Semi-supervised Relation Extraction with Large-scale Word Clustering

Ang Sun **Ralph Grishman** **Satoshi Sekine**

Computer Science Department
New York University
{asun,grishman,sekine}@cs.nyu.edu

Abstract

We present a simple semi-supervised relation extraction system with large-scale word clustering. We focus on systematically exploring the effectiveness of different cluster-based features. We also propose several statistical methods for selecting clusters at an appropriate level of granularity. When training on different sizes of data, our semi-supervised approach consistently outperformed a state-of-the-art supervised baseline system.

1 Introduction

Relation extraction is an important information extraction task in natural language processing (NLP), with many practical applications. The goal of relation extraction is to detect and characterize semantic relations between pairs of entities in text. For example, a relation extraction system needs to be able to extract an *Employment* relation between the entities *US soldier* and *US* in the phrase *US soldier*.

Current supervised approaches for tackling this problem, in general, fall into two categories: feature based and kernel based. Given an entity pair and a sentence containing the pair, both approaches usually start with multiple level analyses of the sentence such as tokenization, partial or full syntactic parsing, and dependency parsing. Then the feature based method explicitly extracts a variety of lexical, syntactic and semantic features for statistical learning, either generative or discriminative (Miller et al., 2000; Kambhatla, 2004; Boschee et al., 2005; Grishman et al., 2005; Zhou et al., 2005; Jiang and Zhai, 2007). In contrast, the kernel based method does not explicitly extract features; it designs kernel functions over the structured sentence representations (sequence, dependency or parse tree) to capture the similarities between different relation instances (Zelenko et al., 2003; Bunescu and Mooney, 2005a; Bunescu and Mooney, 2005b; Zhao and Grishman, 2005; Zhang et al., 2006; Zhou et al., 2007; Qian et al., 2008). Both lines of work depend on effective features, either explicitly or implicitly.

The performance of a supervised relation extraction system is usually degraded by the sparsity of lexical features. For example, unless the example *US soldier* has previously been seen in the training data, it would be difficult for both the feature based and the kernel based systems to detect whether there is an *Employment* relation or not. Because the syntactic feature of the phrase *US soldier* is simply a noun-noun compound which is quite general, the words in it are crucial for extracting the relation.

This motivates our work to use word clusters as additional features for relation extraction. The assumption is that even if the word *soldier* may never have been seen in the annotated *Employment* relation instances, other words which share the same cluster membership with *soldier* such as *president* and *ambassador* may have been observed in the *Employment* instances. The absence of lexical features can be compensated by

Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics, pages 521–529,
Portland, Oregon, June 19-24, 2011.

the cluster features. Moreover, word clusters may implicitly correspond to different relation classes. For example, the cluster of *president* may be related to the *Employment* relation as in *US president* while the cluster of *businessman* may be related to the *Affiliation* relation as in *US businessman*.

The main contributions of this paper are: we explore the cluster-based features in a systematic way and propose several statistical methods for selecting effective clusters. We study the impact of the size of training data on cluster features and analyze the performance improvements through an extensive experimental study.

The rest of this paper is organized as follows: Section 2 presents related work and Section 3 provides the background of the relation extraction task and the word clustering algorithm. Section 4 describes in detail a state-of-the-art supervised baseline system. Section 5 describes the cluster-based features and the cluster selection methods. We present experimental results in Section 6 and conclude in Section 7.

2 Related Work

The idea of using word clusters as features in discriminative learning was pioneered by Miller et al. (2004), who augmented name tagging training data with hierarchical word clusters generated by the Brown clustering algorithm (Brown et al., 1992) from a large unlabeled corpus. They used different thresholds to cut the word hierarchy to obtain clusters of various granularities for feature decoding. Ratinov and Roth (2009) and Turian et al. (2010) also explored this approach for name tagging. Though all of them used the same hierarchical word clustering algorithm for the task of name tagging and reported improvements, we noticed that the clusters used by Miller et al. (2004) were quite different from that of Ratinov and Roth (2009) and Turian et al. (2010). To our knowledge, there has not been work on selecting clusters in a principled way. We move a step further to explore several methods in choosing effective clusters. A second difference between this work and the above ones is that we utilize word clusters in the task of relation extraction which is very different from sequence labeling tasks such as name tagging and chunking.

