# Relation Classification using Entity Sequence Kernels

*Debanjan Ghosh* *Smaranda Muresan*
School of Communication and Information
Rutgers University, New Jersey 08854, USA
{debanjan.ghosh,smuresan}@rutgers.edu

Abstract
This paper presents a novel gap weighted string kernel referred to as an entity sequence kernel for the task of relation extraction. Experiments classifying different relations on a standard dataset show that the entity sequence kernels achieve F1 scores on par with current state-of-the-art techniques. We also present a study of how the *order* of candidate entities influences the *relation directionality* and how various factors might contribute to the accuracy of results for each relation direction.

TITLE AND ABSTRACT IN Bengali (বাংলা)

নামাঙ্কিত সত্তার সম্পর্ক নিষ্কাশন করণ ক্রম কার্নেল

এই গবেষণা পত্রে সম্পর্ক নিষ্কাশনের জন্য একটি নতুন নামাঙ্কিত সত্তার ভারযুক্ত স্ট্রিং কার্নেল উপস্থাপন করা হয়েছে। পরীক্ষামূলক ভাবে নামাঙ্কিত সত্তা কার্নেল্স ব্যবহার করে বিভিন্ন সম্পর্কের উদাহরণগুলি শ্রেণীবদ্ধ করা হয়েছে যার ফলাফল খুবই সন্তোষজনক। কিভাবে বিভিন্ন স্থিতিমাপ এবং সত্তা-প্রার্থীদের অনুবর্তিতা, সম্পর্ক অভিমুখ বিচারে অবদান রাখে আমরা সেটিও এই প্রবন্ধে পরিবেশন করেছি।

Keywords: Relation Extraction, Entity Sequence Kernel, SVM.

Keywords in $L_2$: সম্পর্ক নিষ্কাশন,নামাঙ্কিত সত্তা কার্নেল, SVM.

*Proceedings of COLING 2012: Posters*, pages 391–400,
COLING 2012, Mumbai, December 2012.

## 1 Introduction

Information Extraction (IE) is the task of extracting structured information from unstructured text. Two major sub-tasks of IE are extracting entities such as [John Smith]$_{Pers}$, [New York]$_{Loc}$ and [Google Inc]$_{Org}$ and the relation between these entities, such as *Work_For* relation between [John Smith]$_{Pers}$ and [Google Inc.]$_{Org}$, and *Live_In* relation between [John Smith]$_{Pers}$ and [New York]$_{Loc}$. Extracting relations between entities is still a significantly harder task than recognizing entities, and current state-of-the-art systems achieve inferior results. Consider the following examples of a *Live_In* relation from the corpus introduced by (Roth and Yih, 2004):

(1) *[Actress Angie Dickinson]$_{Pers}$, who was born in [Kulm,N.D.]$_{Loc}$ donated a coat she wore to the 1966 [Academy Awards]$_{Other}$*

(2) *[Modesto]$_{Loc}$, native [George Lucas]$_{Per}$ 's film was released...*

Our task is to extract the *Live_In* relation from the above sentences where the involved named entities are [Actress Angie Dickinson]$_{Per}$ and [Kulm, N.D.]$_{Loc}$ in example (1) and [George Lucas]$_{Per}$ and [Modesto]$_{Loc}$ in example (2). These two examples are illustrative of two key challenges: 1) a sentence can contain multiple entities (e.g., [Academy awards]$_{Other}$ is a named entity in sentence (1), but it is not part of the *Live_In* relation); and 2) each relation has a concept of *directionality*. This is because the arguments in a relation often take different roles and need to be distinguished ( *Live_in*([Actress Angie Dickinson]$_{Pers}$, [Kulm,N.D.]$_{Loc}$] vs. *Live_in*([Modesto]$_{Loc}$,[George Lucas]$_{Pers}$). Identifying the right directionality is key to the task of relation extraction. While few recent work on relation extraction has modeled the directionality of relations (Roth and Yih, 2004; Giuliano et al., 2007; Kate and Mooney, 2010; Zhang et al., 2008), these studies have only reported averaged results. A key contribution of this paper is an in-depth study of relation directionality, showing how various factors might contribute to the accuracy of results for each relation direction.

