptron algorithm (PA) and the state-of-the-art support vector machines (SVM). For PA, a linear discriminative function of a lower-level class is determined with its weight vector derived from that of its upper-level class. For SVM, the guidance is done by, when training a lower-level class, discounting those negative training instances which do not belong to the support vectors of the upper-level class. 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. Moreover, several kinds of class hierarchies either manually predefined or automatically clustered are explored and compared. Evaluation on the ACE RDC 2003 and 2004 corpora 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 much outperforms previous best-reported systems.

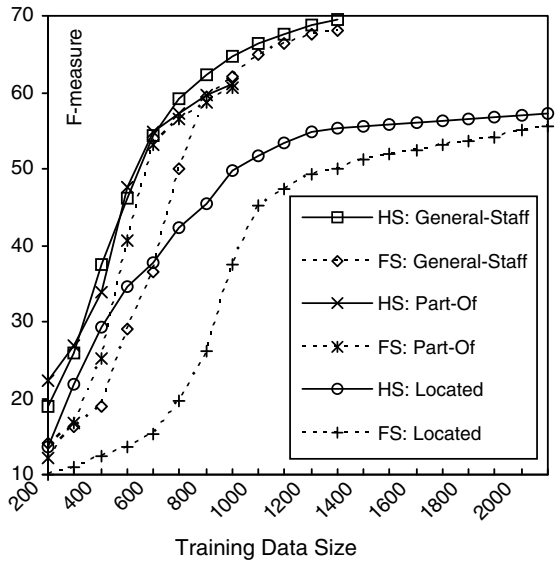

Fig. 1. Comparison on adaptability of the hierarchical strategy vs. the flat strategy for some major relation subtypes on the ACE RDC 2003 corpus (*Note*: FS for the flat strategy and HS for the hierarchical strategy).

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

2. Related work

The semantic relation extraction task was formulated at MUC-7 (1998) and extended at ACE (2000–2005). With the increasing popularity of ACE, it is starting to attract more and more researchers within the Natural Language Processing and Information Processing communities. Representative related works can be classified into three categories according to different approaches they used: generative models (Miller et al., 2000), kernel-based approaches (Zelenko, Aone, & Richardella, 2003; Culotta & Sorensen, 2004; Bunescu & Mooney, 2005a, 2005b; Zhang, Su, Wang, Zhou, & Tan, 2005, 2006), and feature-based approaches (Kambhatla, 2004; Zhao & Grisman, 2005;[1] 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 and relation extraction. The problem is that such integration using a generative approach may impose big challenges such as the need of a large annotated corpus. To overcome the data sparseness problem, generative models typically apply 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 and Sorensen (2004) extended this work to estimate kernel functions between augmented dependency trees and achieved the *F*-measure of 45.8 on the 5 relation types in the ACE RDC 2003 corpus.[2] Bunescu and Mooney (2005a) proposed a shortest path dependency kernel. They argued that the information to model a relationship between two entities can be typically captured by the shortest path between them in the dependency graph. It achieved the *F*-measure of 52.5 on the 5 relation types in the ACE RDC 2003 corpus. Bunescu and Mooney (2005b) proposed a subsequence kernel and applied it in protein interaction and ACE relation extraction tasks. Zhang et al. (2005) adopted clustering algorithms in unsupervised relation extraction using tree kernels. As the state-of-the-art in kernel-based relation extraction, Zhang, Su, Zhang, and Zhou (2006) proposed to apply a convolution tree kernel in relation extraction and explored various relation feature space. It achieved the *F*-measure of 61.9 and 63.6 on the 5 relation types in the ACE RDC 2003 corpus and the 7 relation types in the ACE RDC 2004 corpus respectively. To overcome the data sparseness problem, various scales of sub-trees 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. Kambhatla (2004) employed maximum entropy models with various features derived from word, entity type, mention level, overlap, dependency tree, parse tree and achieved the *F*-measure of 52.8 on the 24 relation subtypes in the ACE RDC 2003 corpus. Zhao and Grisman (2005) combined various kinds of knowledge from tokenization, sentence parsing and deep dependency analysis through SVM and achieved the *F*-measure of 70.3 on the 7 relation types in the ACE RDC 2004 corpus.[3] Zhou et al. (2005) further systematically explored diverse lexical, syntactic and semantic features through SVM and achieved the *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 contexts into the feature vector, not only including complex features but also simple features. In principle, this is similar to the inclusion of various scales of sub-trees in the tree kernel computation.[4] These approaches then depend on adopted learning

[1] Here, we classify this paper into feature-based approaches since the feature space in the kernels of Zhao & Grisman (2005) can be easily represented by an explicit feature vector.

