Automatic Extraction of Hierarchical Relations from Text

Ting Wang[1,2], Yaoyong Li[1], Kalina Bontcheva[1],
Hamish Cunningham[1], and Ji Wang[2]

[1] Department of Computer Science, University of Sheffield, Sheffield, S1 4DP, UK
{T.Wang, Y.Li, K.Bontcheva, H.Cunningham}@dcs.shef.ac.uk
[2] Department of Computer, National University of Defense Technology, Changsha, Hunan, 410073, P.R. China
{tingwang, jiwang}@nudt.edu.cn

Abstract. Automatic extraction of semantic relationships between entity instances in an ontology is useful for attaching richer semantic metadata to documents. In this paper we propose an SVM based approach to hierarchical relation extraction, using features derived automatically from a number of GATE-based open-source language processing tools. In comparison to the previous works, we use several new features including part of speech tag, entity subtype, entity class, entity role, semantic representation of sentence and WordNet synonym set. The impact of the features on the performance is investigated, as is the impact of the relation classification hierarchy. The results show there is a trade-off among these factors for relation extraction and the features containing more information such as semantic ones can improve the performance of the ontological relation extraction task.

1 Introduction

Information Extraction (IE) [4] is a process which takes unseen texts as input and produces fixed-format, unambiguous data as output. It involves processing text to identify selected information, such as particular named entity or relations among them from text documents. Named entities include people, organizations, locations and so on, while relations typically include physical relations (located, near, part-whole, etc.), personal or social relations(business, family, etc.), and membership (employ-staff, member-of-group, etc.).

Until recently, research has focused primarily on use of IE for populating ontologies with concept instances (e.g. [9, 16]). However, in addition to this, many ontology-based applications require methods for automatic discovery of properties and relations between instances. Semantic relations provide richer metadata connecting documents to ontologies and enable more sophisticated semantic search and knowledge access.

One barrier to applying relation extraction in ontology-based applications comes from the difficulty of adapting the system to new domains. In order to overcome this problem, recent research has advocated the use of Machine Learning (ML) techniques for IE. A number of ML approaches have been used for

Y. Sure and J. Domingue (Eds.): ESWC 2006, LNCS 4011, pp. 215–229, 2006.

relation extraction, e.g. Hidden Markov Models (HMM) [7], Conditional Random Fields (CRF) [12], Maximum Entropy Models (MEM) [11]. and Support Vector Machines (SVM) [19]. The experimental results in [19] showed that the SVM outperformed the MEM on the ACE2003 relation extraction data[1].

Zelenko et al [18] proposed extracting relations by computing kernel functions between shallow parse trees. Kernels have been defined over shallow parse representations of text and have been used in conjunction with SVM learning algorithms for extracting person-affiliation and organization-location relations. Culotta et al [5] extended this work to estimate kernel functions between augmented dependency trees.

Zhou et al [19] further introduced diverse lexical, syntactic and semantic knowledge in feature-based relation extraction using SVM. The feature system covers word, entity type, overlap, base phrase chunking, dependency tree and parse tree, together with relation-specific semantic resources, such as country name list, personal relative trigger word list. Their results show that the feature-based approach outperforms tree kernel-based approaches, achieving 55.5% F-measure in relation detection and classification on the ACE2003 training data.

Motivated by the above work, we use the SVM as well and apply a diverse set of Natural Language Processing (NLP) tools to derive features for relation extraction. In particular, several new features are introduced, such as part-of-speech (POS) tags, entity subtype, entity class, entity role, semantic representation of sentences and WordNet synonym set.

In the rest of the paper, we first describe the ACE2004 entity and relation type hierarchy from an ontological perspective (Section 2). Then we give a brief introduction of SVM used as the classifier for relation extraction (Section 3) and explain an extensive set of features used in our experiments (Section 4). Section 5 presents and discusses a series of experiments that investigate the impact of the different features and the classification hierarchy. Finally, we summarise the work and discuss some future directions.

2 The ACE Entity and Relation Hierarchies

Relation extraction from text aims to detect and classify semantic relations between entities according to a predefined entity and relation type system or an ontology. The Automatic Content Extraction (ACE) programme [1] defines this task as Relation Detection and Characterization (RDC). RDC uses the results of named entity recognition, which detects and classifies entities according to a predefined entity type system.

