# Graph Kernels versus Graph Representations: a Case Study in Parse Ranking

Tapio Pahikkala, Evgeni Tsivtsivadze, Jorma Boberg, and Tapio Salakoski

Turku Centre for Computer Science (TUCS)
Department of Information Technology, University of Turku
Lemminkäisenkatu 14 A, FIN-20520 Turku, Finland
firstname.lastname@it.utu.fi

**Abstract.** Recently, several kernel functions designed for a data that consists of graphs have been presented. In this paper, we concentrate on designing graph representations and adapting the kernels for these graphs. In particular, we propose graph representations for dependency parses and analyse the applicability of several variations of the graph kernels for the problem of parse ranking in the domain of biomedical texts. The parses used in the study are generated with the link grammar (LG) parser from annotated sentences of BioInfer corpus. The results indicate that designing the graph representation is as important as designing the kernel function that is used as the similarity measure of the graphs.

## 1 Introduction

Structured data are a commonplace in areas such as natural language processing (NLP). One of the most frequently encountered data structures in NLP are graphs. Kernel methods (see e.g. [1, 2]) have been among the most successful and computationally effective learning algorithms that can take advantage of the structured representation of the data. Recently, kernel functions on instances that are represented by graphs were introduced by [3–7]. Inspired by this research, we propose graph representations for dependency parses and analyse the applicability of the graph kernels for the problem of parse ranking in the domain of biomedical texts.

The link grammar (LG) parser [8] used in our research is a full dependency parser based on a broad-coverage hand-written grammar. The parses are generated by the LG parser applied to BioInfer corpus [9] containing 1100 annotated sentences. Due to the complexity of the biomedical text, the number of parses generated per sentence is large. Recently, we introduced a method for dependency parse ranking [10] that uses regularized least-squares (RLS) algorithm [11] and grammatically motivated features. The method, called RLS ranker, worked notably better giving 0.42 correlation compared to 0.16 of the LG heuristics measured with the Kendall's $\tau_b$ correlation coefficient [12]. In [13], we further developed the method by designing nonlinear kernel functions suitable for the problem.

In this study, we concentrate on designing graph representations, which is often overlooked. We demonstrate that designing an appropriate graph representation has a notable influence on the final results, comparable to the influence of the kernel function. RLS using graph kernels is applied to the proposed graph representations and the results indicate an improved correlation of 0.45. A detailed description of the data, RLS algorithm and the experimental setup used in this paper is given in [10]. We also show that the proposed approach can be considered as a generalization over previously described method [10].

## 2 Graph Kernels

We now give a brief introduction to kernels considered in this study. Formally, let $X$ denote the input space, which can be any set, and $H$ denote the feature space. For any mapping $\Phi : X \rightarrow H, k(x,z) = \langle \Phi(x), \Phi(z) \rangle$ is a kernel function. Following [3], we define a labeled graph representation of data points as follows. Below, $\mathcal{M}_{i \times j}(\mathbb{R})$ denotes the set of real valued matrices of dimension $i \times j$ and $[M]_{i,j}$ denotes the element of matrix $M$ in the $i$-th row and $j$-th column. Let $\mathcal{L} = \{l\}_r, r \in \mathbb{N}^+$ be an enumeration of all possible labels. Let $G = (V, E, h)$ be a graph that consists of the set of vertices $V$, the set of edges $E \subseteq V \times V$, and a function $h : V \rightarrow \mathcal{L}$ that assigns a label to each vertex of a graph. We assume that the edge set of $G$ is represented as an adjacency matrix $A \in \mathcal{M}_{|V| \times |V|}(\mathbb{R})$ whose rows and columns are indexed by the vertices $V$, and $[A]_{i,j}$ is one if the vertices $v_i \in V$ and $v_j \in V$ are connected with an edge, and zero otherwise. We also assume that the function $h$ is represented as a label allocation matrix $L \in \mathcal{M}_{|\mathcal{L}| \times |V|}(\mathbb{R})$ so that its element $[L]_{i,j}$ is one if vertex $v_j \in V$ has a label $l_i \in \mathcal{L}$ and zero otherwise.

