Extracting Relation Descriptors with Conditional Random Fields

Yaliang Li[†], Jing Jiang[†], Hai Leong Chieu[‡], Kian Ming A. Chai[‡]
[†]School of Information Systems, Singapore Management University, Singapore
[‡]DSO National Laboratories, Singapore
{ylli,jingjiang}@smu.edu.sg, {chaileon,ckianmin}@dso.org.sg

Abstract

In this paper we study a novel relation extraction problem where a general relation type is defined but relation extraction involves extracting specific relation descriptors from text. This new task can be treated as a sequence labeling problem. Although linear-chain conditional random fields (CRFs) can be used to solve this problem, we modify this baseline solution in order to better fit our task. We propose two modifications to linear-chain CRFs, namely, reducing the space of possible label sequences and introducing long-range features. Both modifications are based on some special properties of our task. Using two data sets we have annotated, we evaluate our methods and find that both modifications to linear-chain CRFs can significantly improve the performance for our task.

1 Introduction

Relation extraction is the task of identifying and characterizing the semantic relations between entities in text. 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 (Zelenko et al., 2003; Zhou et al., 2005; Bunescu and Mooney, 2005; Zhang et al., 2006). 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 (Agichtein and Gravano, 2000). 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 (Hasegawa et al., 2004; Rosenfeld and Feldman, 2006; Shinyama and Sekine, 2006). More recently, open relation extraction has also been proposed where there is no fixed domain or predefined relation type, and the goal is to identify all possible relations from an open-domain corpus (Banko and Etzioni, 2008; Wu and Weld, 2010; Hoffmann et al., 2010).

These different relation extraction settings suit different applications. In this paper, 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 ACE[1] 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 mentioned in the text. Table 1 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 paper 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 (CRFs) (Lafferty et al., 2001). However, this solution ignores a useful property of the task: the space of possible label sequences is much smaller than that enumerated by a linear-chain CRF. There are two implications. First, the normalization constant in the linear-chain CRF is too large because it also enumerates the impossible sequences. Second, the restriction to the correct space of label sequence per-

[1]Automatic Content Extraction http://www.itl.nist.gov/iad/mig/tests/ace/

Proceedings of the 5th International Joint Conference on Natural Language Processing, pages 392–400,
Chiang Mai, Thailand, November 8 – 13, 2011.

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

Table 1: Some examples of candidate relation instances and their relation descriptors.

mits the use of long-range features without an exponential increase in computational cost.

We compare the performance of the baseline linear-chain CRF model and our special CRF model on two data sets that we have manually annotated. Our experimental results show that both reducing the label sequence space and introducing long-range features can significantly improve the baseline performance.

The rest of the paper is organized as follows. In Section 2 we review related work. We then formally define our task in Section 3. In Section 4 we present a baseline linear-chain CRF-based solution and our modifications to the baseline method. We discuss the annotation of our data sets and show our experimental results in Section 5. We conclude in Section 6.

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 (Bunescu and Mooney, 2005; Zhou et al., 2005; Zhang et al., 2006). 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 (Banko and Etzioni, 2008). Since there are no fixed 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 (Banko and Etzioni, 2008) 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 databases.

In terms of models and techniques, we use standard linear-chain CRF as our baseline, which is the main method used in (Banko and Etzioni, 2008) as well as for many other information extraction problems. The major modifications we propose for our task are the 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 Sarawagi and Cohen (2005). 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.

3 Task Definition

In this section 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 $(w_1, w_2, \ldots, w_n)$

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 $(w_r, \ldots, w_s)$ $(1 \leq r \leq s \leq n)$ 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.

The above definition constrains *ARG-1* and *ARG-2* to single tokens. In our experiments, we will replace the original lexical strings of *ARG-1* and *ARG-2* with the generic tokens `ARG1` and `ARG2`. Examples of sentences with the named entities replaced with argument tokens are shown in the second column of Table 1.

