

Transfer Learning for Adaptive Relation Extraction

Principal Investigators:

Hai Leong Chieu; DSO National Laboratories
Wee Sun Lee; National University of Singapore
Jing Jiang; Singapore Management University

September 13, 2011

Abstract

This project addresses the relation extraction problem in resource-poor domains. The relation extraction problem is the problem of extracting information regarding relationships between entities (e.g. individuals, organizations). This involves extracting the text spans specifying the entities involved and classifying the relationship between these entities into one of several pre-defined classes (e.g. person-organization affiliation). The primary technique for solving such problems is machine learning: given a set of examples of text and the extracted information, a machine-learning algorithm learns a statistical model that can be used to extract information from new text. An obstacle to the widespread application of machine-learning methods is the necessity for relatively large amounts of annotated training data: while free text are abundantly available, it is costly to employ humans to annotate the free text with information that should be extracted from it. The focus of this project is to research into technologies that could enable relation extraction systems to be quickly adapted to resource-poor domains. We apply (1) transfer learning approaches for transferring the models learned in one domain to new domains, and (2) structured learning approaches that could remove the need for powerful pre-processing modules (e.g. parsers, named entity recognizers) that may not be available in resource-poor domains.

Report Documentation Page

Form Approved
OMB No. 0704-0188

Public reporting burden for the collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington VA 22202-4302. Respondents should be aware that notwithstanding any other provision of law, no person shall be subject to a penalty for failing to comply with a collection of information if it does not display a currently valid OMB control number.

1. REPORT DATE 23 SEP 2011	2. REPORT TYPE Final	3. DATES COVERED 14-07-2009 to 14-07-2011			
4. TITLE AND SUBTITLE Transfer Learning for Adaptive Relation Extraction		5a. CONTRACT NUMBER FA23860914123			
		5b. GRANT NUMBER			
		5c. PROGRAM ELEMENT NUMBER			
6. AUTHOR(S) Hai Leong Chieu; Wee Sun Lee; Jing Jiang		5d. PROJECT NUMBER			
		5e. TASK NUMBER			
		5f. WORK UNIT NUMBER			
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) DSO National Laboratories,118230,20 Science Park Drive,Singapore,NA,NA		8. PERFORMING ORGANIZATION REPORT NUMBER N/A			
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) AOARD, UNIT 45002, APO, AP, 96338-5002		10. SPONSOR/MONITOR'S ACRONYM(S) AOARD			
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) AOARD-094123			
12. DISTRIBUTION/AVAILABILITY STATEMENT Approved for public release; distribution unlimited					
13. SUPPLEMENTARY NOTES					
14. ABSTRACT This project addressed the relation extraction problem in resource-poor domains. This problem is one of extracting information regarding relationships between entities, which involves extracting text spans specifying the entities involved and classifying the relationship between these entities into one of several pre-defined classes. The primary technique for solving such problems is machine learning. An obstacle to the widespread application of machine-learning methods is the necessity for relatively large amounts of annotated training data: while free text is abundantly available, it is costly to employ humans to annotate the free text with information that should be extracted from it. The focus of this project was research into technologies that could enable relation extraction systems to be quickly adapted to resource-poor domains. We applied (1) transfer learning approaches for transferring the models learned in one domain to new domains, and (2) structured learning approaches that could remove the need for powerful pre-processing modules (e.g. parsers, named entity recognizers) that may not be available in resource-poor domains.					
15. SUBJECT TERMS Transfer learning					
16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT Same as Report (SAR)	18. NUMBER OF PAGES 56	19a. NAME OF RESPONSIBLE PERSON
a. REPORT unclassified	b. ABSTRACT unclassified	c. THIS PAGE unclassified			

Contents

1	Executive Summary	4
1.1	Introduction	4
1.2	Cross-domain transfer learning	4
1.3	Cross-type transfer learning	4
1.4	Extraction of relation descriptors	5
1.5	Relation argument detection in resource-poor domains	5
1.6	Publications	5
2	SMU Technical Report: Transfer Learning for Adaptive Relation Extraction	6
2.1	Research Team	6
2.2	Introduction	6
2.3	Cross-Domain Relation Extraction	7
2.3.1	Problem definition	7
2.3.2	Domain Similarity Measures	8
2.3.3	Evaluation	9
2.3.4	Multitask Relation Extraction with Domain Similarity Measures	10
2.4	Cross-Type Relation Extraction	12
2.4.1	Problem definition	12
2.4.2	A General MTL Framework	12
2.4.3	Experiment	17
2.5	Extraction of Relation Descriptors	21
2.5.1	Problem definition	21
2.5.2	Related Work	22
2.5.3	Method	23
2.5.4	Features	25
2.5.5	Experiments	26
2.6	Conclusions	30
3	NUS Technical Report: Relation extraction in resource-poor domains	32
3.1	Research Team	32
3.2	Introduction	32
3.3	A Simplified Case	33
3.3.1	Relation Extraction as Sequence Labeling	33
3.3.2	Conditional Random Fields	34
3.3.3	Models	34
3.3.4	Datasets	36
3.3.5	Results and Discussions	37
3.4	Exploiting Tree Structures	39
3.4.1	Motivation	39
3.4.2	Model	39
3.4.3	Variants	40

3.5	Exploiting Long-range Dependencies	41
3.5.1	Motivation	41
3.5.2	Model	41
3.6	Experiments	42
3.6.1	Tree Structures	42
3.6.2	High-order Semi-Markov Features	43
3.6.3	A Combined Model	44
3.7	Conclusion	45
Appendices		47
A Algorithms for the Tree Model		47
A.1	Training	47
A.1.1	Matrix Tree Theorem and Consequences	47
A.1.2	Computing Regularized Log-likelihood	50
A.1.3	Computing Gradient	50
A.2	Decoding	51
B Algorithms for High-order Semi-Markov CRF		51
B.1	Notations	51
B.2	Training	51
B.2.1	Partition Function	51
B.2.2	Expected Feature Sum	52
B.3	Inference	52

1 Executive Summary

1.1 Introduction

DARPA funded the DSO National Laboratories (DSO) over 2 years for the project “Transfer Learning for Adaptive Relation Extraction”. DSO awarded subcontracts to the National University of Singapore (NUS) and the Singapore Management University (SMU) to support the project. DSO also engaged Prof. Leslie Pack Kaelbling from the Massachusetts Institute of Technology (MIT) as a consultant. The objective of this project is to extract relationships between entities, focusing on the research of technologies that would enable the rapid development of adaptive relation extraction systems in resource-poor domains. In Chapter 2 and 3, we provide the technical reports of the work done in collaboration with SMU and with NUS respectively. In the rest of this chapter, we summarize the main achievements.

1.2 Cross-domain transfer learning

The performance of cross-domain transfer learning is deeply related to the similarity between the source and the target domains. We investigated three different measures of domain similarity based on word distribution, language models, and kernels defined on lexicalized parse trees. In Section 2.3, we study how domain similarity between source and target domains is related to the degradation in performance of a system trained on the source domain and tested on a target domain. Using the six genres provided in the ACE 2005 data set as domains, we showed that the ranking of the source domains by cross-domain performance is positively correlated with the ranking of the source domains by their similarity with the target domain. We found that a lexicalized tree kernel-based measure was the most effective in finding good source domains. We attempted to use the matrix of similarity values between domains in a multi-task regularization framework, but experimental results show that while this could improve performance for some domains when training data is small, it often fails to beat the baseline of simply pooling the training data from all domains.

1.3 Cross-type transfer learning

We studied the application of multi-task learning approaches to the problem of cross-type relation extraction. Cross-type relation extraction is the problem where a specific target relation type (e.g. person-organization affiliation relations) is defined but we do not have enough training data for this target type. On the other hand we do have labeled instances for one or a number of other source relation types. The goal is to leverage the training data for the source types for extraction of relation instances of the target type.

In our multi-task learning framework, we assume that there is a low-dimensional feature space which is common across tasks. We build on our previous work of applying a feature selection approach as part of a multi-task learning objective function [16]. In Section 2.4, we generalized this approach to search for a low dimensional feature projection, and the experiments on ACE04

data showed that multi-task models improve the performance over the baselines of learning each task separately, or simply learning all tasks together.

1.4 Extraction of relation descriptors

Relation extraction tasks (such as the tasks defined in ACE) are often defined as extracting the arguments involved in a relation (e.g. person and organization in an affiliation relation). In some cases, users may desire a more specific relation description in the sentence stating the relationship (e.g. the post of the person in the organization). We investigated this problem under a supervised sentence segmentation framework, and we found that standard linear-chain conditional random fields (CRF) model have some potential limitations for this task. We proposed a modified CRF structure that reduces the space of possible label sequences and introduces long-range features. We annotated two data sets to evaluate our methods. In [20] we show that the modified CRF model can perform significantly better than standard linear-chain CRF on two data sets that we collected.

1.5 Relation argument detection in resource-poor domains

Most relation extraction work relies on pre-processing modules such as dependency parsers and named entity recognizers. We address the relation extraction problem without assuming any pre-processing tool. We studied the use of long range dependencies within sentences via high-order semi-Markov conditional random fields [24] and latent tree structures (see Section 3.4). Traditional relation extraction approaches gathers argument pairs using mention detection modules, and classify such pairs to belong to a relation type or not. We show in Section 3.6 that combining the argument-pairs detected with CRF, semi-Markov CRF and a latent tree approach outperforms simply using a mention detection module trained on the ACE 2005 data set.

1.6 Publications

Under this project, we published [20] and [24].

2 SMU Technical Report: Transfer Learning for Adaptive Relation Extraction

2.1 Research Team

The research work in this chapter is performed mainly in SMU. The SMU research team consists of Asst. Prof. Jing Jiang (PI) and Yaliang Li (research engineer). Part of the work reported here was done previously by Dr. Zhisheng Li (postdoctoral research fellow) and Xin Zhao (visiting PhD student). The DSO team consists of Dr. Hai Leong Chieu and Dr. Kian Ming A. Chai.

2.2 Introduction

Information Extraction (IE) is a task in natural language processing whose goal is to automatically extract structured information such as entities and their relations from unstructured text documents. One of the most important subtasks of IE is relation extraction, which identifies semantic relations between entities. For example, the sentence “Larry Page is the CEO of Google” contains the *employment* relation between the *person* entity “Larry Page” and the *organization* entity “Google”. While the entity recognition task helps us find entities from documents (answer the “WHO” question), the relation extraction task detects semantic relationships between these entities (answer the “WHAT” question).

State-of-the-art solutions to relation extraction mostly rely on *supervised learning* [36, 39, 38, 5, 17, 26]. Typically a set of relation types are defined and all instances of these pre-defined relation types are identified and labeled within a corpus. This annotated corpus can then be used for training relation extraction models. Like many other NLP tasks, however, supervised learning approach fails when there is not a sufficient amount of labeled data for training, which is often the case when we need to extract relations from new domains or of new types.

The goal of the T-REX (Transfer learning for Relation EXtraction) project is therefore to develop relation extraction methods that can easily and quickly adapt to new domains and/or new relation types by leveraging existing training data from some old domain(s) and/or relation type(s). From a learning point of view, this adaptation problem is closely related to transfer learning, which has recently attracted much attention in the machine learning community [25]. In this project, we aim to apply and extend existing transfer learning techniques for relation extraction.

In this final report of the T-REX project, we present our exploration in the following three directions:

- **Cross-domain Relation Extraction:** We first present our study on cross-domain relation extraction in Section 2.3. In order to perform cross-domain relation extraction, we study a number of domain similarity measures designed for relation extraction. We check the correlations between these domain similarity measures and cross-domain relation extraction performance in order to evaluate the effectiveness of these domain similarity measures. We then use them to regularize the model parameters in a multi-task learning framework.

Domain	#Files	#Words	#Positive Instances	#Negative Instances	#Sum
NW (newswire)	81	33,459	1,551	41309	42860
BN (broadcast news)	217	52,444	1,734	41422	43156
BC (broadcast conversation)	52	33,874	1,175	28749	29924
WL (weblog)	114	35,529	650	26336	26986
UN (usenet newsgroups)	37	26,371	544	18042	18586
CTS (telephone speech)	34	34,868	503	28185	28688

Table 1: Some statistics of the 6 domains in the ACE 2005 dataset.

- **Cross-type Relation Extraction:** Next, we study cross-type relation extraction in Section 2.4. We observe that different relation types often share some similar syntactic patterns, which leads to the idea of apply multi-task learning for cross-type relation extraction where extraction of each relation type is considered a separate task and extraction of different relation types are considered related tasks. We compare two multi-task learning methods applied to cross-type relation extraction.
- **Extraction of Relation Descriptors:** Finally we study a new relation extraction problem which extracts a specific relation descriptor for a general relation type from a detected relation instance. This study is presented in Section 2.5.

2.3 Cross-Domain Relation Extraction

2.3.1 Problem definition

While an increasing amount of work focuses on designing domain adaptation methods, an important subproblem is how to measure domain similarities. Such similarity measures can help (1) choose a suitable source domain among many for transfer to a target domain, (2) provide a prior expectation on whether transfer will be effective, and (3) set certain parameters in some domain adaptation, transfer learning or multitask learning methods.

Supervised relation extraction has been extensively studied, especially on the ACE benchmark data sets [12, 39, 5, 37, 23]. However, not much attention is given to the setting when training and test domains differ. We use the ACE 2005 dataset [34] to illustrate the problem. The ACE 2005 dataset contains 6 different domains, including newswire and broadcast news transcripts. Some statistics of the data are shown in Table 1. We use a feature-based supervised approach, where the feature set includes standard features such as words, POS tags, entity types, dependency relations and conjunctions of them. To make fair comparison, we use the same number (9,000) of relation instances from each domain for training. Table 2 highlights that the matching source (training) and target (test) domains give the best performance while non-matching source and target domains lead to substantial decrease in performance.

We see on close examination of Table 2 that some source domains give better performance than others given a target domain. For example, BN is the best source domain for the target domain

Source	Target					
	NW	BN	BC	WL	UN	CTS
NW	0.516	0.469	0.406	0.404	0.406	0.292
BN	0.462	0.529	0.453	0.421	0.442	0.412
BC	0.382	0.486	0.525	0.383	0.430	0.458
WL	0.386	0.462	0.402	0.445	0.417	0.336
UN	0.413	0.458	0.442	0.388	0.534	0.419
CTS	0.268	0.272	0.290	0.281	0.327	0.555

Table 2: Average F1 over different relation types in the same-domain and cross-domain settings.

NW. This phenomenon begs the question: can we predict the source domain that can give the best relation extraction performance for a given target domain? Moreover, can the prediction be made without any labeled data from the target domain?

2.3.2 Domain Similarity Measures

We believe that effective domain similarity measures are task-specific. For relation extraction, we observe that there are several factors that may cause performance drop across domains, including the usage of words, entities and sentence patterns.

Word Distribution The major language difference between two domains is arguably the difference between word distributions. We can represent a domain by a unigram language model estimated from a representative corpus from that domain and measure domain similarity through some usual metric such as cosine similarity or KL-divergence. Indeed, this is one of the features considered in [22] for domain adaptation for parsing. Following their work, we define our first measure, *WD*, to be the cosine similarity between the frequency-weighted vectors containing the most frequent K words in each domain. We also consider a variation of *WD* called *NS-WD* where we include only non-stopwords because we suspect the usage of stopwords is not very relevant to relation extraction.

Our experience with relation extraction on the ACE 2005 data shows that the argument entities often provide important clues for classifying their relations. For example, in “*Vatican spokesman*,” the argument word “*spokesman*” is indicative of the employment relation. We suspect that if two domains have more overlapping entity words then it is easier to perform cross-domain relation extraction. So the third measure we consider is the cosine similarity between the vectors containing only the top- K entity words in each domain, which we call *ENT*.

Language Models Intuitively relation extraction relies largely on sentence patterns. In order to capture the difference in sentence patterns across domains, a relatively simple way is to consider larger language units such as bigrams and trigrams rather than single words. We can train a language model on the source domain and test its prediction power on the target domain using

Source	Target					
	NW	BN	BC	WL	UN	CTS
NW	0.1183	0.1089	0.1072	0.1121	0.1073	0.0864
BN	0.1089	0.1219	0.1153	0.1133	0.1131	0.0975
BC	0.1072	0.1153	0.1190	0.1129	0.1143	0.1078
WL	0.1121	0.1132	0.1129	0.1161	0.1132	0.0983
UN	0.1073	0.1131	0.1143	0.1132	0.1158	0.1067
CTS	0.0864	0.0975	0.1078	0.0983	0.1067	0.1361

Table 3: *L-TK* similarity for each pair of domains.

perplexity. Presumably the larger the perplexity the more different the two domains are. Using the SRILM toolkit¹, we experimented with bigram and trigram language models with different smoothing techniques and settled with tigram language models and modified Kneser-Ney discounting method [8]. We call this measure *LM*.

Tree Kernels Sentence patterns can be better captured by parse trees. Recent studies have shown that convolution tree kernels [10] can be effectively used in supervised relation extraction to achieve the state-of-the-art performance [37, 23]. Given the parse trees of two sentences, T_1 and T_2 , The tree kernel $K(T_1, T_2)$ roughly represents the number of common subtrees shared by T_1 and T_2 . We define our tree kernel-based domain similarity measure to be the average tree kernel value over all pairs of sentences in two domains. We consider two variations: *L-TK* uses lexicalized parse trees while *U-TK* uses unlexicalized parse trees. We use the SVM-LIGHT-TK toolkit to compute tree kernels². We show the similarity of *L-TK* in Table 3.

2.3.3 Evaluation

There has not been much work on how to compare domain similarity measures. [22] proposed some domain similarity measures for parsing but did not compare them. Rather, they used them as features in a regression task to predict the cross-domain accuracy. [33] used Pearson product-moment correlation between the similarity metric and the cross-domain accuracy to compare different metrics. We choose a slightly different approach than [33] because we think the cross-domain performance measures may not be *linearly* correlated with a good domain similarity measure. We believe that as long as a domain similarity measure can correctly rank the potential source domains based on how well they transfer to the target domain, it is a good similarity measure. We therefore use the following approach: Given a target domain, we rank the source domains by the cross-domain performance and the domain similarity measure respectively to obtain two ranked lists. We then compute the Spearman’s rank correlation between the two ranked lists. We show the correlation coefficients in Table 4. For *WD*, *NS-WD* and *ENT*, we experimented with different values of K and used the best value of 50 in the end. We can see that both *WD* and *L-TK* are

¹<http://www.speech.sri.com/projects/srilm/>

²<http://disi.unitn.it/moschitti/TK1.0-software/Tree-Kernel.htm>

Target Domain	Correlation Coefficients					
	WD	NS-WD	ENT	LM	L-TK	U-TK
NW	0.7	0.1	0.6	0.6	0.7	0.1
BN	0.7	0.6	0.3	0.7	0.7	0.3
BC	0.7	0.7	0.7	0.7	0.7	0.1
WL	0.5	0.2	-0.3	0.5	0.7	0.5
UN	0.7	0.6	-0.1	0.7	0.7	0.3
CTS	0.9	0.1	0.3	0.7	0.9	0.2
Avg.	0.7	0.38	0.25	0.65	0.73	0.25

Table 4: Spearman’s rank correlation coefficients for different domain similarity measures.

effective, with *L-TK* slightly better. *LM* is also a reasonable measure but not as good. It suggests that lexical usage alone provides a good measure of domain similarity for relation extraction, but sentence patterns as captured in parse trees can further improve the similarity measure. Contrary to our hypotheses, excluding stopwords (*NW-WD*) or including only entity words (*ENT*) does not help.

