A Labeled Graph Kernel for Relationship Extraction

Gonçalo Simões
INESC-ID / IST, PT
goncalo.simoes@ist.utl.pt

David Matos
INESC-ID / IST, PT
david.matos@inesc-id.pt

Helena Galhardas
INESC-ID / IST, PT
helena.galhardas@ist.utl.pt

ABSTRACT
In this paper, we propose an approach for Relationship Extraction (RE) based on labeled graph kernels. The kernel we propose is a particularization of a random walk kernel that exploits two properties previously studied in the RE literature: (i) the words between the candidate entities or connecting them in a syntactic representation are particularly likely to carry information regarding the relationship; and (ii) combining information from distinct sources in a kernel may help the RE system make better decisions. We performed experiments on a dataset of protein-protein interactions and the results show that our approach obtains effectiveness values that are comparable with the state-of-the-art kernel methods. Moreover, our approach is able to outperform the state-of-the-art kernels when combined with other kernel methods.

Categories and Subject Descriptors
H.2 [Information Storage and Retrieval]: Content Analysis and Indexing - Linguistic Processing

General Terms
Algorithms

Keywords
Information Extraction, Machine Learning, Graph Kernels

1. INTRODUCTION
With the increasing use of Information Technologies, the amount of unstructured text available in digital data sources (e.g., email communications, blogs, reports) has grown at an impressive rate. These texts may contain vital knowledge to Human decision making processes. However, it is unfeasible for a human to analyze big amounts of unstructured information in a short time. In order to solve this problem, a typical approach is to transform unstructured information in digital sources into a previously defined structured format.

Information Extraction (IE) is the scientific area that studies techniques to extract semantically relevant segments from unstructured text and represent them in a structured format that can be understood/used by humans or programs (e.g., decision support systems, interfaces for digital libraries). In the past few years, there has been an increasing interest in IE, from industry and scientific communities. In fact, this interest led to huge advances in this area and several solutions were proposed in applications such as Semantic Web and Bioinformatics.

Regardless of the application domain, an IE activity can be modeled as a composition of the following high-level tasks:

- **Segmentation**: divides the text into atomic segments (e.g., words).
- **Entity recognition**: assigns a class (e.g., organization, person) to each segment of the text. Each pair (segment, class) is called an entity.
- **Relationship extraction**: determines relationships (e.g., born_in, works_for) between entities.
- **Entity normalization**: converts entities into a standard format (e.g., convert all dates to a pre-defined format).
- **Co-reference resolution**: determines which entities represent the same object/individual in the real world (e.g., IBM is the same as “Big Blue”).

In the last decade, several techniques to increase the accuracy of these tasks were proposed. In this paper, we focus only on the Relationship Extraction (RE) task. The approaches that are typically used for RE can be divided into two major groups: (i) handcrafted solutions, in which the programs are manually specified by the user through a set of rules; and (ii) Machine Learning solutions, in which the programs are automatically generated by a machine either by explicitly producing rules or by generating a statistical model that is able to produce extraction results with regard to a set of characteristics of the input text.

Most of the first approaches for RE were based on handcrafted rules. Typically, they exploited common patterns and heuristics to extract the desired relationships from
the results of complex Natural Language Processing chains. These solutions were able to produce good results in several specific domains. However, they need a lot of human effort to produce rules for distinct domains.

To overcome this problem of handcrafted solutions, the application of Machine Learning to RE started to receive a lot of attention. Typically, machine learning techniques used for RE are supervised. However, some works have exploited semi-supervised [6][11][13] and unsupervised [10][12] techniques. Supervised approaches to RE are typically based on classifiers that are responsible for determining whether there is a relationship or not between a set of entities.

There are two major lines of works in supervised approaches to RE: (i) feature-based methods, which try to find a good set of features to use in the classification process; and (ii) kernel methods, which try to avoid the explicit computation of features by developing methods that are able to compare structured data (e.g., sequences, graphs, trees). Even though feature-based methods for RE work well [16], there has been an increasing interest in exploiting kernel-based methods, due to the fact that sentences are better described as structures (e.g., sequences of words, parsing trees, dependency graphs).

In this paper, we describe a new supervised approach to RE that is based on labeled dependency graph representations of the sentences. The advantage is that a representation of a sentence as a labeled dependency graph contains rich semantic information that, typically, contains useful hints when discriminating whether a set of entities in a sentence are related. The solution we propose uses kernels to deal with these structures. We propose the application of a marginalized kernel to compare labeled graphs [17]. This kernel is based on walks on random graphs and is able to exploit an infinite dimensional feature space by reducing its computation to the problem of solving a system of linear equations. In order to make this graph kernel suitable for RE, we modified the kernel to exploit the following properties that were previously introduced proposals of kernels for RE: (i) the words between the candidate entities or connecting them in a syntactic representation are particularly likely to carry information regarding the relationship [7]; and (ii) combining information from distinct sources in a kernel may help the RE system make better decisions [14].

