Locality-Convolution Kernel and Its Application to Dependency Parse Ranking

Evgeni Tsivtsivadze, Tapio Pahikkala, 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. We propose a Locality-Convolution (LC) kernel in application to dependency parse ranking. The LC kernel measures parse similarities locally, within a small window constructed around each matching feature. Inside the window it makes use of a position sensitive function to take into account the order of the feature appearance. The similarity between two windows is calculated by computing the product of their common attributes and the kernel value is the sum of the window similarities. We applied the introduced kernel together with Regularized Least-Squares (RLS) algorithm to a dataset containing dependency parses obtained from a manually annotated biomedical corpus of 1100 sentences. Our experiments show that RLS with LC kernel performs better than the baseline method. The results outline the importance of local correlations and the order of feature appearance within the parse. Final validation demonstrates statistically significant increase in parse ranking performance.

1 Introduction

With availability of structured data applicable for statistical and machine learning techniques, application of kernel methods (see e.g. [1, 2]) is shown to have an important role in Natural Language Processing (NLP). Recently, several papers have presented and applied new kernels to NLP problems with promising results. Collins and Duffy [3] described convolution kernels for various discrete structures, encountered in NLP tasks, which allow high dimensional representations of these structures in feature vectors. Suzuki et al. [4] introduced the Hierarchical Directed Acyclic Graph (HDAG) kernel for structured natural language data. It was shown that rich representation of the features through directed graphs in parse, improves performance in various NLP applications. A statistical feature selection approach for the above mentioned kernels was proposed in [5]. This work has been motivated not only by rapidly developing field of the kernel methods and their successful applications in NLP, but also by the importance of incorporating domain knowledge for improving the performance of the learning algorithms.

In this study, we address the problem of dependency parse ranking in the biomedical domain. The parses are generated by the Link Grammar (LG) parser

M. Ali and R. Dapoigny (Eds.): IEA/AIE 2006, LNAI 4031, pp. 610–618, 2006.

[6] which was applied to the Biomedical Dependency Bank (BDB) corpus[1] containing 1100 annotated sentences. The LG parser is a full dependency parser based on a broad-coverage hand-written grammar. It generates all parses allowed by its grammar and applies the built-in heuristics to rank the parses. However, its ranking performance has been found to be poor when applied to biomedical text [7].

Recently, we proposed a method for dependency parse ranking [8] that uses Regularized Least-Squares (RLS) algorithm [9] and grammatically motivated features. The method, called RLS ranker, worked notably better giving 0.42 correlation compared to 0.16 of the LG heuristics. In this paper, we propose a Locality-Convolution (LC) kernel that provides a correlation of 0.46 when used in RLS algorithm. In all experiments, we applied the F-score based parse goodness function [8], and evaluated the ranking performance with Kendall's correlation coefficient τ_b described in [10].

The LC kernel addresses the problem of parse ranking through the following characteristics. Firstly, it possesses the convolution property described by Haussler [11], and operates over discrete structures. Secondly, it calculates similarity between windows spanning over the closely located features. Furthermore, it makes use of the position sensitive function, which takes into account the positions of the features within the windows. The LC kernel function can be considered as a specific instance of the convolution kernels and can be included in many methods, such as the RLS algorithm that we are using in this study.

The paper is organized as follows: in Section 2, we describe the RLS algorithm; in Section 3, we present grammatically motivated features for parse representation; in Section 4, we discuss convolution kernels, in Section 5, we define notions of locality windows, position sensitive feature matching, and finally introduce the LC kernel; in Section 6, we evaluate the applicability of the LC kernel to the task of dependency parse ranking and benchmark it with respect to previous baseline method; we conclude this paper in Section 7.