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

[3] The ACE RDC 2004 corpus defines 7/23 relation types/subtypes between 7 entity types.

[4] This is also similar to linear interpolation in language modeling which normally interpolates the probability of a trigram with those of a unigram and a bigram in a log-linear way.

algorithms to weight and combine each feature effectively. For example, an exponential model and a linear model are applied in the maximum entropy models and support vector machines respectively to combine each feature via the learned weight vector.

In summary, although various approaches have been employed in relation extraction, they implicitly attack the data sparseness problem by using features of different contexts in feature-based approaches or including different scales of sub-trees in tree kernel-based approaches. Until now, there are no explicit ways to capture the hierarchical topology in relation extraction. Currently, all the current approaches apply the flat 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 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 and 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 in a top-down way. Here, two classifier learning approaches are applied: the simple perceptron algorithm (PA) and the state-of-the-art support vector machines (SVM). For PA, a linear discriminative function of a lower-level class is determined with its weight vector derived from that of its upper-level class. For SVM, the guidance is done by, when training a lower-level class, discounting those negative training instances of the given class which do not belong to the support vectors of the upper-level class. Moreover, several kinds of class hierarchies either manually predefined or automatically clustered are explored and compared.

In the following, we will first describe PA and the hierarchical learning strategy using PA, followed by SVM and the hierarchical learning strategy using SVM. Finally, we will consider several ways in building the class hierarchy.

3.1. Perceptron algorithm

The well-known perceptron algorithm (PA) belongs to online learning of linear classifiers, where the learning algorithm represents its tth hypothesis 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 $\hat{y}_t = \text{sign}(w_t \cdot x_t)$[5] and receives the desired label $y_t \in \{-1, +1\}$. What distinguishes different online algorithms is how they update w into w_{t+1} based on the example (x_t, y_t) received at trial t. In particular, PA updates the hypothesis by adding a scalar multiple of the instance when there is an error. Normally, PA initializes the hypothesis as the zero vector $w_1 = 0$. This is usually the most natural choice, lacking any other preference.

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

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

stable performance, and the final weight vector of the discriminative function is averaged over those in the last few iterations (e.g. 5 in this paper).[6]

Basically, PA is only for binary classification. Therefore, we must extend it 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 the algorithm. For comparability, we map the PA output into the probability by using an additional sigmoid model:[7]

$$p(y = 1|f) = \frac{1}{1 + \exp(Af + B)} \tag{1}$$

where $f = w \cdot x$ is the PA output and the coefficients A and B are to be trained using the model trust algorithm 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 returned from the corresponding PA using the additional sigmoid model.

3.2. Hierarchical learning strategy using perceptron algorithm

Assume we have a class hierarchy for the semantic relation extraction 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 its weight vector derived from the weight vector of the upper-level class 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. The intuition behind the hierarchical learning strategy using PA is that, for difficult tasks such as semantic relation extraction on the ACE RDC 2003 and 2004 corpora, the linear discriminative function learnt from PA may be sensitive to the initial weight vector. Please note that this may not apply to easy tasks such as POS tagging and NP chunking.[8] We will justify our intuition in Section 4 using some experiments.

As an example, let us look at the training of the "Spouse" relation subtype (that is "YES.SOCAL.Spouse") in the class hierarchy as shown in Table 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.

[6] Strictly, our perceptron algorithm is a voted version, which is more robust than the original version. The reason we use the voted version is to improve the performance of the perceptron algorithm, which has been justified by Collins (2002).

[7] The prediction scores of different PA classifiers may have different scales (e.g. $Cf(x)$ for any constant C has the same discriminative power as $f(x)$) and the goal of the additional sigmoid model is to make the prediction scores of different classifiers comparable. For different classifiers, their corresponding sigmoid models are learnt separately and may be different (with different A and B).

[8] Freund and Schapire (1990) and Collins (2002) proved that, when the training data is separable, the (voted) perceptron algorithm will converge. Collins (2002) also proved that "the perceptron algorithm can be robust to some training data examples being difficult or impossible to tag correctly". That is, "if the training data is 'close' to being separable with margin δ ... the algorithm will make a small number of mistakes". Furthermore, Freund and Schapire (1990) also proved that, if the perceptron algorithm makes a relatively small number of mistakes on a training sample, it is likely to generalize well to new examples. However, the above statements only apply to easy tasks (when the training data is "close" to be separable), such as POS tagging and NP chunking explored in Collins (2002) with the F-measure of ~95, and may not apply to difficult tasks, such as semantic relation extraction explored in this paper with the F-measure of 60+ on the ACE RDC corpora. Evaluation on the ACE RDC 2003 and 2004 corpora shows that hierarchical learning strategy using the perceptron algorithm (which applies the weight vector of the upper classifier as the initial weight vector of the lower one) performs much better than the flat learning strategy (which normally applies a zero weight vector as its initial weight vector), especially for medium- and least-frequent classes. This suggests that PA is sensitive to the initial weight vector on difficult tasks, such as semantic relation extraction on the ACE RDC corpora, especially for medium- and least-frequent classes. Of course, proving it from a statistical learning theory point-of-view is worthy exploring in the future.