In order to evaluate the model we propose, we performed some experiments with a biomedical dataset called Almed [8]. This dataset is composed of several abstracts from Biology papers. The documents are annotated with interaction relationships between proteins. The results show that the performance of our approach is comparable to the state-of-the-art. Morever, when combining our kernel with other kernel methods, we were able to outperform other state-of-the-art kernel methods.

The rest of the paper is organized as follows. In Section 2 we present the related work. Section 3 defines the problem that we are trying to solve. In Section 4 we describe our method for relationship extraction. In Section 5 we report on the experiments performed. Finally, Section 6 presents the conclusions and some topics for future work.

2. RELATED WORK

The most relevant works in the topic of this paper are the ones that propose kernel methods for RE. In the past ten years, several authors proposed kernels for different syntactic and semantic structures of a sentence. One of the first approaches, presented in 2003 by Zelenko et al. [20], is a kernel based on shallow parse tree representation of sentences. This approach had some problems in what concerns the vulnerability to parsing errors. In order to overcome these problems, Culotta and Sorensen [9] proposed a generalization of this kernel that, when combined with a bag-of-words kernel, is able to compensate the parsing errors.

In 2005, Bunescu and Mooney [7] proposed a kernel based on the shortest path between entities in a dependency graph. The kernel was based on the hypothesis that the words between the candidate entities or connecting them in a syntactic representation are particularly likely to carry information regarding the relationship. The problem of this kernel is the fact that it is not very flexible when comparing candidates, which leads to very low values of recall when the training data is too small. The same authors proposed a different kernel based on subsequences [8]. The subsequences used in this approach could be combinations of words and other tags (e.g., POS tags, Wordnet Synsets). The results of this kernel are very interesting and even today it is still pointed out as a kernel with a very good performance in RE tasks.

Giuliano et al. [14] proposed in 2006 a kernel based only on shallow linguistic information of the sentences. The idea was to exploit two simple kernels that, when combined, were able to obtain very interesting results. The global context kernel compares the whole sentence using a bag-of-n-grams approach. The frequencies of the n-grams are computed in three different locations of the sentence: (i) before the first entity; (ii) between the two entities; and (iii) after the second entity. The local context kernel evaluates the similarity between the entities of the sentences as well as the words in a window of limited size around them. The advantage of this kernel is its simplicity since it does not need deep Natural Language Processing tools to preprocess the sentences in order to compute the kernel. However, its major advantage may very well be a big disadvantage since it is not able to exploit rich syntactic/semantic information like a parsing tree or a dependency graph representation of a sentence (which are structures that can be useful for determining whether a set of entities are related).

In 2008, Airola et al. [2] presented a kernel that combines two graph representations of a sentence: (i) a labeled dependency graph; and (ii) a linear order representation of the sentence. The kernel considers all possible paths connecting any two vertices in the graph. The results obtained are comparable with the state-of-the-art results. However, this kernel is very demanding in terms of computational resources.

In 2010, Tikk et al. [19] performed a study to analyze how a very comprehensive set of kernels for relationship extraction performs when dealing the task of extracting protein-protein interactions. Even though they were not able to determine a clear winner in their comparison, they were still able to outline some very interesting conclusions. First, they notice
that kernels based on dependency parsing tend to obtain better results than kernels based on tree parses. Moreover, they show that a simple kernel, like \[14\], can still obtain results that are at the level of the best kernels based on dependency parsing.

### 3. PROBLEM DEFINITION

In general, the problem of finding an \( n \)-ary relationship between entities can be seen as a classification problem for which the input is a set of \( n \) entities and the output is the type of relationship between them or an indication that they are not related at all.

With this definition, given a text document with all the entities identified, the candidate results are all the sets of \( n \) entities that exist in the text. This approach would generate a huge set of candidates among which very few correspond to actually related entities. For this reason, this configuration would potentially lead to some performance issues (due to the huge amount of candidates) and to some problems in terms of accuracy (due to the unbalancement of the data).

To avoid these issues, we exploit an heuristic that is typically used in related works, which consists in limiting the candidates to sets of entities that can be found in the same sentence.

This way, for one sentence with \( k \) entities, the number of candidates generated for a \( n \)-ary relationship is given by the number of combinations of the \( k \) entities, selected \( n \) at a time, i.e. \( \binom{k}{n} \). For instance, consider the sentence in Figure 1, in which we present an example of a sentence from a biomedical text. Suppose that we aim at finding interaction relationships between proteins. This sentence contains three identified proteins: \( TRADD \), \( RIP \) and \( Fas \). Moreover, there are two interaction relationships between these entities: \( TRADD \) interacts with \( RIP \) and \( RIP \) interacts with \( Fas \).

Given the fact that a protein interaction is a binary relationship, we have a total of \( \binom{3}{1} = 3 \) candidates, which are presented in Figure 2.

Note that it is also possible to use other heuristics to reduce the number of candidates. For instance, in some cases, we may have knowledge about the types of entities that can fulfill a given role in a relationship (e.g. in a relationship between a company and its CEO, it is known that one of the entities must be a a company and the other, a person). Even though these heuristics typically involve some type of prior knowledge about the application domain, they tend to drastically reduce the space of candidates. This fact makes the relationship extraction process a lot easier and helps it produce better results since some of the candidates involving entities that are never related are not used.

