Relation Extraction on the J.D. Power and Associates Sentiment Corpus

by

Gregory Ichneumon Brown

B.S., Cornell University, 2000

A thesis submitted to the

Faculty of the Graduate School of the

University of Colorado in partial fulfillment

of the requirements for the degree of

Master of Science

Department of Computer Science

2011

This thesis entitled:
Relation Extraction on the J.D. Power and Associates Sentiment Corpus
written by Gregory Ichneumon Brown
has been approved for the Department of Computer Science

James H. Martin

Michael C. Mozer

Rodney D. Nielsen

Date

The final copy of this thesis has been examined by the signatories, and we find that both the content and the form meet acceptable presentation standards of scholarly work in the above mentioned discipline.

Brown, Gregory Ichneumon (M.S., Computer Science)

Relation Extraction on the J.D. Power and Associates Sentiment Corpus

Thesis directed by Prof. James H. Martin

Relation extraction is a means of determining relations between entities discussed in text. In this work we apply a number of existing and novel techniques to the J.D. Power and Associates Sentiment Corpus to examine the techniques' effectiveness when looking for relations in noisy blog and social media documents. To classify relations, we use support vector machines with both vector features and tree kernels, the current state of the art methods for relation extraction. Additionally, we extend these methods to examine relations across multiple sentences. Unlike previous systems, this one focuses predominantly on relations between all entities discussed in the text rather than only entities that are mentioned in the same sentence. We present evidence that relation extraction can be improved using an ensemble classification scheme to combine relations between mentions to predict relations between co-reference groups of mentions.

Dedication

To Mekayla, for her patience, inspiration, and knowing just how much to push me and when.

Acknowledgements

I'd like to thank J.D. Power and Associates for their partial sponsering of this research and for constructing the Corpus. I'd also like to thank the annotation team for their great work building such a useful resource. On the research team I'd like to thank Nicholas Nicolov, Jasson Kessler, and Will Headden for all of their ideas and help along the way as I was trying to decide on the best research directions to take.

Contents

Chapter

Tables

Table

Figures

Figure

Chapter 1

Introduction

To accurately summarize, determine the sentiment, or answer questions about a document, it is often necessary to determine the relationships between entities being discussed in the document (such as part-of or member-of). For example, in this simple sentiment example

Example 1.1: I bought a new Mazda3 yesterday. I love the powerful engine.

determining the sentiment the author is expressing about the Mazda3 requires knowing that the engine is a part of the Mazda3 so that the positive sentiment being expressed about the engine can also be attributed to the Mazda3.

In this work, we define relations as they were defined by Winston et al [24], as a relation that exists between two abstract or real objects (entities) in the world. We use two corpora that annotated relations in different ways, but in both cases they discuss relations which are expressed by the text rather than general ontologies of relations. For instance, a car can have a spoiler as a part of it, but not all cars do. In the sentence

Example 1.2: We looked at a BMW 3 Series and a Mazda3, but only the latter had a spoiler.

the phrase "spoiler" would be annotated to be a part of the Mazda3, but would not be a part of the BMW 3 Series despite the fact that generically, a spoiler is a part of a car.

We investigated using syntactic and lexical vectors of features and tree kernels from previous work to see how well the same techniques can be applied to the J. D. Power and Associates (JDPA) Sentiment Corpus [13][14]. Tree kernels provide a method of extracting features directly from the constituency parse tree of a sentence in order to detect relations. Because they operate directly on the parse tree, we find that tree kernels are an effective way to adapt to new data sources, as they are not optimized for any one particular corpus in the same way that vectors of features are.

The JDPA corpus presents a new relation extraction challenge because it contains social media documents. Previous work on relation extraction largely focused on news articles. However, the JDPA Corpus contains documents written by a wide range of authors: professional reviewers, blog writers, and social media/message board writers. We find that this variation in writers also results in a wider range of how they use language.

Two key terms are used throughout this paper that deserve to be properly defined up front: "mention" and "entity". A mention is a word or phrase in the text that refers to a real-world entity. Example 1.1 contains the mentions "I", "Mazda3", "yesterday", "I", and "engine". Expanding on Example 1.1, we can look at the set of mention phrases that all refer to the same entity:

Example 1.3: I bought a new [$_{E_1}$ Mazda3] yesterday. I love the powerful [$_{E_2}$ engine]. [$_{E_2}$ It] makes the [$_{E_1}$ car] feel like a sports car. I love driving [$_{E_1}$ it].

A set of mention phrases that all refer to the same real world entity is called an entity or a co-reference group (we use the two terms interchangeably). In Example 1.3 the mention phrases that are referring to the Mazda3 are marked as E_1 and those referring to the Mazda3's engine are marked with E_2.

Relations can then be considered as either a relation between pairs of mentions, or a relation between pairs of entities. Furthermore, an entity pair is composed of all of the mention pairs between the two entities. So in Example 1.3 where E_1 has three mentions and E_2 has two mentions, there are a total of six mention pairs that compose this single entity pair.

Due to limitations in how relations have been annotated, previous work on relation extraction was limited to only looking at relations between mention pairs occurring in the same sentence. So in Example 1.1 above, there would be no annotated relation indicating that "engine" is a part of "Mazda3". The JDPA Corpus was annotated to indicate relations between all the entities being discussed in the document. Inherently, these annotations cross sentence boundaries. This expands the number of possible mention pairs that can be considered and thus can help boost the number of relations detected within the document.

Similar to other systems, our relation extraction system looks at the pairs of mentions in the documents to determine whether a relation exists between the real world entities. However, our system then combines the results from all of the mention pairs to produce a single entity pair prediction using a simple ensemble voting scheme. Since each mention pair provides a new "look" at whether a relation exists between the entities, combining these results is a novel way to improve how many relations within a document can be extracted. This novel ensemble method improves the performance of the relation extraction system.

Prior work has mostly focused on the number of mention pair relations that can be detected. We argue that the more appropriate metric for measuring the effectiveness of a relation extraction system is to look at the number of correctly classified relations between entity pairs. Since each entity pair can consist of an arbitrary number of mention pairs, the marginal amount of information extracted from the document by correctly detecting a relation between two mentions is zero if the relation between them has already been detected. Section 3.4 provides a more thorough discussion of the metrics for measuring relation extraction.

In summary, in this work we find that:

- Existing relation extraction feature vectors do not work as well on the JDPA Corpus.
- Using a tree kernel to learn models for relation extraction is an effective way to avoid having to re-engineer the vector features.

- Relation extraction can be improved by an ensemble combination of detected relations between mention pairs to produce predicted entity pair relations.
- The traditional metric of comparing correctly classified relations between mention pairs is a less complete relation extraction metric than measuring the correctly classified relations between entity pairs.

Chapter 2

Background

2.1 Relation Corpora

In this section we detail the two corpora we used in our research: the Automatic Content Extraction 2004 Training Corpus (LDC2005T09) and the J.D Power and Associates Sentiment Corpus.

2.1.1 Automatic Content Extraction (ACE) Corpus

The ACE Corpus [16] is one of the most common corpora for performing relation extraction. Multiple versions of the corpus have been released, but in this work we focus on the ACE 2004 Corpus as that seems to be the most common corpus in other relation extraction research. The ACE Corpus is annotated for two different tasks: co-reference resolution and relation extraction. The relation annotations consist of 23 different fine-grained relations, which are divided up into 7 different coarse relation categories (Agent-Artifact, Discourse, Employee-Organization, GeoPolitical-Affiliation, Other-Affiliation, Personal-Social, Physical). The relations are limited to mentions between real-world entities that are used in the same sentence.

In this research we focus primarily on the 7 coarse relations to more easily make comparisons to other research. But we also run our baseline vector features on the fine-grained relations. The results from these relations were used for testing our system and as a comparison point against the JDPA Corpus results. For our work, we use the bnews and nwire documents from the corpus consisting of broadcast news transcripts and newswire articles from a variety of news organizations.

Source	% of Documents
jdpower.com/autos/powersteering/∗	24%
∗.blogspot.com/∗	18%
livejournal.com/∗	18%
∗.wordpress.com/∗	7%
autoblog.com/∗	5%
autochannel.com/news/∗	4%
Other	24%

Table 2.1: Breakdown of automotive document sources in the JDPA Sentiment Corpus.

2.1.2 J.D. Power and Associates Sentiment Corpus

The JDPA Corpus consists of 457 documents containing discussions about cars, and 180 documents discussing cameras [13][14]. In this work we only use the automotive documents. Table 2.1 provides the breakdown of documents from each source.

The annotated mentions in the Corpus are single or multi-word expressions that refer to a particular real world or abstract entity. The mentions are annotated to indicate sets of mentions that constitute co-reference groups referring to the same entity. The relation annotations are between entities in the corpus documents. Table 2.2 summarizes the frequency of the different relations, and provides a comparison to the number of relations in the ACE 2004 Corpus.

2.1.3 Corpora Comparison

The broadcast news transcripts and news articles in the ACE Corpus have a very different writing style from the message board, blogs, and product reviews in the JDPA Corpus. To better understand these differences and the difference between the mentions and relations in the corpora we performed a simple statistical analysis of the documents in the training set.

As already mentioned, the ACE relations are between mentions within the same sentence, while the JDPA Corpus relations are between entities. This means that the JDPA relations are not limited to being between mentions in the same sentence. Table 2.2 shows that in the JDPA Corpus, only 72% of the relations are between entities that have mentions co-occurring in the same

Relation	WordNet Equivalent	Total Count (in Training)	Relations per Sentence	Ave Min Distance	Percent in One Sentence
FeatureOf	Attribute	7656	0.64	12.2	81%
InstanceOf	Hyponym (subset)	1740	0.14	27.1	60%
MemberOf	Meronym	1897	0.16	36.1	60%
PartOf	Meronym	9706	0.81	21.3	67%
Produces	None	1663	0.14	7.4	92%
JDPA Corpus Overall (cars)	NA	22662	1.89	18.9	72%
ACE 2004 Corpus Overall (bnews and nwire)	NA	2930	0.6	3.8	100%

Table 2.2: Statistics about the relations in the corpora generated on the 70% of the data in the training sets. The "Ave Min Distance" is the average of the minimum number of tokens between the two closest mentions of entities that are related to each other. The "Percent in Sentence" indicates the number of related entities that have mentions occurring in the same sentence.

	JDPA Corpus (cars only)	ACE 2004 Corpus (bnews and nwire)
Documents	457	348
Sentences Per Document	37.8	20.0
Tokens Per Document	713	426
Tokens Per Sentence	18.9	21.37
Mentions Per Sentence	4.37	3.46
Mentions Per Entity	1.48	2.36
Mentions With Only One Token	71%	56%
Entities Per Document	111.8	29.2

Table 2.3: Selected document statistics for the JDPA and ACE 2004 Corpora.

sentence. Since most prior research has focused on relations only within a sentence, we expect that detecting relations between mentions in different sentences will be difficult since there is little prior research looking at relations across sentences.