Table 1
Semantic relation statistics in the training data of the ACE RDC 2003 corpus

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

- Train the weight vector of the linear discriminative function for the "YES.SOCIAL" 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.
- Train the weight vector of the linear discriminative function for the "YES.SOCIAL.Spouse" 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 "YES.SOCIAL" relation type vs. all the remaining relation types.
- Return the above trained weight vector as the discriminate function for the "YES.SOCIAL.Spouse" relation subtype.

Please note: considering the argument order of the two mentions, one more step is necessary to differentiate the two different argument orders for a non-symmetric relation subtype. For this issue, please see Section 4.1 in more details.

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 this strategy is that the upper-level class normally has more positive training examples than the lower-level class so that the corresponding discriminative function can be determined more reliably. As a result, 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. Support vector machines

Support Vector Machines (SVM) is a supervised machine learning technique motivated by the statistical learning theory (Vapnik, 1995). Based on the structural risk minimization of the statistical learning theory, SVM seeks an optimal separating hyper-plane to divide the training examples into two classes (positive vs.

negative) and make decisions based on support vectors which are selected as the only effective instances in the training set.

Basically, SVMs are binary classifiers. In this paper, we use the same one vs. strategy described in Section 3.1 to extend SVM to multi-class classification. The reason why we choose SVM is that SVM represents the state-of–the-art in the machine learning community. Moreover, there are good implementations of the algorithm available. In this paper, we use the binary-class SVMLight[9] developed by Joachims (1998). By default, the simple linear kernel is applied unless specified. The final decision of an instance in multi-class classification is determined by the class which has the maximal confidence value returned from the corresponding SVM.

3.4. Hierarchical learning strategy using support vector machines

For SVM, the same principal of the hierarchical learning strategy applies here except that the guidance of related classes in training a discriminative function for a given class is done by discounting those negative training instances of the given class which do not belong to the support vectors of the upper-level class. Here, all the positive training examples of the given class are kept since their number is always much less that of the negative training examples in most classification tasks, such as semantic relation extraction on the ACE RDC 2003 and 2004 corpora.

For an example, let us look at the training of the "Spouse" relation subtype (i.e. "YES.SOCIAL.Spouse") in the class hierarchy as shown in Table 1:

- Train a SVM for the "YES" relation vs. the "NON" relation.
- Train a SVM for the "YES.SOCIAL" relation type vs. all the remaining relation types (including the "NON" relation) by discounting the negative training examples of the "YES.SOCIAL" relation type, which do not belong to the support vectors of the SVM for the "YES" relation vs. the "NON" relation, and keeping the positive training examples of the "YES.SOCIAL" relation type.
- Train a SVM for the "YES.SOCIAL.Spouse" relation subtype vs. all the remaining relation subtypes under all the relation types (including the "NON" relation) by discounting the negative training examples of the "YES.SOCIAL.Spouse" relation subtype, which do not belong to the support vectors of the SVM for the "YES.SOCIAL" relation vs. all the remaining relation types (including the "NON" relation), and keeping the positive training examples of the "YES.SOCIAL.Spouse" relation subtype.
- Return the above trained SVM as the classifier for the "YES.SOCIAL.Spouse" relation subtype.

In the case of using SVM, related classes help by discounting the effect of less-related classes when determining the support vectors for a given class. On the one hand, this can largely reduce the noisy effect. On the other hand, this can largely reduce the search space in training and make SVM training converge much faster. As a result, the support vectors of a given class are much biased to the ones of the upper class which covers the given class and its related classes. Our experimentation shows that similar performance is achieved when the discounting weight ranges from 0.1 to 0.3 for the ACE RDC task using the ACE RDC 2003 and 2004 corpora. Therefore, the discounting weight is set to 0.2 throughout the paper. That is, the cost of a discounted training example is multiplied by 0.2.