Assuming a set of candidate results, Figure 3 describes how the RE extraction task can be represented as a classification problem. The problem can be divided into two main phases: training and execution. In the training phase, the objective is to automatically generate a statistical model that is able to determine whether a given candidate corresponds to a relationship. In order to produce this model, some training examples must be provided to a learning algorithm (e.g., solving a quadratic optimization problem in the case of a SVM classifier). These examples are generated in the same fashion as the candidates, however, they include an additional label that indicates whether they correspond to a relationship.

The execution phase aims at classifying each unlabeled candidate from new untagged documents as containing a relationship or not. This decision is made using the statistical model created in the training phase and a classification algorithm. In the end of the process, the sets of entities in the candidates that are classified as containing a relationship are returned.

### 4. METHOD

In this Section, we present the proposed kernel method. We start by describing the basic idea behind kernel methods for RE in Section 4.1. Then, in Section 4.2, we propose a representation of the candidate sentences as labeled graphs. In Section 4.3, we explain the random walk kernel that was used as the basis for our RE kernel. In Section 4.4, we present the parameters used to modify the random walk kernel for our problem. Finally, in Section 4.5, we propose our kernel for RE.

#### 4.1 Kernel Methods for Relationship Extraction

In some cases, input objects of a classifier may not be easily expressed via feature vectors (e.g., if the range of possible
features is too wide or if the nature of the object does not make it clear how to choose the features). Therefore, the feature engineering process may become painfully hard and lead to high-dimensional feature spaces and consequently to computational problems. Kernel methods are an alternative to feature-based methods that can be used to classify objects while keeping their original representation.

In kernel methods, the idea is to exploit a similarity function (kernel) between input objects. This function, with the help of a discriminative machine learning technique, is used to classify new examples. In order for a similarity function to be an acceptable kernel function, \( K(x, y) \), it must respect the following properties: (i) it must be a bidimensional function over the object space \( X \times X \rightarrow [0, +\infty] \); (ii) it must be symmetric \( (\forall x, y \in X, K(x, y) = K(y, x)) \); and (iii) it must be positive-semidefinite \( (\forall x_1, x_2, \ldots, x_n \in X, \text{the } n \times n \text{ matrix } (K(x_i, x_j))_{ij} \text{ is positive-semidefinite}) \).

RE is an example of a problem for which the inputs may not be easily expressed via feature vectors. As described in Section 3, the inputs of the learning and classification algorithms in supervised RE tasks are sentences. Typically, sentences are better described as structures (e.g., sequences of words, parsing trees, dependency graphs) and it is interesting to use these representations directly.

### 4.2 Labeled Graph Representation of the Sentences

In our approach, we assume that the inputs of the learning and classification algorithms are labeled graph representations of the candidate sentences (see Figure 4). In this graph, each vertex is associated with a word in the sentence and is enriched with additional features of the word. In our representation, the additional features include POS tags, generic POS tags, the lemma of the word and capitalization patterns (however, due to simplicity, we represent only one additional feature in the graph of Figure 4, which is the POS tag). We could use other potentially useful features like hypernyms or synsets extracted from the WordNet. The edges represent semantic relationships between the words. The type of the semantic relationship is represented by the edge label.

Recall that, for a given sentence with \( k \) entities, when searching for a n-ary relationship, the number of candidates that are generated is \( \binom{k}{n} \). In terms of structure (vertexes and edges), the corresponding dependency graph for each of these candidates is always the same. If we used only structural information to compare candidates we could have a problem because we would not be able to distinguish between different candidates generated from the same sentence that are expected to produce different classification results.

For this reason, we used heuristics to enrich our graph representation. First, the entities that are candidate to be related can provide very important clues for detecting if there is a relationship. We define a predicate \( isEntity(v) \), which receives a vertex of the graph and determines whether it is an entity. With this, it is possible for a kernel to use this information in the computation of the similarity between graphs. Second, the shortest path hypothesis, formalized in [7], states that the words between the candidate entities or connecting them in a syntactic representation are particularly likely to carry information regarding their relationship. Analogously to [7] and [2], we exploited this hypothesis by defining a predicate called \( inSP(x) \) that receives as input a node or an edge of the graph and returns true if they belong to the shortest path between the two entities of the graph. Like in the case of the entities, this allows the kernel to treat these vertexes and edges in a special fashion way.

### 4.3 Random Walks Kernel

The random walk kernel used as a basis of our RE kernel was defined in [17] as a marginalized kernel between labeled graphs. The basic idea behind this kernel is the following one: given a pair of graphs, perform simultaneous random walks between the vertexes of the graphs and count the number of matching paths. In a more formal way, the objective of the kernel is to compute the expected number of matching paths between the two graphs.

In order to explain this kernel, we start by defining the graph that is expected as input. Let \( G \) be a labeled directed graph and \( |G| \) be the number of vertexes in the graph. All vertexes in the graph are labeled and \( v_i \) denotes the label of vertex \( i \). The edges of the graph are also labeled and \( e_j \) denotes the label of the edge that connects vertex \( i \) and vertex \( j \). Moreover, we assume two kernel functions, \( K_e(v, v') \) and \( K_e(e, e') \) that are kernel functions between vertexes and edges respectively. Figure 5 presents an example of a graph that can be used as input of the random walk kernel.