[1] SVM has achieved state of the art results in many NLP tasks such as text classification, part of speech tagging and information extraction. This is mainly because, on the one hand, the SVM is an optimal classifier with maximal margin in feature space; on the other hand, NLP tasks typically represent instances by very high dimensional but very sparse feature vectors, resulting in positive and negative examples being distributed into two distinctly different areas of the feature space. As we use a very high dimensional and very sparse feature vector for relation extraction, it can be expected that SVM will have similarly good performance.

In contrast to earlier ACE evaluations, ACE2004 introduced a type and subtype hierarchy for both entity and relations, an important step towards ontology-based IE. Hence, we evaluate our method on the corpus for learning relation hierarchy

2.1 The ACE2004 Entity Hierarchy

Entities are categorized in a two level hierarchy, consisting of 7 types and 44 subtypes. The entity type includes Person, Organisation, Facility, Location and Geo-political Entity (GPE). Subtype refers to sub-concept of entity concept. For example, The entity type Organisation was divided into the subtypes such as Government, Commercial and Educational. For details see [2].

Each entity has been assigned a class which describes the kind of reference the entity makes to something in the world. The class can be one of four values: Negatively Quantified(NEG), Specific Referential(SPC), Generic Referential(GEN), Under-specified Referential(USP). The occurrence of each entity in the dataset is called an entity mention, which can be one of the following: Names(NAM) , Quantified Nominal Constructions(NOM), Pronouns(PRO), Pre-modifier(PRE).

In addition, GPEs are regarded as composite entities comprising of population, government, physical location, and nation (or province, state, county, city, etc.). Consequently, each GPE mention in the text has a mention role which indicates which of these four aspects is being referred to in the given context: of that mention invokes: Person(PER), Organization(ORG), Location(LOC), and GPE.

2.2 The ACE2004 Relation Hierarchy

In an ontology, the concepts are not only organised in a taxonomy representing IS-A relations, but also linked together by semantic relations such as Part-Whole, Subsidiary, LocatedIn, etc. ACE2004 defines a hierarchy of relations with 7 top types and 22 sub-types, shown in Table 1 [3]. There are 6 symmetric relations (marked with star in table) and the remaining ones are asymmetric relations.

Table 1. ACE2004 relation types and subtypes

Type	Subtype
Physical (PHYS)	Located, Near*, Part-Whole
Personal/Social (PER-SOC)	Business*, Family*, Other*
Employment/Membership/ Subsidiary (EMP-ORG)	Employ-Exec, Employ-Staff, Employ-Undetermined, Member-of-Group, Subsidiary, Partner*, Other*
Agent-Artifact (ART)	User/Owner, Inventor/Manufacturer, Other
PER/ORG Affiliation (OTHER-AFF)	Ethnic, Ideology, Other
GPE Affiliation (GPE-AFF)	Citizen/Resident, Based-In, Other
Discourse (DISC)	(none)

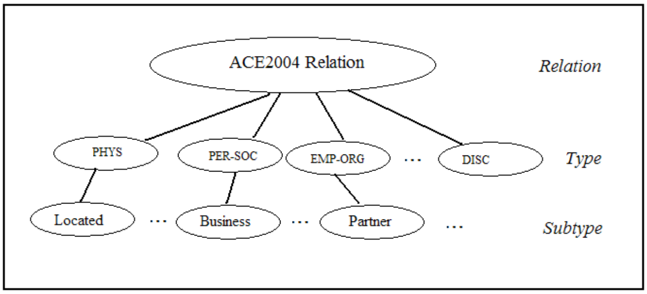

Fig. 1. The hierarchy of the ACE2004 relation types and subtypes

This relation type and subtype hierarchy can also be described as a three levels tree (see Fig 1). In the experiments reported next, we use the ACE2004 corpus to evaluate ontological relation extraction.

3 Using SVM for Relation Extraction

SVM is one of the most successful ML methods, which has achieved the state-of-the-art performances for many classification problems. For example, our experiments in [13] showed that the SVM obtained top results on several IE benchmarking corpora.

As SVM were originally designed for binary classification and relation extraction can be reduced into a multi-class classification problem, we have to extend the SVM for multi-class classification. There exists two approaches to use the SVM for multi-class problem [10]: (i) constructing and combining several SVM binary classifiers via either the one-against-all method or the one-against-one method; (ii) learning a multi-class SVM classifier directly for the multi-class problem. The comparison in [10] shows that one-against-one method is the best in both training time and performance, and suggests that one-against-one method may be more suitable for practical use on large problems than other approaches. Therefore we used the one-against-one method in the experiments (see Section 5.1 for more details).