Though Boschee et al. (2005) and Chan and Roth (2010) used word clusters in relation extraction, they shared the same limitation as the above approaches in choosing clusters. For example, Boschee et al. (2005) chose clusters of different granularities and Chan and Roth (2010) simply used a single threshold for cutting the word hierarchy. Moreover, Boschee et al. (2005) only augmented the *predicate* (typically a verb or a noun of the most importance in a relation in their definition) with word clusters while Chan and Roth (2010) performed this for any lexical feature consisting of a single word. In this paper, we systematically explore the effectiveness of adding word clusters to different lexical features.

3 Background

3.1 Relation Extraction

One of the well defined relation extraction tasks is the Automatic Content Extraction[1] (ACE) program sponsored by the U.S. government. ACE 2004 defined 7 major entity types: PER (Person), ORG (Organization), FAC (Facility), GPE (Geo-Political Entity: countries, cities, etc.), LOC (Location), WEA (Weapon) and VEH (Vehicle). An entity has three types of mention: NAM (proper name), NOM (nominal) or PRO (pronoun). A relation was defined over a pair of entity mentions within a single sentence. The 7 major relation types with examples are shown in Table 1. ACE 2004 also defined 23 relation subtypes. Following most of the previous work, this paper only focuses on relation extraction of major types.

Given a relation instance $x = (s, m_i, m_j)$, where m_i and m_j are a pair of mentions and s is the sentence containing the pair, the goal is to learn a function which maps the instance x to a type c, where c is one of the 7 defined relation types or the type *Nil* (no relation exists). There are two commonly used learning paradigms for relation extraction:

Flat: This strategy performs relation detection and classification at the same time. One multi-class classifier is trained to discriminate among the 7 relation types plus the *Nil* type.

Hierarchical: This one separates relation detection from relation classification. One binary

[1] Task definition: http://www.itl.nist.gov/iad/894.01/tests/ace/
ACE guidelines: http://projects.ldc.upenn.edu/ace/

classifier is trained first to distinguish between relation instances and non-relation instances. This can be done by grouping all the instances of the 7 relation types into a positive class and the instances of *Nil* into a negative class. Then the thresholded output of this binary classifier is used as training data for learning a multi-class classifier for the 7 relation types (Bunescu and Mooney, 2005b).

Type	Example
EMP-ORG	*US president*
PHYS	*a military base in Germany*
GPE-AFF	*U.S. businessman*
PER-SOC	*a spokesman for the senator*
DISC	*each of whom*
ART	*US helicopters*
OTHER-AFF	*Cuban-American people*

Table 1: ACE relation types and examples from the annotation guideline[2]. The heads of the two entity mentions are marked. Types are listed in decreasing order of frequency of occurrence in the ACE corpus.

3.2 Brown Word Clustering

The Brown algorithm is a hierarchical clustering algorithm which initially assigns each word to its own cluster and then repeatedly merges the two clusters which cause the least loss in average mutual information between adjacent clusters based on bigram statistics. By tracing the pairwise merging steps, one can obtain a word hierarchy which can be represented as a binary tree. A word can be compactly represented as a bit string by following the path from the root to itself in the tree, assigning a 0 for each left branch, and a 1 for each right branch. A cluster is just a branch of that tree. A high branch may correspond to more general concepts while the lower branches it includes might correspond to more specific ones.

Brown et al. (1992) described an efficient implementation based on a greedy algorithm which initially assigned only the most frequent words into distinct clusters. It is worth pointing out that in this implementation each word occupies a leaf in the hierarchy, but each leaf might contain more than one word as can be seen from Table 2. The lengths of the bit strings also vary among different words.