Additionally to the graph, this kernel also assumes the existence of three probability distributions: (i) the initial probability distribution, \( p_h(h) \), that corresponds to the probability that a path starts in the vertex \( h \); (ii) the ending probability, \( p_h(h) \), that corresponds to the probability that a path ends in the vertex \( h \); and (iii) the transition proba-
Computing this kernel using a naive approach (i.e., going through all the possible pairs of paths in the kernels), would be computationally expensive for acyclic graphs and impossible for graphs containing cycles. However, [17] demonstrated that this kernel can be efficiently computed by solving a system of linear equations. In order to define this system of linear equations, let us first define the following matrices:

\[ S = \begin{pmatrix} s(1,1') & s(1,2') \\ \vdots & \ddots \\ s(1,|G'|) & s(2,1') \\ \vdots & \ddots & \ddots \\ s(|G|,1') & s(|G|,2') \end{pmatrix}, \quad Q = \begin{pmatrix} q(1,1') \\ q(1,2') \\ \vdots \\ q(|G|,1') \\ q(|G|,2') \end{pmatrix} \]

\[ T = \begin{pmatrix} t(1,1',1,1') & t(1,1',1,2') & \cdots & t(1,1',|G|,|G'|) \\ t(1,1',2,1') & t(1,1',2,2') & \cdots & t(1,2',|G|,|G'|) \\ \vdots & \vdots & \ddots & \vdots \\ t(|G|,|G'|,1,1') & t(|G|,|G'|,1,2') & \cdots & t(|G|,|G'|,|G'|) \end{pmatrix} \]

Where

\[ s(h_1, h'_1) = p_r(h_1)p_r(h'_1)K_v(v_{h_1}, v'_{h'_1}) \]  \hspace{1cm} (4)

\[ q(h_i, h'_i) = p_q(h_i)p_q(h'_i) \]  \hspace{1cm} (5)

\[ t(h_{i-1}, h'_{i-1}, h_i, h'_i) = p_r(h_{i-1})p_r(h'_i)K_v(v_{h_{i-1}}, v'_{h'_i}) \times K_v(v_{h_i}, v'_{h'_i})K_e(e_{h_{i-1}h_i}, e'_{h'_ih'_i}) \]  \hspace{1cm} (6)

The system of linear equations that we need to solve is presented in Equation (7):

\[ (I - T)X = Q \]  \hspace{1cm} (7)

where X is the solution of the system and I is the identity matrix. [17] demonstrated that the random walk kernel between graphs, \( K(G, G') \), can be given by Equation (8):

\[ K(G, G') = \langle S, X \rangle \]  \hspace{1cm} (8)

where \( \langle S, X \rangle \) is the inner product between two vectors.
4.4 Parameters of the Random Walks Kernel for Relationship Extraction

In Section 4.3, we described a kernel for generic labeled graphs. The kernel we propose is a particularization of this one applied to RE.

Recall that our representation of a sentence, presented in Section 4.2, corresponds to a labeled graph where the labels of the vertexes are vectors of tags (containing the word itself, its lemma, POS tags, and orthographic patterns) and the labels of the edges contain simply the type of the semantic relationship between the two entities. Moreover, each vertex and edge contains information about whether it is in the shortest path between the two entities. The vertexes also contain information about whether they are entities.

In order to use the random walk kernel described in Section 4.3, we had to define the kernels between the vertex labels and the kernels between the edge labels. Given the fact that the labels of the vertexes are simply vectors of attributes of the word associated with the vertex, we can use the normalized linear kernel presented in Equation 9:

\[
K_v(v, v') = \frac{c(v, v')}{\sqrt{c(v, v) c(v', v')}}
\]

where \(c(v, v')\) counts the number of common features between the labels of \(v\) and \(v'\).

In order to guarantee that entities can only match in a random walk with other entities and that vertexes contained in a shortest path can only match with vertexes contained in a shortest path, we actually used a slightly modified version of the kernel presented in Equation 9. The modified version is presented in Equation 10:

\[
K_v(v, v') = \begin{cases} 
\frac{c(v, v')}{\sqrt{c(v, v) c(v', v')}} & \text{if } inSP(v) = inSP(v') \wedge isEntity(v) = isEntity(v') \\
0 & \text{otherwise}
\end{cases}
\]

The kernel between the edges is very simple. Since the label for the edges is only a string indicating the type of semantic relationship between the two words, we define this kernel in Equation 11:

\[
K_e(e, e') = \delta(e = e')
\]

where, \(\delta\) is a function that returns 1 if its argument holds and 0 otherwise.

Once again, since we want to differentiate edges in the shortest path from edges outside the shortest path, we added a simple modification to the kernel that is presented in Equation 12:

\[
K_e(e, e') = \begin{cases} 
\delta(e = e') & \text{if } inSP(e) = inSP(e') \\
0 & \text{otherwise}
\end{cases}
\]

Finally, we still need to define the probability distributions necessary to compute the random walk kernel in our problem. Due to the fact that we have no prior knowledge about the probability distributions, we follow the solution proposed in [17] and consider that all the distributions are uniform.