3.5. Building the class hierarchy

We have just described the hierarchical learning strategy using a given class hierarchy. The intuition is that related classes should have similar hyper-planes to separate from other classes and thus have similar discriminative functions. The next problem is how to build a class hierarchy. For our hierarchical learning strategy to work well, the adjacent classes in the class hierarchy should have similar disciminative functions. A simple way is to automatically build it bottom-up using binary hierarchical clustering with the cosine similarity between

[9] Joachims has just released a new version of SVMLight for multi-class classification. However, this paper only uses the binary-class version. For details about SVMLight, please see http://svmlight.joachims.org/.

the disciminative functions of two classes. Here, the discriminative function of a class is learnt using the flat learning strategy. In principle, the weight vector of the discriminative function of a class learnt using PA or SVM can be regarded as linear interpolation of weighed support vectors. Moreover, we have explored to represent each class with its corresponding mean (averaged over its class members) and automatically build it bottom-up using binary hierarchical clustering with the cosine similarity between the average means of two classes.

Normally, a rough class hierarchy can be given manually according to human intuition, such as the two-level one in the ACE RDC 2003 corpus. In order to measure the usefulness of an existing class hierarchy and how well it can fit our hierarchical strategy, we have explored the hierarchical learning strategy using the existing class hierarchy. Finally, in order to explore more commonality among sibling classes in the existing class hierarchy, we make use of binary hierarchical clustering for sibling classes at both the lowest level and all levels. This can be done by first either using the flat learning strategy to learn the discriminative functions for individual classes or calculating their average means and then iteratively combining the two most related classes using the cosine similarity function between the weight vectors of the discriminative functions or average means in a bottom-up way.

- *Hybrid* (*lowest-level*): Binary hierarchical clustering is only done at the lowest level (i.e. the relation subtype level for the ACE RDC task in the paper) while keeping the upper-level class hierarchy. That is, only sibling classes at the lowest level are hierarchically clustered in a binary way.
- *Hybrid* (*all-level*): Binary hierarchical clustering is done at all levels in a bottom-up way. That is, sibling classes at the lowest level are hierarchically clustered in a binary way 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 focuses on the ACE RDC task with evaluation on the ACE RDC 2003 and 2004 corpora.

4.1. Experimental setting

Evaluation is mainly done on the ACE RDC 2003 corpus. The training data consists of 674 documents (~300k words) with 9683 positive relation examples while the held-out testing data consists of 97 documents (~50k words) with 1386 positive relation examples. Table 1 lists various types and subtypes of relations for the 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". Here, we also divide various relation subtypes into three bins: large/middle/small, according to their training data sizes. In this paper, we use 400 as the lower threshold for the large bin and 200 as the upper threshold for the small bin.[10] As a result, the large/medium/small bin includes 5/8/11 relation subtypes in the corpus, respectively.

Detailed evaluation has been also done on the ACE RDC 2004 corpus, which contains 451 documents and 5702 negative relation instances. For comparison with Zhao and Grisman (2005), we only use the same 348 nwire and bnews documents, which contain 125k words and 4400 relation instances. Evaluation is done using 5-fold cross-validation and shows similar tendency with the ACE RDC 2003 corpus. To avoid redundancy, we will only report the final performance on the ACE RDC 2004 corpus.

In this paper, we 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. Moreover, we also explicitly model the argument order of the two mentions involved. For example, when comparing mentions m1 and m2, we distinguish between m1-ROLE.Citizen-Of-m2 and m2-ROLE.Citizen-Of-m1. Note that, in the ACE

[10] 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.

RDC 2003 task, 6 of these 24 relation subtypes are symmetric: "NEAR.Relative-Location", "SOCIAL.Associate", "SOCIAL.Other-Relative", "SOCIAL.Other-Professional", "SOCIAL.Sibling", and "SOCIAL.Spouse". In this way, we model semantic relation extraction in the ACE RDC 2003 task as a multi-class classification task with 43 ($24 \times 2 - 6 + 1$) classes, two for each relation subtype (except the above 6 symmetric subtypes) and a "NONE" class for the case where the two mentions are not related. For the ACE RDC 2004 task, 6 of these 23 relation subtypes are symmetric: "PHYS.Near", "PER-SOC.Business", "PER-SOC.Family", "PER-SOC.Other", "EMP-ORG.Partner", and "EMP-ORG.Other". In this way, we model semantic relation extraction in the ACE RDC 2004 task as a multi-class classification task with 41 ($23 \times 2 - 6 + 1$) classes, two for each relation subtype (except the above 6 symmetric subtypes) and a "NONE" class for the case where the two mentions are not related.