As a metric of the distance between two related entities, we counted the minimum number of tokens between the mentions in the two co-reference groups. For each relation type we then took the average of these counts to calculate the average minimum distance for each relation type. We find that, not surprisingly, the JDPA Corpus relations have a much higher average minimum distance, which we predict will also make it more difficult to predict relations. Notably, the minimum distance with the Produces relation is much smaller, so we expect to be able to predict that relation much more easily than the others. There are many more relation examples to learn from in the JDPA Corpus as compared to the ACE Corpus, but we hypothesize that the increased average distance between the mentions will likely offset any benefit that the increased number of examples provides.

Table 2.3 summarizes some additional statistics about the documents in the corpora. Of particular interest to relation extraction is the increased number of entities in each document of the JDPA Corpus. However, each entity on average is mentioned fewer times than in the ACE Corpus (Table 2.3 Mentions per Entity). This is somewhat unfortunate since the more times an individual entity is mentioned the more chances our ensemble system has of detecting a relation between it and another entity.

2.2 Related Work

In their 1987 paper, Winston et al. created a detailed taxonomy of Part-Whole (Meronymic) relations that combined much linguistic work from that time [24]. This work eventually formed the basis for how the JDPA Corpus relations were defined [14]. However, there have been a wide range of different types of relations explored in the intervening years. We do not here try to fully summarize the literature, especially not the linguistic and cognitive nature of semantic relations. Instead we focus first on some select recent work that has tried various ways of exploring and predicting relations on non-ACE Corpus data, and then summarize the two primary supervised

learning approaches applied to ACE relations that our work builds on: feature vectors and tree kernels.

2.2.1 Relations on Miscellaneous Data Sources

Vieira and Poesio built a heuristic based system for anaphora detection [23]. In addition to anaphora, they also created heuristics specifically for predicting relations between the mentions in the text. However, this relation extraction, by their own admission did not perform as well as they hoped given their results on anaphora detection. The heuristics they used focused on lexical and syntactic patterns in the text. They concluded that semantic knowledge would be a great help in detecting relations.

Girju et al. have also explored entity relations with a number of different methods[7][6][8]. Their initial methods did use semantic knowledge from WordNet to find pairs of words that were related to each other. They then searched for sentences containing both words, and annotated these sentences to indicate whether there was a relation expressed or not. A classifier was then trained and tested on this annotated data[6].

For our work, the most interesting aspects of Girju et al's research are their examination of the parts of speech surrounding relations, the connecting words between related mentions, and how compound noun phrases can express relations. In related work, Nakov and Hearst have also looked at how noun relations can be characterized by specific verbs [18] and how relations can be found to expand the relations within WordNet [9]. This work provides a thorough examination of relations within a sentence that feature vectors can be based on for future work.

On the SemEval 2007 Task 04 (classifying relations between noun phrases) Girju et al. also evaluated the results from a number of different systems to provide guidelines for what type of features helped with determining relations between entities[8]. A very important result from this work was that using WordNet to provide semantic information resulted in about a ten point increase in overall F-score for this task, thus confirming Viera and Poesio's hypothesis that semantic knowledge can greatly help relation extraction. Snow, Jurafsky, and Ng present further evidence that WordNet

information may be helpful in their examination of hyponomy (InstanceOf) relations[22]. Again they focused on relations expressed within a single sentence, but they both extracted patterns for training their classifier and made use of WordNet.

In our own work however, we explicitly do not use external semantic information in order to limit the scope of our research and confine ourselves to finding those features that can extract relations through syntactic and lexical information alone.

2.2.2 Relations on the ACE Corpus

As previously mentioned, the ACE Corpus appears to be the most commonly used corpus for relation extraction. Kambhatla developed a system in 2004 using a maximum entropy model that used lexical and syntactic features such as the parse and dependency trees of the sentence and entity types of the mentions[12]. Zhou et al. further expanded on these features[30], creating a set of features that many others have used as the basis for their own systems[3][10]. In our own work, we use Zhou et al's 2005 features as our baseline vector features.

Jiang and Zhai further expanded on the above features by performing an extensive exploration of the syntactic and lexical feature space[10]. This work certainly improved upon the previous features, and presented a methodology for exploring this feature space that would be necessary for expanding these features to work on different data sets. They also expanded upon the specific features to improve the parse trees and reduce noise in the features.

Finally, Chan and Roth have recently explored adding features outside of just the syntactic and lexical features[3]. Of particular interest to our work is the use of co-reference information to provide additional information about the context around the mention phrases. They also examined adding constraints on the allowable entity types with relations, and expanded the semantic information using Wikipedia.

One difficulty with the feature vector based approach to relation extraction is the time consuming search to build complex features. In particular, the parse trees of sentences can often contain information that is indicative of a relation between mentions, but finding these features

can be quite difficult due to the complexity of the parse trees. Furthermore, different relations are likely to be indicated by different patterns in the parse tree. By using a tree kernel rather than the standard linear or polynomial kernel in a Support Vector Machine (SVM), the parse tree for a sentence can be used directly rather than needing to engineer feature vectors from the parse tree. The tree kernel extracts structural information from the parse trees based upon which substructures in the trees are indicative of the relations[17]. Using a wider range of information than just the vector features also makes it easier for the system to adapt to data sets that are very different from the ACE Corpus. Additionally, the vector features and tree kernels are somewhat complimentary to each other, and so combining a linear kernel with the tree kernel provides significantly improved performance[29][26].

The earliest attempts at using structural SVM kernels to extract structural information were done by Bunescu and Mooney[2], and by Zelenko et al.[25]. Bunescu and Mooney used a subsequence kernel to encode information about the sequence of words, part-of-speech tags, and entity types in the sentence containing two mentions. Zelenko et al seems to be the first application of tree kernels to the domain of relation extraction. They used shallow parse trees rather than the full constituent parse tree. Similar to other work, they decorated their parse tree with entity type information.

Zhang et al[26] performed extensive experiments on how to prune parse trees to be most effective at relation extraction. Zhou et al[31][29] extended this research by exploring how to modify the parse trees to make further improvements. In Chapter 4 we discuss further how we structured our trees based on this prior research.

2.2.3 Relations Across Multiple Sentences

In all of the work mentioned in the previous sub-sections, the relations being detected were between mentions occurring in the same sentence. Although many relations are expressed within a single sentence, there are many cases where two entities that are related to one another do not contain any mentions in the same sentence. In the JDPA Corpus 27% of relations are between

entities that do not have mentions in the same sentence.

There has been relatively little research on relations across sentences, but Gerber and Chai's recent work[5] on extending NomBank to include implicit arguments (arguments from other sentences) presents a number of features and a basis for how to look at relations across sentences. Furthermore, Chan and Roth's previously mentioned work[3] on using co-reference information in relation extraction suggests that adding information about co-referring mentions around the mentions would help when the mentions are in different sentences.

2.2.4 Summary

The takeaway from the prior research seems to be that little to no research has been done with supervised learning techniques across an entire document to find relations between entities. In the context of this other research, the most interesting aspect of our research is that it focuses on relations between entities rather than mentions. This focus also allows examining relations across multiple sentences since the entity relation annotations in the JDPA Corpus are inherently across multiple sentences.

Chapter 3

Experimental Setup

3.1 Data Splits

The automotive documents in the JDPA Corpus were randomly partitioned into a training set (70% of the documents), a development set (10%), and a final test set (20%). The test set has only been used for generating the final results for this paper.

Similarly, the ACE bnews and newswire documents were randomly split into training/development/test sets with 70/10/20 proportions. However, we only used this split when developing our system. Most relation research on the ACE Corpus uses five fold cross validation to report results and so all of our results on the ACE Corpus were tested with five fold cross validation.

3.2 Relation Extraction System

To detect all relations in a document the system would need to compare all pairs of mentions in a document. This is computationally infeasible. On the ACE Corpus we are limited by the dataset to only pairs of mentions that are within the same sentence. For the JDPA Corpus we constrain ourselves to all pairs of mentions within a window of S sentences. In this research we look at a window size of between 1 and 3 sentences. Throughout this paper we will indicate sentence windows by referring to the range of sentences from which the mention pairs are drawn. So a 2-3 sentence window indicates that the extracted mention pairs are either in consecutive sentences or have one sentence in between them. A 1-1 sentence window means all mention pairs are in the same sentence.

The system extracts all pairs of mentions in the sentence window, and then classifies each pair of mentions as either having a relationship, having an inverse relationship, or having no relationship. So for the PartOf relation in the JDPA Corpus we consider both the relation "M_1 is part of M_2" and "M_2 is part of M_1" for a pair of mentions $< M_1, M_2 >$. The following subsections detail the system of how the documents are processed and how classification of these instances is accomplished. We leave the discussion of the actual features used for performing the classification to chapter 4.

For creating features we make use of the gold mentions, entity types, and co-reference groups from the corpora. Predicting this information would be a more realistic task, but in this work (similar to other relation extraction work) we focus solely on the detection of the relations.

3.2.1 Tokenization

Each document is first split into sentences using the OpenNLP Sentence Splitter, and then tokenized using the OpenNLP Tokenizer [1]. Additional custom post processing rules were written to improve the tokenization of the JDPA Corpus documents. In particular rules were written to tokenize the following examples that the standard OpenNLP tokenizer could not handle:

- "Foot/head/leg" into "foot" "/" "head" "/" "leg"
- Typos: "end.The" into "end" "." $< NewSentence >$ "The"
- Itemized lists: "2.It" into "2" "." "It"
- Number and unit: "120hp" into "120" "hp"; "2.0L" into "2.0" "L"
- Parenthesis/Brackets: "good)" into "good" ")"

In general the tokenization of the documents in the JDPA Corpus is a difficult problem because the documents have many typos and do not contain just a series of sentences, but rather also contain lists, headings, and other structure that the authors added to the documents. For instance, one troublesome document consists of a single long sentence with the clauses separated by commas. There is certainly additional work that could be done to improve the tokenization on

this dataset. Additionally, the current structure of separating data simply into sentences may not be the best breakdown when dealing with some of the documents which contain numbered lists or headings in different parts of the document. We speculate that summarizing this structure using a tree kernel may provide additional useful information for the classifier, but leave this to future work.

3.2.2 Parsing and Phrase Chunking

The Stanford Parser [15] is used to generate constituent parse trees and part of speech tags for the sentences in the documents. As per Zhou et al 2005 [30] we also use the Perl script written by Sabine Buchholz from Tilburg University (`http://ilk.kub.nl/~sabine/chunklink`) to detect phrases in the parsed sentences, as well as to build the dependency trees used for determining the head words of phrases.