4.5 Random Walks Kernel for Relationship Extraction

Using the random walk kernel presented in Section 4.3, we produced three variations of the kernel: (i) Full Graph Kernel; (ii) Shortest Path Kernel; and (iii) No Shortest Path Kernel.

The Full Graph Kernel (FGK) corresponds to the application of the random walk kernel to the whole structure described in Section 4.2. The idea of this kernel is to capture the whole view of the graph structure (which is the same for all the candidates generated from a given sentence) but still be able to capture the similarity between interesting properties that are specific to the candidates (i.e., shortest path and entities information).

The Shortest Path Kernel (SPK) aims at exploiting the shortest path hypothesis presented in [7]. The idea is to apply the random walk kernel to the subgraph that corresponds to the shortest path between the entities.

The No Shortest Path Kernel (NSPK) is a variation of FGK where the nodes and edges that belong to the shortest path are not marked as such. For this reason, the only thing that distinguishes the graph structures for candidates generated from a given sentence are the entities.

The kernel we propose is actually based on a very interesting property of kernels: the linear combination of several kernels is itself a kernel. We used this approach because several works empirically demonstrated that combining kernels using this approach typically improves the performance of individual kernels [9, 14].

5. EXPERIMENTS

In this Section, we present the experiments performed in order to evaluate our solution for RE and report on the results obtained. First, we present the relationship extraction task. Then, in Section 5.2, we describe the dataset. Section 5.3 presents the metrics used to evaluate our kernel and Section 5.4 presents the method used to support our claims in what concerns the comparison of the kernels. In Section 5.5, we point out some implementation details of our experiments. In Section 5.6, we report on the performance of the individual kernels presented in Section 5.5, and in Section 5.7, we report on the combination of these kernels. In Section 5.8, we perform a comparison between our solution and other methods. Finally, in Section 5.9, we report on some experiments when combining our kernel with other methods.

5.1 Relationship Extraction Task

In our evaluation, we focused exclusively on the extraction of relationships that correspond to protein-protein interactions. The idea is that, given pairs of entities there is a
relationship between them if the text indicates that the proteins have some kind of biological interaction.

5.2 Dataset
We performed our experiments over a protein-protein interaction dataset called Almed\(^1\). This dataset has been used in previous works to evaluate the performance of relationship extraction systems in the task of extracting protein-protein interactions \(^2\). Almed is composed by 225 Medline abstracts from which 200 describe interactions between proteins and the other 25 do not refer to any interaction. The total number of interacting pairs is 974 and the total number of non-interacting pairs is 4072.

During the evaluation of our model we used a cross-validation strategy that is based on splits of the Almed dataset at the level of document \(^3\). Table 1 presents the number of positive and negative candidates that can be found in the training and testing data of each split.

5.3 Evaluation Metrics
Our experiments are focused on measuring the quality of the results produced when using our kernel. In Information Extraction (and particularly in Relationship Extraction), the quality of the results produced is based on two metrics: recall and precision.

Recall gives the ratio between the amount of information correctly extracted from the texts and the information available in texts. Thus, recall measures the amount of relevant information extracted and is given by Equation 13:

$$\text{recall} = \frac{C}{P}$$  \(13\)

where \(C\) represents the number of correctly extracted relationships while \(P\) represents the total number of relationships that should be extracted. The disadvantage of this measure is the fact that it returns high values when we extract all possible pairs of entities as a relationship regardless of them being related or not.

Precision is the ratio between the amount of information correctly extracted from the texts and all the information extracted. The precision is then a measure of confidence on the information extracted and is given by Equation 14:

$$\text{precision} = \frac{C}{C + \alpha}$$  \(14\)

where \(C\) represents the number of relationships correctly extracted, \(I\) represents the number of relationships incorrectly extracted.

The disadvantage of precision is that we can get high results extracting only information that we are sure to be right and ignoring information that are in the text and may be relevant.

The values of recall and precision may enter in conflict. When we try to increase the recall, the value of precision may decrease and vice versa. The \(F\)-measure was adopted to measure the general performance of a system, balancing the values of recall and precision. It is given by Equation 15:

$$\text{F-measure} = \frac{(\beta^2 + 1) \times P \times R}{\beta^2 \times P + R}$$  \(15\)

where \(R\) represents the recall, \(P\) represents the precision, \(\beta\) is an adaptation value of the equation that allows to define the relative weight of recall and precision. The value \(\beta\) can be interpreted as the number of times that the recall is more important than accuracy. A value for \(\beta\) that is often used is 1, in order to give the same weight to recall and precision. In this case, the \(F\)-measure value is obtained through Equation 16:

$$F_1 = \frac{2 \times P \times R}{P + R}$$  \(16\)

5.4 Significance Tests
In order to support our claims during the comparison of each pair of kernels, we relied significance tests. We used a the paired t-test between each pair of kernels that we wanted to compare directly. Details about this significance test can be found on most statistics text books \(^4\).