2.3.4 Multitask Relation Extraction with Domain Similarity Measures

A straightforward usage of domain similarity measures is to predict which source domain is the best given a target domain. Here we consider another usage, which is to incorporate such measures into a multitask learning framework.

The framework we consider is based on logistic regression. Supervised relation extraction is often treated as a multiclass classification problem where the class labels are the different relation types plus a special label for no relations. Let \mathbf{x} be a feature vector representing a relation instance. Let $y \in \mathcal{Y}$ be a class label. We model $p(y|\mathbf{x})$ with $|\mathcal{Y}|$ weight vectors \mathbf{w}_y in the following way:

$$p(y|\mathbf{x}; \mathbf{w}) = \frac{\exp(\mathbf{w}_y^T \mathbf{x})}{\sum_{y' \in \mathcal{Y}} \exp(\mathbf{w}_{y'}^T \mathbf{x})}.$$

Here \mathbf{w}_y^T is the transpose of \mathbf{w}_y and $\mathbf{w}_y^T \mathbf{x}$ is the dot product of \mathbf{w}_y and \mathbf{x} .

In standard logistic regression, given labeled relation instances $\{(\mathbf{x}_n, y_n)\}_{n=1}^N$, we look for the model parameters that maximize the probability of the data. We often use a Gaussian prior over the parameters as follows:

$$\hat{\mathbf{w}} = \arg \min_{\mathbf{w}} \left[\left(- \sum_{n=1}^N \log p(y_n | \mathbf{x}_n; \mathbf{w}) \right) + \mu \sum_y \frac{\mathbf{w}_y^T \Sigma^{-1} \mathbf{w}_y}{2} \right],$$

where μ is a fixed parameter to control the influence of the prior, and Σ is usually the identity matrix.

x	<i>BL-1</i>	<i>BL-2</i>	<i>WD</i>	<i>L-TK</i>
1	0.3308	0.4626	0.4458	0.4363
2	0.4146	0.5206	0.4923	0.4770
3	0.4330	0.5623	0.5117	0.5074
4	0.4847	0.5757	0.5353	0.5314

Table 5: Average F1 over all domains.

When we have labeled relation instances from different domains, usually we either learn a classification model for each domain independently or we learn a single common classification model. But given the domain differences, a multitask learning approach may be preferred. We can treat the K domains as different tasks and assume a different model \mathbf{w}^k for each task. Given labeled relation instances $\{(\mathbf{x}_n^k, y_n^k)\}$, we again look for the model parameters that maximize the overall probability of the data. However, because these different tasks are related, we can assume that the weight vectors of different domains are correlated. One way is through the following objective function:

$$\{\hat{\mathbf{w}}^k\} = \arg \min_{\{\mathbf{w}^k\}} \left[\left(- \sum_{k=1}^K \sum_{n=1}^{N^k} \log p(y_n^k | \mathbf{x}_n^k, \mathbf{w}^k) \right) + \mu \sum_y \sum_{f \in \mathcal{F}} \frac{\mathbf{v}_{y,f}^T \Sigma^{-1} \mathbf{v}_{y,f}}{2} \right].$$

Here \mathcal{F} is the set of features. $\mathbf{v}_{y,f}$ is defined as $(w_{y,f}^1, w_{y,f}^2, \dots, w_{y,f}^K)^T$, i.e. a vector that consists of the weights for class y and feature f of each domain. Σ is no longer the identity matrix but a positive semi-definite matrix that encodes the similarities between each pair of domains. The idea is that if two domains are similar, their weights for the same feature should also be close to each other.

Given a symmetric domain similarity measure S , we can set

$$\Sigma_{k_1, k_2} = \Sigma_{k_2, k_1} = \frac{S(k_1, k_2)}{\sqrt{S(k_1, k_1) \cdot S(k_2, k_2)}}.$$

If the matrix Σ defined this way is positive semi-definite, we can use it in our learning framework.

We still use ACE 2005 dataset for our experiments. We use the best two similarity measures, *L-TK* and *WD*, to set Σ . After preliminary experiments we set $\mu = 0.25$. Multitask learning is usually more useful when the amount of training data for each task is small. We therefore first randomly divide the data into 5 subsets and then take $x/5$ ($x = 1, 2, 3, 4$) of the data for training and the last $1/5$ for testing. For comparison, we consider two baselines. *BL-1*: The models for each domain are trained independently. *BL-2*: A single model is trained using data across domains.

The results are shown in Table 5. We can see that *BL-1* is always the worst while *BL-2* performs the best in all settings, showing that pooling the training data from different domains together to

Domain	<i>BL-1</i>	<i>BL-2</i>	<i>WD</i>	<i>L-TK</i>
NW	0.2590	0.4717	0.4018	0.3863
BN	0.3767	0.5466	0.5280	0.5563
BC	0.3995	0.5163	0.4632	0.4670
WL	0.2002	0.3649	0.3823	0.3704
UN	0.3627	0.4940	0.5001	0.4722
CTS	0.3865	0.3821	0.3994	0.3661

Table 6: F1 for each domain when $x = 1$.

train a single model is still the best strategy overall. But when the amount of training data is small ($x = 1$), we see that the gap between *BL-2* and multitask learning (*WD* and *L-TK*) is also smaller. We then zoom into the setting with $x = 1$ and examine the performance for each domain. The results are shown in Table 6. We now see that for 4 out of 6 domains (i.e. BN, WL, UN, CTS), at least one of the multitask learning methods outperforms the baselines. This suggests that for some domains multitask learning with a good domain similarity measure can be a preferred choice. However, further studies are needed to find out when a multitask learning approach is preferred.

2.4 Cross-Type Relation Extraction

In this section, we present two general multi-task learning frameworks and how we apply them to the problem of cross-type relation extraction. We empirically compare these methods on the ACE04 data set.

2.4.1 Problem definition

Cross-type relation extraction is the problem where a specific target relation type is defined but we do not have enough training data for this target type. On the other hand we do have labeled instances for one or a number of other source relation types. The goal is to leverage the training data for the source types for extraction of relation instances of the target type.

We treat this problem as a multi-task learning problem. The reason we believe that extraction of these different relation types is related is that different relation types often share some common syntactic structures. Table 7 shows some examples.

In the following subsections we first introduce two general multi-task learning methods and then apply them to cross-type relation extraction.

2.4.2 A General MTL Framework

We assume that there are K related classification tasks which share the same set of class labels and the same feature space. In general for multi-task learning different tasks can have different label sets and/or different feature spaces. For our relation extraction problem, however, because we treat the recognition of instances of each relation type as a binary classification problem, we only have two class labels, namely, the positive class and the negative class. As for features, we can always

Syntactic Pattern	Relation Instance	Relation Type (Subtype)
<i>arg-2</i> <i>arg-1</i>	<i>Arab</i> <i>leaders</i>	OTHER-AFF (Ethnic)
	<i>his</i> <i>father</i>	PER-SOC (Family)
	South Jakarta <i>Prosecution Office</i>	GPE-AFF (Based-In)
<i>arg-1</i> of arg-2	<i>leader</i> of a minority government	EMP-ORG (Employ-Executive)
	the youngest <i>son</i> of ex-director Suharto	PER-SOC (Family)
	the <i>Socialist People's Party</i> of Montenegro	GPE-AFF (Based-In)
<i>arg-1</i> [verb] arg-2	<i>Yemen</i> [sent] planes to Baghdad	ART (User-or-Owner)
	<i>his wife</i> [had] three young children	PER-SOC (Family)
	<i>Jody Scheckter</i> [paced] Ferrari to both victories	EMP-ORG (Employ-Staff)

Table 7: Examples of similar syntactic structures across different relation types. The head words of the first and the second arguments are shown in italic and bold, respectively.

take the union of the feature sets observed in each task as the overall feature set. We therefore assume a single class label set and a single feature set. Let $\mathcal{Y} = \{+, -\}$ denote the class label set. Let each instance be represented by a feature vector from \mathbb{R}^F .

We assume that for each task we have a set of labeled instances for training. We denote the training data for task k as $\mathcal{D}^k = \{\mathbf{x}_i^k, y_i^k\}_{i=1}^{N_k}$ ($1 \leq k \leq K$), where \mathbf{x}_i^k is the feature vector representing the instance, y_i^k is its correct label and N_k is the number of training instances for task k .

Given a data set \mathcal{D}^k , a linear model (predictor) \mathbf{w}^k can be learnt by:

$$\begin{aligned} \hat{\mathbf{w}}^k &= \arg \min_{\mathbf{w}^k} \sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) \\ &= \arg \min_{\mathbf{w}^k} \sum_{i=1}^{N_k} -\log p(y_i^k | \mathbf{x}_i^k, \mathbf{w}^k) \end{aligned}$$

For the scenario we consider here, we assume the model \mathbf{w}^k contains two parts: a high dimensional vector $\mathbf{v}^k \in \mathbb{R}^F$, and a low dimensional vector $\mathbf{u}^k \in \mathbb{R}^H$ ($H \ll F$):

$$\mathbf{w}^k = \mathbf{v}^k + \mathbf{A}^k \mathbf{u}^k$$

where \mathbf{A}^k is an $F \times H$ dimensional matrix which can map the low dimensional vector \mathbf{u}^k into the high dimensional space \mathbb{R}^F .

Thus we get the objective function:

$$\left(\{\hat{\mathbf{v}}^k, \hat{\mathbf{u}}^k, \hat{\mathbf{A}}^k\}_{k=1}^K \right) = \arg \min_{\{\mathbf{v}^k, \mathbf{u}^k, \mathbf{A}^k\}} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, (\mathbf{v}^k + \mathbf{A}^k \mathbf{u}^k)) + r(\mathbf{v}^k, \mathbf{u}^k, \mathbf{A}^k) \right),$$

where $r(\mathbf{v}^k, \mathbf{u}^k, \mathbf{A}^k)$ is a regularization term. Based on different assumptions, we will investigate two different cases:

$$\mathbf{w}^k = \mathbf{v}^k + \mathbf{A} \mathbf{u}^k \tag{1}$$

$$\mathbf{w}^k = \mathbf{v}^k + \mathbf{A} \mathbf{u} \tag{2}$$

Method 1 In this scenario, inspired by [2], we assume that the low dimensional space is shared among different tasks, so we have:

$$\mathbf{w}^k = \mathbf{v}^k + \mathbf{A}\mathbf{u}^k$$

The objective function will be:

$$\left(\{\hat{\mathbf{v}}^k, \hat{\mathbf{u}}^k\}_{k=1}^K, \hat{\mathbf{A}} \right) = \arg \min_{\{\mathbf{v}^k, \mathbf{u}^k\}, \mathbf{A}} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, (\mathbf{v}^k + \mathbf{A}\mathbf{u}^k)) + r(\mathbf{v}^k, \mathbf{u}^k) \right) + r(\mathbf{A})$$

In the following, we consider a special set of regularizers, namely, regularizers based on L_2 norms. As $\mathbf{v}^k = \mathbf{w}^k - \mathbf{A}\mathbf{u}^k$, we have:

$$\left(\{\hat{\mathbf{w}}^k, \hat{\mathbf{u}}^k\}_{k=1}^K, \hat{\mathbf{A}} \right) = \arg \min_{\{\mathbf{w}^k, \mathbf{u}^k\}, \mathbf{A}} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) + \lambda_k \|\mathbf{w}^k - \mathbf{A}\mathbf{u}^k\|^2 \right), \quad (3)$$

$$\text{s.t. } \mathbf{A}^T \mathbf{A} = \mathbf{I}_{H \times H}.$$

Note that the last constraint can be seen as the regularizer for \mathbf{A} .

In order to solve (3), we use the following alternating optimization procedure:

1. Fix $\{\{\mathbf{u}^k\}, \mathbf{A}\}$, and optimize (3) with respect to $\{\mathbf{w}^k\}$.
2. Fix $\{\mathbf{w}^k\}$, and optimize (3) respect to $\{\{\mathbf{u}^k\}, \mathbf{A}\}$.
3. Iterate until the stopping criterion is met.

In the first step, when $\{\{\mathbf{u}^k\}, \mathbf{A}\}$ are fixed, objective function (3) will be:

$$\left(\{\hat{\mathbf{w}}^k\}_{k=1}^K \right) = \arg \min_{\{\mathbf{w}^k\}} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) + \lambda_k \|\mathbf{w}^k - \mathbf{A}\mathbf{u}^k\|^2 \right) \quad (4)$$

which can be solved by some well-established methods. Here, we will use L-BFGS to solve (4).

In the second step, $\{\mathbf{w}^k\}$ are fixed, then the objective function (3) will be:

$$\left(\{\hat{\mathbf{u}}^k\}_{k=1}^K, \hat{\mathbf{A}} \right) = \arg \min_{\{\mathbf{u}^k\}, \mathbf{A}} \sum_{k=1}^K \lambda_k \|\mathbf{w}^k - \mathbf{A}\mathbf{u}^k\|^2 \quad (5)$$

$$\text{s.t. } \mathbf{A}^T \mathbf{A} = \mathbf{I}_{H \times H}$$

With fixed \mathbf{A} , the optimal value for (5) will be achieved at $\hat{\mathbf{u}}^k = \mathbf{A}^T \hat{\mathbf{w}}^k$. By eliminating \mathbf{u}^k , we can rewrite (5) as

$$\hat{\mathbf{A}} = \arg \max_{\mathbf{A}} \sum_{k=1}^K \lambda_k \|\mathbf{A}^T \hat{\mathbf{w}}^k\|^2, \quad \text{s.t. } \mathbf{A}^T \mathbf{A} = \mathbf{I}_{H \times H}$$

Let $\mathbf{W} = [\sqrt{\lambda_1}\hat{\mathbf{w}}^1, \dots, \sqrt{\lambda_K}\hat{\mathbf{w}}^K]$ be an $F \times K$ matrix, we have

$$\hat{\mathbf{A}} = \arg \max_{\mathbf{A}} \text{tr}(\mathbf{A}^T \mathbf{W} \mathbf{W}^T \mathbf{A}), \quad \text{s.t.} \quad \mathbf{A}^T \mathbf{A} = \mathbf{I}_{H \times H} \quad (6)$$

where $\text{tr}()$ is the trace of a particular matrix. The solution of problem (6) can be given by the Singular Value Decomposition (SVD) of \mathbf{W} .

More details about the implementation of this model are described in Algorithm 1.

Algorithm 1: The Pseudocode of Solution for Method 1

Input: $\mathcal{D}_k = \{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k}, H, \lambda_k$
Output: $\{\hat{\mathbf{w}}^k, \hat{\mathbf{u}}^k\}_{k=1}^K, \hat{\mathbf{A}}$

- 1 Initialization: $\{\mathbf{u}'^k = \mathbf{0}_F\}$
- 2 **while** the stop criterion is not met **do**
- 3 /*step 1*/
- 4 **for** $i \leftarrow 1$ **to** K **do**
- 5 solve: $\hat{\mathbf{w}}^k = \arg \min_{\mathbf{w}^k} \sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) + \lambda_k \|\mathbf{w}^k - \mathbf{u}'^k\|^2$
- 6 **end**
- 7 /*step 2*/
- 8 $\mathbf{W} = [\sqrt{\lambda_1}\hat{\mathbf{w}}^1, \dots, \sqrt{\lambda_K}\hat{\mathbf{w}}^K]$
- 9 compute SVD of \mathbf{W} : $\mathbf{W} = \mathbf{V}_1 \mathbf{D} \mathbf{V}_2$
- 10 let the columns of \mathbf{A} be the first H columns of \mathbf{V}_1
- 11 **for** $i \leftarrow 1$ **to** K **do**
- 12 $\mathbf{u}^k = \mathbf{A}^T \mathbf{w}^k$
- 13 $\mathbf{u}'^k = \mathbf{A} \mathbf{u}^k$
- 14 **end**
- 15 **end**

Method 2 In this scenario, both the low dimensional space and the low dimensional vector are shared among different tasks. Here, we want to force the linear classifiers for different tasks to share their model weights for those features that are related to the common syntactic patterns. so we have:

$$\begin{aligned} \mathbf{w}^k &= \mathbf{v}^k + \mathbf{A} \mathbf{u} \\ &= \mathbf{v}^k + \mathbf{u}' \end{aligned}$$

The objective function will be:

$$\left(\{\hat{\mathbf{v}}^k\}_{k=1}^K, \hat{\mathbf{u}}' \right) = \arg \min_{\{\mathbf{v}^k\}, \mathbf{u}'} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, (\mathbf{v}^k + \mathbf{u}')) + r(\mathbf{v}^k) \right). \quad (7)$$

By eliminating \mathbf{v}^k , we can rewrite (7) as:

$$\left(\{\hat{\mathbf{w}}^k\}_{k=1}^K, \hat{\mathbf{u}}' \right) = \arg \min_{\{\mathbf{w}^k\}, \mathbf{u}'} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) + \lambda_k \|\mathbf{w}^k - \mathbf{u}'\|^2 \right). \quad (8)$$

In order to solve (8), we use the following alternating optimization procedure:

1. Fix \mathbf{u}' , and optimize (8) with respect to $\{\mathbf{w}^k\}$.
2. Fix $\{\mathbf{w}^k\}$, and optimize (8) respect to \mathbf{u}' .
3. Iterate until the stopping criterion is met.

In the first step, when \mathbf{u}' is fixed, objective function (8) will be:

$$\left(\{\hat{\mathbf{w}}^k\}_{k=1}^K \right) = \arg \min_{\{\mathbf{w}^k\}} \sum_{k=1}^K \left(\sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) + \lambda_k \|\mathbf{w}^k - \mathbf{u}'\|^2 \right) \quad (9)$$

which is similar to (4), so L-BFGS will be adopted.