For a k-class classification task, the one-against-one method constructs $k(k-1)/2$ classifiers where each one is trained on data from two classes, while the one-against-all method learns k classifier each of which is used to separate one class from all others. Although one-against-one method has to train more classifiers than one-against-all does (on k classifiers), each training data is much smaller, resulting into less total training time than the one-against-all method. We used the *Max Wins* voting strategy to predicate the class: apply every classifier to instance $\mathbf{x}$; if one classifier says $\mathbf{x}$ is in the i-th class, then the vote for the i-th class is incremented by one; in the end $\mathbf{x}$ is classified as the class with the largest number of votes.

We built SVM models for detecting the relations, predicting the type and subtype of relations between every pair of entity mentions within the same sentence. As defined in the ACE evaluation, we only model explicit relations rather than implicit ones. For example, the sentence

$$\textit{Texas has many cars.} \tag{1}$$

explicitly expresses a ART.User/Owner relation between the two entity mentions Texas and many cars. What we need to do is to detect the relation and its type and subtype based on the context information within this sentence. Such context information is usually expressed as a vector consisting of values for some specific attributes, which is called features. Choosing the right features is key to successful application of ML technology.

4 Features for Relation Extraction

Using NLP to derive ML features has been shown to benefit IE results [13]. Features which have been used for relation extractions include word, entity type, mention level, overlap, chunks, syntactic parse trees, and dependency relations [7, 11, 18, 19].

Based on the previous works, we developed a set of features for semantic relation extraction, many of which are adopted from [19]. Moreover, we introduce some new features such as POS tags, entity subtype and class features, entity mention role feature, and several general semantic features. Zhou et al in [19] have designed some relation-specific semantic features, for example, some important trigger words list have been collected from WordNet [15] in order to differentiate the six personal social relation subtypes. However, these lists are too specific to the dataset to be applicable for general purpose relation extraction. Therefore in our method, we introduce instead a set of more general semantic features produced by a semantic analyser and WordNet.

BuChart (which has been renamed to SUPPLE) is a bottom-up parser that constructs syntax trees and logical forms for English sentences [8]. One of its significant characteristics is that it can produce a semantic representation of sentences - called simplified quasilogical form (SQLF). Previously, one of the limitations in applying general semantic information in IE is the relative lack of robustness of semantic analyser. However, BuChart is a general purpose parser that can still produce partial syntactic and semantic results for fragments even when the full sentential parses cannot be determined. This makes it applicable for deriving semantic features for ML-based extraction of semantic relations from large volumes of real text.

WordNet [15] is a widely used linguistic resource which is designed according to psycholinguistic theories of human lexical memory. English nouns, verbs, adjectives and adverbs are organized into synonym sets (called synsets), each representing one underlying lexical concept. In this work, WordNet is used to derive several semantic features based on the synset and hypernym information.

4.1 Using GATE for Feature Extraction

General Architecture for Text Engineering (GATE) [6] is an infrastructure for developing and deploying software components that process human language. It provides or includes from other people a set of NLP tools including tokeniser, gazetteer, POS tagger, chunker, parsers, etc. For the relation extraction task, we make use of a number of GATE components as follows: English Tokeniser, Sentence Splitter, POS Tagger, NP Chunker, VP Chunker, BuChart Parser, MiniPar Parser. To develop more semantic features we also made use of WordNet and derived the synset information as the features.

4.2 Developing Features

In the experiments we tried to use as many NLP features as could be provided by GATE components and which were considered as potentially helpful for modeling the relation extraction task. This is a valid approach due to one advantage of the SVM learning algorithm. Namely, carefully choosing features is crucial for some learning algorithms such as decision trees and rule learning. However, it is no so important for the SVM, because the irrelevant features for one particular binary problem usually distribute evenly over the positive and negative training examples and therefore would have little contribution to the SVM model due to its learning mechanism. Hence, when using SVM, we can put into the feature vector as many features as possible and let the SVM algorithm determine the most useful ones for a given binary classification problem.

In Section 5.3 we will experimentally discuss the contributions of these features to the relation extraction task.

The total feature set consists of 94 features. The rest of the subsection provide an overview of the different types of features we used. Due to space limitations, a complete description is provided in a separate technical report [17].