3.2.3 Relation Classification

The classification of the relation between mention pairs was done using support vector machines implemented with uSVM [21], a variant of SVMLight-TK [17][11]. Section 3.3 details how and why we used uSVM.

For each mention pair $< M_1, M_2 >$ we must consider all of the relations and the inverse relations the two mentions could have to each other. On the JDPA Corpus this means considering 11 classes (5 relations, 5 inverse relations, and 1 $NoRelation$ class). So we train our binary classifiers to detect both "M_1 is part of M_2" and "M_2 is part of M_1" using separate binary classifiers. Each classifier is implemented as a 1 versus rest classifier. Applying these classifiers to the $< M_1, M_2 >$ instances (and their associated features) provides a set of predictions for relations between the mentions.

Additionally, there are a small number of cases in both corpora where some of the relations between the mentions/entities are symmetric. In the JDPA Corpus these are probably errors in the corpus, whereas in the ACE Corpus, some of the relations are sometimes symmetric. We make

no attempt to judge whether the annotations are wrong or not, and simply consider symmetric relations to be another relation class to be predicted (eg S_PART_OF). So for those cases where a symmetric relation occurs we also built classifiers for those symmetric classes. On the JDPA Corpus this results in a total of 14 different classes (3 symmetric ones).

We build three separate sets of classifiers (c0, c1, and c2+) depending on the number of sentences between the two mentions. Each mention pair is classified according to the distance between the mentions: pairs in the same sentence (c0), pairs in consecutive sentences (c1), and pairs with at least one sentence in between (c2+). On the JDPA Corpus this means training a total of 39 binary classifiers ($(14-1)\times 3$) to extract relations for the 14 classes with a window of 3 or more sentences.

Since the ultimate goal is to find relations between the co-reference groups not just the mentions, a second step combines the predicted mention relations between a pair of entities to form a single predicted relation between the entities. This combination is done by a simple voting scheme where each vote for a relation counts once, except for the *NoRelation* class, which is only assigned between two entities if no other relation was predicted between their mentions.

Effectively this converts our system into an ensemble of mention pair classifiers whose results are combined to predict the relations between entities. Across a series of sentences there may only be one or two places where the text indicates a relation between two entities. This voting scheme is biased towards any single detection of a relation and will override any times no relation was detected. Training a classifier to combine the results from the separate mention predictions would quite likely improve upon this system, but we leave this to future work.

3.3 SVM Classification

The uSVM software that we use to produce our binary SVM classifiers uses a uniform sampling technique applied to the cutting planes algorithm to drastically decrease the training time of the models [21]. Without this approximation method, running with tree kernels on this project would be impractical as we estimate that exact training with SVMLight-TK [17][11] would take

over a week to train a single binary classifier. Furthermore, uSVM allows changing the sampling rate to reduce the run time (and of course decrease accuracy). This is useful for quickly testing new features, but also for determining the best parameters to use for training.

As suggested in Severyn and Moschitti's paper on uSVM we use a sampling rate of 100 when searching for optimal training parameters [21]. For our final model training we use a sampling rate of 1000. In Severyn and Moschitti's work they found that a sampling rate of 5000 was necessary to train a model as accurately as the exact approach of SVMLight-TK. However, due to time constraints we were unable to train all of our classifiers at this higher accuracy level. Based on Severyn and Moschitti's work we estimate this reduces our classifier's performance by 1-2 points overall.

SVM tree kernels function in a similar way to using a vector of features with a SVM. Each positive or negative instance in the training set is associated with one or more trees rather than a vector of features. Rather than the SVM directly separating the features with a hyperplane, the trees are then decomposed into a sequence of frequently occurring substructures within the trees. The presence of these substructures then becomes a binary vector summarizing each instance. It is this binary substructure vector which the SVM classification algorithm then operates on[17].

For performing classification with tree kernels we use Moschitti's SST tree kernel, the default tree kernel for uSVM and SVMLight-TK. We did not compare our results to using other tree kernels, although there is some indication that other tree kernel methods may be able to extract more complicated substructures. When combining tree kernels with feature vectors we exclusively used a linear kernel. Many results have shown that a polynomial kernel performs better. We chose a linear combination solely to reduce the training time for our models. Appendix A also provides example command lines for training our models.

When optimizing the training parameters there are actually two different measures to optimize for: the F-Measure for relations between mentions, and the F-Measure for the relations between entities. We found that the maximum F-Measure between mentions often occurred when the recall was slightly higher than the precision. However, for an ensemble classifier such as the one

predicting the relations between entities, having low precision in the individual classifiers will cause the precision of the ensemble to be significantly lower. For this reason when training the individual classifiers that were part of an ensemble we did not choose parameters to maximize the mention F-Measure, but rather chose a point where the precision was reasonably high (above 50%) without reducing the recall too much. Unfortunately, time constraints prevented running enough tests to determine the optimal parameters by searching for the maximum entity relation F-Measure.

In Appendix A we document the specific parameters we used for training our models.

3.4 Evaluation Metrics

Like most other relation extraction work we use precision(P), recall(R), and F-Measure(F) to evaluate our system. We perform this evaluation on two sets of results: relations between mentions, and relations between entities (co-reference groups). For clarity, in the remainder of this paper we refer to these two different metrics as MentRel and EntRel respectively.

Given the total number of true positive (TP), true negative (TN), false positive (FP), and false negative (FN) results for a class, we calculate recall for a single relation type c and an inverse relation type of $\dot{c}$ as follows:

$$Recall_c = \frac{TP_c + TP_{\dot{c}}}{TP_c + FN_c + TP_{\dot{c}} + FN_{\dot{c}}} \tag{3.1}$$

Similarly, we calculate precision for a single class as follows:

$$Precision_c = \frac{TP_c + TP_{\dot{c}}}{TP_c + FP_c + TP_{\dot{c}} + FP_{\dot{c}}} \tag{3.2}$$

To get the overall precision and recall results across all classes C we exclude the results on the $NoRelation$ class, and calculate as follows:

$$Recall_{Overall} = \frac{\sum_{c \in C, c \neq NoRelation} TP_c + TP_{\dot{c}}}{\sum_{c \in C, c \neq NoRelation} TP_c + FN_c + TP_{\dot{c}} + FN_{\dot{c}}} \tag{3.3}$$

$$Precision_{Overall} = \frac{\sum_{c \in C, c \neq NoRelation} TP_c + TP_{\dot{c}}}{\sum_{c \in C, c \neq NoRelation} TP_c + FP_c + TP_{\dot{c}} + FP_{\dot{c}}} \tag{3.4}$$

Finally to calculate the F-Measure for a class c we use the harmonic mean of the precision and recall values:

$$FMeasure_c = \frac{2 \times Recall_c \times Precision_c}{Recall_c + Precision_c} \tag{3.5}$$

In most research, MentRel has been the standard by which those systems have been judged. This is largely a consequence of corpora such as ACE defining their relations between mentions, and so that is the most natural method of reporting results. We evaluate MentRel in order to compare against previous work, as well as for examining the results from individual classifiers since they are directly classifying relations between pairs of mentions. However, the end goal of our system is to find relations between the entities. Chan and Roth also evaluated their ACE relation extraction system on the basis of precision and recall on EntRel [3], and we will do the same.

We believe that the EntRel metric is a better measure of how many relations have been extracted from a set of documents (the ultimate goal of relation extraction). The MentRel metric only measures the effectiveness of the system at determining relations between those mention pairs that have actually been extracted from the document. One could look at all pairs of mentions across the entire document, but doing so results in an exceptionally large number of mention pairs to consider, making this calculation computationally infeasible. The EntRel metric reduces the total number of pairs such that it is more feasible to consider all pairs of entities. Furthermore, MentRel is biased towards relations between entities with a large number of mentions of each entity. EntRel considers each related entity only once.

For instance, consider a hypothetical case where a document has two entities (E_1 and E_2) with a Produces relation between them. Each entity is mentioned 10 times in the document so there are a total of 100 related mention pairs due to this single related entity pair. Also in the document are an additional 9 related entity pairs with each pair related with a PartOf relation.

Each of the 18 entities (E_3 through E_{20}) in these 9 pairs is mentioned only a single time so there are 9 related mention pairs for these 9 related entity pairs. In total the document has 10 related entity pairs and 109 related mention pairs.

Now assume that our system provides 100% recall of the Produces relation, but 0% recall of the PartOf relation. In this case, the overall MentRel recall will be 91.7% (100/109) while the EntRel recall will be 10% (1/10). We contend that the second measure is a much better metric for how much information the system has extracted since it says that only 10% of the relations in the document were found. After the first Produces relation is found between a mention pair involving entities E_1 and E_2, each subsequent relation found between those entities provides significantly less (if not zero) value since the relation has already been found. Certainly subsequent detection of relations increases the confidence in the original prediction, but the MentRel metric treats all mention pair relations equally, which skews the results to focus on those entities with many mentions.

However, since prior work largely uses the MentRel metric, we also report our results using this metric for comparison purposes. The MentRel metric is also a useful measure for how well a classifier is performing at detecting relations between mention pairs since it is a more direct measure of the instances the classifier is working with.

Chapter 4

Relation Features

In our experiments we compare three sets of features, and combinations of those features: baseline vector features, extended vector features, and tree kernel features.

4.1 Baseline Vector Features

We used Zhou et al.'s lexical features [30] as the baseline features of our system similar to what other researchers have done [3]. Additional work has extended these features [10] or incorporated other data sources (e.g. WordNet), but in this paper we focus solely on applying these same lexical features to the JDPA Corpus as a comparison baseline.

In our implementation, the Mention Level, Overlap, Base Phrase Chunking, Dependency Tree, and Parse Tree features are the same as Zhou et al. (except for using the Stanford Parser rather than the Collins Parser). The minor changes we have made are summarized below:

- **Word Features**: Nearly identical. Rather than using a heuristic to determine the head word of the phrase it is chosen to be the noun (or any other word if there are no nouns in the mention) that is the least deep in the parse tree. This change has minimal impact. Also, in generating features, we replace any numbers from the document with "$< NUMBER >$" in order to improve the generalization. This is helpful because of the frequent mentions of performance numbers when discussing cars.
- **Entity Types**: When running on the ACE Corpus these features are the same as Zhou et al. In the JDPA Corpus, some of the entity types indicate the type of the relation

(*CarFeature* and *CarPart*) and so we replace those entity types with "Unknown". The remaining entity types are: *Facility*, *Food*, *GeoPolitical*, *GeoPolitical-City*, *GeoPolitical-Nationalities*, *GeoPolitical-USStates*, *GeoPolitical-Countries*, *Location*, *Organization*, *Person*, *Time*, *Time-Date*, *Time-DaysOfTheWeek*, *Time-Month*, *Time-Year*, *Units*, *Units-Age*, *Units-Money*, *Units-Rate*, *Vehicles*, *Vehicles-Cars*, *Vehicles-SUVs*, *Vehicles-Trucks*

- **Semantic Information**: These features are specific to the ACE relations and so we did not implement them. In Zhou et al.'s work, this set of features increases the overall F-Measure by 1.5.