We also define the following relaxed version of the graph labeling. Let $\mathcal{L}$ be a vector space, for example, $\mathcal{L} = \mathbb{R}^p$. We can then define $h : V \rightarrow \mathcal{L}$ to be a function that assigns a label vector to each vertex of a graph. Its corresponding representation as a label allocation matrix is $L \in \mathcal{M}_{|\mathcal{L}| \times |V|}(\mathbb{R})$ so that the columns of the matrix are the label vectors of the vertices.

Again, we follow [3], and define a class of kernel functions on labeled graphs. Let us consider the $n$th power $A^n$ of the adjacency matrix of the graph $G$. Then, $[A^n]_{i,j}$ is the number of walks of length $n$ from vertex $v_i$ to vertex $v_j$. When we take the labels of the vertices into account, we observe that $[LA^nL^{\mathrm{T}}]_{i,j}$ is the number of walks of length $n$ between vertices labeled $l_i$ and $l_j$. Let $G$ and $G'$ be labeled directed graphs and let $\langle M, M' \rangle_F$ denote the Frobenius product of matrices $M$ and $M'$, that is, $\langle M, M' \rangle_F = \sum_{i,j} [M]_{i,j} [M']_{i,j}$. Let further $\gamma \in \mathcal{M}_{n \times n}(\mathbb{R})$ be a positive semidefinite matrix whose eigen decomposition is $U \Lambda U^{\mathrm{T}}$, where $U$ is a matrix that contains the eigenvectors of $\gamma$ and $\Lambda$ is a diagonal matrix containing the eigenvalues of $\gamma$. We define the kernels $k_n$ between the graphs $G$ and $G'$ as follows

$$\begin{aligned} k_n(G, G') &= \textstyle\sum_{i,j=0}^{n} [\gamma]_{i,j} \langle LA^iL^{\mathrm{T}}, L'A'^jL'^{\mathrm{T}} \rangle_F \\ &= \textstyle\sum_{k,l=1}^{|\mathcal{L}|} \sum_{i,j=0}^{n} [\gamma]_{i,j} [LA^iL^{\mathrm{T}}]_{k,l} [L'A'^jL'^{\mathrm{T}}]_{k,l} \\ &= \textstyle\sum_{k,l=1}^{|\mathcal{L}|} \sum_{i=0}^{n} \phi_{i,k,l}(G) \phi_{i,k,l}(G') \end{aligned} \qquad (1)$$

where it is easy to see that, when $\gamma$ is positive semidefinite, the features are defined as $\phi_{i,k,l}(G) = \sum_{j=0}^{n}[\sqrt{\Lambda}U^{\mathrm{T}}]_{i,j}[LA^{j}L^{\mathrm{T}}]_{k,l}$. By specializing $[\gamma]_{i,j}$ in (1), we obtain several kernel functions with different interpretations. For example, if we set $[\gamma]_{i,j} = \theta^{i}\theta^{j}$, where $\theta \in \mathbb{R}^{+}$ is a parameter, we obtain the kernel

$$\widehat{k_n}(G, G') = \langle L(\sum_{i=0}^{n}\theta^{i}A^{i})L^{\mathrm{T}}, L'(\sum_{i=0}^{n}\theta^{i}A'^{i})L'^{\mathrm{T}}\rangle_F. \tag{2}$$

This kernel can be interpreted as an inner product in a feature space in which there is a feature $\phi_{k,l}$ per each label pair $(k, l)$ so that its value $\phi_{k,l}(G)$ for a graph $G$ is a weighted count of walks of length up to $n$ from the vertices labeled $l$ to the vertices labeled $k$. On the other hand, by setting $[\gamma]_{i,j} = \theta^{i}$ when $i = j$ and zero otherwise, we obtain the kernel