Word Features. This set consists of 14 features including the word list of the two entity mentions and their heads, the two words before the first mention, the two after the second mention, and the word list between them.

POS Tag Features. Because the word features are often too sparse, we also introduce POS tag features. For example, sentence 1 has been tagged as: *Texas* /NNP *has*/VBZ *many*/JJ *cars*/NNS, where NNP denotes proper name, JJ - adjectives, NNS - plural nouns, etc. Similar to the word features, this set of features includes the POS tag list of the two entity mentions and their heads, the two POS tags before the first mention, the two after the second mention, and the tag list in between.

Entity Features. As already discussed, ACE2004 divides entities into 7 types and all the entity mentions has also been annotated with the entity type , subtype and class, all of which have been used to develop features. For each pair of mentions, the combination of their entity types is taken as the entity type feature. For the example sentence above, there are two entity mentions: *Texas* categorized as GPE, and *many cars* as WEH. In addition, the entity hierarchy is used, because subtypes carry more accurate semantic information for the entity mentions. Therefore, the combination of the entity subtypes of the two entity mentions is provided as the entity subtype feature. The subtypes of the two example mentions are State-or-Province and Land. Finally, each entity has also been annotated with a class which describes the kind of reference for the entity. So the entity class is also used in this paper to predicate semantic relations. The classes for the above two mentions are SPC and USP.

Mention Features. This set of features includes the mention type and role information of both mentions, which is also provided by the ACE2004 annotations. For the example sentence, the mention types for the two mentions in sentence (1) are NAM and NOM, while the mention role for Texas is GPE and there is no role information for the second because only GPE entity can take role information.

Overlap Features. The relative position of the two entity mentions can also be helpful for indicating the relationship between them. For these features, we have considered: the number of words separating them, the number of other entity mentions in between, whether one mention contains the other. As the feature indicating whether one mention contains the other is too general, it has been combined with the entity type and subtype of the two mentions to form more discriminating features.

Chunk Features. GATE integrates two chunk parsers: Noun Phrase (NP) and Verb Phrase (VP) Chunker that segment sentences into noun and verb group chunks. For instance, the example sentence (1) is chunked as: [Texas] has [many cars], in which, Texas and many cars are NPs, while has is the VP between them whose type and voice are FVG (means finite verb phrase) and active. The following information has been used as chunk features: whether the two entity mentions are included in the same NP Chunk or VP Chunk, the type and voice information of the verb group chunk in between if there is any.

Dependency Features. In contrast to Kambhatla [11] and Zhou et al [19], who derive the dependency tree from the syntactic parse tree, we apply MiniPar to directly build the dependency tree. MiniPar is a shallow parser which can determine the dependency relationships between the words of a sentence [14]. Fig 2 shows the dependency tree for the example sentence. From the resulting dependency relationships between words, the dependency features are formed, including: combination of the head words and their dependent words for the two entity mentions involved; the combination of the dependency relation type and the dependent word of the heads of the two mentions; the combination of the entity type and the dependent word for each entity mention's head; the name of the dependency relationship between the heads of the two mentions if there is any; the word on which both the heads of the two mentions depend on if there is any; and the path of dependency relationship labels connecting the heads of the two mentions.

Parse Tree Features. The features on syntactic level are extracted from the parse tree. As we mentioned above, we use BuChart to generate the parse tree and the semantic representation of each sentence. Unlike many full parsers which

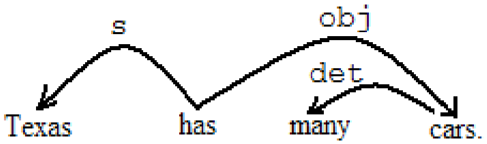

Fig. 2. The dependency tree for the example sentence

would fail if a full sentential parse cannot be found, BuChart can still produce the partial parsing trees and correspondent semantic representations for the fragments. The following list the parse tree in the bracket form for the example sentence,

(s (np (bnp (bnp_core (bnp_head (ne_np (tagged_location_np (list_np "Texas")))))))) (fvp (vp (vpcore (fvpcore (nonmodal_vpcore (nonmodal_vpcore1 (vpcore1 (av (v "has")))))))) (np (bnp (bnp_core (premods (premod (jj "many"))) (bnp_head (n "cars"))))))))

Consequently, from the product of the parser, we extract the following features: the lowest and second lowest phrase labels governing each entity mentions involved; the lowest phrase labels governing both entity mentions; the lowest phrase labels governing the heads of both entity mentions; the path of phrase labels connecting both mentions in the parse tree; and that connecting the heads of both mentions.