[2] http://projects.ldc.upenn.edu/ace/docs/EnglishRDCV4-3-2.PDF

Bit string	Examples
111011011100	***US*** *...*
1110110111011	***U.S.*** *...*
1110110110000	***American*** *...*
1110110111110110	***Cuban***, *Pakistani, Russian ...*
11111110010111	***Germany***, *Poland, Greece ...*
110111110100	***businessman***, *journalist, reporter*
1101111101111	***president,*** *governor, premier...*
1101111101100	***senator***, *soldier, ambassador ...*
11011101110	***spokesman***, *spokeswoman, ...*
11001100	***people***, *persons, miners, Haitians*
110110111011111	***base***, *compound, camps, camp ...*
110010111	***helicopters***, *tanks, Marines ...*

Table 2: An example of words and their bit string representations obtained in this paper. Words in bold are head words that appeared in Table 1.

4 Feature Based Relation Extraction

Given a pair of entity mentions $< m_i, m_j >$ and the sentence containing the pair, a feature based system extracts a feature vector v which contains diverse lexical, syntactic and semantic features. The goal is to learn a function which can estimate the conditional probability $p(c \mid v)$, the probability of a relation type c given the feature vector v. The type with the highest probability will be output as the *class label* for the mention pair.

We now describe a supervised baseline system with a very large set of features and its learning strategy.

4.1 Baseline Feature Set

We first adopted the full feature set from Zhou et al. (2005), a state-of-the-art feature based relation extraction system. For space reasons, we only show the lexical features as in Table 3 and refer the reader to the paper for the rest of the features.

At the lexical level, a relation instance can be seen as a sequence of tokens which form a five tuple *<Before, M1, Between, M2, After>*. Tokens of the five members and the interaction between the heads of the two mentions can be extracted as features as shown in Table 3.

In addition, we cherry-picked the following features which were not included in Zhou et al. (2005) but were shown to be quite effective for relation extraction.

Bigram of the words between the two mentions: This was extracted by both Zhao and Grishman (2005) and Jiang and Zhai (2007), aiming to

provide more order information of the tokens between the two mentions.

Patterns: There are three types of patterns: 1) the sequence of the tokens between the two mentions as used in Boschee et al. (2005); 2) the sequence of the heads of the constituents between the two mentions as used by Grishman et al. (2005); 3) the shortest dependency path between the two mentions in a dependency tree as adopted by Bunescu and Mooney (2005a). These patterns can provide more structured information of how the two mentions are connected.

Title list: This is tailored for the EMP-ORG type of relations as the head of one of the mentions is usually a title. The features are decoded in a way similar to that of Sun (2009).

Position	Feature	Description
Before	BM1F	*first word before* M1
	BM1L	*second word before* M1
M1	WM1	*bag-of-words in* M1
	HM1	*head*[3] *word of* M1
Between	WBNULL	*when no word in between*
	WBFL	*the only word in between when only one word in between*
	WBF	*first word in between when at least two words in between*
	WBL	*last word in between when at least two words in between*
	WBO	*other words in between except first and last words when at least three words in between*
M2	WM2	*bag-of-words in* M2
	HM2	*head word of* M2
M12	HM12	*combination of* HM1 *and* HM2
After	AM2F	*first word after* M2
	AM2L	*second word after* M2

Table 3: Lexical features for relation extraction.

4.2 Baseline Learning Strategy

We employ a simple learning framework that is similar to the *hierarchical* learning strategy as described in Section 3.1. Specifically, we first train a binary classifier to distinguish between relation instances and non-relation instances. Then rather than using the thresholded output of this binary classifier as training data, we use only the annotated relation instances to train a multi-class classifier for the 7 relation types. In the test phase, given a test instance x, we first apply the binary classifier to it for relation detection; if it is detected as a relation instance we then apply the multi-class relation classifier to classify it[4].

5 Cluster Feature Selection

The selection of cluster features aims to answer the following two questions: which lexical features should be augmented with word clusters to improve generalization accuracy? How to select clusters at an appropriate level of granularity? We will describe our solutions in Section 5.1 and 5.2.

5.1 Cluster Feature Decoding

While each one of the lexical features in Table 3 used by the baseline can potentially be augmented with word clusters, we believe the effectiveness of a lexical feature with augmentation of word clusters should be tested either individually or incrementally according to a rank of its *importance* as shown in Table 4. We will show the effectiveness of each cluster feature in the experiment section.