4 Method

4.1 Representation

The relation descriptor extraction task can be treated as a sequence labeling problem. Let $\boldsymbol{x} = (x_1, x_2, \ldots, x_n)$ 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, and the phrase boundary information. 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 (Ramshaw and Marcus, 1995) in sequence labeling, we define $\mathcal{Y} = \{\textit{B-REL}, \textit{I-REL}, \textit{O}\}$. Let $\boldsymbol{y} = (y_1, y_2, \ldots, y_n)$ denote the sequence of labels for $\boldsymbol{x}$. Our task can be reduced to finding the best label sequence $\hat{\boldsymbol{y}}$ among all the possible label sequences for $\boldsymbol{x}$.

4.2 A Linear-Chain CRF Solution

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

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

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

$$Z(\boldsymbol{x}, \Lambda) = \sum_{\boldsymbol{y'}} \exp\Big(\sum_i \sum_k \lambda_k f_k(y'_{i-1}, y'_i, \boldsymbol{x})\Big) \quad (2)$$

is the normalization constant.

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

$$\hat{\Lambda} = \arg\min_{\Lambda} \Bigg(-\sum_j \log p(\boldsymbol{y}_j^*|\boldsymbol{x}_j, \Lambda) + \beta \sum_k \lambda_k^2 \Bigg). \quad (3)$$

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

4.3 Improvement over Linear-Chain CRFs

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

Label sequence constraint

In linear-chain CRFs, the normalization constant Z considers all possible label sequences $\boldsymbol{y}$. 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*, and (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 CRFs is to enforce it only 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 $\boldsymbol{x}$, let $\tilde{\mathbf{Y}}$ denote the set of valid label sequences, i.e., those that have either one or no relation descriptor sequence. We then choose the best sequence $\hat{\boldsymbol{y}}$ as follows:

$$\hat{\boldsymbol{y}} = \arg\max_{\boldsymbol{y} \in \tilde{\mathbf{Y}}} p(\boldsymbol{y}|\boldsymbol{x}, \hat{\Lambda}). \quad (4)$$

Arguably, the more principled way to exploit the constraint is to modify the probabilistic model itself. So at the training stage, we should also consider only $\tilde{\mathbf{Y}}$ by defining the normalization term $\tilde{Z}$ as follows:

$$\tilde{Z}(\boldsymbol{x}, \Lambda) = \sum_{\boldsymbol{y'} \in \tilde{\mathbf{Y}}} \exp\Big(\sum_i \sum_k \lambda_k f_k(y'_{i-1}, y'_i, \boldsymbol{x})\Big). \quad (5)$$

The difference between Equation (5) and Equation (2) is the set of label sequences considered. In other words, while in linear-chain CRFs the correct label sequence competes with all possible label sequences for probability mass, for our task the correct label sequence should compete with only other valid label sequences. In Section 5 we will compare these two different normalization terms and show the advantage of using Equation (5).

Adding long-range features

In linear-chain CRF models, only first-order label dependencies are considered because features are defined over two consecutive labels. Inference in linear-chain CRFs can be done efficiently using dynamic programming. 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 with the order of dependency.

For our task, because of the constraint on the space of label sequences, we can afford to use long-range features. In our case, inference is still efficient because the number of sequences to be enumerated has been drastically reduced due to the constraint. Let $g(\boldsymbol{y}, \boldsymbol{x})$ denote a feature function defined over the entire label sequence $\boldsymbol{y}$ and the observation sequence $\boldsymbol{x}$. We can include such feature functions in our model as follows:

$$p(\boldsymbol{y}|\boldsymbol{x}, \Theta) = \frac{1}{\tilde{Z}(\boldsymbol{x}, \Theta)} \Bigg[\exp\Big(\sum_i \sum_k \lambda_k f_k(y_{i-1}, y_i, \boldsymbol{x}) + \sum_l \mu_l g_l(\boldsymbol{y}, \boldsymbol{x}) \Big) \Bigg], \quad (6)$$

where $\Theta = \{\{\lambda_k\}, \{\mu_l\}\}$ is the set of all model parameters. Both $\{\lambda_k\}$ and $\{\mu_l\}$ are regularized as in Equation (3). Note that although each $f(y_{i-1}, y_i, \boldsymbol{x})$ may be subsumed under a $g(\boldsymbol{y}, \boldsymbol{x})$, here we group all features that can be captured by linear-chain CRFs under f and other real long-range features under g. In Section 5 we will see that with the additional feature functions g, relation extraction performance can also be further improved.