4.2. Experimental results

Table 2 compares the hierarchical learning strategy and the flat learning strategy with different automatically built class hierarchies on the ACE RDC 2003 corpus using the perceptron algorithm (PA) and SVM respectively. Here, two cosine similarity measurements (i.e. between the weight vectors of the discriminative functions and between the average means of the classes), as described in Section 3.5, are applied to automatically build the class hierarchy. It shows that the hierarchical strategy using the class hierarchies automatically built using the discriminative functions and average means outperforms the flat strategy by about 1.6 and 1.3 in *F*-measure respectively. This justifies that our hierarchical learning strategy using automatically built class hierarchies works and that our hierarchical learning strategy has much better discriminative power than the flat learning strategy. It also shows that the hierarchical strategy with the class hierarchy automatically built using the discriminative functions performs slightly better than the one using the average means. This may be due to that the discriminative functions only depends on weighted support vectors and are more robust than the average means, which are averaged over all class members. In the following, we will only consider our class hierarchy built using the discriminative functions by default. As a result, the hierarchical strategy using the automatically built class hierarchy achieves the *F*-measures of 57.5 and 57.4 using PA and SVM respectively.

In order to verify the usefulness of the existing class hierarchy predefined on the ACE RDC corpora, such as the one shown in Table 1, Table 3 compares the performance of the hierarchical learning strategy using different class hierarchies. It shows that the hierarchical learning strategy using the existing class hierarchy performs comparably with the one using the automatically built one. It suggests that the existing class

Table 2
Comparison of the hierarchical learning strategy vs. the flat learning strategy using automatically built class hierarchies on the ACE RDC 2003 corpus

Strategies	PA			SVM		
	P	*R*	*F*	*P*	*R*	*F*
Flat	58.9	53.1	55.9	63.1	49.5	55.5
Hierarchical: Automatic (DF)	63.0	52.8	57.4	63.5	52.5	57.4
Hierarchical: Automatic (mean)	62.6	52.6	57.1	63.2	52.0	57.0

Table 3
Comparison of the hierarchical learning strategy using different class hierarchies on the ACE RDC 2003 corpus

Class hierarchies	PA			SVM		
	P	*R*	*F*	*P*	*R*	*F*
Automatic (DF)	63.0	52.8	57.4	63.5	52.5	57.4
Existing hierarchy	62.7	53.1	57.5	63.7	52.3	57.4
Hybrid (lowest-level)	63.3	53.2	57.8	64.1	52.8	57.9
Hybrid (all-level)	63.3	53.4	57.9	64.3	52.9	58.0

hierarchy provided on the ACE RDC corpora well preserves the similarity between the discriminative functions of classes. It also shows that, the lowest-level hybrid approach, which only automatically updates the existing class hierarchy at the lowest level, slightly improves the performance by about 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 RDC 2003 corpus. As a result, the hierarchical learning strategy using the class hierarchy built with the all-level hybrid approach achieves the *F*-measure of 57.9 and 58.0 using PA and SVM respectively, which outperforms the flat strategy by 2.0–2.5 in *F*-measure. In order to justify the usefulness of our hierarchical learning strategy when a rough class hierarchy is not available and difficult to determine manually, we also experiment using entirely automatically built class hierarchy without considering the existing class hierarchy. Table 3 shows that using automatically built class hierarchy performs comparably with using only the existing one.

With the major goal of resolving the data sparseness problem for the classes with a small amount of training examples, Tables 4 and 5 compare the hierarchical and flat learning strategies on the relation subtypes of different training data sizes using PA and SVM respectively on the ACE RDC 2003 corpus. Here, we divide various relation subtypes into three bins: large/middle/small, according to their available training data sizes. Please see Table 1 for details. Table 4 shows that the hierarchical strategy outperforms the flat strategy by about 0.9/5.2/6.1 in *F*-measure on the large/middle/small bin respectively. This indicates that the hierarchical strategy (which applies the weight vector of the upper classifier as the initial weight vector of the lower one) performs much better than the flat strategy (which normally applies a zero weight vector as its initial weight vector) for those classes with a small or medium amount of annotated examples although the hierarchical strategy only performs slightly better by about 0.9 in *F*-measure than the flat strategy on those classes with a large size of annotated corpus. This suggests that the proposed hierarchical strategy can well deal with the data sparseness problem in the ACE RDC 2003 corpus using PA. This also suggests that the PA classifier is sensitive to the initial weight vector on difficult tasks, such as semantic relation extraction on the ACE RDC 2003 corpus, especially for medium- and least-frequent classes, and that the hierarchical learning strategy provides a better start point for discriminative function learning than the flat learning strategy. Table 5 shows the similar tendency when using SVM.