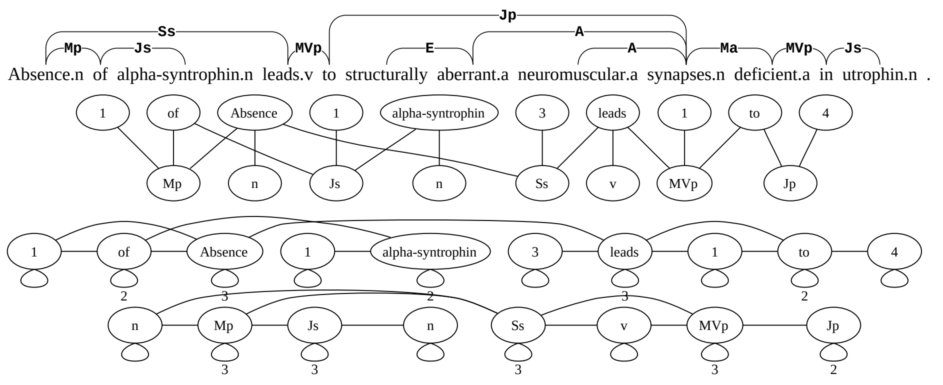
$$\widetilde{k_n}(G, G') = \sum_{i=0}^{n}\theta^{2i}\langle LA^{i}L^{\mathrm{T}}, L'A'^{i}L'^{\mathrm{T}}\rangle_F, \tag{3}$$

which can be interpreted as an inner product in a feature space in which there is a feature $\phi_{i,k,l}$ per each tuple $(i, k, l)$, where $l$ and $k$ are labels and $i$ is a length of a walk. Its value $\phi_{i,k,l}(G)$ for a graph $G$ is $\theta^{i}$ times the count of walks of length $i$ from the vertices labeled $l$ to the vertices labeled $k$. Finally, with certain conditions on the coefficients $[\gamma]_{i,j}$, we can also define $k_{\infty}(G, G') = \lim_{n\to\infty} k_n(G, G')$. One such kernel function is, for example, the exponential graph kernel $k_{exp}(G, G') = \langle Le^{\beta A}L^{\mathrm{T}}, L'e^{\beta A'}L'^{\mathrm{T}}\rangle_F$, where $\beta$ is a parameter and $e^{\beta A}$ can be written as $e^{\beta A} = \lim_{n\to\infty}\sum_{i=0}^{n}\frac{\beta^{i}}{i!}A^{i}$. In this case, the coefficients $[\gamma]_{i,j}$ are determined by the parameter $\beta$ as follows: $[\gamma]_{i,j} = \frac{\beta^{i}}{i!}\frac{\beta^{j}}{j!}$.

## 2.1 Graph Representations of the Parses

The output of the LG parser contains the following information for each input sentence. The linkage consisting of pairwise dependencies between pairs of words termed links. An example of a parsed sentence is presented in Fig.1. In addition to the linkage structure, the parses contain information about the link types (the grammatical roles assigned to the links) used to connect word pairs. The link types present in Fig.1 are *Mp*, *Js*, *Ss*, etc. Further, the parse contains the part-of-speech (PoS) tags of the words, such as verb ($v$), noun ($n$) and adjective ($a$) categories. In Fig.1, the assigned PoS tags, for example, to the words *absence*, *alpha-syntrophin*, *leads* are $n$, $n$ $v$, respectively. Different parses of a single sentence have a different combination of these elements.

In our previous study [10], we proposed several grammatically motivated features that could be extracted from the above described dependency parses, namely the link type, link length, PoS, word & link type, word & PoS, link type & link length, grammatical bigram, and link bigram features that we will describe below in more detail. In this paper, we are using graph kernels as similarity measures of the data points. Therefore, we now define a graph representation for the parses. The representation is designed so that we are able to simulate the previously proposed (and some additional) features with the graph kernel.



Ss
Mp
Js
MVp
Jp
E
A
A
Ma
MVp
Js
Absence.n of alpha-syntrophin.n leads.v to structurally aberrant.a neuromuscular.a synapses.n deficient.a in utrophin.n .
1
of
Absence
1
alpha-syntrophin
3
leads
1
to
4
Mp
n
Js
n
Ss
v
MVp
Jp

**Fig. 1.** Example of parsed sentence (top). Graph representation of the beginning of the parse (middle). Walks of length 2 present in the parse (bottom). Note that some of the loops (the edges starting from and ending to the same node) have weights larger than one, that is, there are several repetitions of the corresponding walks.