4.4 Features

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

Linear-chain features

The linear-chain features are those that can be formulated as $f(y_{i-1}, y_i, \boldsymbol{x})$, i.e., those that depend on $\boldsymbol{x}$ 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 2 shows the feature templates covering only the observations. Each feature shown in Table 2 is further combined with either the value of the current label y_i or the values of the previous and the current labels y_{i-1} and y_i to form zeroth order and first order features. For example, a zeroth order feature is "y_i is *B-REL* and w_i is `the` and w_{i+1} is `president`", and a first order feature is "y_{i-1} is *O* and y_i is *B-REL* and t_i is `N`".

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 $\boldsymbol{y}$ that contains a relation descriptor sequence, let $(w_r, w_{r+1}, \ldots, w_s)$ 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 3. 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).

5 Experiments

5.1 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" relation.[2]

The first data set contains 150 business articles from New York Times. The articles were crawled from the NYT website between November 2009

[2] `http://www.mysmu.edu/faculty/jingjiang/data/IJCNLP2011.zip`

Description	Feature Template	Example
single token	w_{i+j} $(-2 \leq j \leq 2)$	w_{i+1} (next token) is *president*
single POS tag	t_{i+j} $(-2 \leq j \leq 2)$	t_i (current POS tag) is *DET*
single phrase tag	p_{i+j} $(-2 \leq j \leq 2)$	p_{i-1} (previous phrase boundary tag) is *I-NP*
two consecutive tokens	$w_{i+j-1}\&w_{i+j}$ $(-1 \leq j \leq 2)$	w_i is *the* and w_{i+1} is *president*
two consecutive POS tags	$t_{i+j-1}\&t_{i+j}$ $(-1 \leq j \leq 2)$	t_i is *DET* and t_{i+1} is *N*
two consecutive phrase tags	$p_{i+j-1}\&p_{i+j}$ $(-1 \leq j \leq 2)$	p_i is *B-NP* and p_{i+1} is *I-NP*

Table 2: Linear-chain feature templates. Each feature is defined with respect to a particular (current) position in the sequence. i indicates the current position and j indicates the position relative to the current position. All features are defined using observations within a window size of 5 of the current position.

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

Table 3: Long-range feature templates. r and s are the indices of the first word and the last word of the relation descriptor, respectively.

and January 2010. After sentence segmentation and tokenization, we used the Stanford NER tagger (Finkel et al., 2005) 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*.

The second data set comes from a Wikipedia personal/social relation data set previously used in (Culotta et al., 2006). The original data set does not contain annotations of relation descriptors such as "sister" or "friend" between the two PER arguments. We therefore also manually annotated this data set. Similarly, we performed sentence segmentation, tokenization and NER tagging, and took each pair 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*.[3]

[3]Since 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.

We go through each candidate relation instance to find whether there is an explicit sequence of words describing the relation between *ARG-1* and *ARG-2*, and label the sequence of words, if any. Note that we only consider explicitly stated relation descriptors. If we cannot find such a relation descriptor, even if *ARG-1* and *ARG-2* actually have some kind of relation, we still label the instance as *Nil*. For example, in the instance "he is the son of `ARG1` and `ARG2`", although we can infer that *ARG-1* and *ARG-2* have some family relation, we regard this as a negative instance.

A relation descriptor may also contain multiple relations. For example, in the instance "`ARG1` is the CEO and president of `ARG2`", we label "the CEO and president" as the relation descriptor, which actually contains two job titles, namely, CEO and president.

Note that our annotated relation descriptors are not always nouns or noun phrases. An example is the third instance for personal/social relation in Table 1, where the relation descriptor "married" is a verb and indicates a spouse relation.

The total number of relation instances, the number of positive and negative instances as well as the number of distinct relation descriptors in each data set are summarized in Table 4.

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

Table 4: Number 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 descriptors.