2 Regularized Least-Squares Algorithm

Let $\{(x_1, y_1), \ldots, (x_m, y_m)\}$, where $x_i \in P, y_i \in \mathbb{R}$, be the set of training examples. We consider the Regularized Least-Squares (RLS) algorithm as a special case of the following regularization problem known as Tikhonov regularization (for a more comprehensive description, see e.g. [9]):

$$\min_f \sum_{i=1}^{m} l(f(x_i), y_i) + \lambda \|f\|_K^2, \tag{1}$$

where l is the loss function used by the learning machine, $f : P \to \mathbb{R}$ is a function, $\lambda \in \mathbb{R}_+$ is a regularization parameter, and $\|\cdot\|_K$ is a norm in a Reproducing Kernel Hilbert Space defined by a positive definite kernel function K. Here P can be any set, but in our problem, P is a set of parses of the sentences of

[1] http://www.it.utu.fi/∼BDB

the BDB corpus. The target output value y_i is calculated by a parse goodness function, that is, $y_i = f^*(x_i)$, and is predicted with the RLS algorithm. We used the F-score based function f^* as defined in [8]. The second term in (1) is called a regularizer. The loss function used with RLS for regression problems is called least squares loss and is defined as

$$l(f(x), y) = (y - f(x))^2.$$

By the Representer Theorem (see e.g. [12]), the minimizer of equation (1) has the following form:

$$f(x) = \sum_{i=1}^{m} a_i K(x, x_i),$$

where $a_i \in \mathbb{R}$ and K is the kernel function associated with the Reproducing Kernel Hilbert Space mentioned above.

Kernel functions are similarity measures of data points in the input space P, and they correspond to the inner product in a feature space H to which the input space data points are mapped. Formally, kernel functions are defined as

$$K(x, x') = \langle \Phi(x), \Phi(x') \rangle,$$

where $\Phi : P \to H$.

3 Features for Parse Representation

In the case of parse ranking problem, where parses are represented within dependency structure, particular attention to the extracted features is required due to the sparseness of the data. In [8] we proposed features that are grammatically relevant and applicable even when relatively few training examples are available. Grammatical features extracted from a dependency parse, contain information about the linkage consisting of pairwise dependencies between pairs of words (bigrams), the link types (the grammatical roles assigned to the links) and the part-of-speech (POS) tags of the words. An example of a fully parsed sentence is shown in Figure 1.

As in [8], we define seven feature types representing important aspects of the parse, consisting of the following grammatical structures:

Grammatical bigram feature is defined as a pair of words connected by a link. In the linkage example of Figure 1, the extracted grammatical bigrams are *absence—of*, *of—alpha-syntrophin*, etc.

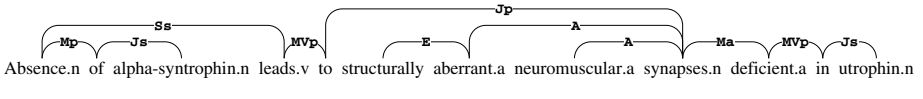

Fig. 1. Example of parsed sentence

Word & POS tag feature contains the word with the POS tag assigned to the word by the LG parser. In Figure 1, the extracted word & POS features are *absence.n*, *alpha-syntrophin.n* etc.
Link type feature represents the type of the link assigned by the LG parser. In the example, they are *Mp*, *Js*, etc.
Word & Link type feature combines each word in the sentence with the type of each link connected to the word, for example, *absence—Mp*, *absence—Ss*, etc.
Link length feature represents the number of words that a link in the sentence spans. In Figure 1, the extracted features of this type are *1*, *1*, etc.
Link length & Link type feature combines the type of the link in the sentence with the number of words it spans. In Figure 1, the extracted features of this type are *1—Mp*, *1—Js*, etc.
Link bigram feature extracted for each word of the sentence is a combination of two links connected to the word, ordered leftmost link first. In the example, the link bigrams are *Mp—Ss*, *Mp—Js*, etc.