***Graph representation.*** Let $p$ be a parse generated from a sentence $s$ and let $G = (V, E, h)$ denote the graph representation of $p$. An example of the graph representation is presented in Fig.1. Let us first define the vertices of $p$. For each word in the sentence $s$, there is a corresponding vertex $v \in V$ and the vertex is labeled with the word (the word vertices and their labels do not depend from the parse). Thus, if a word occurs several times in the sentence $s$, all of the occurrences have their own vertices in the graph but the labels of the vertices are equal. For each link in the parse $p$, there is a corresponding vertex in $V$ that is labeled with its link type. Similarly to the word vertices, a parse may have several occurrences of the same link type. In $p$, each word is assigned a PoS, for which there is a corresponding vertex in $V$ labeled with the PoS. Further, each link in $p$ has a length (the number of words that a link in the sentence spans) for which there is a corresponding vertex in $V$ labeled with the length. In Fig.1, they are the vertices labeled with integers, for example, *1*, *1*, *3*, etc.

The edges of $G$ are defined as follows. A word vertex and its corresponding PoS vertex are connected with an edge. A link vertex and its corresponding length vertex are connected with an edge. If two words are connected with a link in the parse $p$, the corresponding link vertex is connected with an edge to both of the corresponding word vertices. The connection of a word vertex in the graph with a link vertex (for example in Fig.1: *absence—Mp*, *absence—Ss*, *of—Js*, etc.) can be considered as the word & link type feature described in our previous study. Below, we show how these connections are used to create also word bigram and link bigram described previously [10].

***Random walk features.*** Let $G = (V, E, h)$ be a graph representation of a parse. Let us consider the second power of the adjacency matrix $A$ of $G$, that is, the walks of length 2 in the graph representation of the corresponding parse.

Those walks in the graph representation are illustrated in Fig.1. Note that intersection of the sets of the walks of length one and two is empty due to the bipartite property of the graph. Also because of this property, all of the walks of length 2 have both the start and the end vertices in the same subset. Among the walks of length two there is, for example, a walk between two word vertices iff they are connected with a link in the parse, and between two link vertices iff the links are connected to the same word in the parse. Such connections were called grammatical bigrams and link bigrams in [10]. We also obtain walks between between PoS vertices and link vertices, and between word vertices and link length vertices. Finally, a vertex has as many cycles as there are edges connected to it. If we consider the higher powers of the adjacency matrices, we obtain new features, for example, link length pairs in the fourth power. In the higher powers, we also obtain word and link bigrams, where the words and links are not connected to each other in the parse.

***Vector labeled graph representation.*** Using multiple labels for vertices offers a way to incorporate more information into the graph representation. This kind of representations have been considered by [6], for example. In addition to the graph representation presented above, we define the following vector labeled representation. Again, let $p$ be a parse generated from a sentence $s$ and let $G = (V, E, h)$ denote the graph representation of $p$. For each word in the sentence $s$, there is a corresponding vertex $v \in V$ and the vertex is labeled with the word and its PoS. In other words, all the elements of the label vector are zero except the ones indexed by the word and its PoS. For each link in the parse $p$, there is a corresponding vertex in $V$ that is labeled with its link type and its length. Thus, instead of having the PoS and link length vertices as in the previous graph representation of parses, they are used as vertex labels in this representation. We refine the representation further using the following five labels for the link length instead of just one. Let $l$ be the length of a link. Instead of using only $l$, we use $l-2$, $l-1$, $l$, $l+1$, $l+2$. Using this set of labels, we are able to match similarities between links whose lengths are close to each other, while the single label is useful only for exact matching of the link lengths. This representation has certain connections that the previous one does not have, such as the connections between word and link length as well as between PoS and link type. On the other hand, it misses the connections between word and PoS as well as between link type and link length that are in the previous representation.

When we consider the walks of length 2, that is, the label connections obtained using the second power of the adjacency matrix, we observe that the missing connections between word and PoS, for example, are among those walks. Note, however, that there are also connections between words and the PoS of their neighboring words, and there is no way to distinguish between these connections and the connections between words and their own PoS. The same problem arises from the connections between link type and link length that are also among the walks of length 2.

## 3 Experiments

We give a detailed description of the problem and the data in [10]. We evaluate the different variations of the graph kernel by performing a 10-fold cross-validation on the sentence level so that all the parses generated from a sentence would always be in the same fold (see [14] for a fast $n$-fold cross-validation algorithm for RLS). The RLS algorithm has the regularization parameter $\lambda$ that controls the trade-off between the minimization of the training error and the complexity of the regression function. The appropriate values of this parameter is determined together with the kernel parameters by grid search with 10-fold cross-validation.