An interesting question is about the similarity between the linear discriminative functions learned using the hierarchical and flat learning strategies. Tables 4 and 5 compare the cosine similarities between the weight vectors of the linear discriminative functions using the two strategies for different bins, weighted by the

Table 4
Comparison of the hierarchical and flat learning strategies on the relation subtypes of different training data sizes using PA

Bin type (similarity)	Large bin (0.97)			Middle bin (0.91)			Small bin (0.81)		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Flat	61.9	61.7	61.8	60.5	38.2	46.9	33.2	22.1	26.4
Hierarchical	65.6	60.1	62.7	67.1	42.5	52.1	40.7	27.1	32.5

Notes: The figures in the parentheses indicate the cosine similarities between the weight vectors of the linear discriminative functions learned using the two strategies.

Table 5
Comparison of the hierarchical and flat learning strategies on the relation subtypes of different training data sizes using SVM

Bin type (similarity)	Large bin (0.98)			Middle bin (0.90)			Small bin (0.76)		
	P	*R*	*F*	*P*	*R*	*F*	*P*	*R*	*F*
Flat	68.6	57.3	62.4	67.8	34.7	45.9	35.0	24.7	29.0
Hierarchical	69.4	58.5	63.4	68.2	41.8	51.8	42.3	31.2	35.9

Notes: The figures in the parentheses indicate the cosine similarities between the support vectors sets of the discriminative functions learned using the two strategies.

training data sizes of different relation subtypes using PA and SVM respectively. Table 4 shows that the linear discriminative functions learned using the two strategies are very similar (with the cosine similarity about 0.97) 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 about 0.91/0.81 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 also means that the discriminative functions learnt using PA may be sensitive to the initial weight vector, especially for especially for medium- and least-frequent classes. 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. Table 5 shows similar tendency using SVM. Comparing Tables 4 and 5 shows that, for those subtypes belongs to the large bin, the *F*-measure improvement (63.4-62.4) of SVM with hierarchical learning strategy is slightly better (1.0 vs. 0.9) than that (62.7-61.8) of PA, although the cosine similarity between the discriminative functions learnt from SVM using the hierarchical learning strategy and the flat learning strategy is slightly more than that learnt from PA. This may be due to two reasons: (1) the difference is so little that it may be produced by chance; (2) discriminative functions learnt using different algorithms may scale differently.

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. Fig. 1 shows the effect of different training data sizes for some major relation subtypes while keeping all the training examples of remaining relation subtypes on the ACE RDC 2003 corpus using PA and SVM respectively. 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 and 2004 corpora suffer from the lack of training examples. Among the three major relation subtypes, only the subtype ''Located'' achieves steady performance given the current corpus sizes.

Finally, we also compare our system with previous best-reported systems, such as Zhou et al. (2005), and Zhao and Grisman (2005) on the ACE RDC 2003 and 2004 corpora respectively. For the ACE RDC 2003 corpus, Table 6 shows that our system with the hierarchical learning strategy outperforms the previous best-reported system (Zhou et al., 2005) by 2.4 (57.9 vs. 55.5) and 2.5 (58.0 vs. 55.5) in *F*-measure using PA and SVM (with linear kernel), respectively, in relation extraction of the 24 relation subtypes. Moreover, we also evaluate our system using the polynomial kernel with degree $d = 2$ in SVM. It shows that this can further improve the *F*-measure by 2.2 in the extraction of 24 relation subtypes. As a result, our system significantly outperforms Zhou et al. (2005) by 4.7 in relation extraction of the 24 relation subtypes. For the ACE RDC 2004 corpus, Table 6 shows that our system with the hierarchical learning strategy outperforms the previous best-reported system (Zhao & Grisman, 2005) by 0.4 (70.7 vs. 70.3) and 2.5 (70.8 vs. 70.3) in *F*-measure using PA and SVM (with linear kernel) respectively in relation extraction of the 7 relation types. Moreover, we also evaluate our system using the polynomial kernel with degree $d = 2$ in SVM. It shows that this can further improve the *F*-measure by 2.1 in the extraction of 23 relation subtypes. Meanwhile, our system significantly outperforms Zhao and Grisman (2005) by 2.3 in relation extraction of the 7 relation types.