Semantic Features from SQLF. Using relation- or domain- independent semantic features potentially makes the approach easier to adapt to new domains. BuChart provides semantic analysis to produce SQLF for each phrasal constituent. The logical form is composed of unary predicates that denote entities and events (e.g., chase(e1), run(e2)) and binary predicates for properties (e.g. lsubj(e1,e2)). Constants (e.g., e1, e2) are used to represent entity and event identifiers (see [8] for further details). The (somewhat simplified) semantic analysis of the example sentence in SQLF is

location(e2), name(e2,'Texas'), have(e1), time(e1,present), aspect(e1,simple), voice(e1,active), lobj(e1,e3), car(e3), number(e3,plural), adj(e3,many), lsubj (e1,e2)

From the SQLFs, a set of semantic features is generated, one of which is the path of predicate labels connecting the heads of both mentions in the semantic SQLFs. This path may be too specific to be effective and cause data sparseness problem, so we also take some important predicate labels as separate features, such as the first, second, last and penultimate predicates labels in that path.

Semantic Features from WordNet. To exploit more relation-independent semantic features, we use WordNet together with a simple semantic tagging method to find the sense information for the words in each sentence. Tagging words with their corresponding WordNet synsets (e.g. word sense disambiguation - WSD) is a difficult task, which usually can not achieve accuracy as high as other NLP tasks such as POS tagging. However, WordNet's design ensures that synsets are ordered by importance, so a simple and yet efficient heuristic can be used instead of a WSD module, without major accuracy penalty. The heuristic is to take the first synset from WordNet, which matches the POS tag of the given word. Each synset has been assigned an id (consisting of the POS tag and its offset in the WordNet files) which is used in the features. Similar to the word and POS tag features, the features from WordNet include the synset-id list of the two entity mentions and their heads, the two synset-ids before the first mention, the two after the second mention, and the synset-id list in between. With considerations of the data sparseness problem, we also developed a set of more abstract features by using the hypernym information of each synset, which exactly parallel the synset ones by replacing each synset-id with the id of its hypernym synset.

5 Experiment Results and Analysis

We evaluate our method, especially the contribution of the different features, on the ACE2004 training data. As mentioned above, only explicit relations between pairs of entity mentions within the same sentence are considered. We not only evaluate the performance of the system as a whole, but also investigate in detail several factors which have impact on the performance, such as the features set and the relation classification hierarchy.

5.1 Experimental Settings

The ACE2004 training data consists of 451 annotated files (157,953 words) from broadcast, newswire, English translations of Arabic and Chinese Treebank, and Fisher Telephone Speech collection. Among these files, there are 5,914 relation instances annotated which satisfied the experiment set up described above. The distribution of the instances is listed in Table 2.

Table 2. The distributions of the relation instances in ACE 2004 training data

Type	Subtype	Number	Type	Subtype	Number
PHYS	Located	1029	OTHER	Ethnic	53
	Near	141	-AFF	Ideology	55
	Part-Whole	518		Other	75
PER	Business	197	GPE	Citizen/Resident	368
-SOC	Family	178	-AFF	Based-In	333
	Other	69		Other	87
EMP	Employ-Exec	630	ART	User/Owner	273
-ORG	Employ-Undetermined	129		Inventor/Manufacturer	13
	Employ-Staff	694		Other	7
	Member-of-Group	225	DISC		434
	Subsidiary	300			
	Partner	16			
	Other	90	Total		5,914

Following previous work, in order to focus on the performance of the relation extraction only, we suppose that all named entity mentions have been recognised without mistakes and only evaluate the performance of relation extraction on "true" named entity mentions with "true" chaining (i.e. as annotated by the ACE2004 annotators).

Among the 23 relation subtypes (including DISC which has no subtype), there are 6 symmetric ones. So to model the relation extraction task as multi-class classification, we use two labels to denote each non-symmetric relation and only one label for each symmetric one. Also we assign a label to the class of no-relation, which indicates that there is no relation between the two entity mentions. Consequently, in our experiments, relation extraction is modeled as a 41-class classification task, where each pair of entity mentions is assigned one of these 41 relation classes, based on the features discussed in Section 4. In the following experiments, we use the package LIBSVM[2] for training and testing the SVM classifiers with one-against-one method for multi-class classification which has been described in Section 3.

