Appl Intell
DOI 10.1007/s10489-008-0114-2

Locality kernels for sequential data and their applications to parse ranking

Evgeni Tsivtsivadze · Tapio Pahikkala · Jorma Boberg · Tapio Salakoski

Abstract We propose a framework for constructing kernels that take advantage of local correlations in sequential data. The kernels designed using the proposed framework measure parse similarities locally, within a small window constructed around each matching feature. Furthermore, we propose to incorporate positional information inside the window and consider different ways to do this. We applied the kernels together with regularized least-squares (RLS) algorithm to the task of dependency parse ranking using the dataset containing parses obtained from a manually annotated biomedical corpus of 1100 sentences. Our experiments show that RLS with kernels incorporating positional information perform better than RLS with the baseline kernel functions. This performance gain is statistically significant.

Keywords Kernel methods · Parse ranking · Regularized least-squares · Natural language processing

1 Introduction

With availability of structured data applicable for machine learning techniques, application of kernel methods (see e.g., [1–3]) 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. For example, Collins and Duffy [4] described convolution kernels for various discrete structures encountered in NLP tasks, which allow high dimensional representations of these structures. Our work has been motivated not only by rapidly developing field of 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 [5] which is applied to the BioInfer corpus [6] 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]. Thus, our task of parse ranking is to take the output of the LG parser and re-order the ranking.

Recently, we proposed a method for dependency parse ranking [8] that uses the regularized least-squares (RLS) algorithm [9] and grammatically motivated features. The RLS algorithm has performance comparable to the regular support vector machines (see e.g. [10]). Further, unlike for SVMs, there is an efficient method for searching the optimal regularization parameter for RLS with cross-validation [11]. Namely, training RLS with several different values of the regularization parameter can be performed as efficiently as training with only one value. Further, performing cross-validation with any fold partition is as efficient as training a single RLS.

Our method, called here the RLS ranker, worked notably better giving 0.42 correlation compared to 0.16 of the LG heuristics. Furthermore, we evaluated performance of the RLS ranker on different features separately as well as on

E. Tsivtsivadze (✉) · T. Pahikkala · J. Boberg · T. Salakoski
Turku Centre for Computer Science (TUCS), Department of Information Technology, University of Turku, Joukahaisenkatu 3-5 B, 20520 Turku, Finland
e-mail: firstname.lastname@it.utu.fi

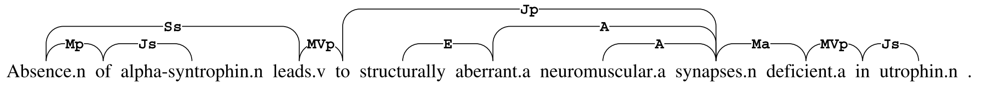
Jp
Ss
A
Mp
Js
MVp
E
A
Ma
MVp
Js
Absence.n of alpha-syntrophin.n leads.v to structurally aberrant.a neuromuscular.a synapses.n deficient.a in utrophin.n .

Fig. 1 Example of parsed sentence

their combination. We have also introduced the locality-convolution (LC) kernel [12] that takes into account positional information within the windows in the parses and further improves performance when used with RLS algorithm. The results showed statistically significant gain in parse ranking performance.

In this paper, we extend the results of [12] by considering a framework for constructing kernels that take advantage of local correlations in sequential data. As baselines we use the locality-improved [13] and spectrum [14] kernels which also are designed for the same kind of purposes. The LC kernel and the baseline kernels are presented in the framework proposed in this paper. In all experiments, we apply the F-score based parse goodness function [8], and evaluate the ranking performance with Kendall's correlation coefficient τ_b described in [15]. To be self contained, these measures are described also in this paper.

The LC kernel addresses the problem of parse ranking through the following characteristics. First, it possesses the convolution property described by Haussler [16], operating over discrete structures. Second, it calculates similarity between windows spanning 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 Sect. 2, we present grammatically motivated features for parse representation; in Sect. 3, we introduce the parse goodness function; in Sect. 4, we describe the RLS algorithm; in Sect. 5, we provide the performance measure applied to parse ranking; in Sect. 6, we discuss convolution kernels, define notions of locality windows, position sensitive feature matching, describe our baseline methods, and finally introduce the LC kernel; in Sect. 7, we evaluate the applicability of the LC kernel to the task of dependency parse ranking and benchmark it with respect to baseline methods; we conclude this paper in Sect. 8.

2 Features for parse representation

In the case of the parse ranking problem, where parses are represented within a 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 Fig. 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 Fig. 1, the extracted grammatical bigrams are *absence—of*, *absence—leads*, *of—alpha-syntrophin*, etc.