In the second step, $\{\mathbf{w}^k\}$ are fixed. Then the objective function (8) will be:

$$\hat{\mathbf{u}}' = \arg \min_{\mathbf{u}'} \sum_{k=1}^K \lambda_k \|\mathbf{w}^k - \mathbf{u}'\|^2 \quad (10)$$

Equation (10) is a weighted least squares problem. The solution can be directly computed.

More details about the implementation of this model are described in Algorithm 2.

Algorithm 2: The Pseudocode of Solution for Method 2

Input: $\mathcal{D}_k = \{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_k}, \lambda_k$

Output: $\{\hat{\mathbf{w}}^k\}_{k=1}^K, \hat{\mathbf{u}}'$

- 1 Initialization: $\hat{\mathbf{u}}' = \mathbf{0}_F$
 - 2 **while** the stop criterion is not met **do**
 - 3 /*step 1*/
 - 4 **for** $i \leftarrow 1$ **to** K **do**
 - 5 | solve: $\hat{\mathbf{w}}^k = \arg \min_{\mathbf{w}^k} \sum_{i=1}^{N_k} L(\mathbf{x}_i^k, y_i^k, \mathbf{w}^k) + \lambda_k \|\mathbf{w}^k - \mathbf{u}'\|^2$
 - 6 **end**
 - 7 /*step 2*/
 - 8 | solve: $\hat{\mathbf{u}}' = \arg \min_{\mathbf{u}'} \sum_{k=1}^K \lambda_k \|\mathbf{w}^k - \mathbf{u}'\|^2$
 - 9 **end**
-

Relation Type	# Positive Instances	# Negative Instances	Sum
PHYS	1179	12190	13369
PER-SOC	363	3727	4090
EMP-ORG	1605	16466	18071
ART	204	2081	2285
OTHER-AFF	141	1566	1707
GPE-AFF	519	5319	5838
DISC	279	2975	3254
Total	4290	44324	48614

Table 8: Some information of the data set.

2.4.3 Experiment

Data set and experiment setup We used the ACE04 data set to evaluate our proposed methods. Each pair of entities within a single sentence is considered a candidate relation instance. After data cleaning, we obtained 4290 positive instances among 48614 candidate relation instances, and the statistical information of the data set is summarized in Table 8. Following [17], we extract unigram and bigram features from a sequence representation of each relation instance. There are seven relation types defined in ACE04. We took each relation type as the target type and used the remaining types as auxiliary types. This gave us seven sets of experiments.

In each set of experiments for a single target relation type, we randomly divided all the data into five subsets, and used one subset for testing while using the other four subsets for training, i.e., each experiment was repeated five times with different training and test sets.

For a specific partition, let \mathcal{S}_{train} denote the union of the source training data, \mathcal{T}_{train} denote the training data of the target type, and \mathcal{T}_{test} denote the test data of the target type. As we assume that for the target relation type there is only a small amount of positive instances, we therefore also randomly choose S positive instances and a proportional number of negative instances (positive vs. negative ratio is 1:10) from \mathcal{T}_{train} to get a smaller target training data set, which can be denoted as \mathcal{T}_{small} . Here we set S to 5.

Given the data generated in this way, we can consider different methods for comparison, which are summarized in Table 9. MT-1 and MT-2 refer to our proposed method 1 and method 2, respectively. BL-MT1 and BL-MT2 also use the multi-task learning framework but have sufficient training instances for both the source and the target relation types. The other methods are straightforward. We use logistic regression to train a liner classifier and then test on the testing data.

Comparison of different methods We first show the comparison of our proposed multi-task transfer learning methods with the baseline methods described above. The performance on each target relation type and the average performance across seven relation types are shown in Table 10. In this set of experiments, the number of positive seed instances for each target relation type is set to 5. The parameters are set to their optimal values ($\lambda = 1$ for all methods, $H = 7$ for BL-MT1 and MT-1).

We can draw the following conclusions from the table:

	Train	Test
BL-supervise	\mathcal{T}_{train}	\mathcal{T}_{test}
BL-big	$\mathcal{S}_{train} \cup \mathcal{T}_{train}$	\mathcal{T}_{test}
BL-MT1	$\mathcal{S}_{train} \cup \mathcal{T}_{train}$	\mathcal{T}_{test}
BL-MT2	$\mathcal{S}_{train} \cup \mathcal{T}_{train}$	\mathcal{T}_{test}
BL-source	\mathcal{S}_{train}	\mathcal{T}_{test}
BL-few	\mathcal{T}_{small}	\mathcal{T}_{test}
BL-small	$\mathcal{S}_{train} \cup \mathcal{T}_{small}$	\mathcal{T}_{test}
MT-1	$\mathcal{S}_{train} \cup \mathcal{T}_{small}$	\mathcal{T}_{test}
MT-2	$\mathcal{S}_{train} \cup \mathcal{T}_{small}$	\mathcal{T}_{test}

Table 9: Methods for comparison.

- BL-few gives a performance lower bound, while BL-supervise gives a performance upper bound.
- BL-big performs worse than BL-supervise as we use the instances from other types to train the linear model directly, while BL-MT1 and BL-MT2 give similar results compared with BL-supervise as the multi-task learning framework can transfer the knowledge from other types.
- BL-source shows that without any transfer or co-training, the performance of source training data is poor.
- BL-small performs poorly compared with BL-few, which shows the necessity of multi-task transfer learning. Compared with BL-few, MT-1 improves the performance, which shows the benefit of our proposed multi-task learning method. But MT-2 performs worse than BL-few. The reason could be that as we do not impose any constraints, the type-dependent features’ weights are also shared between different tasks. We will investigate how to impose such constraints in the next paragraph by changing the value of λ .

The effect of λ Let us now take a look at the effect of using a different λ . All the settings are the same with the previous section, except $\lambda = 1000$ (As the limitation of running time, we just try a different value for parameter λ). The results are summarized in Table 11.

From the table, we can see that when we set λ to 1000, the performance is poor. For our proposed methods MT-1 and MT-2, a large value of λ forces the algorithm to share more common weights between different tasks, which may lead to non-common features to be shared. For BL-MT1 and BL-MT2, the results are similar to the above section, which suggests that these two multi-task learning models are not sensitive to the parameter λ . For all the other baseline methods, a large λ prefers small weights for the classifier, which may make the distinctive features less effective. But these results also prove the conclusion we draw in the above section.

Target Type \mathcal{T}		BL-supervise	BL-big	BL-MT1	BL-MT2	BL-source	BL-few	BL-small	MT-1	MT-2
<i>Physical</i>	P	0.8349	0.7591	0.7732	0.7552	0.4057	0.7187	0.4244	0.6776	0.4837
	R	0.7531	0.6093	0.7503	0.7279	0.1191	0.1218	0.1041	0.2122	0.1178
	F	0.7971	0.6752	0.7611	0.7408	0.1842	0.1925	0.1662	0.3115	0.1883
<i>Personal /Social</i>	P	0.8846	0.7721	0.8723	0.8539	0.5238	0.7104	0.5023	0.6019	0.6780
	R	0.9200	0.7441	0.8629	0.8544	0.1466	0.4262	0.2068	0.4595	0.4963
	F	0.9019	0.7567	0.8664	0.8532	0.2291	0.4996	0.2917	0.5013	0.5576
<i>Employment /Membership /Subsidiary</i>	P	0.8664	0.8173	0.8525	0.8503	0.5285	0.8014	0.5654	0.7851	0.6496
	R	0.8848	0.8130	0.8396	0.8359	0.2242	0.3554	0.2577	0.3894	0.2604
	F	0.8755	0.8151	0.8457	0.8430	0.3148	0.4870	0.3539	0.5157	0.3709
<i>Agent-Artifact</i>	P	0.9268	0.8114	0.9332	0.9045	0.6923	0.9510	0.5201	0.9311	0.6109
	R	0.9268	0.7904	0.8939	0.8388	0.2195	0.4556	0.2075	0.5035	0.2407
	F	0.9268	0.8000	0.9120	0.8700	0.3333	0.6081	0.2948	0.6442	0.3432
<i>PER/ORG Affiliation</i>	P	0.8750	0.8063	0.8529	0.8472	0.9090	0.7821	0.7409	0.7853	0.8133
	R	0.8750	0.7970	0.8762	0.8431	0.4166	0.8042	0.4842	0.8237	0.5404
	F	0.8750	0.8000	0.8636	0.8447	0.5714	0.7890	0.5812	0.7998	0.6366
<i>GPE Affiliation</i>	P	0.8529	0.7977	0.8939	0.8519	0.7580	0.7756	0.7263	0.7152	0.7443
	R	0.8700	0.8113	0.8640	0.8397	0.4700	0.5528	0.4968	0.6120	0.4747
	F	0.8613	0.8041	0.8783	0.8453	0.5802	0.6422	0.5885	0.6542	0.5767
<i>Discourse</i>	P	0.8947	0.7435	0.8529	0.8382	0.3000	0.4524	0.3480	0.4661	0.5704
	R	0.8947	0.6451	0.8822	0.8397	0.1052	0.2225	0.1084	0.2789	0.2641
	F	0.8947	0.6876	0.8666	0.8378	0.1558	0.2973	0.1630	0.3469	0.3591
<i>Average</i>	P	0.8765	0.7868	0.8616	0.8430	0.5882	0.7417	0.5468	0.7089	0.6500
	R	0.8749	0.7443	0.8527	0.8256	0.2430	0.4198	0.2665	0.4685	0.3421
	F	0.8760	0.7627	0.8563	0.8333	0.3384	0.5022	0.3485	0.5391	0.4332

Table 10: Comparison of different methods on ACE 2004 data set. P, R and F stand for precision, recall and F1, respectively. $\lambda = 1$.

Target Type \mathcal{T}		BL-supervise	BL-big	BL-MT1	BL-MT2	BL-source	BL-few	BL-small	MT-1	MT-2
<i>Physical</i>	P	0.8765	0.6999	0.8156	0.7787	0.4745	0.0000	0.4925	0.8181	0.4540
	R	0.3021	0.2701	0.7611	0.6322	0.1191	0.0000	0.1049	0.0918	0.1039
	F	0.4493	0.3891	0.7873	0.6974	0.1904	0.0000	0.1722	0.1528	0.1687
<i>Personal /Social</i>	P	0.8913	0.8327	0.9028	0.8252	0.0000	0.0000	0.0649	0.7943	0.3570
	R	0.5466	0.6022	0.8538	0.7735	0.0000	0.0000	0.0086	0.4867	0.0866
	F	0.6776	0.6976	0.8774	0.7969	0.0000	0.0000	0.0151	0.5599	0.1380
<i>Employment /Membership /Subsidiary</i>	P	0.8413	0.8347	0.8981	0.8770	0.6800	0.0000	0.7136	0.8066	0.6314
	R	0.6909	0.6263	0.8606	0.7977	0.2060	0.0000	0.2207	0.3083	0.2414
	F	0.7587	0.7150	0.8788	0.8353	0.3162	0.0000	0.3367	0.4422	0.3490
<i>Agent-Artifact</i>	P	0.9285	0.8693	0.9477	0.8812	0.9130	0.0000	0.8605	0.9366	0.8385
	R	0.3170	0.6391	0.8840	0.7270	0.5121	0.0000	0.6185	0.3367	0.5003
	F	0.4727	0.7322	0.9129	0.7951	0.6562	0.0000	0.7157	0.4883	0.6196
<i>PER/ORG Affiliation</i>	P	1.0000	0.8926	0.8633	0.8745	0.9285	0.0000	0.8715	0.8014	0.8385
	R	0.4583	0.5912	0.8743	0.8076	0.5416	0.0000	0.5012	0.7981	0.5269
	F	0.6285	0.7035	0.8683	0.8392	0.6842	0.0000	0.6292	0.7962	0.6441
<i>GPE Affiliation</i>	P	0.8571	0.8541	0.8985	0.8592	0.8709	0.0000	0.8482	0.8134	0.8233
	R	0.4200	0.6794	0.8578	0.7980	0.5400	0.0000	0.5888	0.4951	0.5956
	F	0.5637	0.7554	0.8775	0.8266	0.6666	0.0000	0.6931	0.6135	0.6894
<i>Discourse</i>	P	0.0000	0.4271	0.9171	0.7616	0.2000	0.0000	0.3151	0.4898	0.4301
	R	0.0000	0.1086	0.8369	0.6059	0.0350	0.0000	0.0723	0.2006	0.1058
	F	0.0000	0.1723	0.8748	0.6759	0.0579	0.0000	0.1171	0.2819	0.1683
<i>Average</i>	P	0.7707	0.7729	0.8923	0.8368	0.5810	0.0000	0.5952	0.7800	0.6247
	R	0.3907	0.5024	0.8469	0.7351	0.2791	0.0000	0.3021	0.3882	0.3086
	F	0.5072	0.5950	0.8681	0.7809	0.3674	0.0000	0.3827	0.4764	0.3967

Table 11: Comparison of different methods on ACE 2004 data set. P, R and F stand for precision, recall and F1, respectively. $\lambda = 1000$.

Sensitivity of H Another parameter for MT-1 and BL-MT1 is the dimensionality H . To see how the performance may vary as H changes, we try different settings. The results are summarized in Table 12. As we can see from the table, the performance of MT-1 and BL-MT1 are relatively stable, which suggests that these multi-task methods are not very sensitive to the value of H .

Target Type \mathcal{T}		BL-MT1			MT-1		
		$H = 1$	$H = 4$	$H = 7$	$H = 1$	$H = 4$	$H = 7$
<i>Physical</i>	P	0.7500	0.7763	0.7732	0.6934	0.6312	0.6776
	R	0.7143	0.7558	0.7503	0.1603	0.1811	0.2122
	F	0.7314	0.7655	0.7611	0.2513	0.2761	0.3115
<i>Personal /Social</i>	P	0.8589	0.8632	0.8723	0.7174	0.8197	0.6091
	R	0.8446	0.8497	0.8629	0.4094	0.6325	0.4595
	F	0.8513	0.8554	0.8664	0.4917	0.6972	0.5013
<i>Employment /Membership /Subsidiary</i>	P	0.8217	0.8559	0.8525	0.8168	0.8091	0.7851
	R	0.8311	0.8408	0.8396	0.3384	0.3694	0.3894
	F	0.8263	0.8481	0.8457	0.4757	0.5043	0.5157
<i>Agent-Artifact</i>	P	0.9595	0.9563	0.9332	0.9338	0.8993	0.9311
	R	0.8939	0.8541	0.8939	0.3954	0.4115	0.5035
	F	0.9252	0.9014	0.9120	0.5521	0.5621	0.6442
<i>PER/ORG Affiliation</i>	P	0.8471	0.8988	0.8529	0.8590	0.9181	0.7853
	R	0.8718	0.8679	0.8762	0.8021	0.7412	0.8237
	F	0.8585	0.8824	0.8636	0.8268	0.8153	0.7998
<i>GPE Affiliation</i>	P	0.8617	0.8486	0.8939	0.8361	0.8803	0.7152
	R	0.8487	0.8443	0.8640	0.5625	0.4653	0.6120
	F	0.8550	0.8462	0.8783	0.6713	0.6070	0.6542
<i>Discourse</i>	P	0.8868	0.8404	0.8529	0.4627	0.7437	0.4661
	R	0.8695	0.8673	0.8822	0.2268	0.2947	0.2789
	F	0.8769	0.8536	0.8666	0.3024	0.4201	0.3469
<i>Average</i>	P	0.8551	0.8628	0.8616	0.7599	0.8144	0.7089
	R	0.8391	0.8400	0.8527	0.4136	0.4423	0.4685
	F	0.8464	0.8504	0.8563	0.5102	0.5546	0.5391

Table 12: Effect of the dimensionality of low space. P, R and F stand for precision, recall and F1, respectively. $\lambda = 1$.

Changing the number of seed instances Finally, we compare BL-few with MT-1 and MT-2 when the number of positive seed instances increases. We set S from 2 up to 100, and the results are shown in Figure 2.5.5. We can see that when the number of positive seed instances is small, as we expected, MT-1 and MT-2 outperforms baseline method BL-few. When S is large, the problem becomes more like traditional supervised learning problem, in this case, the performance of these three algorithms are quite similar.

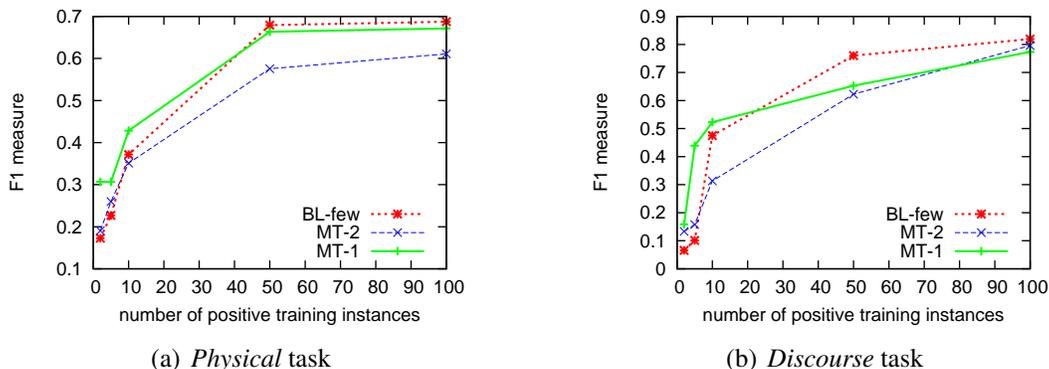


Figure 1: The effect of the amount of training data on the performance.

2.5 Extraction of Relation Descriptors

2.5.1 Problem definition

Depending on the application and the resources available, relation extraction has been studied in a number of different settings. When relation types are well defined and labeled relation mention instances are available, supervised learning is usually applied [36, 39, 5, 37]. When relation types are known but little training data is available, bootstrapping has been used to iteratively expand the set of seed examples and relation patterns [1]. When no relation type is pre-defined but there is a focused corpus of interest, unsupervised relation discovery tries to cluster entity pairs in order to identify interesting relation types [14, 28, 31]. More recently, open relation extraction has also been proposed where there is no fixed domain or pre-defined relation type and the goal is to identify all possible relations from an open-domain corpus [3].

These different relation extraction settings suit different applications. In this section, we focus on another setting where the relation types are defined at a general level but a more specific relation description is desired. For example, in the widely used ACE³ data sets, relation types are defined at a fairly coarse granularity. Take for instance the “employment” relation, which is a major relation type defined in ACE. In ACE evaluation, extraction of this relation only involves deciding whether a person entity is employed by an organization entity. In practice, however, we often also want to find the exact job title or position this person holds at the organization if this information is also mentioned in the text. Table 13 gives some examples. We refer to the segment of text that describes the specific relation between the two related entities (i.e. the two arguments) as the *relation descriptor*. This section studies how to extract such relation descriptors given two arguments.