5.2 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 2.
- **M-CRF-1**: This is our modified linear-chain 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**: This is M-CRF-1 with the addition of the contextual long-range features described in Table 3.
- **M-CRF-3**: This is M-CRF-2 with the addition of the path-based long-range features described in Table 3.
- **M-CRF-4**: This is M-CRF-3 with the addition of the phrase boundary long-range feature described in Table 3.

For the standard linear-chain CRF model, we use the package CRF++[4]. We implement our own version of the modified linear-chain CRF models.

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 testing. We report the average performance across the 10 runs.

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

[4] http://crfpp.sourceforge.net/

We evaluate the performance using two different criteria: overlap match and exact match. Overlap match is a more relaxed criterion: if the extracted 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 extracted relation descriptor be exactly the same as the true relation descriptor 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.

5.3 Method Comparison

In Table 5, we summarize the performance in terms of the various measures on the two data sets. For both the baseline linear-chain CRF model and our modified linear-chain CRF models, we have tuned the regularization parameters and show only the results using the optimal parameter values for each data set, chosen from $\beta = 10^{\gamma}$ for $\gamma \in [-3, -2, \ldots, 2, 3]$.

First, we can see from the table 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 is mostly evident in the improvement of recall. This shows that a larger number of true relation descriptors are extracted when the label sequence space is reduced.

Next we see from the table 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-CRF-3 can significantly outperform M-CRF-1 in all settings. In this case, the improvement is a mixture of both precision and recall. This 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

New York Times	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**

Wikipedia	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 5: Comparison of different methods on the New York Times data set and 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 paired T-test.

significant impact. We hypothesize the following. 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_{i-1} = *is* & y_i = *B-R*". For the phrase boundary long-range feature, since phrase boundary tags have also been used in the linear-chain features, this feature does not provide additional information. In addition, we have found that a large percentage of relation descriptors violate phrase boundaries: 22% in the NYT data set, and 29% in the Wikipedia data set. Therefore, it seems that phrase boundary information is not important for relation descriptor extraction.

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 more varied ways. The lower performance achieved on the Wikipedia data set suggests that extracting relation descriptors is not an easy task even under a supervised learning setting.

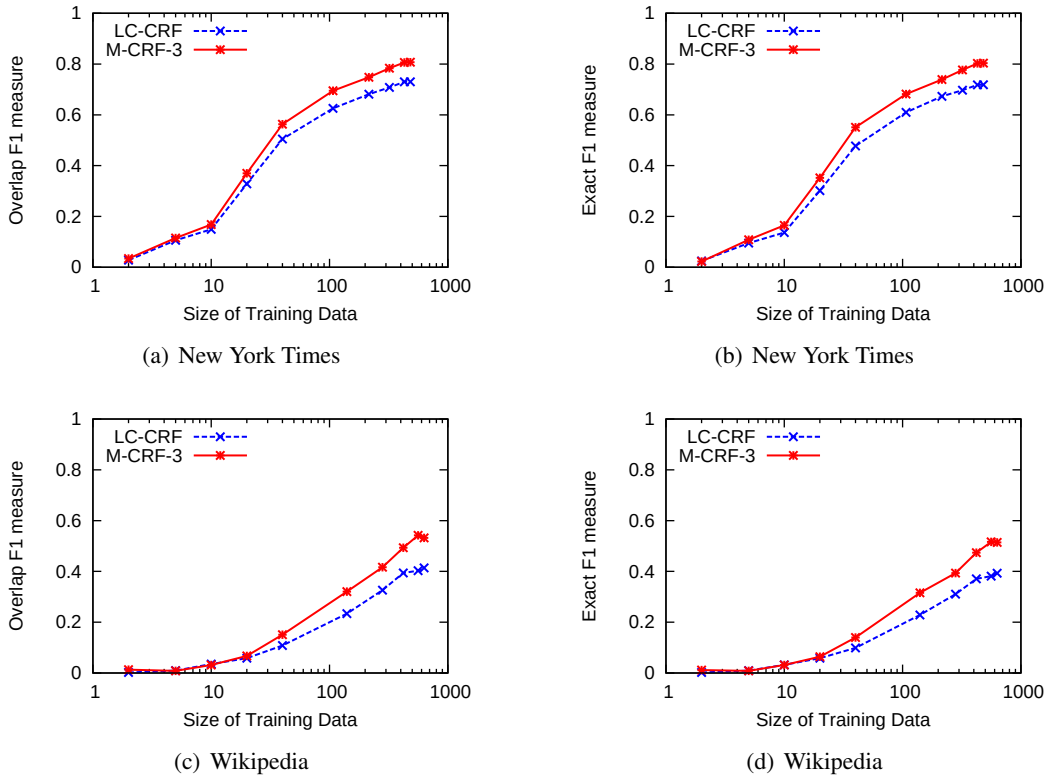
Presumably relation descriptors that are not seen in the training data are harder to extract. We would therefore also like to see how well our model works on such unseen relation descriptors. We find that with 10-fold cross validation, for the NYT data set, on average our model is able to extract approximately 67% of the unseen relation descriptors in the test data using exact match criteria. 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. However, our model is still able to pick up new relation descriptors.