For a given metric presented in Section 5.3, we give as input to the test the result obtained for each split of the dataset. Our claims are based on a significance level of 5%.

5.5 Implementation Details
Our experiments used the SVM package jLIBSVM\(^2\) a Java port of LIBSVM that allows for easy customization when using different kernels. During the experiments, we used most of the default parameters of jLIBSVM. The only exception was the parameter \(C\) of the SVM (which controls the trade-off between the errors of the SVM and the size of the margin). For this parameter, after some empirical experimentation we fixed its value in 50 for all the experiments.

We used the OpenNLP\(^3\) module for sentence detection and

---

1 ftp://ftp.cs.utexas.edu/pub/mooney/bio-data/interactions.tar.gz
2 http://dev.davidsoergel.com/trac/jlibsvm/
3 http://incubator.apache.org/opennlp/
4 http://incubator.apache.org/opennlp/
the Stanford parser\(^3\) for the word segmentation, POS tagging and generation of the labeled dependency graph.

Finally, we used Parallel Colt\(^4\) to perform the matrix operations necessary for our kernel.

### 5.6 Performance of the Individual Kernels

Our first experiment aimed at understanding how each of the individual kernels that we proposed (i.e., FGK, SPK and NSPK introduced in Section 4.5) performs. Table 2 shows the results of this experiment.

| Kernel  | Recall | Precision | \(F_1\) |
|---------|--------|-----------|--------|
| FGK     | 41.51% | 58.94%    | 48.25% |
| SPK     | 43.47% | 56.73%    | 48.86% |
| NSPK    | 37.69% | 58.47%    | 45.39% |

Table 2: Performance of the individual kernels on the AIMed data set.

The results obtained are according to what was expected. First, the individual kernel that obtains the highest value of \(F_1\) is SPK. Knowing how the shortest path hypothesis has been exploited with success in several other works, this comes with no surprise. Even though the average value of \(F_1\) for SPK is higher than that for FGK, the difference is not statistically significant according to the significance tests.

If we look only at the average values of recall and precision presented in Table 2, it seems that SPK is the best kernel in terms of recall and FGK is the best in terms of precision. However, by comparing the results obtained by these two kernels using the significance tests the differences are not significant for both these metrics.

Another result that is not surprising is the fact that the performance of NSPK is very poor. As discussed before, this kernel does not distinguish very well candidates that are generated from the same sentence but are associated with different pairs of entities. This reflects in a drastic drop of the recall value.

### 5.7 Performance of the Combination of Kernels

After analyzing the performance of the individual kernels, we evaluated the performance of the kernels that result from their combination. We considered the following four combinations: (i) FGK + SPK; (ii) FGK + NSPK; (iii) SPK + NSPK; and (iv) ALL = FGK + SPK + NSPK.

Table 3 shows the results of this experiment. Given the performance of the individual kernels reported before, it was expected that the best combination of kernels would be either the one that combines all the individual kernels (ALL) or the one that combines the two best individual kernels (FGK + SPK). In fact, the results show that regarding the average values of recall, precision and \(F_1\), the best combination is actually SPK + NSPK.

| Kernel | Recall | Precision | \(F_1\) |
|--------|--------|-----------|--------|
| FGK + SPK | 45.21% | 59.60%    | 51.83% |
| FGK + NSPK | 40.84% | 57.56%    | 47.34% |
| SPK + NSPK | 46.41% | 60.57%    | 52.31% |
| ALL    | 46.31% | 59.01%    | 51.64% |

Table 3: Performance of the individual kernels on the AIMed data set.

The explanation for this surprising result has to do with the definition of these kernels. On one side, SPK was designed as a good solution to distinguish between candidates generated from the same sentence and associated with different pairs of entities. On the other side, NSPK is good to analyze the whole structure of the dependency graph but it does not distinguish very well the candidates generated from the same sentence. Thus, these two kernels are good at distinguishing very different contexts of the candidates. For this reason, they end up being a good complement to each other.

Even though SPK + NSPK obtained the best average values of recall, precision and \(F_1\), it is important to note that according to the significance tests, it is not fair to claim that it is a superior solution in comparison to FGK + SPK and ALL since the differences for all the metrics were not statistically significant.

Another interesting observation has to do with the terrible results obtained by FGK + NSPK. It is the kernel combination with worst results in all the metrics. Moreover, the significance tests indicated that in terms of recall and \(F_1\) measure, the differences in comparison to the other combinations were significant. These results are also related with the type of information that the two individual kernels try to analyze. Recall that FGK is actually a modified and more refined version of NSPK in which vertexes and edges of the shortest path between the candidate entities are treated differently. For this reason, most of the information exploited by both kernels is the same, which makes their combination a little bit redundant.