One may approach this task as a sequence labeling problem and apply methods such as the linear-chain conditional random fields (LC-CRF) [19]. However, this solution ignores a useful property of the task: the space of possible label sequences is much smaller than that enumerated by an LC-CRF. There are two implications. First, the normalization constant in the LC-CRF is too large because it also enumerates the impossible sequences. Second, the restriction to the correct

³Automatic Content Extraction <http://www.itl.nist.gov/iad/mig/tests/ace/>

Relation	Candidate Relation Instance	Relation Descriptor
Employment (PER, ORG)	... said <i>ARG-1</i> , a vice president at <i>ARG-2</i> , which ...	a vice president
	A <i>ARG-2</i> spokesman , <i>ARG-1</i> , said the company now ...	spokesman
	At <i>ARG-2</i> , by contrast , <i>ARG-1</i> said customers spend on ...	<i>Nil</i>
Social (PER, PER)	<i>ARG-1</i> had an elder brother named <i>ARG-2</i> .	an elder brother
	<i>ARG-1</i> was born at ... , as the son of <i>ARG-2</i> of Sweden ...	the son
	<i>ARG-1</i> later married <i>ARG-2</i> in 1973 , ...	married
	Through his contact with <i>ARG-1</i> , <i>ARG-2</i> joined the Greek_Orthodox_Church .	<i>Nil</i>

Table 13: Some examples of candidate relation instances and their relation descriptors.

space of label sequence permits the use of long-range features without an exponential increase in computational cost. So we propose a modified CRF model for this problem.

Now we define the task of extracting relation descriptors for a given pre-defined class of relations such as “employment.” Given two named entities occurring in the same sentence, one acting as *ARG-1* and the other as *ARG-2*, we aim to extract a segment of text from the sentence that best describes a pre-defined general relation between the two entities.

Formally, let (w_1, w_2, \dots, w_n) denote the sequence of tokens in a sentence, where w_p is *ARG-1* and w_q is *ARG-2* ($1 \leq p, q \leq n, p \neq q$). Our goal is to locate a subsequence (w_r, \dots, w_s) ($1 \leq r \leq s \leq n$) that best describes the relation between *ARG-1* and *ARG-2*. If *ARG-1* and *ARG-2* are not related through the pre-defined general relation, *Nil* should be returned.

Note that the above definition constrains *ARG-1* and *ARG-2* to single tokens. In our experiments, we will replace the original name strings of *ARG-1* and *ARG-2* with the generic tokens ARG1 and ARG2 because we believe that the relation extraction models should be independent on the actual names of the arguments. Examples of sentences with the named entities replaced with argument tokens are shown in the second column of Table 13.

2.5.2 Related Work

Most existing work on relation extraction studies binary relations between two entities. For supervised relation extraction, existing work often uses the ACE benchmark data sets for evaluation [5, 39, 37]. In this setting, a set of relation types are defined and the task is to identify pairs of entities that are related and to classify their relations into one of the pre-defined relation types. It is assumed that the relation type itself is sufficient to characterize the relation between the two related entities. However, based on our observation, some of the relation types defined in ACE such as the “employment” relation and the “personal/social” relation are very general and can be further characterized by more specific descriptions.

Recently open relation extraction has been proposed for open-domain information extraction [3]. Since in the open domain we cannot assume a fixed set of relation types, open relation extraction aims at extracting all possible relations between pairs of entities. The extracted results are $(ARG-1, REL, ARG-2)$ tuples. The TextRunner system based on [3] extracts a diverse set of relations from a huge Web corpus. These extracted predicate-argument tuples are presumably the

most useful to support Web search scenarios where the user is looking for specific relations. However, because of the diversity of the extracted relations and the domain independence, open relation extraction is probably not suitable for populating relational databases or knowledgebases. In contrast, the task of extracting relation descriptors as we have proposed still assumes a pre-defined general relation type, which ensures that the extracted tuples follow the same relation definition and thus can be used in applications such as populating relational database tables.

In terms of models and techniques, we use standard linear-chain CRF as our baseline, which is the main method used in [3] as well as for many other information extraction problems. The major modifications we propose for our task are reduction of the label sequence space and the incorporation of long-range features. We note that these modifications are closely related to the semi-Markov CRF models proposed by [29]. In fact, the modified CRF model for our task can be considered as a special case of semi-Markov CRF where we only consider label sequences that contain at most one relation descriptor sequence.

2.5.3 Method

A Linear-Chain CRF Solution The relation descriptor extraction task can be naturally treated as a sequence labeling problem. Let $\mathbf{x} = (x_1, x_2, \dots, x_n)$ denote the sequence of observations in a relation instance, where x_i is w_i augmented with additional information such as the POS tag of w_i , the phrase boundary information, etc. Each observation x_i is associated with a label $y_i \in \mathcal{Y}$ which indicates whether w_i is part of the relation descriptor. Following the commonly used BIO notation in sequence labeling, we define $\mathcal{Y} = \{B-REL, I-REL, O\}$. Let $\mathbf{y} = (y_1, y_2, \dots, y_n)$ denote the sequence of labels for \mathbf{x} . Our task can be reduced to finding the best label sequence \mathbf{y}^* among all the possible label sequences for \mathbf{x} .

For sequence labeling tasks in NLP, linear-chain conditional random field has been rather successful. It is an undirected graphical model in which the conditional probability of a label sequence \mathbf{y} given the observation sequence \mathbf{x} is

$$p(\mathbf{y}|\mathbf{x}, \Lambda) = \frac{\exp\left(\sum_i \sum_k \lambda_k f_k(y_{i-1}, y_i, \mathbf{x})\right)}{Z(\mathbf{x}, \Lambda)}, \quad (11)$$

where $\Lambda = \{\lambda_k\}$ is the set of model parameters, f_k is an arbitrary feature function defined over two consecutive labels and the whole observation sequence, and

$$Z(\mathbf{x}, \Lambda) = \sum_{\mathbf{y}'} \exp\left(\sum_i \sum_k \lambda_k f_k(y'_{i-1}, y'_i, \mathbf{x})\right) \quad (12)$$

is a normalization constant.

Given a set of training instances $\{\mathbf{x}_j, \mathbf{y}_j^*\}$ where \mathbf{y}_j^* is the correct label sequence for \mathbf{x}_j , we can learn the best model parameters $\hat{\Lambda}$ as follows:

$$\hat{\Lambda} = \arg \min_{\Lambda} \left(- \sum_j \log p(\mathbf{y}_j^*|\mathbf{x}_j, \Lambda) + \beta \sum_k \lambda_k^2 \right). \quad (13)$$

Here $\beta \sum_k \lambda_k^2$ is a regularization term.

Improvement over Linear-Chain CRF We note that while we can directly apply linear-chain CRF to extract relation descriptors, there are some special properties of our task that allow us to modify standard linear-chain CRF to better suit our needs.

Label sequence constraint In linear-chain CRF, the normalization constant Z considers all possible label sequences \mathbf{y} . For the relation descriptor extraction problem, however, we expect that there is either a single relation descriptor sequence or no such sequence. In other words, for a given relation instance, we only expect two kinds of label sequences: (1) All y_i are O . (2) Exactly one y_i is $B-REL$ followed by zero or more consecutive $I-REL$ while all other y_i are O . Therefore the space of label sequences should be reduced to only those that satisfy the above constraint.

One way to exploit this constraint within linear-chain CRF is to only consider it during testing. We can pick the label sequence that has the highest probability in the *valid* label sequence space instead of the entire label sequence space. For a candidate relation instance \mathbf{x} , let $\tilde{\mathbf{Y}}$ denote the set of valid label sequences, i.e. those that has either one or no relation descriptor sequence. We then choose the best sequence $\hat{\mathbf{y}}$ as follows:

$$\hat{\mathbf{y}} = \arg \max_{\mathbf{y} \in \tilde{\mathbf{Y}}} p(\mathbf{y} | \mathbf{x}, \hat{\Lambda}). \quad (14)$$

However, to fully exploit the constraint, at the training stage, we should also consider only $\tilde{\mathbf{Y}}$ by defining the normalization term Z as follows:

$$Z(\mathbf{x}, \Lambda) = \sum_{\mathbf{y}' \in \tilde{\mathbf{Y}}} \exp \left(\sum_i \sum_k \lambda_k f_k(y'_{i-1}, y'_i, \mathbf{x}) \right). \quad (15)$$

Note that the difference between Equation (15) and Equation (12) is the set of label sequences to consider. In other words, while in linear-chain CRF the correct label sequence competes with all possible label sequences for the probability mass, for our task the correct label sequence should compete with only other valid label sequences. In Section 2.5.5 we will compare these two different normalization terms and show the advantage of using Equation (15).

Adding long-range features In linear-chain CRF models, only first-order label dependencies are considered because features are defined over two consecutive labels. More general higher-order CRF models also exist, allowing long-range features defined over more than two consecutive labels. But the computational cost of higher-order CRFs also increases exponentially. For our task, because of the constraint on the space of label sequences, we can afford to define long-range features without blowing up the feature space. Let $g(\mathbf{y}, \mathbf{x})$ denote a feature function defined over the entire label sequence \mathbf{y} and the observation sequence \mathbf{x} . We can include such feature functions in our model as follows:

Description	Feature Template	Example
single token	w_i ($-2 \leq i \leq 2$)	w_1 is <i>president</i>
single POS tag	t_i ($-2 \leq i \leq 2$)	t_1 is <i>N</i>
single phrase tag	p_i ($-2 \leq i \leq 2$)	p_1 is <i>I-NP</i>
two consecutive tokens	$w_{i-1} \& w_i$ ($-1 \leq i \leq -2$)	w_0 is <i>the</i> and w_1 is <i>president</i>
two consecutive POS tags	$t_{i-1} \& t_i$ ($-1 \leq i \leq -2$)	t_0 is <i>DET</i> and t_1 is <i>N</i>
two consecutive phrase tags	$p_{i-1} \& p_i$ ($-1 \leq i \leq -2$)	p_0 is <i>B-NP</i> and p_1 is <i>I-NP</i>

Table 14: Linear-chain feature templates. Each feature is defined with respect to a particular (current) position in the sequence. i indicates the position relative to the current position. All features are defined using observations within a window size of 5 of the current position.

$$p(\mathbf{y}|\mathbf{x}, \Theta) = \frac{1}{Z(\mathbf{x}, \Theta)} \left[\exp \left(\sum_i \sum_k \lambda_k f_k(y_{i-1}, y_i, \mathbf{x}) + \sum_l \mu_l g_l(\mathbf{y}, \mathbf{x}) \right) \right], \quad (16)$$

where $\Theta = \{\{\lambda_k\}, \{\mu_l\}\}$ is the set of all model parameters. Note that although each $f(y_{i-1}, y_i, \mathbf{x})$ can also be regarded as a $g(\mathbf{y}, \mathbf{x})$, here we group all features that can be captured by linear-chain CRF under f and other real long-range features under g . In Section 2.5.5 we will see that with the additional g feature functions relation extraction performance can also be further improved.

2.5.4 Features

We now describe the features we use in the baseline linear-chain CRF model and our modified CRF model.

Linear-chain features The linear-chain features are those that can be formulated as $f(y_{i-1}, y_i, \mathbf{x})$, i.e. those that depend on \mathbf{x} and two consecutive labels only. We use typical features that include tokens, POS tags and phrase boundary information coupled with label values. Let t_i denote the POS tag of w_i and p_i denote the phrase boundary tag of w_i . The phrase boundary tags also follow the BIO notation. Examples include *B-NP*, *I-VP*, etc. Table 14 shows the feature templates covering only the observations. Each feature shown in Table 14 is further combined with either the value of the current label y_0 or the values of the previous and the current labels y_{-1} and y_0 to form zeroth order and first order features. For example, a zeroth order feature can be “ y_0 is *B-REL* and w_0 is *the* and w_1 is *president*”, and a first order feature can be “ y_{-1} is *O* and y_0 is *B-REL* and t_0 is *N*”.

Note that since we represent the two arguments using tokens ARG1 and ARG2, the argument positions are given as part of the observation sequence \mathbf{x} , and hence are implicitly captured by the zeroth and first order features.

Category	Feature Template Description	Example
Contextual Features	word w_{r-1} or POS tag t_{r-1} preceding relation descriptor word w_{s+1} or POS tag t_{s+1} following relation descriptor	, REL REL PREP
Path-based Features	word or POS tag sequence between ARG1 and relation descriptor word or POS tag sequence between ARG2 and relation descriptor word or POS tag sequence containing ARG1, ARG2 and relation descriptor	ARG1 is REL REL PREP ARG2 ARG2 's REL , ARG1
Phrase Boundary Features	whether relation descriptor violates phrase boundaries	1 or 0

Table 15: Long-range feature templates. r and s are the indices of the first word and the last word of the relation descriptor, respectively.

Long-range features Long-range features are those that cannot be defined based on only two consecutive labels. When defining long-range features, we treat the whole relation descriptor sequence as a single unit, denoted as REL. Given a label sequence \mathbf{y} that contains a relation descriptor sequence, let $(w_r, w_{r+1}, \dots, w_s)$ denote the relation descriptor, that is, $y_r = B-REL$ and $y_t = I-REL$ where $(r + 1 \leq t \leq s)$. The long-range features we use are categorized and summarized in Table 15. These features capture the context of the entire relation descriptor, its relation to the two arguments, and whether the boundary of the relation descriptor conforms to the phrase boundaries (since we expect that most relation descriptors consist of a single or a sequence of phrases).

2.5.5 Experiments

Data Preparation Since the task of extracting relation descriptors is new, we are not aware of any data set that can be directly used to evaluate our methods. We therefore annotated two data sets for evaluation, one for the general “employment” relation and the other for the general “personal/social” relation⁴.

The first data set contains 150 business articles from New York Times. The articles were crawled from the NYT website between November 2009 and January 2010. After sentence splitting and tokenization, we used the Stanford NER tagger⁵ to identify PER and ORG named entities from each sentence. For named entities that contain multiple tokens we concatenated them into a single token. We then took each pair of (PER, ORG) entities that occur in the same sentence as a single candidate relation instance, where the PER entity is treated as ARG-1 and the ORG entity is treated as ARG-2. A human annotator manually went through each candidate relation instance to decide (1) whether there is an employment relation between the two arguments and (2) whether

⁴We will make our data sets publicly available.

⁵<http://nlp.stanford.edu/ner/index.shtml>

Data Set	total	positive	negative	distinct descriptors
NYT	536	208	328	140
Wikipedia	700	122	578	70

Table 16: Numbers of instances in each data set. Positive instances are those that have an explicit relation descriptor. The last column shows the number of distinct relation descriptor strings.

there is an explicit sequence of words describing the position or job title held by *ARG-1* in *ARG-2*.

The second data set comes from a Wikipedia personal relation data set previously used in [11]. The original data set⁶ does not contain annotations of relation descriptors such as “sister” or “friend” between the two PER arguments. Our human annotator therefore also manually annotated this data set. Similarly, we performed sentence splitting, tokenization and NER tagging, and took pairs of (PER, PER) entities occurring in the same sentence as a candidate relation instance. Because both arguments involved in the “personal/social” relation are PER entities, we always treat the first PER entity as *ARG-1* and the second PER entity as *ARG-2*⁷.

Note that our annotated relation descriptors are not always nouns or noun phrases. An example can be found in the last but one row of Table 13, where the relation descriptor “married” is a verb and indicates a spouse relation.

The total numbers of relation instances, the numbers of positive and negative instances as well as the numbers of distinct relation descriptor strings in each data set are summarized in Table 16.

Experiment Setup We compare the following methods in our experiments:

- **LC-CRF:** This is the standard linear-chain CRF model with features described in Table 14.
- **M-CRF-1:** This is our modified CRF model with the space of label sequences reduced but with features fixed to the same as those used in LC-CRF.
- **M-CRF-2:** On top of M-CRF-1, in M-CRF-2 we include the contextual long-range features as described in Table 15.
- **M-CRF-3:** On top of M-CRF-2, in M-CRF-3 we include the path-based long-range features as described in Table 15.
- **M-CRF-4:** On top of M-CRF-3, in M-CRF-4 we include the phrase boundary long-range features as described in Table 15.

For the standard linear-chain CRF model, we used the package CRF++⁸. We implemented our own version of the modified CPF models from scratch.

We perform 10-fold cross validation for all our experiments. For each data set we first randomly divide it into 10 subsets. Each time we take 9 subsets for training and the remaining subset for

⁶<http://www.cs.umass.edu/~culotta/data/wikipedia.html>

⁷Since many personal/social relations are asymmetric, ideally we should assign *ARG-1* and *ARG-2* based on their semantic meanings rather than their positions. Here we take a simple approach.

⁸<http://crfpp.sourceforge.net/>

testing. We report the average performance across the 10 runs. It also enables us to perform statistical significance tests.

Based on our preliminary experiments, we found that using a smaller set of general POS tags instead of the standard Penn Treebank POS tag set could slightly improve the overall performance. We therefore only report the performance obtained using the general POS tags. For example, *NN*, *NNP*, *NNS* and *NNPS* are grouped together into a general tag *N*.

We evaluate the performance using two different criteria: overlap match and exact match. Overlap match is a more relaxed criterion: if the predicted relation descriptor overlaps with the true relation descriptor (i.e. having at least one token in common), it is considered correct. Exact match is a much stricter criterion: it requires that the predicted label sequence must be exactly the same as the true label sequence in order to be considered correct. Given these two criteria, we can define accuracy, precision, recall and F1 measures. Accuracy is the percentage of candidate relation instances whose label sequence is considered correct. Both positive and negative instances are counted when computing accuracy. Because our data sets are quite balanced, it is reasonable to use accuracy. Precision, recall and F1 are defined in the usual way at the relation instance level.

Method Comparison In Table 17 and Table 18, we summarize the performance in terms of the various measures on the two data sets, respectively. For both the baseline LC-CRF model and our modified CRF models, there is a regularization parameter β that needs to be manually set. (See Equation (13).) We tuned this parameter and show only the results using the optimal parameter values for each data set.