From Table 2 we can see that the different relation subtypes and types are distributed very unevenly, so we only measure the micro-average of Precision, Recall and F1 measure(which is $2 * Precision * Recall/(Precision + Recall)$), because in such cases macro-average does not reflect the performance reliably.

In each of the following experiments, we performed 5 folds cross validation on the whole data. In every execution, the corpus is spited into a training set (80% of the total files) and a testing set (20% of the total files). All the performance results of Precision, Recall and F1 reported are the means averaged over five runs.

Table 3. The result on different kernel functions

Kernel type	Precision(%)	Recall(%)	F1(%)
Linear	66.41	49.18	56.50
Quadratic	68.96	46.20	55.33
Cubic	71.31	42.39	53.17

5.2 Evaluation on Different Kernels

Since different kernel functions can be used with SVM, in order to select a suitable kernel for relation extraction task, we have implemented experiments to compare three different types of kernels for SVM: the linear, quadratic and cubic kernels. In the comparison, all the features have been used and Table 3 shows the result. The result shows that with all the features the linear kernel is better than both the quadratic and cubic ones. But the t paired test we did using the results from 5-fold cross validation showed that the linear kernel was no significantly better than quadratic kernel at the 95% confidence level (the p-value of the test was 0.088). Since the linear kernel is more simple and efficient than others and obtained better results, in the following experiments we only use the linear kernel with SVM.

[2] See http://www.csie.ntu.edu.tw/~cjlin/libsvm/.

5.3 Evaluation on Features

We carried out the experiments to investigate the impact of different features on the performance by adding them incrementally. The features have been added in the order from the shallow to deep to see the effect of different features. Table 4 presents the results from the experiments. It is possible that not only the individual features but also the combinations of features would affect the performance. Hence, more experiments will be done in the future to figure out the effects of the features on relation extraction. It can be seen from the table, the performance improves as more features are used, until the F1 measure reaches 56.78% which is comparable to the reported best results(55.5%) of [19]on the ACE2003 training data. From the new features introduced in this work, the POS tag features and the general semantic features all contribute to the improvement. The improvement of the general semantic features, including semantic features from SQLF and WordNet, is significant at confidence level 95%, as the p-value of the t paired test on the corresponding data is 0.003. The improvement 1.72% (from 55.06% to 56.78%) was even higher than that brought by some syntactic features such as the chunk, dependency and parse tree features which was only 1.54%(from 53.52% to 55.06%) in total. Therefore, the contribution of the semantic features shows that general semantic information is beneficial for relation extraction and should receive further attention.

The entity features lead to the best improvement in performance. It is not surprised because the relation between two entity mentions is closely related to the entity types of the two mentions. Actually we took advantage of the ACE2004 corpus which included not only the entity types but also entity subtype and class. Therefore we used more entity type features than the previous studies using the ACE2003 corpus which only had entity type. Further investigation shows that the two additional features are in fact very helpful: when only using entity type feature the F1 improvement is 10.29%, and when using the other two features additionally, the improvement increase to 13.83%. This result shows that the more accurate information of the entity mentions we have, the better performance can be achieved in relation extraction.

Table 4. The result on different feature sets

Features	Precision(%)	Recall(%)	F1(%)
Word	57.90	23.48	33.38
+POS Tag	57.63	26.91	36.66
+Entity	60.03	44.47	50.49
+Mention	61.03	45.60	52.03
+Overlap	60.51	48.01	53.52
+Chunk	61.46	48.46	54.19
+Dependency	63.07	48.26	54.67
+Parse Tree	63.57	48.58	55.06
+SQLF	63.74	48.92	55.34
+ WordNet	67.53	48.98	56.78

From Table 4, we can also see that the impact of the deep features is not as significant as the shallow ones. Zhou et al [19] show that chunking features are very useful while the dependency tree and parse tree features do not contribute much. Our results even show that features from word, POS tag, entity, mention and overlap can achieve 53.52% F1, while the deeper features (including chunk, dependency tree, parse tree and SQLF) only give less than 2% improvement over simpler processing. As the number of features impacts directly the required size of training data and the efficiency of training and application (more features need more annotated data for training the model and need more computation resources), there is an interesting trade-off in feature selection for relation extraction.