In this paper, we explore a novel approach of creating substring sequences from corpora annotated with entities for relation extraction. We use intra-sentential information between the entities to create string sequences, which we call *entity sequences*. In our approach, we assume that entity boundaries are known, but the types of entities are unknown. We treat the relation extraction problem as a supervised learning (classification) problem. A modified string kernel is applied over *entity sequences*. This kernel in turn is augmented with SVM to find the decision hyperplane that can separate one relation from the other. We show that semantic and syntactic features (WordNet hypernyms and dependency relations) help the classifier to achieve better results. We also present a preliminary set of experiments using a shortest path dependency kernel similar to the one introduced by Bunescu and Mooney (2005b), which improves our results for three out of the five relations under study. We use the dataset created by Roth and Yih (2004)[1] for two main reasons: 1) it represents a challenging dataset for our task since there are often more than two entities in a sentence, unlike SemEval 2010 dataset[2] and 2) it has been widely used in recent relation extraction research (Roth and Yih, 2004, 2007; Giuliano et al., 2007; Kate and Mooney, 2010) allowing us to compare our results with prior work. This dataset is referred to as the RY dataset.

In Section 2 we describe the method of creating entity sequences for the relation extraction task. Section 3 formally presents our proposed kernel. We discuss the kernel performance in Section

[1] http://cogcomp.cs.illinois.edu/Data/ER/

[2] http://semeval2.fbk.eu/semeval2.php

4 including detailed experiments of relation directionality and comparison with state-of-the-art methods. In Section 5 we briefly review related work.

## 2 Entity Sequence Generation

Given a sentence S that contains a set of entities $e_1, \ldots e_n$ a relation $R_{ij}$ exists between a pair of entities $e_i$ and $e_j$, where $e_i$ is the first entity and $e_j$ is the second entity. Together $e_i$ and $e_j$ are considered *candidate entities*. For example, given the sentence:

(3) a reasonable doubt that [Oswald]$_{Pers}$ was the lone gunman who killed [President Kennedy]$_{Pers}$ and [Officer Tippit]$_{Pers}$ and that there was no coverup by the [Warren Commission]$_{Org}$

There are two *Kill* relations. The first one between [Oswald]$_{e_i}$ (first entity) and [President Kennedy]$_{e_j}$ (second entity) and the second one between [Oswald]$_{e_i}$ (first entity) and [Officer Tippit]$_{e_j}$ (second entity). In this paper, we introduce the concept of the *entity sequence* and describe how it represents entities and relations in a sentence. An entity sequence depends upon the position and occurrence of entities in a sentences. We introduce three terms to represent the word sequences related to a relation: 1) *pre-entity* (the word sequence before the first entity of a relation), 2) *intra-entities* (the word sequence between the two entities) and 3) *post-entity* (the word sequence after the second entity). Thus, an entity sequence (ES) is defined as:

$$ES = [pre] + entity1 + [intra] + entity2 + [post] \quad (1)$$

The pre/post entity word sequences have a maximum length of four words. Each entity sequence can contain a maximum of one relation between candidate entities *entity1* and *entity2*. If an entity sequence contains a relation, then the sequence is considered as a *positive* example for the given relation. Otherwise, it is a *negative* example. A single entity can take part in multiple relations. For example, in Figure 1, [Oswald]$_{Pers}$ is part of two *Kill* relations. In contrast, an entity in a given sentence might not take part in any relation (e.g., [Warren Commission]$_{Org}$). From a given sentence S, it is trivial to create the set of entity sequences by permuting the position of the entities (e.g., Figure 1). However, this has an unwanted consequence of producing an extremely large number of negative entity sequences. Thus, to balance the distribution of the positive and the negative examples in the training set, we selected only those negative entity sequences where at least one of the two entities is a gold standard entity. In Figure 1, we have a total of six entity sequences generated from the candidate sentence. The first ( [Oswald]$e_i$ ... killed [President Kennedy]$e_j$) and the second ([Oswald]$e_i$ ... killed ... and [Officer Tippit]$e_k$) are positive examples for the *Kill* relation, where the rest are negative examples. Sentences in the RY dataset are taken from the TREC corpus and annotated with entities and relations. Our experiments used only the 1,437 sentences that contain at least one relation. There are four types of entities (*Person (Pers)*, *Location (Loc)*, *Organization (Org)* and *Other*) and five types of relations: *Kill*, *Live_In*, *Work_For*, *Located_In*, and *OrgBased_In*. Based on the algorithm of entity sequence generation we created 268 *Kill*, 521 *Live_in*, 401 *Work_For*, 405 *Located_In* and 452 *OrgBased_In* positive entity sequences. Each ES indicates a pair of candidate entities holding a binary relation. Table 1 depicts the relations in the RY dataset, as well as relation directionality. For each entity sequence the candidate entities have assigned a role. For three relations (*Work_For, OrgBased_In* and *Live_In*) the types of the candidate entities are different. For simplicity, we have defined a specific nomenclature for the