Importance	Lexical Feature	Description of lexical feature	Cluster Feature
1	HM	*HM1, HM2 and HM12*	HM1_WC, HM2_WC, HM12_WC
2	BagWM	*WM1 and WM2*	BagWM_WC
3	HC	*a head*[5] *of a chunk in context*	HC_WC
4	BagWC	*word of context*	BagWC_WC

Table 4: Cluster features ordered by *importance*.

The *importance* is based on linguistic intuitions and observations of the contributions of different lexical features from various feature based systems. Table 4 simplifies a relation instance as a three tuple <*Context*, *M1*, *M2*> where the *Context* includes the *Before*, *Between* and *After* from the

[3] The head word of a mention is normally set as the last word of the mention as in Zhou et al. (2005).

[4] Both the binary and multi-class classifiers output normalized probabilities in the range [0,1]. When the binary classifier's prediction probability is greater than 0.5, we take the prediction with the highest probability of the multi-class classifier as the final class label. When it is in the range [0.3,0.5], we only consider as the final class label the prediction of the multi-class classifier with a probability which is greater than 0.9. All other cases are taken as non-relation instances.

[5] The head of a chunk is defined as the last word in the chunk.

five tuple representation. As a relation in ACE is usually short, the words of the two entity mentions can provide more critical indications for relation classification than the words from the context. Within the two entity mentions, the head word of each mention is usually more important than other words of the mention; the conjunction of the two heads can provide an additional clue. And in general words other than the chunk head in the context do not contribute to establishing a relationship between the two entity mentions.

The cluster based semi-supervised system works by adding an additional layer of lexical features that incorporate word clusters as shown in column 4 of Table 4. Take the *US soldier* as an example, if we decide to use a length of 10 as a threshold to cut the Brown word hierarchy to generate word clusters, we will extract a cluster feature *HM1_WC10=1101111101* in addition to the lexical feature *HM1=soldier* given that the full bit string of *soldier* is *1101111101100* in Table 2. (Note that the cluster feature is a nominal feature, not to be confused with an integer feature.)

5.2 Selection of Clusters

Given the bit string representations of all the words in a vocabulary, researchers usually use prefixes of different lengths of the bit strings to produce word clusters of various granularities. However, how to choose the set of prefix lengths in a principled way? This has not been answered by prior work.

Our main idea is to learn the best set of prefix lengths, perhaps through the validation of their effectiveness on a development set of data. To our knowledge, previous research simply uses ad-hoc prefix lengths and lacks this training procedure. The training procedure can be extremely slow for reasons to be explained below.

Formally, let l be the set of available prefix lengths ranging from 1 bit to the length of the longest bit string in the Brown word hierarchy and let m be the set of prefix lengths we want to use in decoding cluster features, then the problem of selecting effective clusters transforms to finding a $|m|$-combination of the set l which maximizes system performance. The training procedure can be extremely time consuming if we enumerate every possible $|m|$-combination of l, given that $|m|$ can range from 1 to the size of l and the size of l equals the length of the longest bit string which is usually 20 when inducing 1,000 clusters using the Brown algorithm.

One way to achieve better efficiency is to consider only a subset of l instead of the full set. In addition, we limit ourselves to use sizes 3 and 4 for m for matching prior work. This keeps the cluster features to a manageable size considering that every word in your vocabulary could contribute to a lexical feature. For picking a subset of l, we propose below two statistical measures for computing the *importance* of a certain prefix length.

Information Gain (IG): IG measures the quality or *importance* of a feature *f* by computing the difference between the prior entropy of classes *C* and the posterior entropy, given values *V* of the feature *f* (Hunt et al., 1966; Quinlan, 1986). For our purpose, *C* is the set of relation types, *f* is a cluster-based feature with a certain prefix length such as *HM1_WC** where * means the prefix length and a value *v* is the prefix of the bit string representation of *HM1*. More formally, the IG of *f* is computed as follows:

$$IG(f) = -\sum_{c \in C} p(c) \log p(c) - \sum_{v \in V} (-p(v) \times \sum_{c \in C} p(c \mid v) \log p(c \mid v)) \quad (1)$$

where the first and second terms refer to the prior and posterior entropies respectively.