Word & POS tag feature contains the word with the POS tag assigned to the word by the LG parser. In Fig. 1, the extracted word & POS features are *absence.n*, *alpha-syntrophin.n*, *leads.v*, etc.

Link type feature represents the type of the link assigned by the LG parser. In the example, they are *Mp*, *Ss*, *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*, *of—Mp*, etc.

Link length feature represents the number of words that a link in the sentence spans. In Fig. 1, the extracted features of this type are *1*, *3*, *1*, etc.

Link length & Link type feature combines the type of the link in the sentence with the number of words it spans. In Fig. 1, the extracted features of this type are *1—Mp*, *3—Ss*, *1—Js*, etc.

Link bigram feature extracted from the parse 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*, *Ss—MVp*, etc. When there are more than two links associated with a word, we extract all link pairs. For example, all link bigrams that have word 'synapses' in common are *Jp—A*, *Jp—A*, *Jp—Ma*, *A—A*, *A—Ma*, *A—Ma*.

To measure the influence of individual features, in our previous study [8], we conducted an experiment where features were introduced to the ranker one by one. We observed that the word & link type was the best feature followed by the grammatical bigram and the link bigram features.

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 $p \in \mathcal{P}$, 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 Fig. 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 $\mathcal{P}$ to the feature space H, $\Phi : \mathcal{P} \rightarrow H$, representing parses through the sequences of the features as follows: $\Phi(p) = (p_1, \ldots, p_r)$. If we denote $p_j = (f_1^{t_j}, \ldots, f_{|p_j|}^{t_j})$, $1 \leq j \leq r$, then

$$\Phi(p) = ((\underbrace{f_1^{t_1}, \ldots, f_{|p_1|}^{t_1}}_{p_1}), \ldots, (\underbrace{f_1^{t_r}, \ldots, f_{|p_r|}^{t_r}}_{p_r})). \tag{1}$$

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 $p \in \mathcal{P}$.

3 F-score based goodness function for parses

The BioInfer corpus is a set of manually annotated sentences, that is, for each sentence of BioInfer, we have a manually annotated correct parse. We define a parse goodness function as

$$f^* : \mathcal{P} \mapsto \mathbb{R}_+$$

which measures the similarity of the parse $p \in \mathcal{P}$ with respect to its correct parse p^*. We propose an F-score based goodness function that assigns a goodness value to each parse based on information about the correct linkage structure. This function becomes the target output value that we try to predict with the RLS algorithm.

Let $L(p)$ denote the set of links with link types of a parse p. The links are considered to be equal if and only if they have the same link type and the indices of the words connected with the links are the same in the sentence in question.

The functions calculating numbers of true positives (TP), false positives (FP) and false negatives (FN) links with link types are defined as follows:

$$\mathrm{TP}(p) = | L(p) \cap L(p^*) |, \tag{2}$$

$$\mathrm{FP}(p) = | L(p) \setminus L(p^*) |, \tag{3}$$

$$\mathrm{FN}(p) = | L(p^*) \setminus L(p) |. \tag{4}$$

For example, let us consider beginning of the parse presented in Fig. 1, namely, starting from the word "Absence" and ending in the word "to". Then the $L(p^*) = \{1 - Mp - 2, 1 - Ss - 4, 2 - Js - 3, 4 - MVp - 5\}$ is the correct set of links with link types and consider $L(p) = \{1 - Ss - 3, 1 - Ss - 4, 2 - E - 3, 4 - MVp - 5\}$ to be the predicted one. Then $\mathrm{TP} = 2$ (correct links: $1 - Ss - 4, 4 - MVp - 5$), $\mathrm{FP} = 2$ (link in a wrong place: $1 - Ss - 3$, link with a wrong link type: $2 - E - 3$), and $\mathrm{FN} = 2$ (missing links: $1 - Mp - 2, 2 - Js - 3$).

We adopt one exception in (3) because of the characteristics of the corpus annotation. Namely, the corpus annotation does not have all links, which the corresponding LG linkage would have: for example, punctuation is not linked in the corpus. As a consequence, links in $L(p)$ having one end connected to a token without links in $L(p^*)$, are not considered in (3). The parse goodness function is defined as an F-score

$$f^*(p) = \frac{2\mathrm{TP}(p)}{2\mathrm{TP}(p) + \mathrm{FP}(p) + \mathrm{FN}(p)}. \tag{5}$$

High values of (5) indicate that a parse contains a small number of errors, and therefore, the bigger $f^*(p)$ is, the better is parse p.

Next we consider the regularized least-squares algorithm by which the measure f^* can be predicted.