**Entity Sequences Generated:**
ES1= a reasonable doubt that [Oswald]$e_i$ was the lone gunman who killed [President Kennedy]$e_j$ and Officer Tippit and
ES2= a reasonable doubt that [Oswald]$e_i$ was the lone gunman who killed President Kennedy and [Officer Tippit]$e_k$ and that there was
ES3= a reasonable doubt that [Oswald]$e_i$ was the lone gunman who killed President Kennedy and Officer Tippit and that there was no coverup by the [Warren Commission]$e_l$
ES4= gunman who killed [President Kennedy]$e_j$ and [Officer Tippit]$e_k$ and that there was
ES5= gunman who killed [President Kennedy]$e_j$ and Officer Tippit and that there was no coverup by the [Warren Commission]$e_l$
ES6= killed President Kennedy and [Officer Tippit]$e_k$ and that there was no coverup by the [Warren Commission]$e_l$

Figure 1: Example of entity sequences for a given sentence.

| Relation | positive | negative | $e_1$ | $e_2$ | Example |
|---|---|---|---|---|---|
| Kill | 191 | 962 | Pers | Pers | $Oswald$ killed $Kennedy$ |
| | 77 | | Pers | Pers | $Tippit$ was killed by $Oswald$ |
| Located_in | 337 | 1234 | Loc | Loc | $DisneyWorld$ in $Florida$ |
| | 68 | | Loc | Loc | $China$'s agricultural producers, $Anhui$ |
| Work_for | 260 | 1280 | Pers | Org | $Emmerich$ vice president of $ABCcorp$ |
| | 141 | | Org | Pers | $Pepco$ executive $SharonPrattDixon$ |
| Orgbased_in | 283 | 1338 | Loc | Org | $USA$ has leaped 34%... $FBI$ reported |
| | 169 | | Org | Loc | $leatherfactory$ in $Caguasu$ |
| Live_in | 376 | 1549 | Pers | Loc | $DavidAbernathy$ is born in $Linden$ |
| | 145 | | Loc | Pers | $Illinois$ born $CharltonHeston$ |

Table 1: Statistics of Entity Sequences in the RY dataset

order of the entities. In the case of *Work_For* relation, the *Pers* entity is $e_1$ and the *Org* entity is $e_2$. This ordering of entities is denoted as $e_1 \rightarrow e_2$. There are 260 examples of this ordering in Table 1. Conversely, there are 141 examples of type $e_2 \rightarrow e_1$ where *Org* is the first entity and *Pers* is the second entity. We have applied some heuristics for *Kill* and *Located_In* because the candidate entities are of the same type. In the first example, one *Pers* entity [Oswald]$e_1$ is acting upon another *Pers* entity [President Kennedy]$e_2$. According to our heuristic thus [Oswald] is denoted as $e_1$ and [President Kennedy] is $e_2$. So the order of the entities is $e_1 \rightarrow e_2$. Similarly, in *[Officer Tippit]* $e_k$*, the Dallas policeman who was killed by [Oswald]*$e_i$ the order of the entities is $e_2 \rightarrow e_1$. For the *Located_in* relation, when a location entity is inside another location entity we denote the contained entity $e_1$ and the container entity as $e_2$. In the example *[Disney World]*$e_i$ *in [Florida]*$e_j$, we have that [Disney World] is $e_1$ and [Florida] is $e_2$. Similarly, in the example *[China]*$e_i$*'s major agricultural producers, [Anhui]*$e_j$, [China] becomes $e_2$ and [Anhui] is $e_1$.