Method	Overlap Match				Exact Match			
	Accu.	Prec.	Rec.	F1	Accu.	Prec.	Rec.	F1
LC-CRF	0.8173	0.8407	0.6548	0.7303	0.8117	0.8373	0.6394	0.7186
M-CRF-1	0.8491 [†]	0.8640	0.7202 [†]	0.7830 [†]	0.8454 [†]	0.8625	0.7124 [†]	0.7774 [†]
M-CRF-2	0.8491	0.8627	0.7202	0.7819	0.8454	0.8617	0.7124	0.7763
M-CRF-3	0.8659[†]	0.9000[†]	0.7364	0.8070[†]	0.8640[†]	0.8992[†]	0.7319[†]	0.8038[†]
M-CRF-4	0.8659	0.9000	0.7364	0.8070	0.8640	0.8992	0.7319	0.8038

Table 17: Comparison of different methods on the New York Times data set. Accu., Prec., Rec. and F1 stand for accuracy, precision, recall and F1 measures, respectively. [†] indicates that the current value is statistically significantly better than the value in the previous row at a 0.95 level of confidence by one-tailed T-test.

First of all, we can see from the tables that by reducing the label sequence space, M-CRF-1 can significantly outperform the baseline LC-CRF in terms of F1 in all cases. In terms of accuracy, there is significant improvement for the NYT data set but not for the Wikipedia data set. We also notice that for both data sets the advantage of M-CRF-1 are mostly evident in the improvement of recall. It shows that by reducing the label sequence space we are able to extract more correct relation descriptors.

Next we see from the tables that long-range features are also useful, and the improvement comes mostly from the path-based long-range features. In terms of both accuracy and F1, M-

Method	Overlap Match				Exact Match			
	Accu.	Prec.	Rec.	F1	Accu.	Prec.	Rec.	F1
LC-CRF	0.8486	0.6513	0.3140	0.4137	0.8457	0.6489	0.2980	0.3931
M-CRF-1	0.8414	0.5648	0.4233 [†]	0.4778 [†]	0.8386	0.5530	0.4072 [†]	0.4609 [†]
M-CRF-2	0.8471	0.5859	0.4260	0.4873	0.8443	0.5741	0.4099	0.4704
M-CRF-3	0.8657 [†]	0.6847 [†]	0.4488	0.5318[†]	0.8628 [†]	0.6823 [†]	0.4327	0.5144[†]
M-CRF-4	0.8671	0.6966	0.4388	0.5278	0.8643	0.6942	0.4228	0.5105

Table 18: Comparison of different methods on the Wikipedia data set. Accu., Prec., Rec. and F1 stand for accuracy, precision, recall and F1 measures, respectively. [†] indicates that the current value is statistically significantly better than the value in the previous row at a 0.95 level of confidence by one-tailed T-test.

CRF-3 can significantly outperform M-CRF-1 in all settings. In this case, the improvement is a mixture of both precision and recall. It shows that by explicitly capturing the patterns between the two arguments and the relation descriptor, we can largely improve the extraction performance. On the other hand, neither the contextual long-range features nor the phrase boundary long-range features exhibit any impact significantly. We hypothesize the following reasons: for contextual long-range features, they have already been captured in the linear-chain features. For example, the long-range feature “*is REL*” is similar to the linear-chain feature “ $w_{-1} = is \ \& \ y_0 = B-R$ ”. For the phrase boundary long-range feature, since phrase boundary tags have also been used in the linear-chain features, this feature may not provide further help. In addition, we examined our annotated data and found that for the NYT data set around 22% of the positive instances actually have relation descriptors violating phrase boundaries, and in the Wikipedia data set the percentage increases to around 29%. Therefore phrase boundaries are not an important factor when extracting relation descriptors.

Overall, performance is much higher on the NYT data set than on the Wikipedia data set. Based on our observations during annotation, this is due to the fact that the “employment” relations expressed in the NYT data set often follow some standard patterns, whereas in Wikipedia the “personal/social” relations can be expressed in various different ways. The relatively low performance achieved on the Wikipedia data set suggests that extracting relation descriptors is not an easy task even under a supervised learning setting.

Presumably relation descriptors that are not seen in the training data are harder to extract. We would therefore also like to see how well our model works on such unseen relation descriptors. We find that with 10-fold cross validation, for the NYT data set, on average our model is able to extract approximately 67% of the unseen relation descriptors in the test data, and for the Wikipedia data set this percentage is approximately 27%. Both numbers are lower than the overall recall values the model can achieve on the entire test data, showing that unseen relation descriptors are indeed harder to extract. But it still shows that our model is able to pick up new relation descriptors.

The Effect of Training Data Size For the previous experiments, we always used 90% of the data for training and the remaining 10% for testing. We now take a look at how the performance

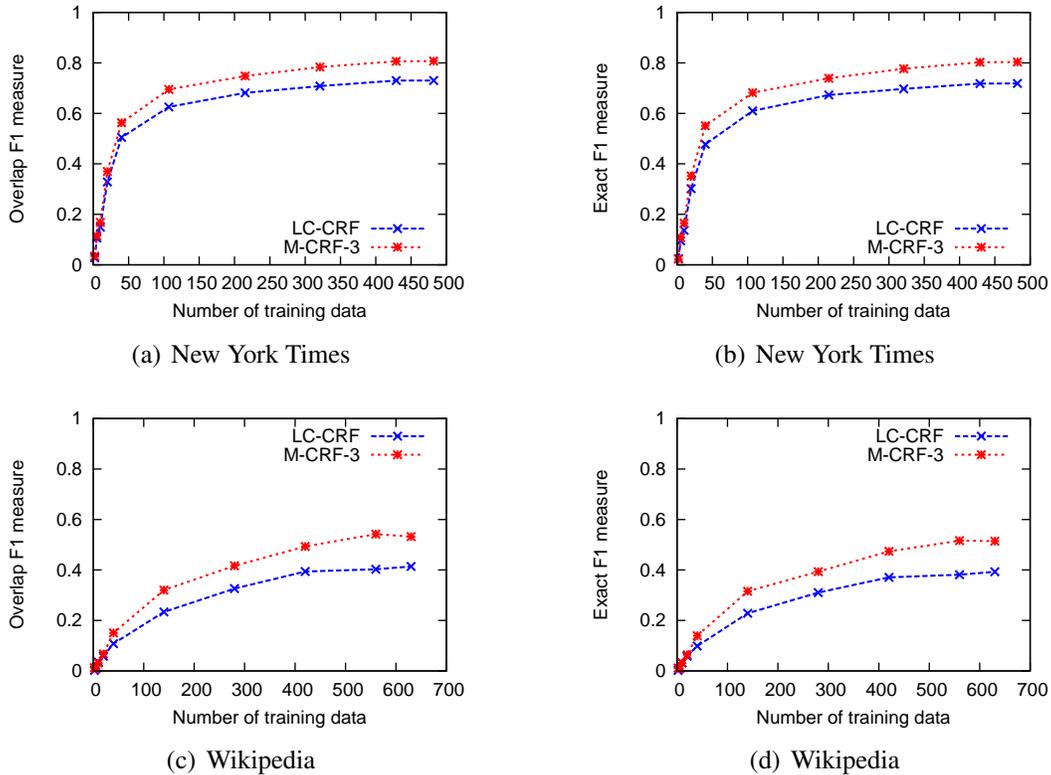


Figure 2: The effect of the training data size

changes with different numbers of training instances. We vary the training data size from only a few instances (2, 5, and 10) to 20%, 40%, 60% and 80% of the entire data set. The results are shown in Figure 1.

As we can expect, when the number of training instances is small, the performance on both data sets is low. The figure also shows that between the two data sets the Wikipedia data set is the more difficult one, which is consistent with our observations with the data as mentioned earlier.

The modified CRF model consistently outperforms the baseline LC-CRF model, which is not surprising. We can see that with the modified CRF model to achieve the same level of performance the required training data size can be much smaller compared with the baseline LC-CRF model. For example, Figure 2(b) shows that with 482 training instances the LC-CRF model gives an exact F1 of around 0.719, but to achieve this level of F1, the modified CRF model only needs fewer than 215 training instances.

2.6 Conclusions

This is the final report of the DSO funded project on transfer learning for adaptive relation extraction. We presented three pieces of work:

- **Cross-domain Relation Extraction:** First, we proposed and compared a number of domain

similarity measures for relation extraction and found that a lexicalized tree kernel-based measure was the most effective in finding good source domains. We then applied the domain similarity measures in a multitask relation extraction framework and found that when the amount of training data was small multitask learning could help some domains.

- **Cross-type Relation Extraction:** Next, we studied cross-type relation extraction problem. We propose two multi-task transfer learning methods. The experiments on ACE04 data showed that multi-task models could improve the performance.
- **Extraction of Relation Descriptors:** Finally we studied relation extraction under a new setting: The relation types are defined at a general level but more specific relation descriptors are desired. Based on the special properties of this new task, we found that standard linear-chain CRF models have some potential limitations for this task. We subsequently proposed some modifications to linear-chain CRF in order to suit our task better. We annotated two data sets to evaluate our methods. The experiments showed that by restricting the space of possible label sequences and introducing certain long-range features, the performance of the modified CRF model can perform significantly better than standard linear-chain CRF.

3 NUS Technical Report: Relation extraction in resource-poor domains

3.1 Research Team

The research work in this chapter is performed mainly in NUS. The NUS research team consists of Assoc. Prof. Wee Sun Lee (PI), PhD students Nan Ye and Viet Cuong Nguyen. The DSO team consists of Dr. Hai Leong Chieu and Dr. Kian Ming A. Chai.

3.2 Introduction

The task of information extraction (IE) is concerned with converting unstructured information sources (such as news articles) to structured information representation which has simple semantics and is easily manipulable (such as database records). The widespread use of computers and the Internet has made large amount of unstructured information sources easily available, and has spurred interests in various specialized IE tasks, such as tracking disease outbreak [13], extracting protein-protein interaction from papers [4], extracting information from merchant web sites for comparison shopping [15], and semantic role labeling [6]. Examples of IE tasks include Named Entity Recognition (NER), Relation Extraction (RE), and Event Detection. NER involves recognizing named entities (NEs) such as persons, organizations, locations. RE involves extracting some relationships, which is usually prespecified, between NEs. Example relations include the organization-affiliation relation – whether a person works for an organization, and familial relation – whether two people are family members. The goal of IE is to discover domain-independent technology for various extraction tasks.

In this project, we consider extraction of binary relations in sentences from plain texts in resource-poor domains, and limit us to building relation extraction systems without using parsers. We assume there is a set of example sentences with the true relations marked out, and use machine learning techniques to train an extraction system. In addition, we shall only consider RE at the sentence level, in consideration that the dataset we used, that is, the ACE (Automatic Content Extraction) 2005 corpus, only tags relation instances with explicit clues within a sentence⁹.

Previous works on using machine learning approach for RE mainly treat RE as a classification problem. The NEs are assumed to be marked out, then a classifier is trained to classify the relation between any two NEs, using various context information extracted from texts – such as the entity types of the NEs, the words contained in or surrounding the NEs. For example, consider the sentence *Peter is working for IBM.*, which contains an instance of the organization-affiliation relation. In this case, it is assumed that *Peter* and *IBM* have been labeled as PERSON and ORGANIZATION respectively. A useful feature in this case is the phrase *working for*, which indicates the organization-affiliation relation between *Peter* and *IBM*.

We explore models which can incorporate structures in the data in this project.

In Section 3.3, we first consider a simplified case of RE in which only sentences containing at most one instance of any relation under consideration. We first describe an approach based

⁹See <http://projects ldc.upenn.edu/ace/docs/English-Relations-Guidelines.v5.8.3.pdf>

on Conditional Random Fields (CRFs) [19]. Using the baseline techniques, we get an F-score of around 39% on a subset of sentences containing at most one relation. If we use hand-labeled entity-type tags instead of machine tagged entity-types, we are able to obtain an F-score of around 54%. This indicates that one potential area for improvement is to improve the automatic entity-type tagger. We also found that a common source of error is to tag sentences containing no relation with some relation tags. Assuming a perfect classifier that is able to classify whether a sentence has a relation (and its type), the F-score can be improved to 64%.

We then consider two approaches for relation extraction. One approach, described in Section 3.4, is designed to capture the structure of relations in a sentence to jointly extract all the relations in one sentence. This approach is related to [18], which makes use of the matrix tree theorem to perform dependency tree parsing. Another approach, described in Section 3.5, uses CRF with high-order semi-Markov features to capture long-range dependencies between the two relation arguments. The algorithms extend ideas in first-order semi-Markov CRF [29] and high-order CRFs [35]. In Section 3.6, we demonstrate the empirical performance of the two approaches on the ACE 2005 dataset [34], and demonstrates that combining the outputs of different systems can be very effective: The precisions and recalls of the combined model are at least comparable to the individual systems, and the F1 scores are at least 4% better. Section 3.7 concludes the report by discussing important factors and difficulties in building an RE system.

3.3 A Simplified Case

We consider extracting relations from sentences containing at most one relation in this section. We first show how to convert RE to a sequence labeling problem in Section 3.3.1, then we proceed to describe the conditional random fields (CRFs), a model which has been successfully applied for various sequence labeling problems in Section 3.3.2. Section 3.3.3 describes a basic CRF together with variants for RE. Section 3.3.4 describes the dataset used for evaluation. Section 3.3.5 contains experimental results with discussions.

3.3.1 Relation Extraction as Sequence Labeling

Consider extracting *is-Affiliated-with-Organization* relation instances from texts. An instance of this relation is contained in the following sentence: *Senator Christopher Dodd of Connecticut made the announcement today that he would not be the 10th candidate for the nomination.*, where *Christopher Dodd* is affiliated with *Connecticut*.

In this case, we can label *Christopher* and *Dodd* with *Arg1*, *Connecticut* with *Arg2* and other words with *O*. If there is at most one instance of the relation under consideration, then such labeling can always be done. Note that a trained tagger sometimes may tag several phrases as *Arg1* or *Arg2*. We take all *Arg1-Arg2* pairs as the predicted relation instances.

A tagger is first trained for each relation type. Given a sentence, their predicted relations are merged as the output.

3.3.2 Conditional Random Fields

A conditional random field induces a conditional probability distribution of label sequences given an observation sequence, as described below. Assume a fixed set of possible observations, and a fixed set of possible labels. Let \mathbf{x} and \mathbf{y} denote an observation sequence and a label sequence respectively. Let $|\cdot|$ be the function which maps a sequence to its length. A set of features f_1, \dots, f_m , which are real-valued functions of the form $f_i(\mathbf{x}, \mathbf{y}, t)$ (defined only when $|\mathbf{x}| = |\mathbf{y}|$, and $1 \leq t \leq |\mathbf{x}|$), are first chosen. Each f_i is associated with a real-valued weight λ_i which is to be determined. Let $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)$, and $W_{\vec{\lambda}}(\mathbf{x}, \mathbf{y}) = \exp(\sum_i \sum_t \lambda_i f_i(\mathbf{x}, \mathbf{y}, t))$, then using $W_{\vec{\lambda}}$ we can induce the following probability distribution: $P_{\vec{\lambda}}(\mathbf{y}|\mathbf{x}) = \frac{W_{\vec{\lambda}}(\mathbf{x}, \mathbf{y})}{\sum_{\mathbf{y}'} W_{\vec{\lambda}}(\mathbf{x}, \mathbf{y}')}$. $P_{\vec{\lambda}}(\mathbf{y}|\mathbf{x})$ is considered to be 0 whenever $|\mathbf{y}| \neq |\mathbf{x}|$.

In practice the weight vector $\vec{\lambda}$ is often obtained by maximizing the regularized log-likelihood of a training set $\{(\mathbf{x}_1, \mathbf{y}_1), \dots, (\mathbf{x}_n, \mathbf{y}_n)\}$, that is, the weight vector is chosen to be

$$\vec{\lambda}^* = \arg \max_{\vec{\lambda}} [\prod_{i=1}^n \log P_{\vec{\lambda}}(\mathbf{y}_i|\mathbf{x}_i) - \sum_{i=1}^m \frac{\lambda_i^2}{2\sigma^2}]$$

, where σ controls the degree of regularization. Maximizing the likelihood favors models which fit well with the training data, while regularization discourages models with exceptionally large λ_i 's.

Given a model $P_{\vec{\lambda}^*}$, one common way to predict the label sequence for any given observation sequence \mathbf{x} is to predict the most likely label sequence, that is, $\arg \max_{\mathbf{y}} P_{\vec{\lambda}^*}(\mathbf{y}|\mathbf{x})$. Another commonly used prediction method is to predict the label sequence $(y_1, \dots, y_{|\mathbf{x}|})$ such that y_i is the most likely label at the i th position. Generally the first method performs better than the second one, and we use the first method for all experiments done.

Learning and inference in CRF is generally intractable, but there are some interesting tractable cases. In particular, one such case is the class of linear-chain CRFs, which are CRFs using only features depending on at most two consecutive labels. There are various efficient algorithms for learning and inference in linear-chain CRFs. In our experiments, inferring the marginal probabilities are done using an analogue of the forward-backward algorithm used in HMM [27], and inferring the most likely sequence is done using an analogue of the Viterbi algorithm in HMM [27] as well. Learning is done by maximizing the regularized log-likelihood using the gradient-based L-BFGS method [30], with the gradient computed using the marginal probabilities. Though linear-chain CRFs only capture dependencies for at most two consecutive labels, they have been found to be expressive enough for problems such as Parts of Speech (POS) tagging and Phrase Chunking. In addition, they have been used in some relation extraction tasks with success as well [3, 11]. The tractability and expressiveness of linear-chain CRFs make them suitable models for initial investigation.

3.3.3 Models

We assume the data consists of sentences with the head words of the arguments labeled using *Arg1*, *Arg2* tags, and the other words labeled using *O*. We start with a basic CRF model with the following features. The zeroth order features (features which depend on only one label) are listed below:

- Current, previous, next word and its capitalization pattern;
- Words at most k positions before current word;
- Words at most k positions after current word;
- Combinations of word and capitalization patterns:
 - n -grams in words.

The first-order features (features which depend on two consecutive labels) are listed below:

- Transition features without any observations;
- Transition features with first or second word or its capitalization pattern;
- Capitalization patterns for the two words.

When additional tags (such as Named Entity (NE) tags, POS tags) are available, we add the current, next, previous tag and their combinations as additional zeroth order features.

Various models can be derived from the above basic model by adding new information, and/or changing the way the dataset is used. The modifications that have been evaluated can be broadly classified into three categories: handling imbalance of positive and negative examples, adding additional tags, and adding long-range information to the labels. We list the modifications below.

Handling Dataset Imbalance

- **BALANCE:** Generally there are far more negative examples (sentences which containing no relation instance) than positive examples. We remove a random subset of negative examples to make the number of negative examples the same as the number of positive examples.

Adding Additional Tags

- **NE:** Add golden (human labeled) NE tags.
- **POS:** Add POS tags generated by the Stanford POS tagger ¹⁰.