Table 6
Comparison of our system with other best-reported systems (the figures outside/inside the parentheses are for extraction of the relation types/subtype)

System	2003			2004		
	P	*R*	*F*	*P*	*R*	*F*
Ours: Hierarchical + PA (linear)	63.3 (77.9)	53.4 (60.2)	57.9 (67.9)	69.5 (83.4)	55.9 (61.4)	62.0 (70.7)
Ours: Hierarchical + SVM (linear)	64.3 (82.2)	52.9 (58.2)	58.0 (68.1)	74.2 (82.6)	54.5 (62.0)	62.8 (70.8)
Ours: Hierarchical + SVM (polynomial)	70.8 (82.8)	52.3 (59.3)	60.2 (69.1)	74.8 (83.4)	57.3 (64.1)	64.9 (72.6)
Zhou et al. (2005): Flat + SVM (linear)	63.1 (77.2)	49.5 (60.7)	55.5 (68.0)	– (–)	– (–)	– (–)
Zhao and Grisman (2005): Flat + SVM (composite polynomial)	– (–)	– (–)	– (–)	– (69.2)	– (70.5)	– (70.3)

5. Comparison and discussion

To our best knowledge, our work is the first in semantic relation extraction exploring the hierarchical learning strategy although hierarchical learning has been explored much by previous works, especially in the hierarchical document classification task. Representative works include (Koller & Sahami, 1997; McCallum, Rosenfeld, & Ng, 1998; Weigend, Wiener, & Pederson, 1999; Dumais & Chen, 2000; Vural & Dy, 2004; Dekel, Keshet, & Singer, 2004).

Most previous works on hierarchical classification decouples this problem into independent classification problems by assigning and training a classifier for each class in the class hierarchy. Koller and Sahami (1997) learnt and applied classifiers separately at each class of the hierarchy independently by focusing only on a very small set of critical features among a huge number of available features for each class in the hierarchy to reduce the computational cost and the complexity of each classifier. For an N class problem, Vural and Dy (2004) produced an $N-1$ node binary decision tree where each node represents a decision boundary formed by each of the $N-1$ SVM binary classifiers. Weigend et al. (1999) explored a two-level hierarchical structure present in the semantic space of topics to improve performance in text classification. The proposed architecture matches the hierarchical structure of the topic space with the first level predicting the probabilities of the meta-topic groups and the second level focusing on finer discriminations within the group. Dumais and Chen (2000) worked on a two-level hierarchical structure of web content (In both levels, they used one vs. others strategy to distinguish the sibling classes from each other.) and explored the benefits of using different sets of features at each level of the hierarchy. To accommodate the semantics imposed by the hierarchical structure, some researchers have imposed statistical similarity constraints between the discriminative functions of adjacent classes in the hierarchy (McCallum et al., 1998; Dekel et al., 2004). McCallum et al. (1998) enforced statistical similarities using a traditional technique (i.e. shrinkage).[11] Dekel et al. (2004) explored a large margin approach to hierarchical classification in the spirit of Bayesian approaches. They imposed the similarity constraints between adjacent classes by adjusting the learning rate according to their distance in the tree during the iterative training process when an error occurs. (The closer two classes in the hierarchy, the bigger the learning rate when an error occurs.)

In comparison, this paper proposes a hierarchical learning strategy to model the commonality among related classes. Here, the similarity between the discriminative functions of classes is used to measure their relatedness, based on which a class hierarchy can be built automatically using binary hierarchical clustering in a bottom-up way. It shows that the automatically built class hierarchies work well on the ACE RDC 2003 and 2004 corpora. It also shows that the existing class hierarchies on the ACE RDC 2003 and 2004 corpora well preserve this characteristic and can be well applied in our hierarchical learning strategy. For each class in a class hierarchy, a discriminative function is determined in a top-down way using the simple perceptron algorithm and the state-of-the-art Support Vector Machines. In this way, the upper-level discriminative function can effectively guide the lower-level discriminative function learning. This kind of hierarchical learning strategy is much different from above-mentioned previous works and obviously a new one.

6. Conclusion

This paper proposes a novel hierarchical learning strategy to deal with the data sparseness problem in semantic 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 simple perceptron algorithm and the state-of-the-art Support Vector Machines. In this way, the upper-level discriminative function can effectively guide the lower-level discriminative function learning. Evaluation on the ACE RDC 2003 and 2004 corpora shows that the hierarchical learning strategy performs much better than the flat learning strategy in resolving the critical data sparseness problem in semantic relation extraction.

In the future work, we will explore to justify our hierarchical learning strategy in a theoretical way, e.g. from the statistical learning theory point-of-view. Moreover, we will explore the hierarchical learning strategy

[11] In probabilistic setting, statistical similarities can be also enforced using other traditional techniques such as back-off estimates (Katz, 1987).

using other guidance principle. Finally, just as indicated in Fig. 1, most relation subtypes in the ACE RDC 2003 corpus (arguably the largest annotated corpus in semantic relation extraction) suffer from the lack of training examples. Therefore, a critical research in semantic relation extraction is how to rely on semi-supervised learning approaches (e.g. bootstrapping) to alleviate its dependency on a large amount of annotated training examples and achieve better and steadier performance.