In our experiments, we evaluate the kernels $\widehat{k_n}$ and $\widetilde{k_n}$ up to the third power with different parameters. In addition, we evaluate the following version of the exponential graph kernel $\bar{k}_{exp}(G, G') = \langle LAe^{\beta A}L^{\mathrm{T}}, L'A'e^{\beta A'}L'^{\mathrm{T}}\rangle_F$. We multiply the exponentiated adjacency matrix $e^{\beta A}$ by $A$, because then by setting $\beta = 0$, we obtain the original adjacency matrix $A$ as a special case, and we can set the preferred weight of the higher powers by a grid search on $\beta$.

We start by evaluating the graph kernel $k_0$ with $[\gamma]_{0,0} = 1$, that is, the zeroth power of the adjacency matrix. The obtained performance of the RLS ranker with $k_0$ is 0.377. This corresponds to a kernel that counts the elementary features that are present in both parses, that is, the words, link types, PoS tags, and link lengths. Recall that the word vertices only depend on the sentence, and therefore they are useless for the parse ranking. In our previous study [10], we also found out that the other elementary features are not as good as the combination features. We continue by evaluating $k_1$ with $[\gamma]_{i,j} = 1$ when $i = j = 1$ and zero otherwise. In other words, we use only the original adjacency matrices of the graphs determined by the edges defined in Sect.2. The result of this experiment is 0.406 correlation points. The performance differences in these two experiments are in correspondence to our previous study [10], in which we observed that the ranking performances with the four elementary features were low compared to their combinations that are present in the first power of the adjacency matrices.

The second powers of the adjacency matrices contain the elementary features present in the zeroth power, the rest of the combination features proposed in the previous study that are present in the first and the second powers, and also some new combination features in the second power. To get a weighted combination of the first and the second power features, we evaluate the following kernel function

$$k(G, G') = \langle L(A + \theta A^2)L^{\mathrm{T}}, L'(A' + \theta A'^2)L'^{\mathrm{T}}\rangle_F, \qquad (4)$$

where $\theta$ is a parameter for which we perform a grid search in range $2^{-5}, 2^{-4}, \ldots, 2^5$. The above kernel is equal to $\widehat{k_2}$ in (2) except that we exclude the zeroth power. Note that due to the bipartite property, the kernel is also equal to $\widetilde{k_2}$ in (3) with the zeroth power excluded. The performance with the best $\theta$ parameter is 0.429 correlation points.

To analyse the usefulness of the walk features of length longer than 2, we evaluate the following two kernels

$$k(G, G') = \langle L(A + \theta A^2 + \theta^2 A^3)L^{\mathrm{T}}, L'(A' + \theta A'^2 + \theta^2 A'^3)L'^{\mathrm{T}}\rangle_F, \tag{5}$$

and

$$\begin{aligned} k(G, G') = \langle LAL^{\mathrm{T}}, L'A'L'^{\mathrm{T}}\rangle_F + \theta\langle LA^2L^{\mathrm{T}}, L'A'^2L'^{\mathrm{T}}\rangle_F \\ +\theta^2\langle LA^3L^{\mathrm{T}}, L'A'^3L'^{\mathrm{T}}\rangle_F, \end{aligned} \tag{6}$$

where the best $\theta$ parameters are found with a grid search. The first kernel is similar to $\widehat{k_3}$ in (2) with the zeroth power excluded, and the second kernel is similar to $\widetilde{k_3}$ in (3) with the zeroth power excluded. The performance using the first and the second kernels with the best parameters are 0.422 and 0.429 correlation points, respectively. We also performed some experiments with the fourth powers of the adjacency matrices but the results were worse than the above two. The performance with the exponential graph kernel $\bar{k}_{exp}(G, G')$ is 0.425 correlation points. According to the results, is seems that the walks of length larger than 2 are not useful features when this kind of graph representation is used. In fact, they seem to be even harmful when they are mixed with the lower order features, for example, in the kernel (5).