## 3 Entity Sequence Kernel and SVM

Once we generate *entity sequences* from the given sentences, the next task is to adopt the proper machine learning algorithm for the relation extraction task. Every relation is split into two *sub relations* ($e_1 \rightarrow e_2$ and $e_2 \rightarrow e_1$) depending upon the order of the candidate entities. All negative examples are categorized together in a single category. We utilize a modified version of the gap weighted sequence kernel (Lodhi et al., 2002) for the relation extraction task. Our data set (entity sequences) is nothing but a carefully selected sequences of words, where the order of the words is of prime importance. A conventional BoW feature vector representation (e.g., binary value features) is unaware of the word order and hence it will be difficult for a traditional

classifier (e.g., a standard vector kernel) to classify entity sequences. Instead, gap weighted sequence kernels (Lodhi et al., 2002) are a perfect fit to handle instances where the order of the word sequences is essential. Thus, this kernel is a natural choice for our classification task.

Given two *entity sequences* $s$ and $t$, an Entity Sequence Kernel $K_{es}$ counts the number of subsequences of length n common to both $s$ and $t$. Formally, let $F_i$ be the feature space over the *words* in an ES. Similarly, we consider other disjoint feature spaces $F_j, F_k, ..., F_l$ (e.g., stem, POS tags, chunk tags) (Bunescu and Mooney, 2005a) where the set of all possible feature vectors $F_\times = F_i \times F_j \times F_k \times ... \times F_l$. For any two feature vectors $x, y \in F_\times$ let $sim(x, y)$ computes the number of similar (i.e., common) features between $x$ and $y$. Given two entity sequences $s$ and $t$ over the finite set $F_\times$, let $|s|$ denote the length of $s = s_1...s_{|s|}$. Let $\mathbf{i} = (i_1, ..., i_{|i|})$ be a sequence of $|i|$ indices in $s$ where the length l(**i**) is $i_{|\mathbf{i}|} - i_1 + 1$. Similarly, **j** is a sequence of $|j|$ indices in $t$. The kernel function $K_{es}(s, t, \lambda)$ that calculates the number of weighted sparse subsequences of length n (say, n=2:bigram) common to both $s$ and $t$, is defined as:

$$K_{es}(s, t, \lambda) = \sum_{\mathbf{i}:|\mathbf{i}|=n} \sum_{\mathbf{j}:|\mathbf{j}|=n} \prod_{k=1}^{n} sim(s_{i_k}, t_{j_k}) \lambda^{l(\mathbf{i})+l(\mathbf{j})} \quad (2)$$

The recursive computation can be computed in $O(kn|s||t|)$ time. The gap between the words is penalized with a suitable decay factor $\lambda$ $(0 < \lambda < 1)$. This decay factor in turn compensates for matches between lengthy word sequences. The design of the kernel $K_{es}$ is created by the $pre$, $intra$, and $post$ patterns, which have already been found useful in previous work of relation extraction (Giuliano et al., 2007; Bunescu and Mooney, 2005a). We define two separate kernels to effectively use the candidate entities and the word sequence before and after them. The *relation kernel* $K_{rel}$ measures the similarity between $s$ and $t$ by adding up the evidences of various sub kernels over the word sequences ($pre$, $post$ and $intra$): $K_{rel} = K_{prei} + K_{int} + K_{ipost}$, where $K_{prei}$ consists of *pre-entity* and *intra-entity* substrings, $K_{int}$ consists of *intra-entity* substring, and $K_{ipost}$ consists of *intra-entity* and *post-entity* substrings. The *entity kernel*, $K_{ent}$ measures the similarity between the candidate entities ($K_{ent} = K_{e_1} + K_{e_2}$) where $K_{e_1}$ is the kernel for the first entity, and $K_{e_2}$ is the kernel for the second entity. The final entity sequence kernel is $K_{es} = K_{rel} + K_{ent}$.