5.4 Experiments on Hierarchical Classification

As already discussed, ACE2004 defined both an entities and relations hierarchy, which provides a data resource for evaluating our method for ontology-based IE. The significant contribution of entity subtype and class features demonstrated above shows that the entity hierarchy information is important for relation extraction. As shown in Fig 1 the relation hierarchy has three levels, so we ran experiments to evaluate our method with these different classification levels: subtype classification – 23 relations at leaf level, type classification – 7 relations at middle level, relation detection – predicating if there is relation between two entity mentions, which can be treated as a binary classification task. The experiments on the three different classification levels have been done separately. In each experiment, the classifier is trained and tested on the corresponding relation labels (e.g. 23, 7, or 1 relations). All these experiments made use of the complete feature set and Table 5 shows the averaged overall results.

Table 5. The result on different classification levels

Level	Precision(%)	Recall(%)	F1(%)
Subtype classification	67.53	48.98	56.78
Type classification	71.41	60.03	65.20
Relation detection	73.87	69.50	71.59

The results show that performance on relation detection level is the highest while that on subtype classification is the lowest. The Precision, Recall and F1 all show the same trend, revealing that it is more difficult to classify on deeper levels of the hierarchy because there are less examples per class and also the classes are getting more similar as the classification level gets deeper. This has been supported by the more detailed results for the relations type EMP-ORG and its subtypes, as shown in Table 6. The performance for the type EMP-ORG when classifying on the type level is the best among all 7 relation types: 77.29% Precision, 75.00% Recall and 76.01% F1 averaged over 5 folds cross validation. However, the performance on the 7 subtypes within EMP-ORG when classifying at subtype level is not only much lower than the result for EMP-ORG overall

Table 6. The result on the subtypes of EMP-ORG

Subtypes	Num	Precision(%)	Recall(%)	F1(%)
Employ-Exec	630	71.37	63.9	67.16
Employ-Undetermined	129	68.76	43.23	51.2
Employ-Staff	694	64.39	60.97	62.25
Member-of-Group	225	62.16	38.55	46.85
Subsidiary	300	83.81	65.29	72.79
Partner	16	0	0	0
Other	90	33.33	5.89	9.9

Table 7. The F1(%) results on different feature sets and classification levels

Features	Relation detection	Type classification	Subtype classification
Word	61.25	41.61	33.38
+POS Tag	60.03	44.13	36.66
+Entity	63.31	57.84	50.49
+Mention	65.57	59.45	52.03
+Overlap	66.84	61.36	53.52
+Chunk	66.61	62.52	54.19
+Dependency	70.01	63.61	54.67
+Parse Tree	70.42	64.05	55.06
+SQLF	70.08	64.24	55.34
+WordNet	71.59	65.20	56.78

but also rather unstable: from zero for Partner to 72.79% for Subsidiary. The two biggest subtypes Employ-Exec and Employ-Staff get only 67.16% and 62.25 % F1 which are much lower than the 76.01% on type level for their parent type EMP-ORG. We consider that the zero result for Partner is mainly due to too few instances. Therefore, the closer distance between the classes at subtype level causes the performance to decrease and become unstable.

We also investigated the influence of different feature sets on the different classification levels (see Table 7). For all of the three classification levels, the improvements are almost stable as more features are introduced and the best performance is achieved with the complete feature set (there is only one exception when add SQLF features in relation detection). But the improvement at various levels is different: as more features are used, the improvement in relation detection is only 10.34% (from 61.25% to 71.59%), while the improvement in type and subtype classification is much more significant: 23.59% (from 41.61% to 65.20%) and 23.40% (from 33.38% to 56.78%). Such difference suggests that features provide more significant effect for classification on deep level. Furthermore, the impact of the SQLF and WordNet synset features on different levels also shows that semantic knowledge will play more important role in extracting fine granularity relations.

6 Conclusions

In this paper we investigated SVM-based classification for relation extraction and explored a diverse set of NLP features. In comparison to previous work, we introduce some new features, including POS tag, entity subtype and class features, entity mention role features and even general semantic features which all contribute to performance improvements. We also investigated the impact of different types of feature and different relation levels.