5.4 The Effect of Training Data Size

In the previous experiments, we have used 90% of the data for training and the remaining 10% for testing. We now take a look at how the performance 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 the Wikipedia data set is the more difficult than the NYT data set. This is consistent with our observation in the previous section.

The modified linear-chain CRF model consistently outperforms the baseline linear-chain CRF model. For similar level of performance, the modified linear-chain CRF model requires less training data than the baseline linear-chain CRF model.



(a) New York Times (b) New York Times

(c) Wikipedia (d) Wikipedia

Figure 1: Performance of LC-CRF and M-CRF-3 as the training data size increases.

For example, Figure 1(b) shows that the modified linear-chain CRF model achieve 0.72 F1 with about 215 training instances, while the baseline linear-chain CRF model requires about 480 training instances for a similar F1.

6 Conclusions

In this paper, 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 CRFs 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 linear-chain CRF model can perform significantly better than standard linear-chain CRFs.

Currently our work is only based on evaluation on two data sets and on two general relations. In the future we plan to evaluate the methods on other general relations to test its robustness. We also plan to explore how this new relation extraction task can be used within other NLP or text mining applications.

Acknowledgments

This material is based on research sponsored by the Air Force Research Laboratory, under agreement number FA2386-09-1-4123. The U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation thereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of the Air Force Research Laboratory or the U.S. Government.

References

Eugene Agichtein and Luis Gravano. 2000. Snowball: Extracting relations from large plain-text collections. In *Proceedings of the Fifth ACM Conference on Digital Libraries*, pages 85–94, June.

Michele Banko and Oren Etzioni. 2008. The tradeoffs between open and traditional relation extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics*, pages 28–36.

Razvan Bunescu and Raymond Mooney. 2005. 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, October.

Aron Culotta, Andrew McCallum, and Jonathan Betz. 2006. 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.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 363–370, June.

Takaaki Hasegawa, Satoshi Sekine, and Ralph Grishman. 2004. Discovering relations among named entities from large corpora. In *Proceedings of the 42nd Meeting of the Association for Computational Linguistics*, pages 415–422, July.

Raphael Hoffmann, Congle Zhang, and Daniel S. Weld. 2010. Learning 5000 relational extractors. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 286–295, July.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. 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.

Lance A. Ramshaw and Mitchell P. Marcus. 1995. Text chunking using transformation-based learning. In *Proceedings of the Third ACL Workshop on Very Large Corpora*, pages 82–94.

Benjamin Rosenfeld and Ronen Feldman. 2006. 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.

Sunita Sarawagi and William W. Cohen. 2005. Semi-Markov conditional random fields for information extraction. In *Advances in Neural Information Processing Systems 17*, pages 1185–1192.

Yusuke Shinyama and Satoshi Sekine. 2006. 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.

Fei Wu and Daniel S. Weld. 2010. Open information extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, July.

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

Min Zhang, Jie Zhang, and Jian Su. 2006. Exploring syntactic features for relation extraction using a convolution tree kernel. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 288–295, June.

GuoDong Zhou, Jian Su, Jie Zhang, and Min Zhang. 2005. Exploring various knowledge in relation extraction. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, pages 427–434, June.