4.2 Extended Vector Features

An extended set of features were created to augment the baseline vector features. Inspired by Chan and Roth's work [3], this set of features largely focused on indicating which words around the two mentions were in the same co-reference group as either of the mentions and the distance from the mentions to co-referring mentions. These features especially start to help when the mentions are not in the same sentence. We also added bi-gram and part-of-speech tags as features since there is a lot of research showing that particular patterns of words and POS tags can be indicative to relations.

However, in the course of deciding which features to use across sentences it became clear that the possible features were too numerous to fully explore in the course of this project. There does not seem to be much relation extraction research that suggests effective features across multiple sentences. For this reason we turned to tree kernels to try and improve the relation extraction across sentences.

The extended features that were created are better than our baseline vector features, so we used them in our classifiers. A complete description of the features is in Appendix B.

4.3 Tree Kernel Features

As already discussed, much prior work has had success using tree kernels to detect relations [26][29][31]. Tree kernels were particularly useful to us because not much work has been done looking at relations across sentences, and so creating tree structures out of the multiple sentences and then allowing the learning algorithm to find the best features was more practical. An additional benefit is that the prior work was all done on a very different dataset, and using tree structures would remove some of the bias that our baseline vector features have towards the newswire and broadcast news transcripts in the ACE Corpus.

We used different trees depending upon whether the two mentions being classified were in the same sentence, in two consecutive sentences, or had at least one sentence between them. In the following subsections we detail how these trees are constructed.

4.3.1 Mentions in one sentence

How to structure the parse tree to determine the relation between two mentions in a sentence has been rather extensively studied. In particular Zhang et al. [26] showed that truncating the constituent parse to only include those nodes containing the two mentions as well as the nodes between the mentions (which they called the partial tree) was the most effective tree structure of seven that they experimented with. The partial tree (PT) is augmented by adding a node for each mention that is the parent of all of the words in that mention. We also looked at extending the PT to include any co-referenced mentions within the original parse tree, but in our limited experiments adding this additional context reduced the classification performance.

Zhou et al. [29] further explored a number of other methods of changing the partial tree to improve performance. We implemented three of the suggestions from their work: removing single-in and single-out nodes to compress the tree, expanding the context of the tree to include the predicate for certain mention pairs, and adding the entity types of the two mentions at the top of the parse tree. Neither compressing the tree, nor including the predicate improved the

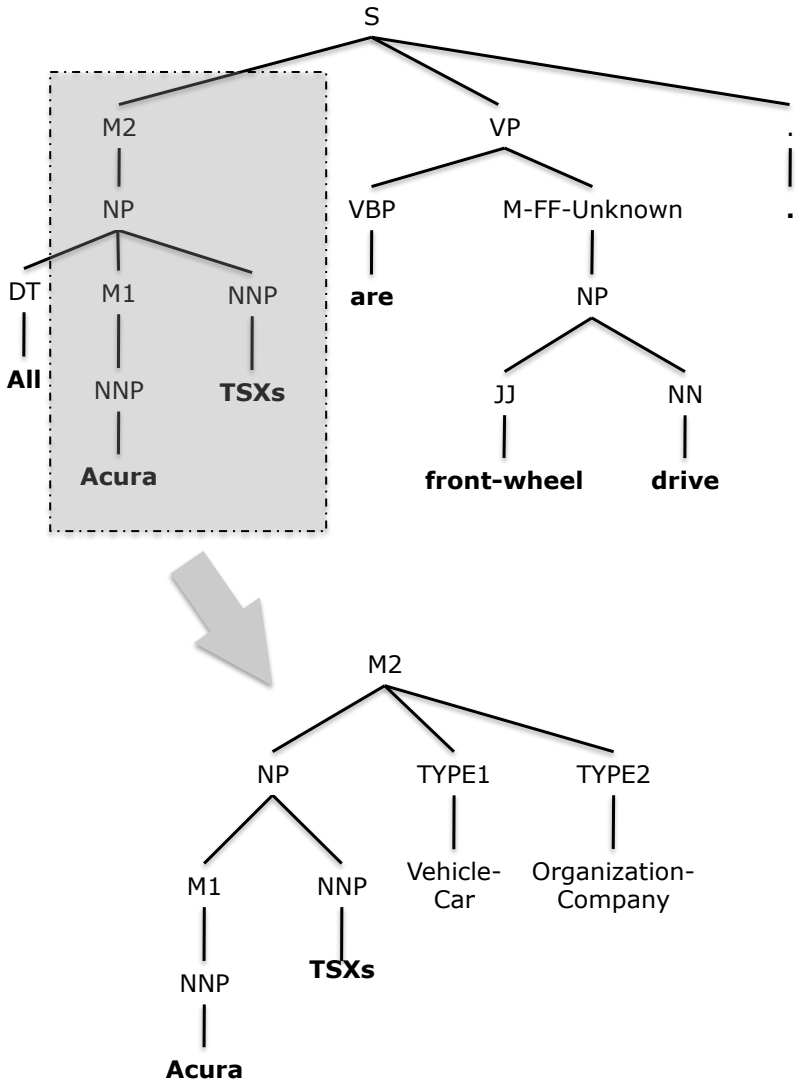

Figure 4.1: Production of the feature tree for the mentions "Acura" and "Acura TSXs" from the constituency parse of the sentence "All Acura TSXs are front-wheel drive." The mention phrase "front-wheel drive" is not in the co-reference group for either M_1 or M_2 and so its mention node contains the string "FF".

classification performance in our experiments on limited portions of the JDPA Corpus. However, adding the entity type at the top of the tree did substantially improve the results. We hypothesize that the other two methods did not work either because of the differences between the corpora, or because our parse trees were not of high enough quality, resulting in too much noise.

In figure 4.1 we show the full parse tree and final partial tree for the sentence

Example 4.1: All [$_{M_2}$ [$_{M_1}$ Acura] TSXs] are [$_{\text{M-FF-Unknown}}$ front-wheel drive].

where the two mentions being tested for a relation are "Acura" and "Acura TSXs". In addition to marking the other mention in the sentence "front-wheel drive" with its entity type, we have also added a binary encoding ("FF", "TF", "FT", or "TT") to indicate whether that mention is in the same co-reference group as M_1 or M_2. Further detail on the co-reference encoding is described in Appendix B.

4.3.2 Mentions in Consecutive Sentences

When two mentions occur in consecutive sentences there can often be an implicit relationship between the two mentions. For instance, in the sentences

Example 4.2: [$_{\text{M-FF-Person}}$ I] love the [$_{M_1}$ styling]. [$_{M_2}$ It] [$_{\text{M-FF-Unknown}}$ looks] and [$_{\text{M-FF-Unknown}}$ feels] like a foreign [$_{\text{M-FF-Vehicles-Car}}$ car].

The mention "styling" is implicitly a feature of the car being referred to with the mention "It". To build the trees for our features, we create tree T_1 by taking the constituent parse tree of the first sentence and removing all nodes to the left of the mention phrase that are not parents of the mention (Figure 4.2). Similarly, we build tree T_2 by removing all nodes to the right of the second sentence's mention phrase which are not parents of that phrase.

We experimented with then combining T_1 and T_2 into a single tree by adding a joint top node, however, this decreased classification performance. Therefore we used each tree separately

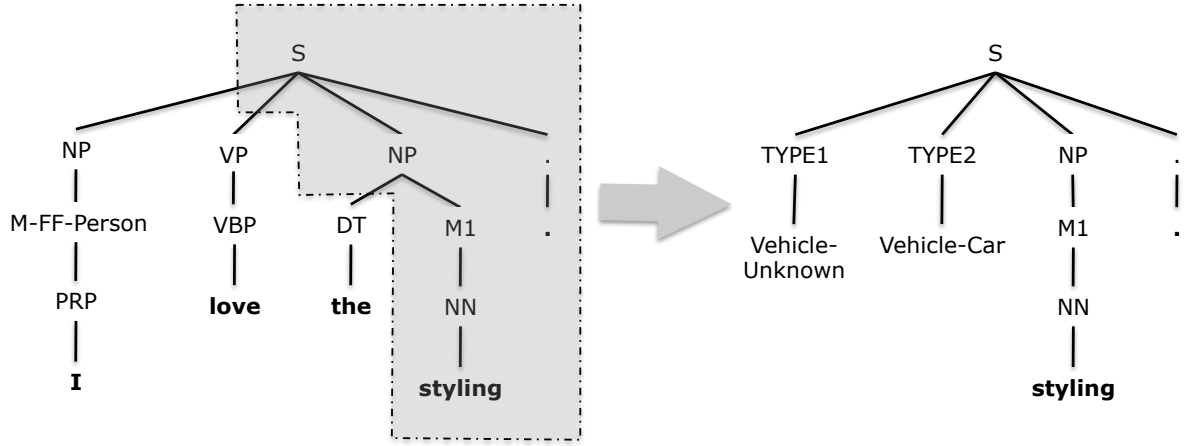

Figure 4.2: Tree production of T_1 with a single mention, "styling", in the parse tree.

as a feature for the mention pair, and attached to the top of each tree the entity types of the two mentions. Figure 4.3 depicts the the resulting trees T_1 and T_2 used as features for classifying the relation between the mentions "styling" and "It" in the above example.

4.3.3 Mentions with at Least One Sentence in Between

To illustrate relations across three sentences we will use the following example:

Example 4.3: This [$_{M_1}$ car] is good. [$_{\text{M-TF-Vehicle-Car}}$ It]'s [$_{\text{M-FF-Unknown}}$ powerful], has great [$_{\text{M-FF-Unknown}}$ handling] and is very [$_{\text{M-FF-Unknown}}$ fuel efficient]. My [$_{\text{M-FF-Person}}$ friend] has [$_{M_2}$ one].

With additional sentences between the two mentions, the trees for the sentences containing mentions M_1 and M_2 are produced the same as trees T_1 and T_2 above. To classify the relation between the mentions "car" and "one" (an InstanceOf relation) we generate the trees T_1 and T_2 as seen in Figure 4.4. However, using full trees for the sentence in between the two mentions would almost certainly add too much noise to be useful. This would become even more true as the number of sentences between the mentions increases above one.