4 Regularized least-squares algorithm

Let $\{(x_1, y_1), \ldots, (x_M, y_M)\}$, where $x_i \in \mathcal{P}, y_i \in \mathbb{R}$, 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]):

$$\min_f \sum_{i=1}^{M} l(f(x_i), y_i) + \lambda \| f \|_K^2, \tag{6}$$

where l is the loss function used by the learning machine, $f : \mathcal{P} \rightarrow \mathbb{R}$ 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 $\mathcal{P}$ can be any set, but in our problem, $\mathcal{P}$ is a set of parses of the sentences of the BioInfer 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. The second term in (6) 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., [17]), the minimizer of (6) 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 $\mathcal{P}$, 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 $\Phi : \mathcal{P} \to H$.

5 Performance measure for ranking

In this section, we present the performance measures used to evaluate the parse ranking methods. We follow Kendall's definition of rank correlation coefficient [15] and measure the degree of correspondence between the true ranking and the ranking output by an evaluated ranking method. If two rankings are equal, then correlation is $+1$, and on the other hand, if one ranking is the inverse of the other, correlation is -1.

The problem of the parse ranking can be formalized as follows. Let s be a sentence of BioInfer, and let $\mathcal{P}_s = \{v_1, \ldots, v_n\} \subseteq \mathcal{P}$ be the set of all parses of s produced by the LG parser. We apply the parse goodness function f^* to provide the target output variables for the parses by defining the following preference function

$$R_{f^*}(v_i, v_j) = \begin{cases} 1 & \text{if } f^*(v_i) > f^*(v_j), \\ -1 & \text{if } f^*(v_i) < f^*(v_j), \\ 0, & \text{otherwise} \end{cases}$$

which determines the ranking of the parses $v_i, v_j \in \mathcal{P}_s$. We also define a preference function $R_f(v_i, v_j)$ in a similar way for the regression function f learned by the RLS algorithm. In order to measure how well the ranking R_f is correlated with the target ranking R_{f^*}, we adopt Kendall's commonly used rank correlation measure τ. Let us define the score S_{ij} of a pair v_i and v_j to be the product

$$S_{ij} = R_f(v_i, v_j) R_{f^*}(v_i, v_j).$$

If score is $+1$, then the rankings agree on the ordering of v_i and v_j, otherwise score is -1. The total score is defined as

$$S = \sum_{1 \le i < j \le n} S_{ij}.$$

The number of all different pairwise comparisons of the parses of $\mathcal{P}_s$ that can be made is $\binom{n}{2} = \frac{1}{2} \cdot n(n-1)$. This corresponds to the maximum value of the total score, when agreement between the rankings is perfect. The correlation coefficient τ_a defined by Kendall is

$$\tau_a = \frac{S}{\frac{1}{2} \cdot n\,(n-1)}.$$

While τ_a is well applicable in many cases, there is an important issue that is not fully addressed by this coefficient—tied ranks, that is, $f^*(v_i) = f^*(v_j)$ or $f(v_i) = f(v_j)$ for some i, j. To take into account possible occurrences of tied ranks, Kendall proposes an alternative correlation coefficient

$$\tau_b = \frac{S}{\frac{1}{2}\sqrt{\sum_{i,j} R_{f^*}(v_i, v_j)^2 \cdot \sum_{i,j} R_f(v_i, v_j)^2}}, \tag{7}$$

where $1 \le i, j \le n$. With tied ranks the usage of τ_b is more justified than τ_a. For example, if both rankings are tied except the last member, then $\tau_b = 1$ indicating complete agreement between two rankings, while $\tau_a = \frac{2}{n}$. This can be demonstrated by the following simple example. Let $\mathcal{P}_s = \{v_1, \ldots, v_4\}$, $f^*(v_1) = \cdots = f^*(v_3) = 3$, $f^*(v_4) = 4$ and let the predicted values be $f(v_1) = \cdots = f(v_3) = 2$, $f(v_4) = 3$. Then it is easy to see that $S = 3$, and thus $\tau_a = \frac{1}{2} = \frac{2}{n}$, where $n = 4$. The denominator of τ_b is 3, and thus $\tau_b = 1$. Then for large values of n this measure is very close to 0, and therefore inappropriate. Due to many ties in the data, we use the correlation coefficient τ_b to evaluate performance of our ranker.

6 Kernels for sequential data

In this section, we first discuss the convolution kernels. Next, a framework for constructing kernels that take advantage of local correlations in sequential data is considered and the locality-convolution (LC) kernel is introduced. The LC kernel is based on the convolution framework as we demonstrate in Sect. 6.2. Finally, we present the locality-improved and spectrum kernels that we use as baselines.