Embedding Long-range Information into Labels

- **RELABEL-NE:** Relabel the sentences to remember the NE tags for the arguments by appending the NE tags to the current labels. For example, if the relation tags are $O, Arg1, O, Arg2, O$, and the NE tags are O, PER, O, LOC, O , then the new relation tags are $O, Arg1-PER, O-PER, Arg2-LOC, O$.
- **RELABEL-ARGS:** Relabel the sentences to remember the arguments that has appeared by appending the list of arguments appeared to current labels. For example, $O, Arg1, O, Arg2, O$ is converted to $O, Arg1-1, O-1, Arg2-12, O-12$.

¹⁰<http://nlp.stanford.edu/software/tagger.shtml>

	+	-	Org-Aff	Gen-Aff	Part-Whole	Art	Phys	Per-Soc	All
Train	3418	12360	1764	640	924	640	1387	766	6121
	2560		978	395	486	351	743	427	3380
Test	1472	5446	770	317	377	232	581	322	2599
	1117		441	184	204	131	294	192	1446

Table 19: There are two rows for both train and test data, with the first for original dataset and the second for the filtered dataset. The -/+ column is the number of negative/positive sentences in the dataset (a sentence is positive if it contains a relation instance for any of the six relations). Each entry in the next six columns shows the number of positive sentences for the corresponding relation, where the bracketed number is the number of positive sentences containing at least two instances for that relation. The last column shows the total number of relation instances.

- RELABEL-B: If the boundary of the noun phrases containing the arguments are given, then we can again append the information that a boundary has been observed to the tags of words within the noun phrases. Note that one argument can be contained in the other, thus for each word in the noun phrases, we append the ordered list of boundaries that has been observed.
- TYING: After relabeling the dataset, O , $Arg1$, $Arg2$ have a few variants. These variants should be considered to be the same in features which are intended to capture the same kind of dependencies. For example, if we are interested to learn whether the word *be* should be a non-argument, then

$$f(\mathbf{x}, \mathbf{y}, t) = \begin{cases} 1, & x_t=be \text{ and } y_t=O; \\ 0, & \text{otherwise.} \end{cases} \text{ and } g(\mathbf{x}, \mathbf{y}, t) = \begin{cases} 1, & x_t=be \text{ and } y_t=O-1; \\ 0, & \text{otherwise.} \end{cases}$$

should be treated equally and should have the same weight. Setting these features to share the same weight is called parameter tying. In our experiments, we tie the parameters for non-tag zeroth order features which uses variants of O .

3.3.4 Datasets

We evaluated the models in Section 3.3.3 on the ACE 2005 [34] corpus. The articles are drawn from 6 different sources: bc (Broadcast Conversation), bn (Broadcast News), cts (Conversation Telephone Speech), nw (Newswire), un (Usenet), wl (Weblog). There are six labeled relations: PART-WHOLE, PHYS (located or near), ORG-AFF, (organizational affiliation), GEN-AFF (general affiliation), PER-SOC (personal-social relation), ART (artifacts).

We construct the training and test data by putting the first 70% of the sentences from each source into the training data and putting the remaining 30% into the test data. We then remove sentences which contain at least two instances of the same relation from both the training and test data. Table 19 shows statistics for the number of relation instances in the original dataset and the filtered dataset.

	All	Art	Gen-Aff	Org-Aff	Part-Whole	Per-Soc	Phys
BASE	24.28	12.33	12.56	35.52	17.36	41.51	6.27
BEST	53.92	49.12	45.60	64.46	55.12	54.95	44.79

Table 20: Best results obtained by adding BALANCE, POS, NE, RELABEL-NE to the baseline.

	All	Art	Gen-Aff	Org-Aff	Part-Whole	Per-Soc	Phys
BEST	53.92	49.12	45.60	64.46	55.12	54.95	44.79
-BALANCE	48.82	37.93	36.00	62.66	42.76	56.03	35.59

Table 21: Effect of not balancing the negative examples.

3.3.5 Results and Discussions

We performed experiments using various combinations of the modifications described above, and determine a best combination, then we study the effect of each modification by either removing it from the best model (if it is used in the best model) or adding it to the best model (if it is not used in the best model). We assumed that human-labeled entity type labels are available as the baseline for analysis.

We measure the the F1 score, which is the harmonic mean of precision (number of correct predicted relations over number of predicted relations) and recall (number of correct predicted relations over number of true relations), that is, $\frac{2 * precision * recall}{precision + recall}$.

Best Model BASE is the basic model in Section 3.3.3. The best model is the model obtained by adding the combination of BALANCE, POS, NE, and RELABEL-NE to BASE. The F1 scores for all relation types and for each relation relation type are shown in the table above. The performance improvement in terms of the F1 for all relation types is about 30%.

Balancing Negative Examples It can be seen that balancing the dataset so that the number of sentences without relations is the same as the number of sentences with relations is essential. Note that balancing greatly reduces the dataset size and thus make learning much more efficient.

Entity Type and Part-of-speech Entity-type tags turned out to have a large impact on performance. But POS tags have a minor effect.

	All	Art	Gen-Aff	Org-Aff	Part-Whole	Per-Soc	Phys
BEST	53.92	49.12	45.60	64.46	55.12	54.95	44.79
-NE	30.62	21.19	23.72	37.79	30.66	38.63	22.22
-POS	53.35	48.46	46.57	63.85	55.30	52.69	44.16

Table 22: Effects of removing entity tags and part-of-speech tags.

	All	Art	Gen-Aff	Org-Aff	Part-Whole	Per-Soc	Phys
BEST	53.92	49.12	45.60	64.46	55.12	54.95	44.79
-RELABEL-NE	41.61	36.00	34.84	53.70	36.48	49.18	30.69
+RELABEL-ARG	52.64	50.95	44.86	67.17	52.66	47.19	44.15
+RELABEL-B	52.35	41.24	45.07	64.56	53.58	52.39	42.67
+TYING	52.82	45.71	46.04	62.71	55.58	52.61	44.22

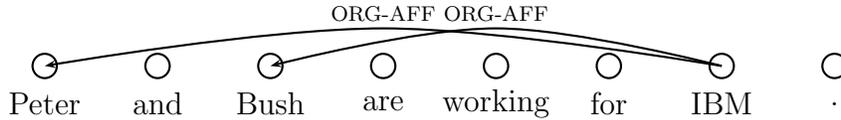
Table 23: Effects of storing information in the tags.

	All	Art	Gen-Aff	Org-Aff	Part-Whole	Per-Soc	Phys
BEST	53.92	49.12	45.60	64.46	55.12	54.95	44.79
+TrainedNE	38.98	29.60	35.47	46.60	35.20	46.31	31.97
+CorrectedNE	39.44	32.26	36.25	47.26	36.08	46.04	31.07
+CLASSIFIER	63.77	54.90	59.86	70.94	63.44	67.48	56.13

Table 24: Performance of using trained instead of human-labeled and effect of perfect classifier.

Storing Long Distance Information in the Tags By storing the entity type in the tag of the first argument, the second argument can be matched using the correct type. This has a large effect on performance as seen by the effect of removing it from the best model. On the other hand, adding even more information about the order of the argument and noun phrase boundaries appears to cause overfitting. However, we also found when RELABEL-NE is not used, RELABEL-ARGS and RELABEL-B improves performance, so these labels do contain useful information. Parameter typing does not seem to affect the performance much in this case, but it should be noted that it is still an important consideration in model design, and is essential if there are many important features which are semantically the same.

Effects of Human-Labeled Entity Tags and Classifier Using a trained entity tagger in place of human labels degraded performance significantly. This indicates that improving the entity tagger is one way to improve performance. One of the difficulties in tagging entities is deciding the tags of pronouns. This may require the use of co-reference. To test the importance of this, we use human labeled tags for pronouns and retain the machine tags for the other entities (CorrectedNE). The relative performance of TrainedNE and CorrectedNE suggests that using coreference resolution to build a better NE tagger may not be very helpful. We also investigated using human provided information on the presence (and type) of relation in the sentence in the form of a classifier. The performance of CLASSIFIER is about 10% better than BEST, and this suggests a possible avenue to gain performance improvement.



3.4 Exploiting Tree Structures

3.4.1 Motivation

Relation extraction from a sentence is closely related to dependency parsing for a sentence: NLP dependencies can be viewed as binary relations, and thus dependency parsing can be thought of as a special type of relation extraction task; In addition, it often happens that the set of relations in a sentence is one-to-one or one-to-many, which is a characteristic that motivated the concept of dependency tree. This motivates us to formulate the problem of relation extraction as learning and prediction of tree structures from a partial set of observed edges, as described below.

A sentence and the relations present in it can be visualized as a directed graph with labeled edges: Each NE and each other word is a node, and there is an edge from the first argument to the second argument for each relation instance. The graph for the simple sentence *Peter and Bush are working for IBM.* is shown below.

The above graph can be viewed as a subgraph in the complete weighted directed graph on the same set of nodes, with a labeled weighted edge for each pair of nodes and each possible relation, where the weight of an edge indicates the strength of the evidence supporting the edge label. Extracting relations can then be considered as searching for an *optimal edge weight assignment scheme* and searching for an *optimal subgraph* in the complete weighted directed graph.

3.4.2 Model

To reduce the complexity of searching for an optimal subgraph, we make the assumption that there are hidden binary relations between the nodes, which form a directed spanning tree together with the observed relation edges. This assumption only allows one-to-many relations to be considered, but is generally valid at the sentence level as mentioned above. In Figure 3.4.1, we orient the edge so that it points to the employee, then this encodes the assumption that one person is affiliated with one organization only, which is not always but generally true.

Edge weight assignment is done using a set of weighted features of the form $f(s, i, j, l)$, where s denotes a sentence, and (i, j, l) is a labeled edge, with i, j, l being the index of the head, the index of the tail, and the label respectively. An example feature is

$$f(s, i, j, l) = \begin{cases} 1, & s(i) = Peter \wedge s(j) = IBM \wedge l = ORG - AFF; \\ 0, & otherwise. \end{cases}$$

where $s(i)$ denotes the i -th node of s . Let f_1, \dots, f_m be the set of features chosen, and $\lambda_1, \dots, \lambda_m$ the corresponding weights. Let $\vec{F}(s, i, j, l) = (f_1(s, i, j, l), \dots, f_m(s, i, j, l))$, and $\vec{\lambda} = (\lambda_1, \dots, \lambda_m)$, then the weight $\omega(e)$ of a labeled edge $e = (i, j, l)$ is defined to be $\exp(\vec{\lambda} \cdot \vec{F}(s, i, j, l))$. We shall use G_s to denote the complete weighted directed graph constructed for a sentence s . A set of relation

on \mathbf{s} is a set of labeled edges on $G_{\mathbf{s}}$. The set of relations in \mathbf{s} is denoted by $R_{\mathbf{s}}$, which is a subset of edges of $G_{\mathbf{s}}$.

Optimal feature weight vector is chosen by maximizing a probability parameterized by $\vec{\lambda}$, as described below. Define the weight $\omega(T)$ of a spanning tree T in $G_{\mathbf{s}}$ as $\prod_{e \in T} \omega(e)$. Let $\mathcal{T}(G_{\mathbf{s}})$ denote the set of spanning trees on $G_{\mathbf{s}}$, and define the probability of $T \in \mathcal{T}(G_{\mathbf{s}})$ as

$$P_{\vec{\lambda}}(T|\mathbf{s}) = \frac{\omega(T)}{\sum_{T' \in \mathcal{T}(G_{\mathbf{s}})} \omega(T')}$$

For a set of relation R , we say a spanning tree of $G_{\mathbf{s}}$ is *consistent* with R if it contains all edges in R but with all other edges being a non-relation edge. Let $\mathcal{T}(G_{\mathbf{s}}, R)$ be the set of spanning trees of $G_{\mathbf{s}}$ consistent with R . Define the probability

$$P_{\vec{\lambda}}(R|\mathbf{s}) = \frac{\sum_{T \in \mathcal{T}(G_{\mathbf{s}}, R)} \omega(T)}{\sum_{T \in \mathcal{T}(G_{\mathbf{s}})} \omega(T)}$$

If the training examples are $\mathcal{S} = \{(\mathbf{s}_1, R_{\mathbf{s}_1}), \dots, (\mathbf{s}_n, R_{\mathbf{s}_n})\}$, then the optimal weight vector $\vec{\lambda}^*$ is chosen to be

$$\vec{\lambda}^* = \arg \max_{\vec{\lambda}} \mathcal{L}_{\vec{\lambda}}(\mathcal{S})$$

where $\mathcal{L}_{\vec{\lambda}}(\mathcal{S}) = \sum_{(R_{\mathbf{s}}, \mathbf{s}) \in \mathcal{S}} \ln P_{\vec{\lambda}}(R_{\mathbf{s}}|\mathbf{s}) - \sum_{i=1}^m \frac{\lambda_i^2}{2\sigma^2}$ is the regularized log-likelihood of \mathcal{S} . The regularization term $\sum_{i=1}^m \frac{\lambda_i^2}{2\sigma^2}$ is used to prevent overfitting, with σ being a parameter chosen to control the tradeoff between model complexity (as measured by the regularization term) and how well the model fits to training examples.

Efficient training and prediction algorithms are given in Appendix 1.

3.4.3 Variants

In the above discussion, we only consider directed graphs. The same technique can be used to model tree structures over undirected graphs, with some minor changes. We leave out the technical details here.

Since we are mainly interested to learn the relations between the named entities, we can also choose only the NEs rather than all the phrases as the nodes of the graphs.

Though the tree model may capture the structure of the relations in the graphs, this is only an approximation as not all relations form a tree in practice. In contrast, previous classification approach does not assume any structure in the relations in the graphs. We combine both approaches to form a hybrid model as follows: A tree model and a classifier are trained separately, but for decoding, a tree is first obtained using the tree model, then high-confidence edges predicted using the classifier are also included.

3.5 Exploiting Long-range Dependencies

3.5.1 Motivation

For relation extraction from plain text alone, especially in a new domain without a good parser, we may need to do both segmentation and argument identification jointly. Thus, sequence labeling models such as linear Conditional Random Field [19] (CRF) or semi-Markov Conditional Random Field [29] would be useful for the task. However, both linear CRF and semi-Markov CRF have a major drawback: they cannot capture the long-range dependencies or the label patterns of the input sequence. Consider again the example in previous section: *Peter is working for IBM*. The label pattern *Arg1+ O+ Arg2+* will be learned from this example, and this pattern says that Arg1's should be followed by some O's, then some Arg2's. This can be useful to avoid label sequences such as *Arg1+ O+ Arg1+*. Such information cannot be captured in linear CRFs and semi-Markov CRFs.

In this section, we extend the semi-Markov Conditional Random Field to include high-order semi-Markov features which depend on two or more previous segment labels. We use techniques similar to the algorithms in [35] to train and predict the labels. As in [35], we also make the *pattern sparsity assumption*, that is, the number of observed segment label patterns of length k is much smaller than $|\mathcal{Y}|^k$, where \mathcal{Y} is the set of all labels in the model.

Relation extraction is casted as sequence labeling in a slightly different way as compared with the method in Section 3.3. If a word in a sentence is the first argument of a relation, we tag it as *Arg1*. If it is the second argument, we tag it as *Arg2*. If the word is the first argument of a relation and it is also the second argument of another relation of the same type, we tag it as *Arg1Arg2*. Otherwise, we tag it as *O*, which means the word is not part of any relation.

After the arguments are identified, we construct all *Arg1-Arg2* pairs, and use a trained classifier to predict whether each of them is a true relation. The subset of relations predicted to be true is output as the relations. This classification step is not present in the approach in Section 3.3.

The classifier is trained using the training data by splitting it into two equal halves, then one tagger is trained for each partition, and then used to generate all candidate relations (triplets of *Arg1-Arg2-type*) on the other partition as in Section 3.3. These candidate relations are tagged as true or false using the true relation instances in the training data. This set of tagged candidate relations is the training set for the classifier.

3.5.2 Model

Let $\mathbf{s} = (s_1, \dots, s_p)$ be a segmentation of an input sequence \mathbf{x} , where each $s_j = (u_j, v_j, y_j)$ is a segment, consisting of start position u_j , end position v_j , and label $y_j \in \mathcal{Y}$. Assume we have m features f_1, \dots, f_m , and each feature f_i is associated with a segment label pattern \mathbf{z}^i such that

$$f_i(\mathbf{x}, \mathbf{s}, t) = \begin{cases} g_i(\mathbf{x}, u_t, v_t) & \text{if } y_{t-|\mathbf{z}^i|+1} \dots y_t = \mathbf{z}^i \\ 0 & \text{otherwise} \end{cases}$$

	+	-	Org-Aff	Gen-Aff	Part-Whole	Art	Phys	Per-Soc	All
Train	3418	12360	1764	640	924	640	1387	766	6121
Directed	3123	3418	1535	544	780	487	1053	678	5077
Undirected	3400	3418	1743	634	909	630	1351	752	6019

Table 25: First row shows statistics for the original training data. Second row and third row show statistics after filtering non-forest examples, when directions are considered and ignored, respectively.

A high-order semi-CRF is defined as

$$P(\mathbf{s}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \exp\left(\sum_{i=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_i f_i(\mathbf{x}, \mathbf{s}, t)\right)$$

where $Z_{\mathbf{x}} = \sum_{\mathbf{s}} \exp\left(\sum_{i=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_i f_i(\mathbf{x}, \mathbf{s}, t)\right)$.

Efficient training and prediction algorithms are given in Appendix 2.

3.6 Experiments

We study the effects of modeling the tree structures and high-order semi-Markov features on the ACE 2005 corpus.

As in Section 3.3, we construct the training and test data by putting the first 70% of the sentences from each source into the training data and putting the remaining 30% into the test data.

3.6.1 Tree Structures

In this section, we compare the effectiveness of the classification approach (using a MaxEnt classifier), the tree model, and the hybrid model, using the same set of features. The comparison demonstrates that modeling the structure for the relations can be useful.

As in Section 3.3, for all three models, we first randomly balance the number of positive and negative examples so as to speed up training and improve performance. For the tree models, we then remove non-forest examples from the training data. The statistics for the processed data is shown in Table 25.

We list the features used. Uni-gram features include: words in the phrase, the phrase itself, POS tags for the words in the phrase, POS tag sequence for the phrase, NE tag of the phrase, previous and next i th phrases ($1 \leq i \leq 3$). Bi-gram features include: combination of the phrases, combination of the NE tags, combination of the POS tags, prepositions between the two phrases, whether there is any entity between the two phrases, and whether there is any entity of the same type as the first/second phrase between the two phrases.