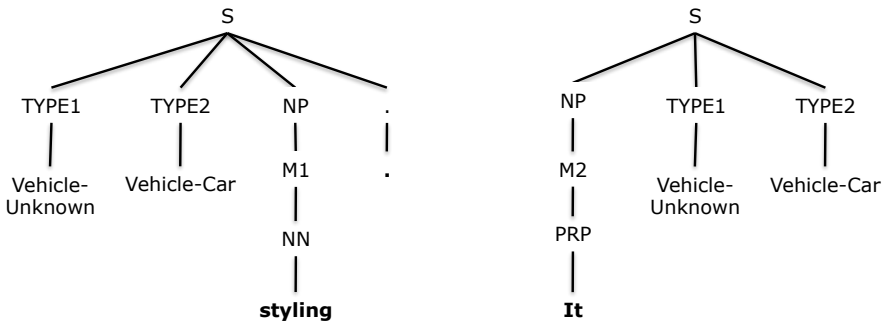

Figure 4.3: Feature trees T_1 and T_2 for the mentions "styling" and "It" in the consecutive sentences: "I love the styling. It looks and feels like a foreign car."

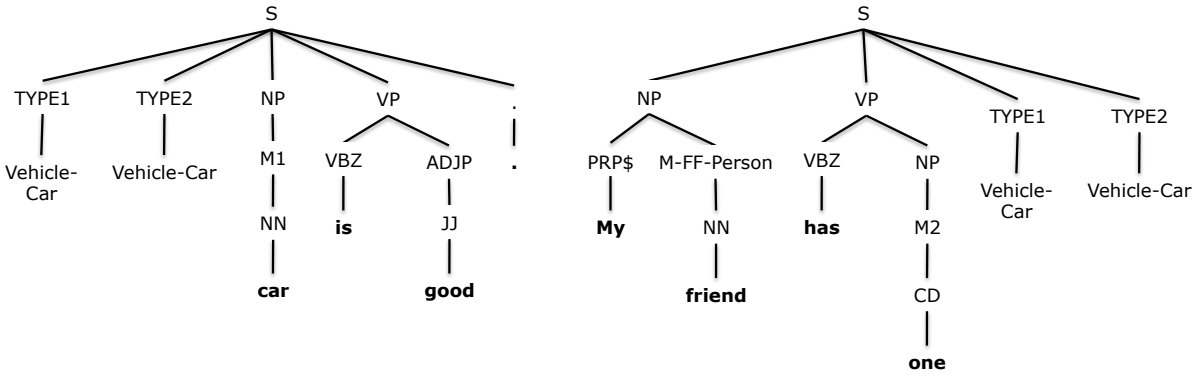

Figure 4.4: Feature trees T_1 and T_2 for the mentions "car" and "one" in the non-consecutive sentences: "This car is good. It's powerful, has great handling and is very fuel efficient. My friend has one."

Based loosely on the concept of centering[20], where the author of a document focuses on one or two entities before moving on to others, we created a tree structure to try and capture the entities being mentioned in the intervening sentences. The inter-tree T_I contains a parent node for each sentence in between the two mention sentences. Each sentence node then contains a sequence of children corresponding to the mentions in that sentence. These mention nodes indicate if the mention is co-referring to one (or both) of the mentions being classified. Additionally, any verbs in the sentence are added in among the sequence of mentions. Figure 4.5 depicts the generation of the inter-tree for the above example sentence. In limited experiments on our development set, adding T_I to our instances improved the MentRel F-Measure by 3.5 on mention pairs with one sentence in between them (a window of 3 sentences).

4.4 Feature Combinations

Prior research has found that combining vector and tree features together can improve classification accuracy because the two types of features tend to be complementary and reinforce one another [26][29][31]. We also investigate combining our tree kernels with the vector features to improve classification accuracy. In our experiments we look at the following combinations of features:

- Baseline Vector
- Baseline + Extended Vector
- Tree
- Baseline Vector + Tree
- Baseline Vector + Extended Vector + Tree

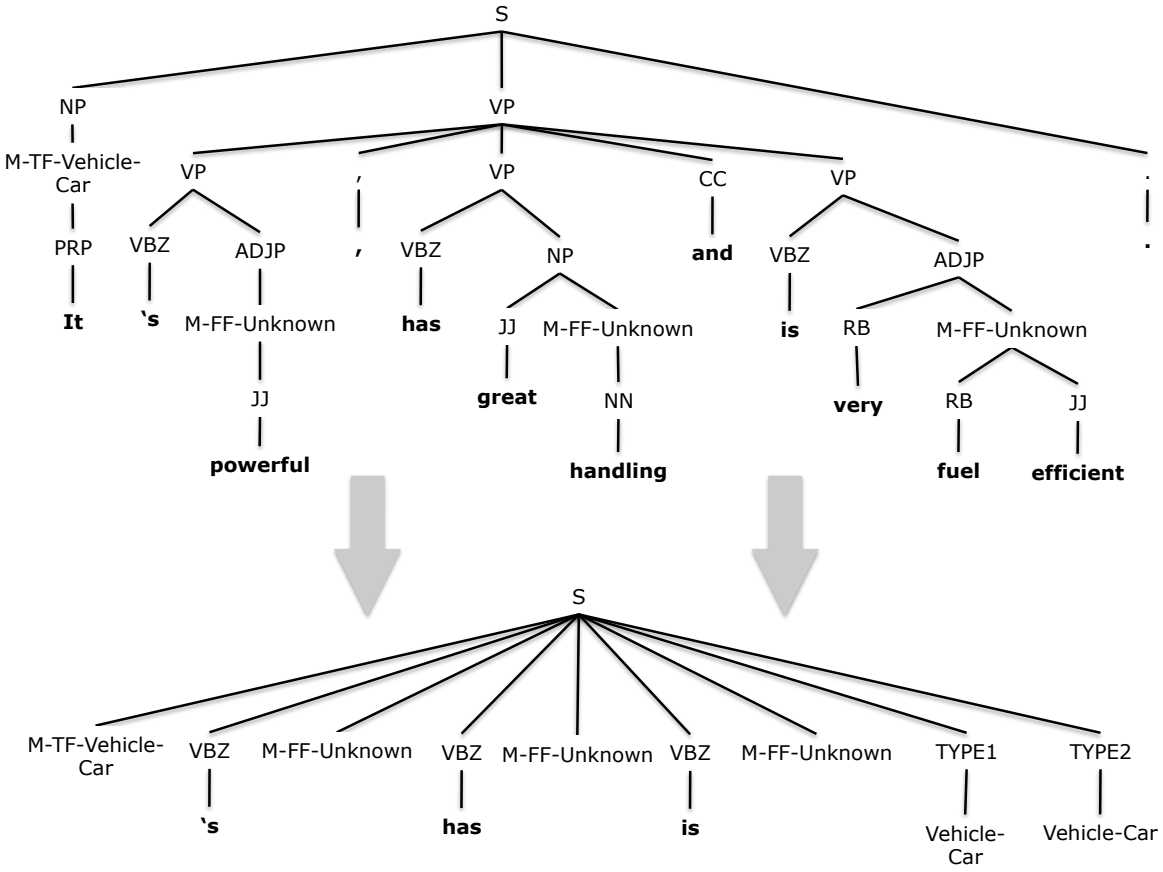

Figure 4.5: Inter-tree production for the sentence "It's powerful, has great handling and is very fuel efficient."

Chapter 5

Results and Discussion

5.1 ACE Results

Table 5.1 shows our results on both the ACE 2004 and JDPA Corpora test sets. The ACE MentRel results on our system were run with 5 fold cross validation across the seven coarse-grained relations. As a comparison, Table 5.2 lists some of the best results by other researchers on the ACE Corpus. This second table also lists (in parentheses) the results other researchers achieved on fine-grained relations. We did not examine fine-grained relations using our system for all combinations of our features. But on our baseline vector features we got a MentRel F-Measure of 50.5 (P=54.8, R=46.9) using 5 fold cross validation, about 5 points less than Zhou et al's 2005 results.

Our baseline features, a reimplementation of the Zhou et al 2005 features[30], performs about 10 points worse than Zhou's results on the ACE-2003 or ACE-2004 coarse-grained relations. As discussed in the features section (Section 4.1) we do not include a few features which accounts for

Feature Set	Sentence Window	ACE 2004 Corpus			JDPA Corpus		
		P	R	F	P	R	F
Baseline	1-1	61.4	57.3	59.3	35.6	49.2	41.3
Baseline + Extended	1-1	54.6	55.2	54.8	37.2	50.2	42.7
Tree	1-1	55.8	55.1	55.4	46.9	38.2	42.1
Tree + Baseline	1-1	61.8	60.0	**60.9**	50.0	38.1	43.3
Tree + Baseline + Extended	1-1	57.9	60.0	58.9	40.1	54.7	**46.3**

Table 5.1: Overall MentRel results for ACE and JDPA Corpora for mentions occurring in the same sentence.

a shortfall of about 1.5 points. Using uSVM's sampling technique to reduce the training time also probably costs us about 2 points, and we only use linear kernels while some of the other research has been able to get modest improvements by using polynomial kernels. These differences though, only account for at most a 4 point difference. We speculate that some of the remainder is due to differences in the parser and tokenization methods that we are using.

Similarly, we see that our non-composite tree kernel gets a much lower score when compared to Zhou et al's 2010 tree kernel that our tree is based off of (with the entity type attached to the top of the tree). This difference of 13 points further suggests that the problem is in the parsing of the documents. Indeed, Zhou et al noted that: "the final performance of semantic relation extraction may change greatly with a different range of syntactic parsing errors."[29] In their research they use the Charniak parser[4] in order to reduce the errors in their parse trees. Furthermore, they perform pre-processing on the sentences, and post-processing of the parse trees in order to improve the parsing and particularly to prevent mentions from being spread erroneously across different branches of the parse tree. We find it likely that Zhou et al's particular attention to the quality of their parse trees has greatly helped their system in competing with others.

The extended features significantly reduce the system's performance on the ACE Corpus whether they are combined with a tree kernel or not. These features were optimized specifically to improve performance on the JDPA Corpus, so it is interesting to see how much they hurt performance on the ACE data. We believe that the extended features add too much noise when being applied to the ACE Corpus. The JDPA Corpus contains many more mention pairs and positive relation examples, so perhaps the increased number of examples enables these features to be useful. Or perhaps the baseline features are so well optimized for the ACE Corpus that any additional noise from other features significantly reduces performance.

Overall, we are surprised at the large difference between our results and Zhou's with approximately the same features. The fact that we get significantly worse performance with both tree kernels and our baseline vector features suggests that the problem is common to both methods, thus leading to our suspicions about the parser and tokenization.