For each prefix length in the set l, we can compute its IG for a type of cluster feature and then rank the prefix lengths based on their IGs for that cluster feature. For simplicity, we rank the prefix lengths for a group of cluster features (a group is a row from column 4 in Table 4) by collapsing the individual cluster features into a single cluster feature. For example, we collapse the 3 types: *HM1_WC*, *HM2_WC* and *HM12_WC* into a single type *HM_WC* for computing the IG.

Prefix Coverage (PC): If we use a short prefix then the clusters produced correspond to the high branches in the word hierarchy and would be very general. The cluster features may not provide more informative information than the words themselves. Similarly, if we use a long prefix such as the length of the longest bit string, then maybe only a few of the lexical features can be *covered* by clusters. To capture this intuition, we define the PC of a prefix length *i* as below:

$$PC(i) = \frac{count(f_{c_i})}{count(f_l)} \quad (2)$$

where f_l stands for a lexical feature such as *HM1* and f_{c_i} a cluster feature with prefix length *i* such as *HM1_WCi*, $count(*)$ is the number of occurrences of that feature in training data.

Similar to IG, we compute PC for a group of cluster features, not for each individual feature.

In our experiments, the top 10 ranked prefix lengths based on IG and prefix lengths with PC values in the range [0.4, 0.9] were used.

In addition to the above two statistical measures, for comparison, we introduce another two simple but extreme measures for the selection of clusters.

Use All Prefixes (UA): UA produces a cluster feature at every available bit length with the hope that the underlying supervised system can *learn* proper weights of different cluster features during training. For example, if the full bit representation of "Apple" is "000", UA would produce three cluster features: *prefix1=0*, *prefix2=00* and *prefix3=000*. Because this method does not need validation on the development set, it is the laziest but the fastest method for *selecting* clusters.

Exhaustive Search (ES): ES works by trying every possible combination of the set *l* and picking the one that works the best for the development set. This is the most cautious and the slowest method for selecting clusters.

6 Experiments

In this section, we first present details of our unsupervised word clusters, the relation extraction data set and its preprocessing. We then present a series of experiments coupled with result analyses.

We used the English portion of the TDT5 corpora (LDC2006T18) as our unlabeled data for inducing word clusters. It contains roughly 83 million words in 3.4 million sentences with a vocabulary size of 450K. We left case intact in the corpora. Following previous work, we used Liang's implementation of the Brown clustering algorithm (Liang, 2005). We induced 1,000 word clusters for words that appeared at least twice in the corpora. The reduced vocabulary contains 255K unique words. The clusters are available at http://www.cs.nyu.edu/~asun/data/TDT5_BrownWC.tar.gz.

For relation extraction, we used the benchmark ACE 2004 training data. Following most of the previous research, we used in experiments the *nwire* (newswire) and *bnews* (broadcast news) genres of the data containing 348 documents and 4374 relation instances. We extracted an instance for every pair of mentions in the same sentence which were separated by no more than two other mentions. The non-relation instances generated were about 8 times more than the relation instances.

Preprocessing of the ACE documents: We used the Stanford parser[6] for syntactic and dependency parsing. We used chunklink[7] to derive chunking information from the Stanford parsing. Because some *bnews* documents are in lower case, we recover the case for the head of a mention if its type is NAM by making the first character into its upper case. This is for better matching between the words in ACE and the words in the unsupervised word clusters.

We used the OpenNLP [8] maximum entropy (maxent) package as our machine learning tool. We choose to work with maxent because the training is fast and it has a good support for multi-class classification.

6.1 Baseline Performance

Following previous work, we did 5-fold cross-validation on the 348 documents with hand-annotated entity mentions. Our results are shown in Table 5 which also lists the results of another three state-of-the-art feature based systems. For this and the following experiments, all the results were computed at the relation mention level.

System	P(%)	R(%)	F(%)
Zhou et al. (2007)[9]	78.2	63.4	70.1
Zhao and Grishman (2005)[10]	69.2	71.5	70.4
Our Baseline	73.4	67.7	70.4
Jiang and Zhai (2007) [11]	72.4	70.2	71.3

Table 5: Performance comparison on the ACE 2004 data over the 7 relation types.