6.1 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 [16], we briefly describe the convolution kernel framework. Let us consider $x \in X$ as a composite structure such that $x_1, \dots, x_N$ 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 $X_1, \dots, X_n$ as countable sets, however, they can be more general separable metric spaces [16]. Let us denote shortly $\widehat{x} = x_1, \dots, x_N$. Then the relation "$x_1, \dots, x_N$ are parts of x" can be expressed as a relation R on the set $X_1 \times \cdots \times X_N \times X$ such that $R(\widehat{x}, x)$ is true if $\widehat{x}$ are the parts of x. Then we 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 $\widehat{x} = x_1, \dots, x_N$ are the parts of x and $\widehat{y} = y_1, \dots, y_N$ are the parts of y. If we have some kind of kernel functions

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

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:

$$K(x, y) = \sum_{\widehat{x} \in R^{-1}(x)} \sum_{\widehat{y} \in R^{-1}(y)} \prod_{n=1}^{N} K_n(x_n, y_n). \tag{8}$$

There have been several different convolution kernels reported and applied in NLP, for example, string kernel [18], word-sequence kernel [19], tree kernels (see e.g., [4, 20]) and graph kernels (see e.g., [21–23]). Furthermore, general examples of convolution kernels can be found in [16] as well. 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 Sect. 2.

6.2 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 $p, q \in \mathcal{P}$ and let $p_j = (f_1^{t_j}, \dots, f_{|p_j|}^{t_j})$ and $q_j = (g_1^{t_j}, \dots, g_{|q_j|}^{t_j})$ be their subparses consisting of the features of the same type t_j as described in Sect. 2. Next we consider how to define a similarity between the subparses p_j and q_j. For simplicity, we omit superscript t_j in the features of subparses, because we consider them to belong to the type t_j. At each position we compare subparses locally within a small window of length $2w + 1$. The similarity of the subparses p_j and q_j is obtained with the kernel

$$G(p_j, q_j) = \sum_{i=1}^{|p_j|} \sum_{k=1}^{|q_j|} \kappa(i, k), \tag{9}$$

where κ is a kernel. To measure the similarity between whole parses p and q, we measure the correspondence of their subparses within each feature type and then sum them up:

$$K(p, q) = \sum_{j=1}^{r} G(p_j, q_j), \tag{10}$$

where r is the number of the feature types.

By defining $\kappa(i, k)$ in the general formulation (9), we obtain different similarity functions between parses. If we set $\kappa(i, k) = \delta(f_i, g_k)$, where

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

then (9) equals to the number of matching features in the two subparses. To pay attention also to the nearby features of the matching features within a certain locality window centered at the matching features, we use an appropriate real valued $(2w + 1) \times (2w + 1)$ matrix P in the definition of $\kappa(i, k)$. The purpose of P is to weight the matches within the windows with respect to the positions of the matches (see [24–26] for a similar approach). However, here we do not weight the individual positions within one window as with the baseline kernels described in Sect. 6.3. Instead, we define weights for all position pairs within two windows compared.

A simple realization of this idea would be

$$\kappa(i, k) = \delta(f_i, g_k) \sum_{m,l=-w}^{w} [P]_{m,l} \delta(f_{i+m}, g_{k+l}). \tag{11}$$

Note that the rows and the columns of the positional weight matrix P are indexed from $-w$ to w. Furthermore, we consider features as mismatched when the indices $i + m$ and $g + k$ are not valid, e.g. $i + m < 1$ or $i + m > |p_j|$.

When we set $P = A$, where A is a matrix whose all elements are ones, we get κ that counts the matching features in the two windows. This corresponds to the case when the order of the features inside the windows is disregarded. As another alternative, we can construct a function that requires the matching feature positions to be exactly the same. We obtain it by $P = I$, where I denotes the identity matrix.

Furthermore, when P is a diagonal matrix whose elements are weights increasing from the boundary to the center of the window, we obtain a kernel that is related to the locality improved kernel proposed in [13]. However, if we do not require strict position matching, but rather penalize the features that match but have a different position within the windows, one can use a positional similarity matrix whose off-diagonal elements are nonzero and smaller than the diagonal elements. We can obtain such a matrix, for example, by

$$[P]_{m,l} = e^{-\frac{(m-l)^2}{2\theta^2}}, \tag{12}$$

where $\theta \geq 0$ is a parameter.