Several features are used in computing $sim(s, t)$ such as original word, stem, POS, chunk information, dependency and WordNet hypernym features. Various preprocessing steps (sentence detection, POS tagging, chunking) are performed using the JTextPro[3] package. Rita.WordNet[4] is used as the WordNet library to compute the similar hypernyms between words. Stanford Dependency Parser[5] is utilized to extract the dependency features. Often the entity sequences are just sequences of words which are non-grammatical as an utterance. Consequently, a parser will behave unexpectedly while parsing these sequences. Thus, we ran the Stanford Parser over the original sentences instead of the entity sequences. The grammatical relation with the *governing* token is used as a feature for the words. All the experiments are conducted using the LibSVM (Chang and Lin, 2001) package customized to augment the entity sequence kernel. The decay factor $\lambda$ was set to 0.5 empirically. To reduce the data imbalance problem the cost factor $W_i$ was set to be the ratio between the number of negative and positive examples.

We have also performed an initial set of experiments using the shortest path dependency

[3] http://jtextpro.sourceforge.net/

[4] http://www.rednoise.org/rita/wordnet/documentation/

[5] http://nlp.stanford.edu/software/lex-parser.shtml

| Approach | Direction | Kill | | | Located In | | | Live In | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| BL | $e_1 \rightarrow e_2$ | 65.5 | 73.5 | 69.3 | 74.1 | 67.6 | 70.7 | 59.5 | 61.7 | 60.6 |
| BL | $e_2 \rightarrow e_1$ | 87.1 | 70.1 | 77.6 | 68.0 | 32.4 | 43.9 | 80.5 | 40.9 | 54.2 |
| BL + WN | $e_1 \rightarrow e_2$ | 68.8 | 76.7 | 72.5 | 82.5 | 79.4 | 80.9 | 60.8 | 62.2 | 61.5 |
| BL + WN | $e_2 \rightarrow e_1$ | 92.6 | 64.6 | 76.6 | 71.0 | 29.7 | 41.7 | 84.5 | 41.2 | 55.4 |
| BL + Dep + WN | $e_1 \rightarrow e_2$ | 75.0 | 76.7 | 75.9 | 81.6 | 79.3 | 80.4 | 60.5 | 60.7 | 60.7 |
| BL + Dep + WN | $e_2 \rightarrow e_1$ | 94.6 | 66.7 | 78.1 | 73.3 | 29.6 | 42.2 | 85.9 | 41.3 | 55.8 |

| Approach | Direction | OrgBased In | | | Work For | | |
|---|---|---|---|---|---|---|---|
| | | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| BL | $e_1 \rightarrow e_2$ | 56.1 | 42.7 | 48.5 | 70.2 | 72.6 | 71.4 |
| BL | $e_2 \rightarrow e_1$ | 79.7 | 77.4 | 78.6 | 83.8 | 45.5 | 59.0 |
| BL + WN | $e_1 \rightarrow e_2$ | 69.9 | 60.5 | 64.9 | 67.3 | 72.4 | 69.8 |
| BL + WN | $e_2 \rightarrow e_1$ | 88.1 | 80.8 | 84.3 | 87.5 | 52.0 | 65.2 |
| BL + Dep + WN | $e_1 \rightarrow e_2$ | 54.2 | 44.8 | 49.1 | 69.5 | 75.9 | 72.6 |
| BL + Dep + WN | $e_2 \rightarrow e_1$ | 83.0 | 80.8 | 81.9 | 87.8 | 54.1 | 67.0 |

Table 2: Performance of entity sequence kernels ($K_{es}$) on both relation directions.

kernel of Bunescu and Mooney (2005b) modified for our settings. We have modified the Entity Sequence Kernel to use only those words that occur on the shortest dependency path between the mentioned entities. Using the Stanford Dependency Parser we create the shortest path between two entities ($e_i$ and $e_j$) in the undirected version of the dependency graph. Thus an entity sequence - shortest path ($ES_{SP}$) is defined as $ES_{SP} = entity1 + [sp] + entity2$, where $[sp]$ represents the words which appear on the shortest dependency path between entity1 and entity2. We define the shortest path entity sequence kernel $K_{es\_sp} = K_{rel\_sp} + K_{ent}$, where $K_{rel\_sp}$ is based on the words present in $[sp]$. In Table 3, we notice that for three out of the five relations, $K_{es\_sp}$ kernel outperforms the original $K_{es}$ kernel.