For each model, either true NE tags or predicted NE tags may be used, the directions of relations may be kept or ignored, and the candidate arguments considered may be only the NEs or all phrases. This gives 8 test settings for each model. The performance of the three models under these settings are shown in Table 26.

	Directed	MaxEnt	Tree	Hybrid
TrueNE	NE	60.45/46.98/52.87	53.64/53.91/53.77	60.61/49.90/54.74
	Phrases	59.22/46.94/52.37	55.58/52.14/53.80	61.10/49.79/54.87
TrainedNE	NE	56.08/31.78/40.57	49.50/36.44/41.98	56.87/34.40/42.87
	Phrases	57.06/32.67/41.55	46.14/39.28/42.44	55.11/34.86/42.71
	Undirected			
TrueNE	NE	61.11/51.77/56.05	55.59/57.78/56.66	60.70/54.04/57.18
	Phrases	59.41/51.89/55.39	53.96/59.85/56.75	59.47/54.85/57.07
TrainedNE	NE	55.66/35.51/43.36	50.75/39.18/44.22	55.49/37.48/44.74
	Phrases	53.38/36.05/43.04	46.48/41.38/43.79	53.90/37.67/44.35

Table 26: Precisions, recalls and F1 measures for the three approaches.

	Part-Whole	Phys	Org-Aff	Gen-Aff	Per-Soc	Art	Average
SemiCRF	38.51	33.40	60.78	31.35	53.46	40.07	42.92
High-order SemiCRF	42.76	42.00	64.08	35.38	57.29	48.79	48.38

Table 27: Argument detection F1 scores for semiCRF tagger and high-order semiCRF tagger

It can be seen that for all cases, in terms of F1, the hybrid model is the best, followed by the tree model, then followed by classifier model. The classifier model has higher precision but lower recall than the tree model. The hybrid model’s precision and recall are in between those of the classifier model and the tree model.

3.6.2 High-order Semi-Markov Features

For both first-order and high-order semiCRFs, we use the following set of zeroth order features: current, previous, next word and its capitalization pattern; words at most k positions before and after the current word; combinations of word and capitalization patterns; n-grams in words; current, previous, next NE tags and their combinations; current, previous, next POS tags and their combinations. We also use the following set of first-order features: transition features without any observation, transition features with first or second words or its capitalization pattern, capitalization patterns for the two words. Additionally, we include the second-order transition features without observation to the high-order semi-Markov CRF tagger. The F1 scores of the semi-Markov CRF tagger and high-order semi-Markov CRF tagger on the test set are shown in Table 27.

Using a slightly different train-test partition, we demonstrated that high-order semi-Markov features can capture additional long-range dependencies as compared with first-order semi-Markov features. We showed that high-order semiCRF performs better than first-order semiCRF in terms of the F1 scores for argument detection. This is reported in [24], and we reproduce the scores below.

The table above shows that high-order semi-CRF performs better than semi-CRF for all relation types, thus high-order transition features can capture some additional long-range dependencies.

We also evaluated the semiCRF’s performance in terms of relations identified. As described in

		CRF	SemiCRF
TrueNE	Before	47.43/47.94/47.68	33.85/49.40/40.18
	After	65.49/40.75/50.24	63.59/39.25/48.54
TrainedNE	Before	42.02/36.00/38.78	30.59/37.27/33.61
	After	61.13/29.61/39.90	62.36/28.84/39.44

Table 28: Precisions, recalls, and F1s for the linear-chain CRF and the high-order semiCRF

Section 3.5, we need to build a classifier that predicts whether a candidate relation output by the semiCRF is a true relation. We build a MaxEnt classifier \mathcal{C} using the features described below. For a candidate relation (a_1, a_2, t) , where a_1, a_2 and t are first argument, second argument and the relation type of the candidate instance, we construct the following features: $a_1 + t, a_2 + t, a_1 + a_2 + t, n_1 + t, n_2 + t, n_1 + n_2 + t$ (n_1 and n_2 are the entity types of a_1 and a_2 respectively), a feature indicating whether the candidate relation is predicted by both \mathcal{T} and \mathcal{F} , $w + t$ for all words w between the arguments, probability of the candidate relation as measured by the tree model, the entity type distribution for the arguments according to the NE tagger trained on the training data.

For prediction on a set of candidate relations, \mathcal{C} uses a method which is different from the usual classification rule which outputs the most likely labels. Instead, \mathcal{C} first compute for each candidate relation the probability that it is positive. Then choose n such that classifying the top n candidates results in the largest expected F1 (see [7] for details), and classify the top n candidates as positive. We simply remark that our experiments showed that this is better than the usual classification rule here.

Note that this classification step can be applied to the linear chain CRF model trained as described in Section 3.3 as well. Table 28 shows the scores of both linear chain CRF and high-order semiCRF using true NE tags or predicted NE tags, before applying the classification step and after the classification step.

The classification step improves the precisions significantly, but lowers the recalls. Note that the linear chain CRF has relatively good performance because it remembers the NE tags of the first argument seen in the labels, while high-order semiCRF only remembers the transition patterns.

3.6.3 A Combined Model

For the method of first generating candidate relations then classifying them, we can certainly consider candidate relations generated using models other than CRFs. We construct a combined system by using as candidate relations all the relations output by the classifier models, the tree models, the hybrid models (considering relation directions, but either using NEs or all phrases as possible arguments), and the candidate relations output by the linear-chain CRF and the semiCRF. That is, we use 8 different sets of outputs to form a candidate relation set. We then apply the classification step as in the above experiments.

Table 29 shows the precision, recall and F1 of the combined system. The precision, recall and F1 score of the combined model are better than or at least comparable to all the above models.

We also remark that our evaluations on other combinations of the outputs show that classification on the outputs of the classifier models, the tree models, or the hybrid models alone generally

	TrueNE	TrainedNE
Combined	64.64/53.87/58.76	61.09/38.51/47.24

Table 29: Precisions, recalls and F1 scores for the combined model.

do not lead to better performance. In addition, using all of the 8 different sets of outputs leads to better performance for both the cases of using true NE tags and predicted NE tags.

3.7 Conclusion

We highlight a few important factors and difficulties for building an effective relation extraction system for plain texts.

A useful observation is that our results showed that combining the outputs of several different systems can be useful. Thus it can be useful to build an ensemble of systems which capture different information, and then combine the outputs of these ensembles.

From the perspective of breaking RE as a pipeline of tasks which are intuitively simpler, our analysis shows that we will be able to improve performance if we can classify whether a relation (and the type) is present in a sentence. However, our experience suggests that detecting whether a sentence contains a relation instance is not necessarily easier than extracting the relation instances together with their types.

Regarding the information useful for RE, typical systems often incorporate information from NE recognition, POS tagging, phrase chunking, and parsing. For relations which can be distinguished by the NE types of the arguments (for example, the organization-affiliation relation is between people and organizations, and the personal-social relation must be between people), NE tags are certainly very useful. A typical relation extraction system is often constructed as a pipeline in which the first step is NE recognition, then followed by discovering relations between the NEs. Our results have shown that NE tags is one of the most important features for relation extraction. POS tags are also useful, while parse trees and phrase tags are harder to use and generally do not yield much improvements. In addition, parse trees can contain a lot of errors for informal texts, such as those in bc, cts. Exploiting shallow syntactic information is easier and may often be sufficient.

Relations can also be hard to extract because of the complex structures and rich semantics of natural languages. An example of difficulties with complex structures is a sentence like “A, who ..., is the father of B”, which can make the relation “A is the father of B” difficult to extract by making the *who* clause complicated. An example of difficulties with rich semantics is the phrase structure “A of B”, which can be used to express various relations, as in “a packed room of 200 intelligent, practical, driven idealists” (PHYS), “a student of Plato” (PER-SOC), “president of IBM” (ORG-AFF). While features remembering the exact words can be used to solve these cases once they are seen, it appears that there are no automatically constructed simple features can lead to good performance on the underlying general cases.

While the basic units of information used can be the same (such words, their POS tags, NE tags), modeling additional structures between the relations can be useful. For example, the tree model allows the relations to compete with each other, which can lead to more accurate feature

weights, and this may be the reason why it has better performance than the classifier approach. Using high-order semi-Markov features allows us to capture some useful long-range dependencies. However, so far we can only use semiCRF to identify the relation arguments, and extraction of the relations requires an additional step of pairing up the arguments using a classifier. The later step does not appear to be easy.

Certainly, RE at the document level, while more difficult, is more interesting and useful in practice. Additional document level analysis, such as coreference resolution and discourse analysis, are expected in this case. We have not studied this yet.

Appendix A Algorithms for the Tree Model

This section presents an efficient training algorithm for the tree model (that is, determining the optimal feature weights $\vec{\lambda}^*$) and an efficient algorithm for making predictions on given sentences using a given model.

A.1 Training

We can make use of standard gradient descent technique, like the L-BFGS algorithm [21], to solve the optimization problem $\arg \max_{\vec{\lambda}} \mathcal{L}_{\vec{\lambda}}(\mathbf{s}_1, \dots, \mathbf{s}_n)$. This requires efficient computation of the value and gradient of $\mathcal{L}_{\vec{\lambda}}$ as a function of λ . Both can be done efficiently by exploiting the matrix tree theorem, which expresses the total weight of spanning trees of a directed graph as the determinant a simple matrix. We shall first introduce the matrix tree theorem, and derive a few consequences of it.

A.1.1 Matrix Tree Theorem and Consequences

Throughout this section, $G = (V, E, \omega)$ denotes a weighted directed graph, where $V = \{1, 2, \dots, n\}$, $E = \{(i, j, l) : 1 \leq i, j \leq n, 1 \leq l \leq L\}$ (L is the number of distinct labels), and $\omega : E \rightarrow \mathbb{R}$. We only consider graphs with the following restrictions: (a) ω is log-linear, that is, there exists some $\vec{\lambda} \in \mathbb{R}^m$ and a function $\vec{F} : E \rightarrow \mathbb{R}^m$ such that for all i, j, l , $\omega(i, j, l) = \exp(\vec{\lambda} \cdot \vec{F}(i, j, l))$; (b) $\omega(i, i, l) = 0$ for all i, l ; (c) The labels $1, \dots, L$ are partitioned into two categories: relation labels and non-relation labels.

We shall use w_{ijl} to denote $\omega(i, j, l)$, and w_{ij} to denote $\sum_{l=1}^L w_{ijl}$. The *Laplacian* $L(G)$ of G is the matrix such that its (i, j) -th entry is $L(G)(i, j) = \begin{cases} -w_{ij}, & i \neq j; \\ \sum_{k=1}^n w_{kj}, & i = j. \end{cases}$ The determinant, inverse and transpose of a matrix A is denoted by $|A|$, A^{-1} and A^T respectively. The (i, j) -minor of A , that is, the matrix obtained by deleting the i -th row and the j -th column of A , is generally denoted by A_{ij} .

P1.Sum of all spanning trees

Theorem 1. (*Matrix tree theorem*) ([32],p140) *The total weight of all spanning trees of G with node m as the root node is $(-1)^{k+m} |L^{(k,m)}(G)|$, where k is any of $1, \dots, |V|$, and $L^{(k,m)}(G)$ is the (k, m) -minor of $L(G)$.*

As a simple corollary of the matrix tree theorem, we have the following.

Corollary 2. *Suppose $1 \leq i \leq |V|$. Let $L^{(i)}(G)$ be the matrix obtained by replacing the i -th row of $L(G)$ with all 1's, then $|L^{(i)}(G)|$ is the total weights of all spanning trees of G .*

Since the determinant of a $k \times k$ matrix can be computed in $O(k^3)$ time using Gaussian elimination, the above theorem leads to an $O(|V|^3)$ time algorithm to find the total weight of all spanning trees.

P2. Sum of all spanning trees consistent with given relation edges

We give two algorithms to compute $\sum_{T \in \mathcal{T}(G,R)} \omega(T)$ for a set of relation edges R that form a directed forest. An edge (i, j, l) of G is said to be *consistent* with R if $(i, j, l) \in R$, or l is a non-relation label and there is no l' such that $(i, j, l') \in R$. Let $G|R$ denote the subgraph of G containing all and only edges consistent with R . $G|R$ is called the *consistency graph* for (G, R) .

Theorem 3. $\sum_{T \in \mathcal{T}(G,R)} \omega(T) = \sum_{R' \subseteq R} (-1)^{|R'|} \sum_{T \in \mathcal{T}(G|R-R')} \omega(T)$.

Proof. Note that $\sum_{T \in \mathcal{T}(G|R)} \omega(T) - \sum_{T \in \mathcal{T}(G|R-R')} \omega(T)$ is the total weight of spanning trees containing at least one edge from R' and with all other edges being non-relation edges. Apply the principle of inclusion-exclusion and simplify the expression, then we obtain the above formula. \square

The above theorem leads to an $O(2^{|R|}|V|^3)$ algorithm to compute $\sum_{T \in \mathcal{T}(G,R)} \omega(T)$. Though not a polynomial time algorithm, it is generally efficient enough in practice, as the number of relations in a sentence is usually quite small. There is in fact a simple polynomial time algorithm based on edge contraction, as shown in Algorithm 1.

Algorithm 1 Algorithm for total weight of consistent spanning trees

Input: G, R .

Output: $\sum_{T \in \mathcal{T}(G,R)} \omega(T)$.

- 1: Construct $G|R$;
 - 2: Construct G' with nodes $v_1, \dots, v_{|V|}$ such that $v_i = \{i\}$, and for every edge (i, j, l) in $G|R$, there is a corresponding edge (v_i, v_j, l) with the same weight;
 - 3: $W = 1$;
 - 4: **for** each edge (i, j, l) in R **do**
 - 5: Identify the nodes $u, v \in G'$ such that $i \in u$ and $j \in v$;
 - 6: $W = W \cdot \omega(i, j, l)$;
 - 7: Remove all incoming edges on v ;
 - 8: For each edge (v, t, l') , where $t \neq u$, create an edge (u, t, l') with the same weight as (v, t, l') , and delete (v, t, l') ;
 - 9: **return** $W \cdot \sum_{T \in \mathcal{T}(G')} \omega(T)$.
-

The above idea can be implemented using simple arithmetic operations on matrices, rather than explicitly performing operations on a graph. To be precise, it can be used to find a $|V| \times |V|$ matrix $S = (s_{ij})$ such that $|S| = \sum_{T \in \mathcal{T}(G,R)} \omega(T)$. We shall call S a *tree sum matrix* for (G, R) .

Algorithm 2 shows how S can be constructed. In addition, we construct a $|V| \times |V| \times L$ array C such that $C(i, j, k) = \begin{cases} \omega(i, j, k) & (i, j, k) \text{ is in a spanning tree of } G \text{ consistent with } R; \\ 0 & \text{otherwise;} \end{cases}$, which

we call the *consistency matrix* for (G, R) .

P3. Gradients of $\sum_{T \in \mathcal{T}(G)} \omega(T)$ and $\sum_{T \in \mathcal{T}(G,R)} \omega(T)$ with respect to $\vec{\lambda}$

Algorithm 2 Construction of the consistency matrix and the tree sum matrix

Input: ω, R

Output: S, C as described above.

- 1: For each (i, j, l) , set $C(i, j, l)$ and $A(i, j, l)$ to be $\omega(i, j, l)$ if $(i, j, l) \in G|R$, and 0 otherwise;
- 2: Set $id(i) = i$ for all node i ;
- 3: **for** $(i, j, l) \in R$ **do**
- 4: $contract((i, j, l), C, A)$;
- 5: Let S be the Laplacian of the matrix with its (i, j) th entry being $\sum_l A(i, j, l)$.
- 6: Find the smallest r such that $id(r) = r$;
- 7: Set $S(r, j) = 1$ for each j such that $id(j) = j$;

procedure $contract((i, j, l), C, A)$

- 1: Set $C(i', j, l') = 0$ for all i', l' such that $i' \neq i$ or $l' \neq l$;
 - 2: Set $C(j', i', l') = 0$ for all i', j' such that $id(i') = id(i)$ and $id(j') = id(j)$, and for all l' ;
 - 3: Set $A(i', j, l') = 0$ for all i', l' ;
 - 4: $A(id(i), k, l') = A(id(i), k, l) + A(j, k, l')$ for all l and all $k \neq id(i)$;
 - 5: Set $A(j, k, l') = 0$ for all k, l' ;
 - 6: $A(j, j, l) = C(i, j, l)$;
 - 7: Set $id(j') = id(i)$ for all j' with $id(j') = id(j)$;
-

A consequence of Algorithm 2 is that both $\sum_{T \in \mathcal{T}(G)} \omega(T)$ and $\sum_{T \in \mathcal{T}(G, R)} \omega(T)$ can be written as the determinant of a matrix of the form (w'_{ij}) where w'_{ij} is the sum of a few edge weights of G . We describe how to find the gradient of a sum of spanning tree weights expressible in such determinant form. Let $\mathcal{T}_P(G)$ be the set of spanning trees of G satisfying some property P . Let $S(G) = \sum_{T \in \mathcal{T}_P(G)} \omega(T)$ and suppose $S(G) = |(w'_{ij})|$ for some $|V| \times |V|$ matrix (w'_{ij}) , with each w'_{ij} being the sum of a few edge weights of G . Let $\vec{F}(i, j, l) = (f_1(i, j, l), \dots, f_m(i, j, l))$. Note

that $\frac{\partial \omega(T)}{\partial w_{ijl}} w_{ijl} = \begin{cases} \omega(T), & (i, j, l) \in T; \\ 0, & (i, j, l) \notin T. \end{cases}$, and $\frac{\partial w_{ijl}}{\partial \lambda_k} = w_{ijl} f_k(i, j, l)$. Thus we have

$$\frac{\partial \omega(T)}{\partial w_{ijl}} \frac{\partial w_{ijl}}{\partial \lambda_k} = \frac{\partial \omega(T)}{\partial w_{ijl}} w_{ijl} f_k(i, j, l) = \begin{cases} \omega(T) f_k(i, j, l), & (i, j, l) \in T; \\ 0, & (i, j, l) \notin T. \end{cases}$$

$$\frac{\partial S(G)}{\partial \lambda_k} = \sum_{T \in \mathcal{T}_P(G)} \frac{\partial \omega(T)}{\partial \lambda_k} = \sum_{i, j, l} \sum_{T \in \mathcal{T}_P(G)} \frac{\partial \omega(T)}{\partial w_{ijl}} \frac{\partial w_{ijl}}{\partial \lambda_k} = \sum_{i, j, l} \sum_{T \in \mathcal{T}_P(G), (i, j, l) \in T} \omega(T) f_k(i, j, l)$$

Since $\sum_{T \in \mathcal{T}_P(G), (i, j, l) \in T} \omega(T) = \sum_{T \in \mathcal{T}_P(G)} \omega(T) - \sum_{T \in \mathcal{T}_P(G - (i, j, l))} \omega(T) = S(G) - S(G - (i, j, l))$, we can thus use the assumption that $S(G)$ is the determinant of a $|V| \times |V|$ matrix to compute $\sum_{T \in \mathcal{T}_P(G), (i, j, l) \in T} \omega(T)$ for all i, j, l , in time $O(|V|^2 L \times |V|^3)$.