Finally, we wanted to compare the combination kernels with the individual kernels to understand whether it pays off to use the combinations. For each metric, we compared the combination kernels with the individual kernel with the highest value of the metric as presented in Table 2. First, in what concerns recall, we observe that the differences between SPK and most of the combinations is not significant. The only exception is SPK + NSPK. In what concerns precision, we compared with FGK and we observed that the gains from using the combinations in this case are not significant. For, the comparison regarding \(F_1\), most of the combination kernels significantly outperform SPK. The only exception is FGK + NSPK. In fact, if we compare FGK + NSPK with both kernels that originate it, we notice that the differences in terms of \(F_1\) between them are not statistically significant. This is interesting because it illustrates how combining two kernels does not necessarily mean that the results will improve.

\(^3\)http://nlp.stanford.edu/software/lex-parser.shtml
\(^4\)http://sites.google.com/site/piotrwendykier/software/
Table 4: Performance of the individual kernels on the AImed data set

| Kernel | Recall   | Precision | F₁   |
|--------|----------|-----------|------|
| 14     | 47.74%   | 62.09%    | 53.49%|
| 8      | 41.15%   | 66.68%    | 50.60%|
| SPK + NSPK | 46.41%   | 60.57%    | 52.31%|

Table 5: Performance of the individual kernels on the AImed data set

| Kernel            | Recall   | Precision | F₁   |
|-------------------|----------|-----------|------|
| SPK + NSPK + 14   | 49.38%   | 64.12%    | 55.43%|
| SPK + NSPK + 8    | 45.67%   | 67.96%    | 54.23%|
| 14 + 8            | 45.21%   | 69.07%    | 54.12%|
| SPK + NSPK + 14 + 8 | 46.66%   | 68.36%    | 55.14%|

5.8 Comparison with Other Methods

In order to compare the performance of our solution with other methods, we implemented two additional kernels described in the literature: (i) a kernel based on shallow linguistic information of the sentences, [14]; and (ii) a kernel based on subsequences, [8]. During these experiments we always compared these kernels with our combination of kernels that showed better performance on the average values of the recall, precision and $F_1$: [SPK + NSPK]. Table 4 shows the results of this experiment.

The most evident conclusion obtained by observing the results is that our solution is still outperformed by the shallow linguistic information kernel in terms of average values of the metrics. However, the significance tests for all the metrics indicate that the differences between [SPK + NSPK] and [14] are not significative.

If we compare [SPK + NSPK] with [8], the results are very different. In fact, the results of the significance tests show that there are significative differences between these two kernels in terms of recall and precision ([SPK + NSPK] is better in terms of recall and [8] is better in terms of precision). However, in terms of $F_1$, the differences are not significative (even though [SPK + NSPK] obtains an higher average value of $F_1$).

The differences of the results of precision and recall of [SPK + NSPK] and [14] in comparison to [8] are something worth mentioning: the precision values are not as high as in the subsequences kernel but the values of recall are significantly higher. This is interesting because it goes against a typical trend in works on supervised RE in which the values of precision tend to be very high but the values of recall tend to be very low.

5.9 Combination with Other Kernel Methods

Finally, we performed some experiments to evaluate how combining [SPK + NSPK] with other methods influences the results. Once again, we used the two kernels that we compared our solution to in Section 5.8. Table 5 presents the results of this experiment.

By analyzing the results obtained in this experiment, we observe that the best combination is the one that joins [SPK + NSPK] with [14]. Moreover, even the combination of [SPK + NSPK] with [8] is able to outperform the combination of [14] and [8].

In order to understand these results, recall that [14] is based on several kernels including information of n-grams in three different locations of the sentence: before the first entity, between the entities and after the second entity. Knowing that n-grams are among the subsequences of the sentence, it is easy to understand that there is some overlapped information when combining these two kernel.

When these kernels are combined with [SPK + NSPK], we are joining information from completely different sources: sequences and dependency graph. For this reason, the kernel we propose is very interesting when used in combinations with kernels from different sources.

We also wanted to determine whether the difference of the results of these combinations in comparison to the individual kernels was significative. Thus, we performed significative tests between [SPK + NSPK], [14], [8] and all their combinations presented in Table 5.

In what concerns recall, the differences between the combinations, [SPK + NSPK] and [14], [8] are not significative. However, the tests indicate that all the combinations are able to outperform [8]. This comes with no surprise knowing that the differences in terms of average value of recall were very high.

Regarding precision, the significative tests show that combining [SPK + NSPK] with all the other kernels have a significant impact. The tests also obtain the same result for [14]. With [8] the results are different: none of the combinations is able to significantly outperform [8].

When comparing the results of the significative tests for $F_1$, there is only one combination that is able to clearly outperforms [SPK + NSPK] and [14]. This combination is actually the one that combines both these kernels. In all the other cases, the differences are not significative. Regarding [8], all the combinations are able to significantly outperform it in terms of $F_1$.

6. CONCLUSIONS AND FUTURE WORK

This paper proposes a solution for Relationship Extraction (RE) based on labeled graphs kernels. The proposed kernel is a particularization of the Random Walk Kernel for generic labeled graphs presented in [17]. In order to make the kernel suitable for RE tasks, we exploited two properties typically used in this line of work: (i) the words between the candidate entities or connecting them in a syntactic representation are particularly likely to carry information regarding the relationship; and (ii) combining information from different sources in a kernel may help the RE system to make better decisions. Our experiments show that the performance of our solution is comparable with the state-of-the-art on RE. Moreover, we showed that combining our solution with other methods for RE leads to significant gains in terms of performance.