As a natural continuation of [8], we propose projecting parses into feature sequences in order to take into account local correlations between parses. To avoid sparsity, for each parse, we make one sequence consisting of homogeneous features per each type instead of a single sequence containing the features of all types. We define these homogeneous feature sequences as follows. Let r be the number of types, and let $\{t_1, \ldots, t_r\}$ be the set of types. In our case, $r = 7$ and the corresponding feature types are described above. For example, t_1 is the grammatical bigram type consisting of all the grammatical bigram features of the particular parse. Let us consider a parse $p \in P$, and let $p_j, 1 \leq j \leq r$, be the sequence of the features of type t_j in the parse p in the order of their appearance. For example, in the case of Figure 1, $p_1 =$ *absence—of*, *absence—leads*, *of—alpha-syntrophin*, etc. The order of features is also preserved for all other types: $p_3 =$ *Mp*, *Ss* etc. or $p_4 =$ *absence—Mp*, *absence—Ss*, *of—Mp*, *of—Js* etc. For the basic types – POS tag, Word, Link type, and Link length features – as well as for the complex features representing conjunctions of the basic types, the order of appearance is determined by the indices of the words they are related to. For example, if there exist two grammatical bigrams having a common first word, the decision of the feature positions within the sequence is based on the index of the second word.

Now we can define a mapping Φ from the parse space P to the feature space H, $\Phi : P \to H$, representing parses through the sequences of the features as follows: $\Phi(p) = (p_1, \ldots, p_r)$. If we denote $p_j = (f_1^{t_j}, \ldots, f_{|p_j|}^{t_j})$, we get

$$\Phi(p) = ((\underbrace{f_1^{t_1}, \ldots, f_{|p_1|}^{t_1}}_{p_1}), \ldots, (\underbrace{f_1^{t_r}, \ldots, f_{|p_r|}^{t_r}}_{p_r})). \tag{2}$$

Here, the length of the sequences p_j, denoted as $|p_j|$, as well as the individual features $f_i^{t_j}$ depend on the parse p. We call the sequences p_j subparses of $p \in P$.

4 Convolution Kernels

The convolution kernels are usually built over discrete structures. They are defined between the input objects by applying convolution sub-kernels for the parts of the objects. Following [11], we briefly describe convolution kernel framework. Let us consider $x \in X$ as a composite structure such that $x_1, \ldots, x_N$ are its parts, where x_n belongs to the set X_n for each $1 \leq n \leq N$, and N is a positive integer. We consider $X_1, \ldots, X_n$ as countable sets, however, they can be more general separable metric spaces [11]. Let us denote shortly $\widehat{x} = x_1, \ldots, x_N$. Then the relation "$x_1, \ldots, x_N$ are parts of x" can be expressed as a relation R on the set $X_1 \times \ldots, X_N \times X$ such that $R(\widehat{x}, x)$ is true if $\widehat{x}$ are the parts of x. Then we can define $R^{-1}(x) = \{\widehat{x} : R(\widehat{x}, x)\}$. Now let us suppose that $x, y \in X$ and there exist decompositions such that $\widehat{x} = x_1, \ldots, x_N$ are the parts of x and $\widehat{y} = y_1, \ldots, y_N$ are the parts of y. If we have some kind of kernel functions

$$K_n(x_n, y_n) = \langle \Phi(x_n), \Phi(y_n) \rangle, 1 \leq n \leq N,$$

to measure similarity between elements of X_n, then the kernel $K(x, y)$ measuring the similarity between x and y is defined to be the following generalized convolution:

$$K(x, y) = \sum_{\widehat{x} \in R^{-1}(x)} \sum_{\widehat{y} \in R^{-1}(y)} \prod_{n=1}^{N} K_n(x_n, y_n). \tag{3}$$

There have been several different convolution kernels reported and applied in NLP, for example, string kernel [13], tree kernel [3], word-position kernel ([14], [15]) and HDAG kernel [5]. The LC kernel function proposed in this paper satisfies the properties of the above convolution approach and it is built over discrete and homogeneous sequences of the features described in the Section 3.