Further work on using machine learning for relation extraction needs to address several issues. Firstly, although the ACE2004 entity and relation type system provides a hierarchy organization which is somewhat like ontology, it is still very limited for large-scale ontology-based IE. We plan to extend our method and evaluate it on bigger scale ontology. Another interesting future work is to integrate the automatic named entity recognition with relation extraction, which would be more realistic than the experiments described in this paper where relation extraction was based on the gold standard named entities. It would also be interesting to compare the SVM model with other variants of the SVM (such as the SVM with uneven margins) as well as with other ML approaches (such as CRF, MEM and so on) for relation extraction.

Acknowledgements. This research is supported by the EU-funded SEKT project (www.sekt-project.com) , the National Natural Science Foundation of China (60403050) and the National Grand Fundamental Research Program of China under Grant No. 2005CB321802. Thanks to Guodong Zhou for his helpful information about his work.

References

1. ACE. See http://www.nist.gov/speech/tests/ace/
2. Annotation Guidelines for Entity Detection and Tracking (EDT) Version 4.2.6, http://www.ldc.upenn.edu/Projects/ACE/docs/EnglishEDTV4-2-6.PDF. (2004)
3. Annotation Guidelines for Relation Detection and Characterization (RDC) Version 4.3.2, http://www.ldc.upenn.edu/Projects/ACE/docs/EnglishRDCV4-3-2.PDF. (2004)
4. Appelt, D.: An Introduction to Information Extraction. Artificial Intelligence Communications, **12(3)** (1999) 161-172
5. Culotta, A., Sorensen, J.: Dependency tree kernels for relation extraction. Proceedings of 42th Annual Meeting of the Association for Computational Linguistics. 21-26 July Barcelona, Spain (2004)
6. Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V.: GATE: A Framework and Graphical Development Environment for Robust NLP Tools and Applications. Proceedings of the 40th Anniversary Meeting of the Association for Computational Linguistics. Philadelphia, July (2002)
7. Freitag, D., and McCallum A.: Information extraction with HMM structures learned by stochastic optimization. Proceedings of the 7th Conference on Artificial Intelligence (AAAI-00) and of the12th Conference on Innovative Applications of Artificial Intelligence (IAAI-00), 584-589,Menlo Park, CA. AAAI Press (2000)

8. Gaizauskas, R., Hepple, M., Saggion, H., Greenwood, M.A., Humphreys, K.: SUPPLE: A Practical Parser for Natural Language Engineering Applications. Technical report CS–05–08, Department of Computer Science, University of Sheffield (2005)
9. Handschuh, S., Staab, S., Ciravegna, F.: S-CREAM — Semi-automatic CREAtion of Metadata. Proceedings of the13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Siguenza, Spain(2002)
10. Hsu, C.-W., Lin, C.-J.: A comparison of methods for multi-class support vector machines , IEEE Transactions on Neural Networks, 13(2). (2002)415-425
11. Kambhatla, N.: Combining lexical, syntactic and semantic features with Maximum Entropy models for extracting relations. Proceedings of 42th Annual Meeting of the Association for Computational Linguistic. 21-26 July Barcelona, Spain (2004)
12. Lafferty, J., McCallum, A., Pereira. F.: Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In Proc. 18th International Conf. on Machine Learning, Morgan Kaufmann, San Francisco, CA (2001) 282-289
13. Li, Y., Bontcheva, K., Cunningham, H.: SVM Based Learning System For Information Extraction. In Proceedings of Sheffield Machine Learning Workshop, Lecture Notes in Computer Science. Springer Verlag (2005)
14. Lin, D.: Dependency-based Evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain, May (1998)
15. Miller, A., ”WordNet: An On-line Lexical Resource”, Special issue of the Journal of Lexicography, vol. 3, no. 4(1990)
16. Motta, E., VargasVera, M., Domingue, J., Lanzoni, M., Stutt, A., Ciravegna, F.: MnM: Ontology Driven Semi-Automatic and Automatic Support for Semantic Markup. Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management (EKAW02), Siguenza, Spain(2002)
17. Wang, T., Bontcheva, K., Li, Y., Cunningham, H.: D2.1.2. Ontology-Based Information Extraction. SEKT Deliverable D2.1.2. (2005). http://www.sekt-project.org/rd/deliverables/index_html/
18. Zelenko, D., Aone, C., Richardella, A.: Kernel methods for relation extraction. Journal of Machine Learning Research (2003) 1083-1106
19. Zhou G., Su, J., Zhang, J., Zhang, M.: Combining Various Knowledge in Relation Extraction, Proceedings of the 43th Annual Meeting of the Association for Computational Linguistics (2005)