Interesting topics for future work include the study of differ-
ent parameterizations of the Random Walk Kernel for RE. Namely, we want to try different kernels for vertex and edge labels as well as different probability distributions associated to the vertexes and the transitions. Moreover, it would be interesting to compare this kernel directly with other methods and test the combination of other kernels with ours. Finally, we would also like to test our solution with other datasets, namely the ACE dataset, which is composed by documents containing a wide variety of relationships (e.g., CEO_OF, Located_In) involving several types of entities (e.g., person, organization, location).

7. REFERENCES

[1] E. Agichtein, L. Gravano, J. Pavel, V. Sokolova, and A. Voskoboynik. Snowball: a prototype system for extracting relations from large text collections. In SIGMOD ’01: Proceedings of the 2001 ACM SIGMOD international conference on Management of data, 2001.

[2] A. Airola, S. Pyysalo, J. Björne, T. Pahikkala, F. Ginter, and T. Salakoski. A Graph Kernel for Protein-Protein Interaction Extraction. In BioNLP 2008: Current Trends in Biomedical Natural Language Processing, 2008.

[3] C. Aone, L. Halverson, T. Hampton, and M. Ramos-Santacruz. SRA: Description Of The Ie2 System Used for MUC-7. In Proceedings of the Seventh Message Understanding Conferences (MUC-7), 1998.

[4] R. Baumgartner, T. Eiter, G. Gottlob, M. Herzog, and C. Koch. Information extraction for the semantic web. In Reasoning Web, volume 3564 of Lecture Notes in Computer Science, pages 95–96. Springer Berlin / Heidelberg, 2005.

[5] G. E. P. Box, W. G. Hunter, J. S. Hunter, and W. G. Hunter. Statistics for Experimenters: An Introduction to Design, Data Analysis, and Model Building. John Wiley & Sons, 1978.

[6] S. Brin. Extracting patterns and relations from the World Wide Web. In EDBT ’98: WebDB Workshop at 6th International Conference on Extending Database Technology, 1998.

[7] R. Bunescu and R. Mooney. A shortest path dependency kernel for relation extraction. In Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP-05), 2005.

[8] R. Bunescu and R. Mooney. Subsequence Kernels for Relation Extraction. In Advances in Neural Information Processing Systems 18, 2006.

[9] A. Culotta and J. Sorensen. Dependency tree kernels for relation extraction. In ACL ’04: Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics, 2004.

[10] K. Eichler, H. Hemsen, and G. Neumann. Unsupervised relation extraction from web documents. In LREC 2008: Proceedings of the 6th edition of the International Conference on Language Resources and Evaluation, 2008.

[11] O. Etzioni, M. Cafarella, D. Downey, S. Kok, A.-M. Popescu, T. Shaked, S. Soderland, D. S. Weld, and A. Yates. Web-Scale Information Extraction in KnowItAll. In Proceedings of the 13th international conference on World Wide Web, 2004.

[12] A. Fader, S. Soderland, and O. Etzioni. Identifying Relations for Open Information Extraction. In EMNLP 2011: Proceedings of the Conference on Empirical Methods in Natural Language Processing, 2011.

[13] Y. Fang and K. C.-C. Chang. Searching patterns for relation extraction over the web: rediscovering the pattern-relation duality. In WSDM’11: Proceedings of the fourth ACM international conference on Web search and data mining, 2011.

[14] C. Giuliano, A. Lavelli, and L. Romano. Exploiting shallow linguistic information for relation extraction from biomedical literature. In Proceedings of EACL 2006, 11st Conference of the European Chapter of the Association for Computational Linguistics, 2006.

[15] K. Humphreys, R. Gaizauskas, S. Azzam, C. Huyck, B. Mitchell, H. Cunningham, and Y. Wilks. University Of Sheffield: Description Of The Lasie-Ii System As Used For MUC-7. In Proceedings of the Seventh Message Understanding Conferences (MUC-7), 1998.

[16] J. Jiang and C. Zhai. A systematic exploration of the feature space for relation extraction. In Proceedings of Human Language Technologies: The Conference of the North American Chapter of the Association for Computational Linguistics, 2007.

[17] H. Kashima, K. Tsuda, and A. Inokuchi. Marginalized kernels between labeled graphs. In Proceedings of the Twentieth International Conference on Machine Learning, 2003.

[18] A. McCallum. Information Extraction: Distilling Structured Data from Unstructured Text. ACM Queue, 3(9):48–57, 2005.

[19] D. Tikk, P. Thomas, P. Palaga, J. Hakenberg, and U. Leser. A comprehensive benchmark of kernel methods to extract protein–protein interactions from literature. PLoS Computational Biology, 6(7):e1000837, 2010.

[20] D. Zelenko, C. Aone, A. Richardella, J. K, T. Hofmann, T. Poggio, and J. Shawe-taylor. Kernel methods for relation extraction. Journal of Machine Learning Research, 3:2003, 2003.