5 Locality-Convolution Kernel

The Locality-Convolution kernel has the following properties that we believe are of importance for the ranking task: *i*) the use of feature sequences extracted in the order of the appearance in the parse, *ii*) construction of locality window around matching features, and *iii*) position sensitive evaluation of the features within the window. Below we define these properties formally.

Let us consider parses $p, q \in P$ and let $p_j = (f_1^{t_j}, \ldots, f_{|p_j|}^{t_j})$ and $q_j = (g_1^{t_j}, \ldots, g_{|q_j|}^{t_j})$ be their subparses. They consist of the features of the same type t_j as described in Section 3. Next we consider how to define a similarity between the subparses p_j and q_j. For the sake of simplicity, we write in the feature sequences supersript j instead of t_j.

The similarity of the subparses p_j and q_j is obtained with the following kernel:

$$K(p_j, q_j) = \sum_{s \in S_j} \kappa(s) = \sum_{i=1}^{|p_j|} \sum_{k=1}^{|q_j|} \kappa(f_i^j, g_k^j) \delta(f_i^j, g_k^j), \tag{4}$$

where $S_j = \left\{(f_i^j, g_k^j) | 1 \le i \le |p_j|, 1 \le k \le |q_j|, f_i^j = g_k^j\right\}$ is the set of all equal pairs of the features present in the subparses p_j and q_j, κ is a kernel, and

$$\delta(x, y) = \begin{cases} 0, & \text{if } x \neq y \\ 1, & \text{if } x = y. \end{cases}$$

If we set $\kappa \equiv 1$, then the equation (4) equals to $|S_j|$. To weight more those cases where there are exact matches near each other within a certain locality window, we define κ as follows. First, for each $s = (f_i^j, g_k^j) \in S_j$, we create a window around f_i^j and g_k^j of length w, and define

$$w_s = \left\{(f_m^j, g_l^j) | \ \ s = (f_i^j, g_k^j) \in S_j, |m - i| \le \left\lfloor \frac{w}{2} \right\rfloor, |l - k| \le \left\lfloor \frac{w}{2} \right\rfloor, (f_m^j, g_l^j) \in S_j \right\}.$$

A simple realization of the weighting idea would be:

$$\kappa(s) = |w_s|, \tag{5}$$

where $|w_s|$ is the number of all identical feature pairs between two locality windows. As another alternative, we construct a function that requires the matching feature positions to be exactly the same:

$$\kappa(s) = \sum_{(f_m^j, g_l^j) \in w_s} \delta(m, l). \tag{6}$$

The kernel (4) with exact position matching κ described in (6) is related to the locality improved kernel proposed in [16]. However, if we would not require strict position matching, but rather penalize the features that match but have a different position within the windows, one can use the following position sensitive version of the kernel function:

$$\kappa(s) = \prod_{(f_m^j, g_l^j) \in w_s} (e^{-\left(\frac{(m-l)}{\theta}\right)^2} + 1), \tag{7}$$

where $\theta \ge 0$ is a parameter. Note that (5), (6), and (7) become equal to 1, 1, and 2, respectively, when the length of the locality window is one. The choice of an appropriate κ might be a matter closely related to the domain of the study. In Section 6 we show that additional information captured with (7) is useful and improves the ranking performance. Drawing a parallel between the proposed kernel and the convolution approach, one can distinguish between "structures" and "different decompositions" constructed by our kernel. By substituting the position sensitive κ defined in (7) into (4), we obtain:

$$K_{LC}(p_j, q_j) = \sum_{s \in S_j} \prod_{(f_m^j, g_l^j) \in w_s} (e^{-\left(\frac{(m-l)}{\theta}\right)^2} + 1), \tag{8}$$

which we call the Locality-Convolution kernel. Conceptually, the LC kernel enumerates all the substructures representing pairs of windows built around the matching features in the subparses and calculates their inner product. The LC kernel

is able to treat not only exact matching of the features, but also matching within the locality windows, therefore making the similarity evaluation more precise.