Author	ACE Dataset	Features	P	R	F
Zhou et al 2005 [30]	2003	Linear Kernel	77.2 (63.1)	60.7 (49.5)	68.0 (55.5)
Zhou et al 2010 [29]	2004	Linear Kernel	78.2 (-)	63.4 (-)	70.1 (-)
Jiang and Zhai 2007 [10]	2004	Linear Kernel	74.6 ()	71.3 (-)	72.9 (-)
Bunescu and Mooney 2005 [2]	2003	Seq. Kernel	65.5 (-)	43.8 (-)	52.5 (-)
Zhou et al 2010 [29]	2004	Tree Kernel	76.8 (-)	62.1 (-)	68.6 (-)
Zhang et al 2006 [26]	2003	Tree Kernel	76.3 (62.4)	63.0 (48.5)	68.7 (54.6)
Zhao and Grisham 2005 [28]	2004	Composite	69.2 (-)	70.5 (-)	70.4 (-)
Zhou et al 2010 [29]	2004	Composite	83.1 (71.2)	73.5 (64.2)	77.8 (67.5)

Table 5.2: ACE overall MentRel obtained by other researchers on the 7 type relations, and 23 subtype relations in parenthesis.

Relation	P	R	F	Total Positive Instances
FeatureOf	40.7	60.4	48.7	1736
InstanceOf	49.8	33.9	40.3	348
MemberOf	34.3	9.6	14.9	387
PartOf	37.3	52.2	43.5	1919
Produces	43.3	82.0	56.7	724
Overall	40.1	54.7	46.3	5114

Table 5.3: MentRel results on the JDPA test set for mentions in the same sentence using the combined Baseline, Extended, and Tree features.

Relation	P	R	F	Total Related Entity Pairs	Related Entity Pairs Outside Single Sentence Window
FeatureOf	38.6	49.1	43.2	1667	382
InstanceOf	46.4	20.3	28.2	419	183
MemberOf	36.0	5.8	10.0	553	259
PartOf	35.0	34.6	34.8	2122	793
Produces	38.5	79.9	51.9	368	43
Overall	37.3	38.3	37.8	5129	1660

Table 5.4: EntRel results on the JDPA test set using the combined Baseline, Extended, and Tree features to predict mentions in the same sentence.

5.2 JDPA MentRel Results

As compared to the ACE MentRel results, the JDPA results are quite a lot worse. Table 5.3 shows the JDPA mention results of each relation using our composite kernel using the baseline and extended features combined with the tree kernel. These results are from looking only at mention pairs that occur in the same sentence.

It is important to point out that the JDPA relations are more similar to the ACE fine-grained relations than to the ACE coarse-grained relations. Still, our system is not performing nearly as well as other systems perform on the ACE fine-grained relations (46 vs. 67). As previously discussed we attribute some of this difference to parsing and other issues; however, it seems likely that a significant portion is also due to the difference between well edited news articles in ACE as opposed to the blogs and social media documents from the JDPA Corpus. In an analysis that limits the JDPA Corpus test documents to only those that were written by JDPA professional authors for their Power Steering blog, we found that using our baseline features our system achieved a 49.9 MentRel F-Measure, 8.6 points higher than our results on the entire test set. This suggests that higher editing standards do indeed contribute to better relation extraction performance, presumably due to more uniform use of language.

From Table 5.3 it becomes clear that there is an interesting difference between the results on the MemberOf and InstanceOf relations. As shown in Table 2.2 these two relations occur much less frequently than the others, and often do not occur within the same sentence, further reducing the number of examples to learn from. But our system performs 25 points better on the InstanceOf relation. It is possible this is an artifact of the dataset we are using. Since all of the documents are about cars there will often be discussion of a particular car which is an instance of a previously mentioned model, whereas the MemberOf relation may have a wider range of related entities.

The Produces relation performs by far the best. It seems likely that this is due to the short average distance between mention pairs for this relation. Very often this relation is between two overlapping mention phrases such as when a car is referred to both by the manufacturer and by

the model in a single, multi-word noun phrase.

In addition to the suspected problems we have with our parsing across the ACE Corpus, a further problem arises when using the JDPA Corpus: many "sentences" are not complete English sentences but fragments or headings or list items. For instance, the following fragments' part of speech tags were generated with our parser:

Example 5.1: [$_{\text{DT}}$ No] [$_{\text{NN}}$ radiator] [$_{\text{NN}}$ fan] [$_{\text{VBP}}$ issue] [. .]

Example 5.2: [$_{\text{UH}}$ No] [$_{\text{VB}}$ worn] [$_{\text{NN}}$ piston] [$_{\text{NNS}}$ rings] [. .]

In both cases the parser wrongly tags a word as a verb when there is no verb in the fragment. The JDPA Corpus contains many sequences of text which do not fall into the category of being true sentences. Often the structure of the text (such as lists of car features) is more indicative of the relationship within the text than any sentence structure. Our current system's approach does not allow any way to encode these sorts of structural relationships between lines of text, but it seems that future work could encode this information using tree kernels in a similar way to how our inter-sentence tree was constructed.

Our results also confirm the generally accepted idea that optimized vector features will perform better than a tree kernel itself. Using just the linear baseline features works better than the tree kernel on ACE, and the baseline+extended features work better than the tree kernel on the JDPA Corpus. In both cases the vector features have been optimized to work well on their respective datasets. But those optimized features do not perform as well on the data from the other corpus. On the JDPA data, the baseline features perform worse than the tree kernel, whereas the reverse is true on the ACE data that the baseline features were optimized for. Furthermore, our results confirm that combining feature vectors with tree kernels produces results which are better than either set of features by themselves.

Sentence Window	MentRel P	MentRel R	MentRel F	EntRel P	EntRel R	EntRel F	Untested Relations
1-1	40.1	54.7	46.3	37.3	38.3	37.8	1660
1-2	42.5	35.3	38.6	37.6	39.1	38.3	834
1-3	44.0	29.9	35.6	37.8	40.1	**38.9**	525
2-2	58.6	13.5	21.9	56.6	5.8	10.5	2892
3-3	56.6	15.3	24.1	56.4	5.1	9.4	3399

Table 5.5: Overall MentRel and EntRel results on the JDPA Corpus using the Baseline + Extended + Tree feature set. The sentence window gives the range of sentences from which mention pairs to classify were extracted. So "2-2" indicates that the system was run only on those mention pairs in two consecutive sentences (i.e., using only classifier c1).

5.3 JDPA EntRel Results

As we argued in Section 3.4 we feel that the EntRel metric is a much better means of measuring how much information a relation extraction system has extracted from a set of documents. From an information point of view (assuming one has the co-referring groups of mentions) once a single relation between a pair of mentions is known, knowing the relation between any additional mention pairs between those two entities provides one with no additional information. Table 5.4 shows our detailed EntRel results on mentions within a single sentence, and Table 5.5 summarizes our EntRel and MentRel results on the JDPA Corpus across multiple sentence windows. An additional benefit of the EntRel metric is it considers relations between entities that are not detected because they are outside of the sentence window. Whereas the mention relation metric only considers those relations that are within the sentence window. This difference accounts for why the EntRel results are lower than MentRel. For instance, by evaluating the EntRel metric for the 1-1 sentence window only on those entities that are mentioned in a single sentence (i.e. excluding the 1660 untested relations) we get an EntRel F-Measure of 45.0, significantly higher than the 37.9 score when including the untested relations.

For this reason, we see that as the sentence window increases the number of untested entity relations decreases, resulting in a modest increase in the overall EntRel rate. This increase is despite the fact that the MentRel rate decreases as the sentence window increases. Overall, the system

extracts more information about relations as the sentence window increases.

5.4 JDPA Ensemble Entity Pair Classification

The ensemble combination of mention relation predictions to form the entity relation predictions also has implications for how EntRel improves as the sentence window increases. Table 5.6 depicts the percentage of entity relations which have Y mention pairs expressed within a window of X sentences. So with a sentence window of 1-1 (i.e. only looking at mention pairs in the same sentence), 54.79% of entities with a relation between them have only one place where both entities are mentioned in the same sentence. From this data it can be seen that as the sentence window increases the number of relations with more than one mention pair increases from about 20% to over 50%. This increase means that our ensemble classifier has to deal more and more with combining separate mention pair votes to determine the relation between two entities.

As stated in Section 3.3, for our single sentence relation detection we have optimized our learning parameters largely to maximize the MentRel metric. With a sentence window of 1-1, maximizing the MentRel seems to also approximately maximize EntRel. However, as the sentence window is increased, the increase in the number of mention pairs per entity relation results in increased classification errors. This is due to the precision of the results being lower when the mention pairs are in different sentences.

Since the MentRel precision indicates the probability that a particular mention pair prediction is correct we optimized the c1 and c2+ classifiers so that the MentRel precision was above 50%. This increases the likelihood that a mention pair prediction will be correct and that it will help rather than hurt when combining the mention pair predictions with our voting scheme. Quite a bit of further work should be done to improve on the best ways to combine the multiple mention pair predictions into a single entity pair prediction. From our current work, the key point is that combining the results from multiple mention pairs can improve the total number of entity relations that get extracted from a document.

As further evidence that combining mention pair predictions into a single entity pair predic-

Mention Pair Count	% of Relations at Window Size				
	1-1	1-2	1-3	1-4	1-5
0	27.63%	13.64%	8.58%	6.20%	4.58%
1	54.79%	45.60%	41.87%	39.70%	38.72%
2	11.28%	18.85%	17.02%	15.71%	14.39%
3	3.34%	8.75%	10.49%	9.65%	8.98%
4	1.44%	5.38%	7.03%	7.92%	7.92%
5	0.38%	2.69%	4.22%	4.67%	5.07%
6	0.34%	1.92%	3.61%	4.88%	4.97%
7	0.13%	0.66%	1.71%	2.37%	3.31%
8	0.20%	0.68%	1.62%	2.42%	2.92%
9	0.05%	0.32%	0.70%	1.25%	1.73%
10+	0.42%	1.51%	3.16%	5.24%	7.41%

Table 5.6: Frequency of occurrence of mention pairs that are a part of a relation as the sentence window size increases. The "Mention Pair Count" is the number of times a mention pair for a particular relation occurs. The "% of Relations at Window Size" is the overall percentage of relations with that number of mention pairs at the given window size. The data was generated by examining all relations in the JDPA Training set which has a total of 22622 relations.