[6] http://nlp.stanford.edu/software/lex-parser.shtml
[7] http://ilk.uvt.nl/team/sabine/chunklink/README.html
[8] http://opennlp.sourceforge.net/
[9] Zhou et al. (2005) tested their system on the ACE 2003 data; Zhou et al. (2007) tested their system on the ACE 2004 data.
[10] The paper gives a recall value of 70.5, which is not consistent with the given values of P and F. An examination of the correspondence in preparing this paper indicates that the correct recall value is 71.5.
[11] The result is from using the *All* features in Jiang and Zhai (2007). It is not quite clear from the paper that whether they used the 348 documents or the whole 2004 training data.

Note that although all the 4 systems did 5-fold cross-validation on the ACE 2004 data, the detailed data partition might be different. Also, we were doing cross-validation at the document level which we believe was more natural than the instance level. Nonetheless, we believe our baseline system has achieved very competitive performance.

6.2 The Effectiveness of Cluster Selection Methods

We investigated the tradeoff between performance and training time of each proposed method in selecting clusters. In this experiment, we randomly selected 70 documents from the 348 documents as test data which roughly equaled the size of 1 fold in the baseline in Section 6.1. For the baseline in this section, all the rest of the documents were used as training data. For the semi-supervised system, 70 percent of the rest of the documents were randomly selected as training data and 30 percent as development data. The set of prefix lengths that worked the best for the development set was chosen to select clusters. We only used the cluster feature *HM_WC* in this experiment.

System	F	△	Training Time (in minute)
Baseline	70.70		1
UA	71.19	+0.49	1.5
PC3	71.65	+0.95	30
PC4	71.72	+1.02	46
IG3	71.65	+0.95	45
IG4	71.68	+0.98	78
ES3	71.66	+0.96	465
ES4	71.60	+0.90	1678

Table 6: The tradeoff between performance and training time of each method in selecting clusters. PC3 means using 3 prefixes with the PC method. △ in this paper means the difference between a system and the baseline.

Table 6 shows that all the 4 proposed methods improved baseline performance, with UA as the fastest and ES as the slowest. It was interesting that ES did not always outperform the two statistical methods which might be because of its overfitting to the development set. In general, both PC and IG had good balances between performance and training time. There was no dramatic difference in performance between using 3 and 4 prefix lengths.

For the rest of this paper, we will only use PC4 as our method in selecting clusters.

6.3 The Effectiveness of Cluster Features

The baseline here is the same one used in Section 6.1. For the semi-supervised system, each test fold was the same one used in the baseline and the other 4 folds were further split into a training set and a development set in a ratio of 7:3 for selecting clusters. We first added the cluster features individually into the baseline and then added them incrementally according to the order specified in Table 4.

System		F	△
1	Baseline	70.4	
2	1 + HM_WC	71.5	+ 1.1
3	1 + BagWM_WC	71.0	+ 0.6
4	1 + HC_WC	69.6	- 0.8
5	1 + BagWC_WC	46.1	- 24.3
6	2 + BagWM_WC	71.0	+ 0.6
7	6 + HC_WC	70.6	+ 0.2
8	7+ BagWC_WC	50.3	- 20.1

Table 7: Performance [12] of the baseline and using different cluster features with PC4 over the 7 types.

We found that adding clusters to the heads of the two mentions was the most effective way of introducing cluster features. Adding clusters to the words of the mentions can also help, though not as good as the heads. We were surprised that the heads of chunks in context did not help. This might be because ACE relations are usually short and the limited number of long relations is not sufficient in generalizing cluster features. Adding clusters to every word in context hurt the performance a lot. Because of the behavior of each individual feature, it was not surprising that adding them incrementally did not give more performance gain.

For the rest of this paper, we will only use *HM_WC* as cluster features.

6.4 The Impact of Training Size

We studied the impact of training data size on cluster features as shown in Table 8. The test data was always the same as the 5-fold used in the baseline in Section 6.1. no matter the size of the training data. The training documents for the

[12] All the improvements of F in Table 7, 8 and 9 were significant at confidence levels >= 95%.