To measure the similarity between whole parses, we measure the correspondence of their subparses within each type and then sum them up:

$$K(p,q) = \sum_{j=1}^{r} K_{LC}(p_j, q_j). \tag{9}$$

Finally, we wrap the kernel inside a Gaussian function $K^{\mathcal{G}}$ in the following way:

$$K^{\mathcal{G}}(p,q) = e^{\left(-\frac{K(p,p)-2K(p,q)+K(q,q)}{\sigma^2}\right)}, \tag{10}$$

where σ is a parameter controlling the width of Gaussian function. The wrapping has also a normalization effect, which is useful because the size of the parses, that is, the total number of features in the parse is not a constant. Expression (10) represents the kernel function used in the experiments of this study.

6 Experiments

The performance of the RLS ranker with the LC kernel proposed in this paper was compared to the baseline method, the RLS ranker described in [8]. Throughout our experiments we have been using BDB corpus which consists of 1100 annotated sentences. It was split into two datasets containing 500 and 600 sentences. The first dataset was used for the parameter estimation and the second one was reserved for the final validation. For each sentence, there is a certain amount of parses generated by the LG parser. Due to the computational complexity, we restricted the number of parses per sentence to 5 in training and to 20 in testing. When more parses were available for a sentence, we sampled randomly the necessary amount, when fewer were available, all parses were used.

We used the Kendall's correlation coefficient τ_b [10] as a performance measure in all experiments. The parse goodness function that determines the true ranks of the parses and the ranking procedure are described in [8].

The RLS algorithm has the regularization parameter λ that controls the trade-off between the minimization of the training error and the complexity of the regression function. In addition, the LC kernel uses θ parameter that determines the width of the position sensitive function and w, the size of the locality window, constructed around the matching features in both subparses. Finally, σ controls the width of the Gaussian function, into which the LC kernel is wrapped. The appropriate values of these parameters were determined by grid search with 10-fold cross-validation.

6.1 Evaluation of LC Kernel

The evaluation of the kernel function was conducted on the dataset consisting of 500 sentences. We observed a stable performance of the RLS ranker with the LC kernel providing a correlation of 0.45 against 0.42 of the RLS ranker reported

in [8]. The values of the parameters found by grid search are: $\lambda = 2^{-10}$, $w = 3$, $\sigma = 155$, and $\theta = 0.5$. The small value of θ for the LC kernel in equation (8) indicates that the positional matching of the features inside the locality window is more important contributor than the position insensitive one. The optimal size of the window appeared to be small. To find out whether some of the feature types would prefer longer windows, we also tried different sizes for different features. There was, however, no notable improvement in performance.

6.2 Final Validation

In order to test the statistical significance of the ranking performance difference between the two methods, we conducted a two-tailed paired t-test. The rankers were trained on the parameter estimation data and the 600 sentences reserved for the final validation were considered as independent trials. The performance of the RLS ranker with the LC kernel on the validation data is 0.46 and the improvement is statistically significant ($p < 0.05$) when compared to 0.42 correlation obtained using the RLS ranker as reported in [8].

7 Conclusions

This paper introduces the Locality-Convolution (LC) kernel and its application to the dependency parse ranking with Regularized Least-Squares algorithm. The proposed LC kernel uses feature sequences extracted in the order of the appearance in the parse, constructs local windows around matching features in the sequences in order to capture local correlations between the parses, and performs position sensitive evaluation of the features within the window. The usage of the LC kernel is not restricted for the parse ranking tasks, but can be applied everywhere where dependency structures, position sensitive matching, or local correlations play an important role.

We compared the ranking performance with the proposed kernel function to the results reported in our previous work [8]. The results show statistically significant improvement from 0.42 to 0.46 in correlation. A straightforward way to obtain even higher correlations is to increase the number of training examples. In [8], we show that by increasing the number of parses per sentence it is possible to achieve a better performance, however, the improvement obtained by increasing the number of sentences is larger.