Number of Mention Pairs	Total Mention Pairs		Non-NULL Mention Pairs	
	Entity Count	F	Entity Count	F
1	58543	37.3	4044	46.7
2	11647	39.1	767	56.3
3	3914	42.0	293	61.2
4	2550	42.0	143	67.5
5	1280	48.3	54	74.2
6	942	57.3	51	77.8
7	368	43.5	19	84.8
8	355	51.6	16	74.1
9	188	57.5	13	76.2
10+	885	48.8	39	78.1

Table 5.7: A breakdown of the EntRel results for the 1-3 sentence window based on the number of mention pairs that were classified. Columns two and three provide a breakdown for those entities that had the specified number of mention pairs occurring within the sentence window. Columns four and five indicate the breakdown based on the number of mention pairs that were classified to have a relation other than NoRelation (i.e. Non-NULL).

tion improves performance, we created a breakdown of the EntRel results based upon the number of mentions that occur within the sentence window. In the second and third columns of Table 5.7 we show the EntRel F-Measure for those entity pairs that have a certain number of mention pairs occurring within the three sentence window. Quite clearly, as the number of mention pairs for a particular entity pair increases, the EntRel F-Measure increases from the high 30s with a single mention pair, up to the high 40s and mid 50s when there are 8 or more mention pairs. The more opportunities that the system has to classify pairs of mentions from two entities, the more likely it is that the system will correctly determine the relationship between the entities.

The fourth and fifth columns of Table 5.7 breaks down the EntRel results based on the number of those mention pair predictions within the sentence window that were predicted to have a relation other than NoRelation. These results make it even more clear that the more often the system has more than one mention pair prediction, the better the system will perform. As previously discussed, this area of combining mention pair predictions to create entity pair predictions is ripe for the application of a classifier rather than a voting scheme for combining these predictions. But even our simple voting ensemble demonstrates the usefulness of combining mention pair predictions.

Chapter 6

Conclusions and Future Research

6.1 Conclusions

In this thesis we have applied a number of existing and novel techniques to the problem of relation extraction on the J.D. Power and Associates Sentiment Corpus. We implemented Zhou et al's set of vector features[30] and found that they do not seem to perform nearly as well on the relations defined within the JDPA Corpus. This disparity appears to be both due to the types of relations and data in the JDPA Corpus, and also that these vector features are at least somewhat tuned to the ACE Corpus. To support this we found that a tree kernel can beat these vector features on the JDPA Corpus while on the ACE Corpus a tree kernel is unable to perform as well as the optimized feature vectors. This confirms the conventional wisdom that optimized feature vectors will beat a tree kernel, but also that combining vectors with tree kernels will perform better than either method by itself.

We find that tree kernels are an effective method of extracting relations from new datasets and avoid the problems of over-fitting the features to match any one particular corpus. In examining features for relation extraction we found that co-reference information can improve the Zhou et al features on the JDPA Corpus, but these features significantly reduce performance on the ACE Corpus.

Overall our results on both the ACE and JDPA Corpora seem fairly low. We attribute at least some of this error to poor parsing and tokenization of the documents based on some anecdotal reviews of parse trees and the reported methods which other researchers have used to improve the

parsing on their own systems. Our ACE results are worse than other researchers whether we use tree kernels or Zhou's 2005 features, providing further evidence that the tokenization and parsing that is common to both methods is partly to blame.

We have also argued that the primary metric used by most researchers for reporting relation extraction results (F-measure of relations between mention pairs) has underlying flaws when the ultimate goal is to determine the relations between entities being discussed in a document. This mention pair metric is sensible given the annotations which were done on the ACE Corpora, but seems inappropriate when the annotated relations are between entities in the documents rather than mentions. Future relation annotation projects should annotate the relations between entities rather than limiting the annotations to relations between mentions in the same sentence.

Our system predicts relations between entities using an ensemble voting method to combine the predictions of relations between pairs of mentions. Although significant further work could be done in how to perform this ensemble classification, we showed that by increasing the number of mention pairs considered, the overall performance of finding entity relations can be improved. In our system, we consider a window of sentences in which all pairs of mentions are classified according to their relation. Combining the predictions from multiple mention pairs yields improved entity relation extraction results.

6.2 Future Research

Since this research brings up a number of interesting directions to pursue, we present here a list of the individual aspects that appear to need further research.

- The effect of parsing and tokenization on relation extraction, in particular how they affect tree kernels.
- Combining mention pair predictions to create entity pair relation predictions. In particular, creating an ensemble classifier to combine these results seems like an obvious way to improve the relation extraction.

- Using tree kernels to summarize additional structure within a document such as headings, paragraphs, and itemized lists. A lot of structure in the documents is lost by simply considering a document a sequence of sentences, and in at least some cases this structure seems like it would be useful for relation extraction.

- Although in this paper we have discussed how the EntRel metric provides a better assessment of how much information a relation extraction system extracts, we have done so only within the context of a system that has complete knowledge about both the mention phrases and the sets of mention phrases that constitute the entities within the document. How this metric, and this type of system in general works on the more general task of joint mention, entity, and relation detection is unknown, and has not been studied.

- Detection of relations between mentions in different sentences could certainly benefit from future work on determining the best features to use.

- There are a number of other types of tree kernels, and some research has suggested that other kernels may be better at extracting complex features from the parse trees.

- Our system builds separate classifiers based on whether the mentions are in the same sentence, consecutive sentences, or have sentences in between. In retrospect, it is not clear this was a good decision. The added complexity may be unnecessary, and it is possible that combining the examples together may provide the classifier with additional training instances.

Bibliography

[1] Apache Software Foundation. OpenNLP. http://incubator.apache.org/opennlp/, version 1.5.0.

[2] R Bunescu and R Mooney. A shortest path dependency kernel for relation extraction. HLT '05 Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing, pages 724–731, 2005.

[3] Y S Chan and D Roth. Exploiting background knowledge for relation extraction. COLING '10 Proceedings of the 23rd International Conference on Computational Linguistics, pages 152–160, 2010.

[4] E Charniak. Immediate-head parsing for language models. Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'2001), pages 129–137, 2001.

[5] M Gerber and J Chai. Beyond Nombank: A study of implicit arguments for nominal predicates. ACL '10 Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, 2010.

[6] R Girju, A Badulescu, and D Moldovan. Automatic discovery of part-whole relations. Computational Linguistics, pages 83–135, 2006.

[7] R Girju, D Moldovan, M Tatu, and D Antohe. On the semantics of noun compounds. Computer speech & language, pages 479–496, 2005.

[8] R Girju, P Nakov, V Nastase, S Szpakowicz, P Turney, and D Yuret. Semeval-2007 task 04: Classification of semantic relations between nominals. Proceedings of the 4th International Workshop on Semantic Evaluations, pages 13–18, 2007.

[9] M.A Hearst. Automated discovery of wordnet relations. WordNet: an electronic lexical database, pages 131–151, 1998.

[10] J Jiang and C Zhai. A systematic exploration of the feature space for relation extraction. Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics (NAACL-HLT'07), pages 113–120, 2007.

[11] T Joachims. Making large-scale SVM learning practical. Advances in Kernel Methods - Support Vector Learning, 1999.

[12] N Kambhatla. Combining lexical, syntactic, and semantic features with maximum entropy models for extracting relations. Proceedings of the ACL 2004 on Interactive poster and demonstration sessions, 2004.

[13] J Kessler, M Eckert, L Clark, and N Nicolov. The ICWSM 2010 JDPA Sentiment Corpus for the automotive domain. 4th International AAAI Conference on Weblogs and Social Media Data Challenge Workshop (ICWSM-DCW 2010), 2010.

[14] J Kessler and N Nicolov. Targeting sentiment expressions through supervised ranking of linguistic configurations. 3rd International AAAI Conference on Weblogs and Social Media, 2009.

[15] D Klein and C Manning. Accurate Unlexicalized Parsing. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pages 423–430, 2003.

[16] A Mitchell, S Strassel, S Huang, and R Zakhary. ACE 2004 Multilingual Training Corpus. Linguistic Data Consortium, Philadelphia, 2005.

[17] A Moschitti. Making tree kernels practical for natural language learning. Proceedings of the Eleventh International Conference on European Association for Computational Linguistics, 2006.

[18] P Nakov and M Hearst. Using verbs to characterize noun-noun relations. Artificial Intelligence: Methodology, Systems, and Applications, pages 233–244, 2006.

[19] M Poesio. The MATE/GNOME proposals for anaphoric annotation, revisited. Proceedings of SIGDIAL, 2004.

[20] M Poesio, B Di Eugenio, R Stevenson, and J Hitzeman. Centering: A parametric theory and its instantiations. Computational Linguistics, pages 309–363, 2004.

[21] A Severyn and A Moschitti. Large-Scale Support Vector Learning with Structural Kernels. ECML PKDD'10 Proceedings of the 2010 European conference on Machine learning and knowledge discovery in databases, 2010.

[22] R Snow, D Jurafsky, and A Ng. Learning syntactic patterns for automatic hypernym discovery. Advances in Neural Information Processing Systems (NIPS 2004), 2005.

[23] R Vieira and M Poesio. An empirically based system for processing definite descriptions. Computational Linguistics, 2000.

[24] M E Winston, R Chaffin, and D Herrmann. A taxonomy of part-whole relations. Cognitive Science, pages 417–444, 1987.

[25] D Zelenko, C Aone, and A Richardella. Kernel methods for relation extraction. The Journal of Machine Learning Research, pages 1083–1106, 2003.

[26] M Zhang, J Zhang, and J Su. Exploring syntactic features for relation extraction using a convolution tree kernel. HLT-NAACL '06 Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, pages 288–295, 2006.

[27] M Zhang, G Zhou, and A Aw. Exploring syntactic structured features over parse trees for relation extraction using kernel methods. Information Processing & Management, pages 687–701, 2008.

[28] B Zhao and R Grishman. Extracting relations with integrated information using kernel-based methods. Proceedings of the Annual Meeting of the Association for Computational Linguistics (ACL'2005), pages 419–426, 2005.

[29] G Zhou, L Qian, and J Fan. Tree kernel-based semantic relation extraction with rich syntactic and semantic information. Information Sciences, pages 1313–1325, 2010.

[30] G Zhou, J Su, J Zhang, and M Zhang. Exploring various knowledge in relation extraction. ACL '05 Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics, pages 427–434, 2005.

[31] G Zhou, M Zhang, DH Ji, and QM Zhu. Tree kernel-based relation extraction with context-sensitive structured parse tree information. EMNLP-CoNLL'2007, pages 728–736, 2007.