## 4 Results and Discussion

Table 2 presents the results for each of the five relations, including directionality ($e_1 \rightarrow e_2$ vs $e_2 \rightarrow e_1$). All scores are averaged over a 5-fold cross validation set. *BL* denotes baseline features (word, stem, chunk, POS), *Dep* is the dependency feature and *WN* is the WordNet hypernym similarity. For three relations (*Kill*, *Work_For* and *Live_In*) the $e_1 \rightarrow e_2$ relations have a higher recall but lower precision where the $e_2 \rightarrow e_1$ have significantly higher precision for all the relations. For the *Kill* relation, there are 191 examples of $e_1 \rightarrow e_2$ direction and only 77 examples of $e_2 \rightarrow e_1$ (Table 1). This imbalance in number explains the general trend to express a *Kill* relation in text. The entity order $e_1 \rightarrow e_2$ (*Oswald killed Kennedy*) is more common than $e_2 \rightarrow e_1$ (*Kennedy was killed by Oswald*).

In addition, a larger number of negative examples are created for $e_1 \rightarrow e_2$ relations than for the $e_2 \rightarrow e_1$ relations. For *Kill* relation there are around 650 negative examples for the direction $e_1 \rightarrow e_2$ , i.e., 70% of all the negative sequences. These negative examples are similar in syntactic structure to the positive examples, which leads the classifier to misclassify the negative examples as $e_1 \rightarrow e_2$. This explains the low precision. In addition, from the perspective of a sequence kernel, it considers all possible subsequences for matching, implementing a partial (fuzzy) matching. Table 2 for $e_1 \rightarrow e_2$ represents the effect of disjoint feature scopes of every features (POS, Chunk, Dep, WordNet). Each features adds up and expands the feature scope of the sequence kernels by allowing fuzzy matching, which in turn improve the recall. For the $e_2 \rightarrow e_1$ direction, the number of negative examples is small and thus there are fewer false

| Approach | Kill | | | Located In | | | Live In | | |
|---|---|---|---|---|---|---|---|---|---|
| | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| $K_{es}$ BL + WN | 80.7 | 70.7 | 75.4 | 77.7 | 51.6 | 62.1 | 72.7 | 51.7 | 60.5 |
| $K_{es}$ BL + Dep + WN | 84.8 | 71.7 | 77.8 | 77.4 | 54.5 | 64.0 | 73.2 | 52.0 | 61.0 |
| $K_{es_{sp}}$ BL + WN | 80.0 | 73.7 | 76.7 | 65.9 | 71.1 | **68.4** | 65.1 | 58.5 | **61.6** |
| $K_{es_{sp}}$ BL + Dep + WN | 82.0 | 75.1 | **78.4** | 65.5 | 69.6 | 67.5 | 64.9 | 56.3 | 60.3 |
| KM10 Pipeline | 91.1 | 61.2 | 73.1 | 71.5 | 57.0 | 62.3 | 68.1 | 56.6 | 61.7 |
| KM10 CardPyramid | 91.6 | 64.1 | 75.2 | 67.5 | 56.7 | 58.3 | 66.4 | 60.1 | **62.9** |
| RY07 Pipeline | 73.0 | 81.5 | 76.5 | 52.5 | 56.4 | 50.7 | 58.9 | 50.0 | 53.5 |
| RY07 Joint | 77.5 | 81.5 | 79.0 | 53.9 | 55.7 | 51.3 | 59.1 | 49.0 | 53.0 |
| G10 $M_C \vert K^*_{SL}$ | 82.5 | 77.2 | **79.8** | 78.1 | 59.0 | **67.2** | 71.8 | 53.4 | 61.2 |