In the future, we plan to investigate the task of dependency parse ranking by learning the ranks directly instead of regressing the parse goodness function. We also aim to consider the convolution properties of the LC kernel on a more basic level of features by conducting different projections and to explore graph kernels that are able to evaluate locality relations within dependency parse structures.

Acknowledgments

We would like to thank CSC, the Finnish IT center for science, for providing us extensive computing resources. This work has been supported by Tekes, the Finnish National Technology Agency.

References

1. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA (2004)
2. Scholkopf, B., Smola, A.J.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge, MA, USA (2001)
3. Collins, M., Duffy, N.: Convolution kernels for natural language. In Dietterich, T.G., Becker, S., Ghahramani, Z., eds.: NIPS, MIT Press (2001) 625–632
4. Suzuki, J., Hirao, T., Sasaki, Y., Maeda, E.: Hierarchical directed acyclic graph kernel: Methods for structured natural language data. In: ACL. (2003) 32–39
5. Suzuki, J., Isozaki, H., Maeda, E.: Convolution kernels with feature selection for natural language processing tasks. In: ACL. (2004) 119–126
6. Sleator, D.D., Temperley, D.: Parsing english with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA (1991)
7. Pyysalo, S., Ginter, F., Pahikkala, T., Boberg, J., Järvinen, J., Salakoski, T., Koivula, J.: Analysis of link grammar on biomedical dependency corpus targeted at protein-protein interactions. In Collier, N., Ruch, P., Nazarenko, A., eds.: Proceedings of the JNLPBA workshop at COLING'04, Geneva. (2004) 15–21
8. Tsivtsivadze, E., Pahikkala, T., Pyysalo, S., Boberg, J., Mylläri, A., Salakoski, T.: Regularized least-squares for parse ranking. In: Proceedings of the 6th International Symposium on Intelligent Data Analysis, Springer-Verlag (2005) 464–474 Copyright Springer-Verlag Berlin Heidelberg 2005.
9. Poggio, T., Smale, S.: The mathematics of learning: Dealing with data. Amer. Math. Soc. Notice **50** (2003) 537–544
10. Kendall, M.G.: Rank Correlation Methods. 4. edn. Griffin, London (1970)
11. Haussler, D.: Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz (1999)
12. Schölkopf, B., Herbrich, R., Smola, A.J.: A generalized representer theorem. In Helmbold, D., Williamson, R., eds.: Proceedings of the 14th Annual Conference on Computational Learning Theory and and 5th European Conference on Computational Learning Theory, Berlin, Germany, Springer-Verlag (2001) 416–426
13. Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., Watkins, C.J.C.H.: Text classification using string kernels. Journal of Machine Learning Research **2** (2002) 419–444
14. Pahikkala, T., Pyysalo, S., Ginter, F., Boberg, J., Järvinen, J., Salakoski, T.: Kernels incorporating word positional information in natural language disambiguation tasks. In Russell, I., Markov, Z., eds.: Proceedings of the Eighteenth International Florida Artificial Intelligence Research Society Conference, Menlo Park, Ca., AAAI Press (2005) 442–447 Copyright AAAI Press, http://www.aaai.org/.
15. Pahikkala, T., Pyysalo, S., Boberg, J., Mylläri, A., Salakoski, T.: Improving the performance of bayesian and support vector classifiers in word sense disambiguation using positional information. In Honkela, T., Könönen, V., Pöllä, M., Simula, O., eds.: Proceedings of the International and Interdisciplinary Conference on Adaptive Knowledge Representation and Reasoning, Espoo, Finland, Helsinki University of Technology (2005) 90–97
16. Zien, A., Ratsch, G., Mika, S., Scholkopf, B., Lengauer, T., Muller, K.R.: Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics **16** (2000) 799–807