As described in [18], there is a much more efficient algorithm, which takes only $O(|V|^3)$ time if each w'_{ij} is the sum of at most some constant number of edge weights of G . This is based

on the fact that for $W = (w'_{ij})$ (with the w'_{ij} 's being viewed as independent variables), we have $\frac{\partial |W|}{\partial w'_{ij}} = (-1)^{i+j} W_{ij}$ (obtained using expansion by minor) and $W^{-1} = \frac{1}{|W|} ((-1)^{i+j} W_{ij})^T$ (the standard formula for the inverse of a non-singular matrix). Hence we have the following

$$\frac{\partial S(G)}{\partial \lambda_k} = \frac{\partial |W|}{\partial \lambda_k} = \sum_{i,j} \frac{\partial |W|}{\partial w'_{ij}} \frac{\partial w'_{ij}}{\partial \lambda_k} = \sum_{i,j} (-1)^{i+j} W_{ij} \frac{\partial w'_{ij}}{\partial \lambda_k} = |W| \sum_{i,j} W_{ji}^{-1} \frac{\partial w'_{ij}}{\partial \lambda_k}$$

Each $\frac{\partial w'_{ij}}{\partial \lambda_k}$ can be efficiently computed as w'_{ij} is the sum of a few edge weights of G , thus by first computing W^{-1} (in $O(|V|^3)$ time using Gaussian elimination), we can then find out $\frac{\partial S(G)}{\partial \lambda_k}$ efficiently.

For the case when $S(G)$ is $\sum_{T \in \mathcal{T}(G)} \omega(T)$, we can choose $W = L^{(1)}(G)$, then $w'_{ij} = \begin{cases} 1, & i = 1; \\ \sum_{i',l} w_{i'jl}, & i = j > 1; \\ -\sum_l w_{ijl}, & i > 1 \wedge i \neq j \end{cases}$

thus we can compute $\frac{\partial S(G)}{\partial \lambda_k}$ values easily.

For the case when $S(G)$ is $\sum_{T \in \mathcal{T}(G,R)} \omega(T)$, there is a simple algorithm to compute all $\frac{\partial S(G)}{\partial \lambda_k}$ values using the tree sum matrix S , the consistency matrix C , and id in Algorithm 2. From $\frac{\partial S(G)}{\partial \lambda_k} = \sum_{i,j,l} \sum_{T \in \mathcal{T}_P(G), (i,j,l) \in T} \omega(T) f_k(i,j,l)$, we have $\frac{\partial S(G)}{\partial \lambda_k} = S(G) \sum_{i,j,l} P(i,j,l) f_k(i,j,l)$, where $\mu(i,j,l)$ is the marginal probability that (i,j,l) is included in a spanning tree satisfying property P , which is $\sum_{T \in \mathcal{T}_P(G), (i,j,l) \in T} \omega(T) / S(G)$. In addition, it can be shown that $\mu(i,j,l) = [(1 - \delta_{r,id(j)}) S_{id(i),id(i)}^{-1} - (1 - \delta_{r,id(i)}) S_{id(j),id(i)}^{-1}] C(i,j,l)$, where $\delta_{i,j}$ is the Dirac delta function.

A.1.2 Computing Regularized Log-likelihood

Computing the function value requires computation of $\sum_{T \in \mathcal{T}(G_s)} \omega(T)$ and $\sum_{T \in \mathcal{T}(G_s,R)} \omega(T)$, which can be done efficiently using the method described in previous section.

A.1.3 Computing Gradient

The main difficulty in computing the gradient of $\mathcal{L}_{\tilde{\lambda}}$ lies in computing partial derivatives of the form $\frac{\partial \ln P_{\tilde{\lambda}}(R_s | \mathbf{s})}{\partial \lambda_i}$. This expression can be rewritten in the following form.

$$\begin{aligned} \frac{\partial \ln P_{\tilde{\lambda}}(R_s | \mathbf{s})}{\partial \lambda_i} &= \frac{\partial [\ln \sum_{T \in \mathcal{T}(G_s, R_s)} \omega(T) - \ln \sum_{T \in \mathcal{T}(G_s)} \omega(T)]}{\partial \lambda_i} \\ &= \frac{1}{\sum_{T \in \mathcal{T}(G_s, R_s)} \omega(T)} \frac{\partial \sum_{T \in \mathcal{T}(G_s, R_s)} \omega(T)}{\partial \lambda_i} - \frac{1}{\sum_{T \in \mathcal{T}(G_s)} \omega(T)} \frac{\partial \sum_{T \in \mathcal{T}(G_s)} \omega(T)}{\partial \lambda_i} \end{aligned}$$

Now we can apply the results in previous section directly to compute the gradients.

A.2 Decoding

Given a sentence s and a trained model with parameter $\vec{\lambda}^*$, we find the directed maximum spanning tree on G_s using the Chu-Liu's algorithm [9], and output the relation edges on the maximum spanning tree as predicted relations.

Appendix B Algorithms for High-order Semi-Markov CRF

In this section, we present efficient algorithms for training and inference for high-order semi-Markov CRF.

B.1 Notations

We use \mathcal{Z} to denote the segment label pattern set $\{\mathbf{z}^1, \dots, \mathbf{z}^M\}$, which is the set of distinct segment label patterns of the m features. The forward-state set $\mathcal{P} = \{\mathbf{p}^1, \dots, \mathbf{p}^{|\mathcal{P}|}\} = \mathcal{Y} \cup \{\mathbf{z}_{1:k}^j\}_{0 \leq k < |\mathbf{z}^j|, 1 \leq j \leq M}$ consists of all the labels and proper prefixes of the segment label patterns. And, the backward-state set $\mathcal{S} = \{\mathbf{s}^1, \dots, \mathbf{s}^{|\mathcal{S}|}\} = \mathcal{P}\mathcal{Y}$ consists of the elements of \mathcal{P} concatenated with a label in \mathcal{Y} , with duplicates removed. As in [35], we also make use of the longest suffix relation $\mathbf{z}_1 \leq_{\mathcal{A}}^s \mathbf{z}_2$ on a set \mathcal{A} , which is true if \mathbf{z}_1 , among all the elements of \mathcal{A} , is the longest suffix of \mathbf{z}_2 .

B.2 Training

Given a training set \mathcal{T} , we maximize the regularized log-likelihood

$$\mathcal{L}_{\mathcal{T}} = \sum_{(\mathbf{x}, \mathbf{s}) \in \mathcal{T}} \log P(\mathbf{s}|\mathbf{x}) - \sum_{i=1}^m \frac{\lambda_i^2}{2\sigma^2}$$

We describe below the algorithms to compute all the necessary components of $\mathcal{L}_{\mathcal{T}}$ and its partial derivatives

$$\frac{\partial \mathcal{L}_{\mathcal{T}}}{\partial \lambda_i} = \tilde{E}(f_i) - E(f_i) - \frac{\lambda_i}{\sigma^2}$$

B.2.1 Partition Function

Let $\mathbf{p}_{j, \mathbf{p}^i}$ be the set of all partial segmentations starting from 1 to j which contain \mathbf{p}^i as the longest suffix. We define the forward variables $\alpha_{\mathbf{x}}(j, \mathbf{p}^i)$ as follows

$$\alpha_{\mathbf{x}}(j, \mathbf{p}^i) = \sum_{\mathbf{s} \in \mathbf{p}_{j, \mathbf{p}^i}} \exp\left(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)\right)$$

Let $\Psi_{\mathbf{x}}(u, v, \mathbf{p}) = \exp(\sum_{i: \mathbf{z}^i \leq_{\mathcal{P}} \mathbf{p}} \lambda_i g_i(\mathbf{x}, u, v))$ and L be the longest possible length of a segment. We can compute $\alpha_{\mathbf{x}}(j, \mathbf{p}^i)$ by

$$\alpha_{\mathbf{x}}(j, \mathbf{p}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{p}^k, y): \mathbf{p}^i \leq_{\mathcal{P}} \mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d, j, \mathbf{p}^k y) \alpha_{\mathbf{x}}(j-d-1, \mathbf{p}^k)$$

The partition function can now be computed by $Z_{\mathbf{x}} = \sum_{\mathbf{p}^i} \alpha_{\mathbf{x}}(|\mathbf{x}|, \mathbf{p}^i)$. The time complexity to compute all the values of $\Psi_{\mathbf{x}}$ is $O(mT^2|\mathcal{Y}||\mathcal{P}|)$. After computing all the values of $\Psi_{\mathbf{x}}$, we can compute all the values of $\alpha_{\mathbf{x}}$ in $O(T^2|\mathcal{Y}||\mathcal{P}|)$ time.

B.2.2 Expected Feature Sum

Let \mathbf{s}_{j,s^i} be the set of all partial segmentations starting from j to $|\mathbf{x}|$ whose longest previous pattern is \mathbf{s}^i . The backward variables $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ are defined as

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{\mathbf{s} \in \mathbf{s}_{j,s^i}} \exp\left(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)\right)$$

We can compute $\beta_{\mathbf{x}}(j, \mathbf{s}^i)$ by

$$\beta_{\mathbf{x}}(j, \mathbf{s}^i) = \sum_{d=0}^{L-1} \sum_{(\mathbf{s}^k, y): \mathbf{s}^k \leq_{\mathcal{S}} \mathbf{s}^i y} \Psi_{\mathbf{x}}(j, j+d, \mathbf{s}^i y) \beta_{\mathbf{x}}(j+d+1, \mathbf{s}^k)$$

Let $P(u, v, \mathbf{z}|\mathbf{x})$ be the marginal probability that (u, v) is a segment and its label pattern is \mathbf{z} . We can compute $P(u, v, \mathbf{z}|\mathbf{x})$ by

$$P(u, v, \mathbf{z}|\mathbf{x}) = \frac{1}{Z_{\mathbf{x}}} \sum_{(\mathbf{p}^i, y): \mathbf{z} \leq_{\mathcal{S}} \mathbf{p}^i y} \alpha_{\mathbf{x}}(u-1, \mathbf{p}^i) \Psi_{\mathbf{x}}(u, v, \mathbf{p}^i y) \beta_{\mathbf{x}}(v+1, \mathbf{p}^i y)$$

The model expectation for feature f_i is:

$$E(f_i) = \sum_{(\mathbf{x}, \mathbf{s}) \in \mathcal{T}} \sum_{u \leq v} P(u, v, \mathbf{z}^i) g_i(\mathbf{x}, u, v)$$

Similar to computing $\alpha_{\mathbf{x}}$, the time complexity to compute all the values of $\beta_{\mathbf{x}}$ is $O(T^2|\mathcal{Y}||\mathcal{S}|)$. After computing all the values of $\Psi_{\mathbf{x}}$, $\alpha_{\mathbf{x}}$, and $\beta_{\mathbf{x}}$, we can compute all the marginal probabilities in $O(T^2|\mathcal{Z}||\mathcal{P}|)$ time.

B.3 Inference

We use a Viterbi-like algorithm for calculating the most likely label sequence. In this algorithm, we define $\delta_{\mathbf{x}}(j, \mathbf{p}^i)$ as follows

$$\delta_{\mathbf{x}}(j, \mathbf{p}^i) = \max_{\mathbf{s} \in \mathbf{s}_{j, \mathbf{p}^i}} \exp\left(\sum_{k=1}^m \sum_{t=1}^{|\mathbf{s}|} \lambda_k f_k(\mathbf{x}, \mathbf{s}, t)\right)$$

We can compute $\delta_{\mathbf{x}}(j, \mathbf{p}^i)$ by using dynamic programming

$$\delta_{\mathbf{x}}(j, \mathbf{p}^i) = \max_{(d, \mathbf{p}^k, y): 0 \leq d \leq L-1 \wedge \mathbf{p}^i \leq_{\mathcal{S}} \mathbf{p}^k y} \Psi_{\mathbf{x}}(j-d, j, \mathbf{p}^k y) \delta_{\mathbf{x}}(j-d-1, \mathbf{p}^k)$$

The best segmentation can be traced back from $\max_{\mathbf{p}^i} \delta_{\mathbf{x}}(|\mathbf{x}|, \mathbf{p}^i)$. The time complexity for decoding is $O(T^2|\mathcal{Y}||\mathcal{P}|)$.

References

- [1] Eugene Agichtein and Luis Gravano. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, June 2000.
- [2] Rie Kubota Ando and Tong Zhang. A framework for learning predictive structures from multiple tasks and unlabeled data. *Journal of Machine Learning Research*, 6:1817–1853, November 2005.
- [3] Michele Banko and Oren Etzioni. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 28–36, 2008.
- [4] Razvan Bunescu, Ruifang Ge, Rohit J. Kate, Edward M. Marcotte, Raymond J. Mooney, Arun K. Ramani, and Yuk Wah Wong. Comparative experiments on learning information extractors for proteins and their interactions. *Artif. Intell. Med.*, 33(2):139–155, 2005.
- [5] Razvan Bunescu and Raymond Mooney. A shortest path dependency kernel for relation extraction. In *Proceedings of the Human Language Technology Conference and the Conference on Empirical Methods in Natural Language Processing*, pages 724–731, 2005.
- [6] Xavier Carreras and Lluís Màrques. Introduction to the conll-2004 shared task: Semantic role labeling. In *Proceedings of CoNLL-2004*, pages 89–97. Boston, MA, USA, 2004.
- [7] Kian Ming Adam Chai. Expectation of f-measures: tractable exact computation and some empirical observations of its properties. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 593–594, New York, NY, USA, 2005. ACM.
- [8] Stanley Chen and Joshua Goodman. An empirical study of smoothing techniques for language modeling. Technical Report TR-10-98, Harvard University, 1998.
- [9] Y. J. Chu and T. H. Liu. On the shortest arborescence of a directed graph. *Science Sinica*, 14:1396–1400, 1965.
- [10] Michael Collins and Nigel Duffy. Convolution kernels for natural language. In *Advances in Neural Information Processing Systems 14*. 2002.
- [11] Aron Culotta, Andrew McCallum, and Jonathan Betz. Integrating probabilistic extraction models and data mining to discover relations and patterns in text. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 296–303, June 2006.
- [12] Aron Culotta and Jeffrey Sorensen. Dependency tree kernels for relation extraction. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 423–429, 2004.

- [13] Ralph Grishman, Silja Huttunen, and Roman Yangarber. Information extraction for enhanced access to disease outbreak reports. *J. of Biomedical Informatics*, 35(4):236–246, 2002.
- [14] Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 415–422, July 2004.
- [15] Bin He, Mitesh Patel, Zhen Zhang, and Kevin Chen-Chuan Chang. Accessing the deep web. *Commun. ACM*, 50(5):94–101, 2007.
- [16] Jing Jiang. Multi-task transfer learning for weakly-supervised relation extraction. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP*, pages 1012–1020, Singapore, August 2009.
- [17] Jing Jiang and ChengXiang Zhai. A systematic exploration of the feature space for relation extraction. In *Proceedings of the Human Language Technologies Conference*, pages 113–120, 2007.
- [18] Terry Koo, Amir Globerson, Xavier Carreras, and Michael Collins. Structured prediction models via the matrix-tree theorem. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 141–150, 2007.
- [19] John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, June 2001.
- [20] Yaliang Li, Jing Jiang, Hai Leong Chieu, and Kian Ming A. Chai. Extracting relation descriptors with modified conditional random fields. In *Proceedings of the 5th International Joint Conference on Natural Language Processing of the AFNLP*. Association for Computational Linguistics, 2011.
- [21] D. C. LIU and J. NOCEDAL. On the limited memory BFGS method for large scale optimization. *Math. Programming*, 45(3, (Ser. B)):503–528, 1989.
- [22] David McClosky, Eugene Charniak, and Mark Johnson. Automatic domain adaptation for parsing. In *Proceedings of Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 28–36, 2010.
- [23] Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. Convolution kernels on constituent, dependency and sequential structures for relation extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 1378–1387, 2009.

- [24] Viet Cuong Nguyen, Nan Ye, Wee Sun Lee, and Hai Leong Chieu. Semi-markov conditional random field with high-order features. In *ICML Workshop on Structured Sparsity: Learning and Inference*, 2011.
- [25] Sinno Jialin Pan and Qiang Yang. A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359, October 2010.
- [26] Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics*, pages 697–704, 2008.
- [27] Lawrence R. Rabiner. *A tutorial on hidden Markov models and selected applications in speech recognition*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1990.
- [28] Benjamin Rosenfeld and Ronen Feldman. URES : An unsupervised Web relation extraction system. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 667–674, July 2006.
- [29] Sunita Sarawagi and William W. Cohen. Semi-markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, pages 1185–1192. 2005.
- [30] Fei Sha and Fernando Pereira. Shallow parsing with conditional random fields. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 282–289, 2003.
- [31] Yusuke Shinyama and Satoshi Sekine. Preemptive information extraction using unrestricted relation discovery. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 304–311, June 2006.
- [32] W. Tutte. *Graph theory*. Addison-Wesley, 1984.
- [33] Vincent Van Asch and Walter Daelemans. Using domain similarity for performance estimation. In *Proceedings of the 2010 Workshop on Domain Adaptation for Natural Language Processing*, pages 31–36, 2010.
- [34] Christopher Walker, Stephanie Strassel, Julie Medero, and Kazuaki Maeda. ACE 2005 multilingual training corpus. Linguistic Data Consortium, Philadelphia, 2006.
- [35] Nan Ye, Wee Sun Lee, Hai Leong Chieu, and Dan Wu. Conditional random fields with high-order features for sequence labeling. In *NIPS'09: Advances in Neural Information Processing Systems 22*, pages 1393–1400, Cambridge, MA, 2009. MIT Press.
- [36] Dmitry Zelenko, Chinatsu Aone, and Anthony Richardella. Kernel methods for relation extraction. *Journal of Machine Learning Research*, 3:1083–1106, 2003.

- [37] Min Zhang, Jie Zhang, and Jian Su. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference*, pages 288–295, 2006.
- [38] Shubin Zhao and Ralph Grishman. Extracting relations with integrated information using kernel methods. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 419–426, 2005.
- [39] GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434, 2005.