| Approach | OrgBased In | | | Work For | | |
|---|---|---|---|---|---|---|
| | Prec. | Rec. | $F_1$ | Prec. | Rec. | $F_1$ |
| $K_{es}$ BL + WN | 79.0 | 70.7 | **74.6** | 77.4 | 62.2 | 69.0 |
| $K_{es}$ BL + Dep + WN | 68.6 | 62.8 | 62.3 | 78.7 | 65.0 | **71.2** |
| $K_{es_{sp}}$ BL + WN | 68.4 | 69.1 | 68.7 | 71.7 | 65.2 | 68.3 |
| $K_{es_{sp}}$ BL + Dep + WN | 68.8 | 68.8 | 68.8 | 74.3 | 65.1 | 69.4 |
| KM10 Pipeline | 70.6 | 60.2 | 64.6 | 74.1 | 66.0 | 69.7 |
| KM10 CardPyramid | 66.2 | 64.1 | 64.7 | 73.5 | 68.3 | 70.7 |
| RY07-Pipeline | 77.8 | 42.1 | 54.3 | 60.8 | 44.4 | 51.2 |
| RY07 Joint | 79.8 | 41.6 | 54.3 | 72.0 | 42.3 | 53.1 |
| G10 $M_C \vert K^*_{SL}$ | 68.3 | 61.5 | 64.7 | 75.4 | 67.1 | **71.0** |

Table 3: Comparison with existing state-of-the-art systems (Average F1)

positives, which explains the higher precision. But since there are a small number of training examples (for a 5-fold cross validation, each fold has an average of 60 training instance), the recall is low. In the case of *OrgBased_In* and *Located_In* we observe two outcomes. First, for $e_1 \rightarrow e_2$ the precision is little higher than the recall, unlike the other three relations. Even if we add features in the sequence kernel the concept of fuzzy matching is not helping to improve the recall for these two relations. We notice that a lot of of positive examples for these two relations are connected either by a single word or by punctuation. Consider the following examples, a) *[Havana]$e_i$ [Radio Rebelda Network]$e_j$ in spanish GMT...*, and b) *[George Lucas]$e_i$ , a [Modesto]$e_j$*. Even if we add POS, Chunk, Dep features besides the word features, it will not help the classifier to improve the recall as there is not much useful information available. A single letter token like *a* is like a stop-word and does not help in classification. Thus, the recall does not change for these cases. Second, we notice that the dependency feature is not contributing much to these two relations. Stanford Parser does not recognize punctuation as relation markers. To test our hypothesis we observe that for these two relations (containing shorter sequences) the original gap sequence word kernel performs close to the baseline kernel (around 58% F1 for *Located_In* and around 62.1% for *OrgBased_In*). However, for *OrgBased_In* we achieve a very high precision at the same time. WordNet hypernym help to match non-obvious terms like (*Federation* and *Nation*), (*Citizen* and *National*). In the case of $e_2 \rightarrow e_1$ for *Located_In*, entities are mostly linked via the "possession" type of dependency relation. However, a lot of negative examples are linked like that; so by adding the dependency feature for $e_2 \rightarrow e_1$, we observe that the precision slightly decreases.

In order to compare the results with state of the art systems (Kate and Mooney, 2010; Roth and Yih, 2007; Giuliano et al., 2007), Table 3 shows the average scores of the relation directions (however our folds of the 5-fold cross validation are not the same as their folds which were not available). For Entity Sequence Kernels we present the results of *BL+WN* and *BL+WN+Dep*.

For Roth and Yih (2007), we report the results they obtain using their most sophisticated model which they call "E $\leftrightarrow$ R" (RY07 in Table 3). For Kate and Mooney (2010) we show both the results of their card-pyramid method that performs joint modeling of entities and relations, and their pipeline approach (KM10 in Table 3). For (Giuliano et al., 2007) (G10 in Table 3) we use the results of their $M_C|K^*_{SL}$ model which is the closest to ours (it uses entity boundaries but no entity types during training). Our method performs the best for three out of the five relations (*OrgBased_In*, *Work_For*, *Located_In*). For the other two relations our average F1 is very close to the best results, being 3rd for *Kill*, 2nd for *Live_In*.