Appendix A

Model Training

A.1 Parameter Optimization

In training our SVM models with uSVM we performed a search of the learning parameters to find those which maximized the MentRel F-Measure. In searching for these parameters we used uSVM's "-q" option to reduce the sample size to 100 and speed the search. According to Severyn and Moschitti's research on uSVM parameter optimization, parameters chosen with a lower sampling rate should perform well at a higher sample rate[21]. For the most part we found this to be true, though there were a few cases where optimized parameters did not perform much if any better at the higher sampling rate. When training our final classifiers we used a sampling size of 1000 which most likely reduces our overall performance by 1-2 points from an exact SVM model.

Although we were training multiple binary classifiers, in all cases we used the same learning parameters across all classes. This was done to reduce the complexity and number of options to consider, but likely resulted in slightly worse results, especially given the disparity in number of training examples for each class. Three particular learning parameters were optimized in this search:

- `-j`: This parameter controls the cost applied to incorrectly classified positive examples in the binary classifier. Effectively this controls the trade-off between precision and recall in the learning process. A high precision and low recall indicates that the cost value needs to

Corpus	Features	Classifier	Parameters		
			j	c	T
ACE	Baseline	c0	6.5	0.17	-
ACE	Baseline + Extended	c0	2.5	0.004	-
ACE	Tree	c0	6.5	0.16	-
ACE	Baseline + Tree	c0	6.5	0.44	0.45
ACE	Baseline + Extended + Tree	c0	6.5	0.079	0.43
JDPA	Baseline	c0	3.36	1.05	-
JDPA	Baseline + Extended	c0	5.5	0.005	-
JDPA	Tree	c0	8	0.7	-
JDPA	Baseline + Tree	c0	4	0.05	0.95
JDPA	Baseline + Extended + Tree	c0	8.3	0.051	0.56
JDPA	Baseline + Extended + Tree	c1	3	0.03	0.2
JDPA	Baseline + Extended + Tree	c2+	3	0.035	0.84

Table A.1: The final parameters used when training the SVM classifiers.

be higher, and similarly a high recall indicates the cost value is too high.

- `-c`: The margin parameter controls the SVM margin width of the classifier.
- `-T`: For composite tree and linear kernels, this parameter controls how much the tree and linear kernels respectively contribute to the SVM kernel function being optimized.

The following list provides command line option examples for the various types of classifiers we used:

- **Linear Kernel:** `-q 1000 -t 0 -c 0.02 -j 6.5`
- **Tree Kernel:** `-q 1000 -t 5 -c 0.02 -j 6.5`
- **Composite Linear/Tree Kernel:** `-q 1000 -t 5 -c 0.035 -j 5 -T 0.84 -S 0 -C +`

Appendix B

Extended Vector Features

A prominent feature among the extended features is the encoding of whether a word is co-referring to either M_1 or M_2. Hence for each word there are four possible states plus an additional NULL state for when the word does not exist:

- FF: The word is not in the same co-reference group of either M_1 or M_2.
- TF: The word is in the same co-reference group as M_1 but not M_2.
- FT: The word is in the same co-reference group as M_2 but not M_1.
- TT: The word is in the same co-reference group of both M_1 and M_2 (also indicating that M_1 and M_2 are in the same co-reference group).
- –: The word in question does not exist.

In the following description of the features we use CR to indicate this five-state variable for a particular word.

B.1 Co-Reference Location

- CR12: True if M_1 and M_2 are co-refs of each other
- CRFL+ET12: CR for the one and only word between the mentions combined with the entity types.

- CRF+ET12: CR for each of the two words following M_1 (25 possible states) combined with the entity types
- CRL+ET12: CR for each of the two words preceding M_2 (25 possible states) combined with the entity types
- CRO+#SB+ET12: CR for all other words between M_1 and M_2 ORed together (5 states) combined with the entity types and the number of sentences between the mentions (see below)
- CRB+ET12: CR for each of the three words preceding M_1 (125 possible states) combined with the entity types
- CRA+ET12: CR for each of the three words following M_2 (125 possible states) combined with the entity types
- AM2F+CRA2: Combination of word after M_2 and CR of second word after M_2
- WMF+CRF2: Combination of word after M_1 and CR of second word after M_1
- AM2F+CRA3: Combination of word after M_2 and CR of third word after M_2 (attempt to detect M_2 PRP DT NP)
- WMF+CRF3: Combination of word after M_1 and CR of third word after M_1 (attempt detect M_1 PRP DT NP)
- POSSM1+CRB2+ET12: Binary indication of whether word before M_1 is possessive and the CR of that word, combined with the entity types
- POSSM1+CRB2+ET12: Binary indication of whether word before M_2 is possessive and the CR of that word, combined with the entity types

B.2 Co-Reference Distance

- #WBCR1: Number of words between M_1 and closest co-ref of M_2 (-SELF-,-OVERLAP-, 0, 1, 2, 3, 4, 5, 6+)
- #WBCR2: Number of words between M_2 and closest co-ref of M_1 (-SELF-,-OVERLAP-, 0, 1, 2, 3, 4, 5, 6+)
- #MBCR1: Number of mentions between M_1 and closest co-ref of M_2 (-SELF-,-OVERLAP-, 0, 1, 2, 3, 4+)
- #MBCR2: Number of mentions between M_2 and closest co-ref of M_1 (-SELF-,-OVERLAP-, 0, 1, 2, 3, 4+)
- #SBCR1: Number of mentions between M_1 and closest co-ref of M_2 (-SELF-, 0, 1, 2, 3, 4, 5+)
- #SBCR2: Number of mentions between M_2 and closest co-ref of M_1 (-SELF-, 0, 1, 2, 3, 4, 5+)

B.3 Co-Reference Parse Tree

- PTPCR1: The parse tree path between M_1 and the closest co-ref of M_2
- PTPCR2: The parse tree path between M_2 and the closest co-ref of M_1

B.4 Co-Reference Phrase Chunking

- CPPCR1: Path of phrase labels connecting M_1 and closest co-ref of M_2 in the phrase chunking
- CPPCR2: Path of phrase labels connecting M_2 and closest co-ref of M_1 in the phrase chunking

- CPPHCR1: Path of phrase labels augmented with head words connecting M_1 and closest co-ref of M_2 in the phrase chunking
- CPPHCR2: Path of phrase labels augmented with head words connecting M_2 and closest co-ref of M_1 in the phrase chunking

B.5 Other Features

- BIWB+#WB: Bi-grams of words between mentions (if less than 6 words between) combined with the number of words between.
- #SB: Number of sentences between mentions (0, 1, 2, 3, 4, 5+)
- POSM1: Bag of Part-of-Speech terms in M_1
- POSM2: Bag of Part-of-Speech terms in M_2
- POSBFL+ET12: The only POS in between when only one word in between, combined with the entity types of the mentions.
- POSBF+ET12: First POS in between when at least two words in between, combined with the entity types of the mentions.
- POSBL+ET12: Last POS in between when at least two words in between, combined with the entity types of the mentions.
- POSBO+ET12: Other POS in between except first and last words when at least three words in between, combined with the entity types of the mentions.
- POSB1+ET12: POS of word before M_1, combined with the entity types of the mentions.
- POSA1+ET12: POS of word after M_2, combined with the entity types of the mentions.

B.6 System Development Testing

In the course of developing our system and the sets of features that we examined a number of experiments were run on our development sets both to verify that the system was working properly and to try and optimize the features that the system used. In this section we will attempt to outline some of the primary experiments that were run and their outcomes. Because these experiments were run only on the development set it is quite likely that some of them are not statistically significant. Still, it is hoped that this background can provide some information for future researchers. These results are presented in chronological order since one experiment tended to build upon the previous ones as the features changed over time.

- Early testing of Baseline+Extended Features on a sentence window from 1-1 up to 1-3 sentences (MentRel F1 = 44.34, 39.29, 37.91; EntRel F1 = 37.63, 39.18, 39.29). These results led to building a separate classifier depending on whether the mention pair was in the same sentence, consecutive sentences, or had a sentence in between. In retrospect it is not clear using separate classifiers was necessarily the correct decision.

- Attempted to discover combinations of features that improved results by looking at the mutual information for pairs of features. However, the identified combinations of features actually decreased performance. In retrospect, we believe the mutual information metric used was not adequate because it gave equal weight to both positive and negative indicators of relations and positive indicators of relations would probably be more important.

- Used the Baseline features to only detect whether a relation exists between two mention pairs (MentRel F1 = 49.56) and in a separate test to distinguish between the type of relation when it was already known that a relation exists (MentRel F1 = 76.26). From this result it is clear that the true difficulty in relation extraction comes largely from the actual detection of whether a relation exists between the mentions. Distinguishing between relation types (at least for this set of relations) is much easier. This is probably because

the entity types provide a lot of information as to what type of relation is involved.

- Modified the Baseline features to replace all numbers in the lexical features with the generic symbol "$< NUMBER >$". This increased results from 39.99 to 40.43 so it was retained.

- Added a binary feature to indicate if there was a determiner (and the type of the determiner) before each of the mentions. This feature had no effect on the results, and the feature was removed.

- Tested adding only the bi-grams from the extended features to the baseline features. No impact on results, but a more limited version of this features was later added back in.

- Tested adding only the co-reference features from the extended features to the baseline features. Improved MentRel F-1 by 2.5

- Tested adding only the POS features from the extended features to the baseline features. Improved MentRel F-1 by 0.25

- At this point we abandoned further feature vector engineering efforts in order to focus on trying to develop tree kernel methods to apply to relation extraction. Initially our results with tree kernels were quite poor until we recognized the increased importance that the training parameters played in helping the SVM tree kernel to work.

- Due to observed errors in the sentence splitting and tokenization as well as a desire to use a publicly available library we converted the system to use the OpenNLP sentence splitter (F1 = 36.84) and tokenizer rather than the proprietary JDPA tokenizer (F1 = 35.94). This change helped a bit due to better sentence splitting.

- Added co-reference information to parse tree to indicate co-referring mentions. Improved MentRel F-1 by 0.74

- In Zhou et al's 2010 work they reduced the complexity of some trees by compressing single-in-single-out nodes to remove the node. However, in our system, simplifying the tree

resulted in a reduction of MentRel F1 by 0.4.

- We also tried to expand the partial tree to include any other mentions within the sentence which were co-referring to one of the mentions in the mention pair. However, this expanded context reduced performance of MentRel F1 by 2.4.

- In developing our inter-tree for summarizing sentences between the mentions we tested our c2 classifier both with and without the inter-tree. With the inter-tree our MentRel F1 for this classifier improved by 3.5.

- As an alternative to our simple voting scheme we tried allowing classifications of mention pairs with NoRelation to count as a vote towards there being no relation between the entities. This reduced EntRel by 3.5.

- As an alternative to our simple voting scheme we tried summing the margin output value for each mention pair prediction. The hope was that strong indications of relations would overcome the weak indications that there was no relation. However, this method still reduced EntRel by 3.0 from our simple voting scheme.