We also evaluate the vector labeled graph representation presented in Sect.2. The results with the original adjacency matrix and with the kernel (4) are 0.364 and 0.377, respectively. They are clearly worse than the ones with the single labeled graph representation and therefore we do not conduct more experiments with the vector labeled representation. The low performance is probably due to the noisy combination features that we discuss in Sect.2.1. The results indicate that designing the graph representation is as important as designing the kernel function.

In order to validate the results with an unseen test set, we conduct a final validation experiment on 600 sentences reserved for that purpose. We select the single labeled graph representation and the kernels (4) and (6) with the best performing parameter combinations, that is, the settings that give the highest ranking performance in the parameter estimation experiments. The ranker is trained with the parameter estimation data and the results with the kernels (4) and (6) are 0.444 and 0.447 correlation points, respectively.

## 4 Conclusion

We propose graph representations for dependency parses and analyse the applicability of several variations of the graph kernels for the problem of parse ranking in the domain of biomedical texts. We use the graph kernels to generate features that are based on the start and end labels of random walks between vertices in the graphs. The feature vector corresponding to a data point is determined by its graph representation and the kernel function. Both of them have to be selected carefully in order to ensure a good performance of the machine learning method. Several kernel functions have already been proposed for a data that consists of graphs. In addition, our results underline the importance of the design of a good graph representation for the data points. The performance achieved is promising when compared to our previous studies [10] and could be even further improved by designing representations capturing additional prior knowledge about the problem to be solved.

## Acknowledgments

This work has been supported by Tekes, the Finnish Funding Agency for Technology and Innovation. We would like to thank CSC, the Finnish IT center for science, for providing us extensive computing resources and the anonymous reviewers for their insightful comments.

## References

1. Schölkopf, B., Smola, A.J.: Learning with kernels. MIT Press (2002)
2. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004)
3. Gärtner, T.: Exponential and geometric kernels for graphs. In: NIPS Workshop on Unreal Data: Principles of Modeling Nonvectorial Data. (2002)
4. Gärtner, T., Flach, P.A., Wrobel, S.: On graph kernels: Hardness results and efficient alternatives. In Schölkopf, B., Warmuth, M.K., eds.: COLT'03, Springer (2003) 129–143
5. Kashima, H., Inokuchi, A.: Kernels for graph classification. In: ICDM Workshop on Active Mining. (2002)
6. Suzuki, J., Sasaki, Y., Maeda, E.: Kernels for structured natural language data. In Thrun, S., Saul, L.K., Schölkopf, B., eds.: NIPS 16, MIT Press (2003)
7. Vishwanathan, S., Smola, A.J., Vidal, R.: Binet-cauchy kernels on dynamical systems and its application to the analysis of dynamic scenes. International Journal of Computer Vision (2006) To appear.
8. Sleator, D.D., Temperley, D.: Parsing english with a link grammar. Technical Report CMU-CS-91-196, Carnegie Mellon University, Pittsburgh, PA (1991)
9. Pyysalo, S., Ginter, F., Heimonen, J., Björne, J., Boberg, J., Järvinen, J., Salakoski, T.: Bioinfer: A corpus for information extraction in the biomedical domain (2006) Submitted.
10. Tsivtsivadze, E., Pahikkala, T., Pyysalo, S., Boberg, J., Mylläri, A., Salakoski, T.: Regularized least-squares for parse ranking. In Famili, A.F., Kok, J.N., Peña, J.M., Siebes, A., Feelders, A.J., eds.: IDA'05, Springer (2005) 464–474
11. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Notices of the American Mathematical Society (AMS) **50**(5) (2003) 537–544
12. Kendall, M.G.: Rank Correlation Methods. 4. edn. Griffin (1970)
13. Tsivtsivadze, E., Pahikkala, T., Boberg, J., Salakoski, T.: Locality-convolution kernel and its application to dependency parse ranking. In Ali, M., Dapoigny, R., eds.: IEA-AIE'06, Springer (2006) 610–618
14. Pahikkala, T., Boberg, J., Salakoski, T.: Fast n-fold cross-validation for regularized least-squares. In: Proceedings of the Ninth Scandinavian Conference on Artificial Intelligence (SCAI 2006). (2006) To appear.