## 5 Related Work

Recently there has been a lot of research on relation extraction using kernel methods. In this section we review mainly two lines of work closely related to ours.

In Section 4 we have introduced several state-of-the-art approaches to relation extraction which have used the RY corpus. Roth and Yih (2007) have adopted an integer linear programming framework for joint extraction of entity and relations. Kate and Mooney (2010) have implemented a card-pyramid parsing technique where each candidate sentence is represented as a binary directed graph. The entities are placed on the leaf nodes and relations are on the higher levels in the graph. Giuliano et al. (2007) et al. have studied the relation extraction problem using a pipeline architecture, similar in nature to our approach but using linear kernel with only basic features. They ran an independent NER to recognize the entities in the sentences and used these new recognized entities as possible entity mentions for the relation extraction. However, we have not conducted any NER experiments to recognize entities and thus have used the available correct boundaries of entities in our research. Both Kate and Mooney (2010) and Giuliano et al. (2007) have mentioned the directionality issue of the relations but they only presented the micro-average F1 scores.

In terms of methodology, the closest approaches to ours are the ones using sequence kernels for relation extraction. Inspired by the string kernel of Lodhi et al. (2002), Bunescu and Mooney (2005a) created subsequence patterns between entities to extract top-level relations from the ACE dataset. In our work we use entity sequence kernes, and only consider the entity boundaries as given, and not entity types as in (Bunescu and Mooney, 2005a). Bunescu and Mooney (2005b) present a shortest path (between the entities) dependency tree kernel and evaluate it on the ACE 2002 dataset. However, as pointed out by (Giuliano et al., 2007) due to the varied datasets (e.g. ACE, SemEval) employed for these research it is a hard task to compare one against another. The generic trend is usually similar — sequence kernels have more flexibility and thus gap sequence kernels find similar subsequences and often results in a higher recall (Wang, 2008).

## 6 Conclusion

We have presented an approach for relation extraction using semantic and syntactic features augmented with an entity sequence kernel. To the best of our knowledge, this paper presents the first in depth study of how the *order* of the candidate entities influences *relation directionality* and how various factors might contribute to the accuracy of results for each relation direction. Our proposed entity sequence kernel outperforms state-of-the-art methods for three out of the five relations under study. We plan to further explore the shortest path dependency kernel with different kernel combination schemes in future work.

## References


Bunescu, R. and Mooney, R. J. (2005a). Subsequence kernels for relation extraction. Proceedings of the 19th Conference on Neural Information Processing Systems (NIPS).

Bunescu, R. C. and Mooney, R. J. (2005b). A shortest path dependency kernel for relation extraction. Vancouver, BC. Proceedings of the Joint Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP).

Chang, C.-C. and Lin, C.-J. (2001). Libsvm: a library for support vector machines.

Giuliano, C., Lavelli, A., and Romano, L. (2007). Relation extraction and the influence of automatic named-entity recognition. *ACM Transactions on Speech and Language Processing (TSLP)*, 5(2).

Kate, R. J. and Mooney, R. J. (2010). Joint entity and relation extraction using card-pyramid parsing. Number 203-212, Uppsala, Sweden. Proceedings of the Fourteenth Conference on Computational Natural Language Learning (CoNLL-2010).

Lodhi, H., Saunders, C., Taylor, J. S., Christianini, N., and Watkins, C. (2002). Text classification using string kernels. *Journal of Machine Learning Research*, 2:419–444.

Roth, D. and Yih, W. (2004). A linear programming formulation for global inference in natural language tasks. *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 1–8.

Roth, D. and Yih, W. (2007). Global inference for entity and relation identification via a linear programming formulation. *Introduction to Statistical Relational Learning*.

Wang, M. (2008). A re-examination of dependency path kernel for relation extraction. In Proceedings of IJCNLP.

Zhang, M., Zhoua, G., and Aw, A. (2008). Exploring syntactic structured features over parse trees for relation extraction using kernel methods. *Information Processing and Management*, 44(2):687–701.