# docs	F of Relation Classification			F of Relation Detection		
	Baseline	PC4 (△)	Prefix10(△)	Baseline	PC4(△)	Prefix10(△)
50	62.9	63.8(+ 0.9)	63.7(+0.8)	71.4	71.9(+ 0.5)	71.6(+0.2)
75	62.8	64.6(+ 1.8)	63.9(+1.1)	71.5	72.3(+ 0.8)	72.5(+1.0)
125	66.1	68.1(+ 2.0)	67.5(+1.4)	74.5	74.8(+ 0.3)	74.3(-0.2)
175	67.8	69.7(+ 1.9)	69.5(+1.7)	75.2	75.5(+ 0.3)	75.2(0.0)
225	68.9	70.1(+ 1.2)	69.6(+0.7)	75.6	75.9(+ 0.3)	75.3(-0.3)
≈280	70.4	71.5(+ 1.1)	70.7(+0.3)	76.4	76.9(+ 0.5)	76.3(-0.1)

Table 8: Performance over the 7 relation types with different sizes of training data. Prefix10 uses the single prefix length 10 to generate word clusters as used by Chan and Roth (2010).

Type	P		R		F	
	Baseline	PC4 (△)	Baseline	PC4 (△)	Baseline	PC4 (△)
EMP-ORG	75.4	77.2(+1.8)	79.8	81.5(+1.7)	77.6	79.3(+1.7)
PHYS	73.2	71.2(-2.0)	61.6	60.2(-1.4)	66.9	65.3(-1.7)
GPE-AFF	67.1	69.0(+1.9)	60.0	63.2(+3.2)	63.3	65.9(+2.6)
PER-SOC	88.2	83.9(-4.3)	58.4	61.0(+2.6)	70.3	70.7(+0.4)
DISC	79.4	80.6(+1.2)	42.9	46.0(+3.2)	55.7	58.6(+2.9)
ART	87.9	96.9(+9.0)	63.0	67.4(+4.4)	73.4	79.3(+5.9)
OTHER-AFF	70.6	80.0(+9.4)	41.4	41.4(0.0)	52.2	54.6(+2.4)

Table 9: Performance of each individual relation type based on 5-fold cross-validation.

current size setup were randomly selected and added to the previous size setup (if applicable). For example, we randomly selected another 25 documents and added them to the previous 50 documents to get 75 documents. We made sure that every document participated in this experiment. The training documents for each size setup were split into a real training set and a development set in a ratio of 7:3 for selecting clusters.

There are some clear trends in Table 8. Under each training size, PC4 consistently outperformed the baseline and the system Prefix10 for relation classification. For PC4, the gain for classification was more pronounced than detection. The mixed detection results of Prefix10 indicated that only using a single prefix may not be stable.

We did not observe the same trend in the reduction of annotation need with cluster-based features as in Koo et al. (2008) for dependency parsing. PC4 with sizes 50, 125, 175 outperformed the baseline with sizes 75, 175, 225 respectively. But this was not the case when PC4 was tested with sizes 75 and 225. This might due to the complexity of the relation extraction task.

6.5 Analysis

There were on average 69 cross-type errors in the baseline in Section 6.1 which were reduced to 56 by using PC4. Table 9 showed that most of the improvements involved EMP-ORG, GPE-AFF, DISC, ART and OTHER-AFF. The performance gain for PER-SOC was not as pronounced as the other five types. The five types of relations are ambiguous as they share the same entity type GPE while the PER-SOC relation only holds between PER and PER. This reflects that word clusters can help to distinguish between ambiguous relation types.

As mentioned earlier the gain of relation detection was not as pronounced as classification as shown in Table 8. The unbalanced distribution of relation instances and non-relation instances remains as an obstacle for pushing the performance of relation extraction to the next level.

7 Conclusion and Future Work

We have described a semi-supervised relation extraction system with large-scale word clustering. We have systematically explored the effectiveness of different cluster-based features. We have also demonstrated that the two proposed statistical methods are both effective and efficient in selecting clusters at an appropriate level of granularity through an extensive experimental study.