Acknowledgements

This research is supported by Project 60673041 under the National Natural Science Foundation of China and Project 2006AA01Z147 under the "863" National High-Tech Research and Development of China. We would also like to thank the critical and insightful comments from the two anonymous reviewers.

References

ACE (2000–2005). Automatic content extraction. http://www.ldc.upenn.edu/Projects/ACE/.

Bunescu, R., & Mooney, R. J. (2005a). A shortest path dependency kernel for relation extraction. In *HLT/EMNLP'2005, 6–8 October 2005, Vancouver, BC* (pp. 724–731).

Bunescu, R., & Mooney, R. J. (2005b). Subsequence kernels for relation extraction. In *NIPS'2005, Vancouver, BC, December 2005*.

Collins, M. (2002). Discriminative training methods for hidden Markov models: Theory and experiments with perceptron algorithms. *EMNLP'2002* (pp. 1–8).

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

Dekel, O., Keshet, J., & Singer, Y. (2004). Large margin hierarchical classification. In *ICML'2004*.

Dumais, S. T. & Chen, H. (2000). Hierarchical classification of Web content. *SIRIR'2000* (pp. 256–263).

Freund, Y., & Schapire, R. (1990). Large margin classifier using the perceptron algorithm. *Machine Learning, 37*(3), 277–296.

Gao, J. F., Nie, J. Y., Wu, G. G., & Cao, G. H. (2004). Dependence language model for information retrieval. In *SIGIR'2004, Sheffield, UK, 25–29 July 2004*.

Hirschman, L., & Gaizauskas, R. (2001). Natural Language question answering: The view from here. *Natural Language Engineering, 7*(4), 275–300.

Joachims, T. (1998). Text categorization with support vector machines: Learning with many relevant features. In *ECML'1998. 21–23 April 1998*. Chemnitz, Germany.

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

Katz, S. (1987). Estimation of probabilities from sparse data for the language model component of a speech recognizer. *IEEE Transaction on Acoustics, Speech and Signal Processing, 35*, 400–401.

Koller, D., & Sahami, M. (1997). Hierarchically classifying documents using very few words. In *ICML'1997* (pp. 171–178).

McCallum, A. K., Rosenfeld, R. Mitchell, T. M. & Ng, A. Y. (1998). Improving text classification by shrinkage in a hierarchy of classes. In *ICML'1998* (pp. 359–367).

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

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

Platt, J. (1999). Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In J. Smola, P. Bartlett, B. Scholkopf, & D. Schuurmans (Eds.), *Advances in large margin classifiers*. MIT Press.

Strzalkowski, T., & Carballo, J. P. (1998). Natural language information retrieval: TREC-5 Report. In *5th Text retrieval conference*.

Vapnik, V. (1995). *The nature of statistical learning theory*. NY, USA: Springer-Verlag.

Vural, V., & Dy, J. G. (2004). A hierarchical method for multi-class support vector machines. In *ICML'2004* (pp. 831–838).

Weigend, A. S., Wiener, E. D., & Pederson, J. O. (1999). Exploiting hierarchy in text categorization. *Information Retrieval, 1*(3), 192–216.

Zelenko, D., Aone, C., & 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. & Tan, C. L. (2005). Discovering relations from a large raw corpus using tree similarity-based clustering. In *IJCNLP'2005, 11–16 October 2005, Jeju Island, South Korea. Lecture Notes in Computer Science, LNCS* (Vol. 3651, pp. 378–389).

Zhang, M., Su, J., Zhang, J., & Zhou, G. D. (2006). A composite kernel to extract relations with both flat and structured features. In *COLING-ACL'2006, Sydney, Australia* (pp. 825–832).

Zhao, S. B., & Grisman, R. (2005). Extracting relations with integrated information using kernel methods. In *ACL'2005, University of Michgan-Ann Arbor, USA, 25–30 June 2005* (pp. 419–426).

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

Zhou, G. D., Su, J., & Zhang, M. (2006). Modeling commonality among related classes in relation extraction. In *COLING-ACL'2006, Sydney, Australia* (pp. 121–128).

Zhou, G. D., Zhang, M., & Fu, G. H. (2006). Hierarchical learning strategy in relation extraction using support vector machines. in *The third Asian information retrieval symposium (AIRS'2006), 16–18 October 2006*. *Lecture Notes in Computer Science, LNCS* (Vol. 4182, pp. 67–78).