Based on the experimental results, we plan to investigate additional ways to improve the performance of relation detection. Moreover, extending word clustering to phrase clustering (Lin and Wu, 2009) and pattern clustering (Sun and Grishman, 2010) is worth future investigation for relation extraction.

References

Rie K. Ando and Tong Zhang. 2005 A Framework for Learning Predictive Structures from Multiple Tasks and Unlabeled Data. *Journal of Machine Learning Research*, Vol 6:1817-1853.

Elizabeth Boschee, Ralph Weischedel, and Alex Zamanian. 2005. Automatic information extraction. In *Proceedings of the International Conference on Intelligence Analysis*.

Peter F. Brown, Vincent J. Della Pietra, Peter V. deSouza, Jenifer C. Lai, and Robert L. Mercer. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Razvan C. Bunescu and Raymond J. Mooney. 2005a. A shortest path dependency kenrel for relation extraction. In *Proceedings of HLT/EMNLP*.

Razvan C. Bunescu and Raymond J. Mooney. 2005b. Subsequence kernels for relation extraction. In *Proceedings* of *NIPS*.

Yee Seng Chan and Dan Roth. 2010. Exploiting background knowledge for relation extraction. In *Proc. of COLING*.

Ralph Grishman, David Westbrook and Adam Meyers. 2005. NYU's English ACE 2005 System Description. *ACE 2005 Evaluation Workshop.*

Earl B. Hunt, Philip J. Stone and Janet Marin. 1966. *Experiments in Induction*. New York: Academic Press, 1966.

Jing Jiang and ChengXiang Zhai. 2007. A systematic exploration of the feature space for relation extraction. In *Proceedings of HLT-NAACL-07*.

Nanda Kambhatla. 2004. Combining lexical, syntactic, and semantic features with maximum entropy models for information extraction. In *Proceedings of ACL-04*.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple Semi-supervised Dependency Parsing. In *Proceedings of ACL-08: HLT*.

Percy Liang. 2005. Semi-Supervised Learning for Natural Language. Master's thesis, Massachusetts Institute of Technology.

Dekang Lin and Xiaoyun Wu. 2009. Phrase Clustering for Discriminative Learning. In *Proc. of ACL-09.*

Scott Miller, Heidi Fox, Lance Ramshaw, and Ralph Weischedel. 2000. A novel use of statistical parsing to extract information from text. In *Proc. of NAACL.*

Scott Miller, Jethran Guinness and Alex Zamanian. 2004. Name Tagging with Word Clusters and Discriminative Training. In *Proc. of HLT-NAACL.*

Longhua Qian, Guodong Zhou, Qiaoming Zhu and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction . In *Proc. of COLING.*

John Ross Quinlan. 1986. Induction of decision trees. *Machine Learning*, 1(1), 81-106.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of CoNLL-09.*

Ang Sun. 2009. A Two-stage Bootstrapping Algorithm for Relation Extraction. In *RANLP-09.*

Ang Sun and Ralph Grishman. 2010. Semi-supervised Semantic Pattern Discovery with Guidance from Unsupervised Pattern Clusters. In *Proc. of COLING.*

Joseph Turian, Lev-Arie Ratinov, and Yoshua Bengio. 2010. Word representations: A simple and general method for semi-supervised learning. In *Proceedings of ACL.*

Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. 2003. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106.

Zhu Zhang. 2004. Weakly supervised relation classification for information extraction. In *Proc. of CIKM'2004*.

Min Zhang, Jie Zhang, Jian Su, and GuoDong Zhou. 2006. A composite kernel to extract relations between entities with both flat and structured features. In *Proceedings of COLING-ACL-06*.

Shubin Zhao and Ralph Grishman. 2005. Extracting relations with integrated information using kernel methods. In *Proceedings of ACL.*

Guodong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of ACL-05.*

Guodong Zhou, Min Zhang, DongHong Ji, and QiaoMing Zhu. 2007. Tree kernel-based relation extraction with context-sensitive structured parse tree information. In *Proceedings of EMNLPCoNLL-07.*