Furthermore, we define the following multiplicative version of the κ function:

$$\kappa(i,k) = \delta(f_i, g_k) \prod_{m,l=-w}^{w} \left([P]_{m,l}\delta(f_{i+m}, g_{k+l}) + 1\right). \tag{13}$$

The choice of an appropriate κ is a matter closely related to the domain of the study. In Sect. 7, we show that positional information captured with (13) is useful and improves the ranking performance. When using (13) with (12) in (9), we obtain the kernel which we call the locality-convolution (LC) kernel. The proposed function is indeed a valid kernel due to positive semidefiniteness of matrix P and kernel closure properties. Conceptually, our kernel enumerates all the substructures representing pairs of windows built around the matching features in the subparses and calculates their similarity. 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 suitable for the task.

Let us consider the LC kernel from convolution perspective by using the notation introduced in Sect. 6.1 and defining the set X and the relation R. Let X be the set of all subparses. We define $R(\widehat{x}, x)$ to be true iff $\widehat{x}$ is a window (as defined in Sect. 6.2) of the subparse x. Then $R^{-1}(x) = \{\widehat{x} : R(\widehat{x}, x)\}$ is the set of all the windows of x. We observe the analogy in calculation of subparse similarity by the LC kernel and (8) assuming in the latter $N = 1$.

6.3 Related kernels

A sequence kernel called the locality-improved kernel described in [13] concerns recognition of so called translation initial sites (TIS) in the sequence. Zien et al. [13] proposed incorporating basic biological hypothesis that while certain local correlations are relevant in recognition, dependencies between distant positions are of the minor importance or even do not exist. By incorporating this knowledge into feature space one can expect to obtain better results, rather than with general kernels. Using the notations defined in Sect. 6.2, we can formulate the locality-improved kernel as

$$G(p_j, q_j) = \left(\sum_{i=1}^{n} \kappa(i)\right)^{d_2}, \tag{14}$$

where

$$\kappa(i) = \left(\sum_{m=-w}^{w} [P]_{m,m}\delta(f_{i+m}, g_{i+m})\right)^{d_1}, \tag{15}$$

$d_1, d_2, n \in \mathbb{N}_+$ and $P = D$ with all except diagonal elements equal to 0. The locality-improved kernel (14) is not applicable in the situations where sequences of different length are being compared. We use a straightforward modification of (14) that allows its application to sequences of different length:

$$G(p_j, q_j) = \left(\sum_{i=1}^{|p_j|}\sum_{k=1}^{|q_j|} \kappa(i,k)\right)^{d_2}, \tag{16}$$

where

$$\kappa(i,k) = \left(\sum_{m=-w}^{w} [P]_{m,m}\delta(f_{i+m}, g_{k+m})\right)^{d_1}. \tag{17}$$

The difference between (14) and (16) is that in the former the centers of the two compared windows are always in the same position, while they can be in arbitrary positions in the latter. Nonzero elements of D are defined to increase from the boundaries to the center of the window. Thus, sequences are compared locally at each position by constructing a small window of length $2w + 1$, and the matching features are counted and multiplied with weights $[D]_{m,m}$.

Next we consider the spectrum kernel introduced in [14] (see also [1]). When comparing two sequences, a natural way is to count the common contiguous subsequences that are contained in both of them. Informally, the spectrum kernel of an order h can be considered as a histogram of frequencies of all its contiguous substrings of length h. In other words, the h-spectrum kernel function is an inner product of the h-spectra of the two evaluated sequences. The spectrum kernel is obtained by using

$$\kappa(i,k) = \prod_{l=1}^{h} \delta(f_{i+l}, g_{k+l}), \tag{18}$$

in (9). In our parse space the feature space generated by (18) is very sparse. Therefore, instead of using (9), we sum up all spectrum kernels up to window length h:

$$G(p_j, q_j) = \sum_{u=1}^{h}\sum_{i=1}^{|p_j|}\sum_{k=1}^{|q_j|} \kappa(i,k) \prod_{l=1}^{h} \delta(f_{i+l}, g_{k+l}). \tag{19}$$

7 Experiments

Throughout our experiments we have been using the BioInfer corpus which consists of 1100 annotated sentences. It

Table 1 Results of the parameter estimation and final validation experiments

Kernel	Positional matrix	Best parameters on training data	Performance on training data (τ_b)	Performance on test data (τ_b)
(11)	$P = A$	$\lambda = 2^{-2}, w = 1$	0.396	
	$[P]_{m,l} = e^{-\frac{(m-l)^2}{2\theta^2}}$	$\lambda = 2^4, w = 1, \theta = 0.5$	0.421	
	$P = I$	$\lambda = 2^5, w = 1$	0.413	
LC (13)	$P = A$	$\lambda = 2^1, w = 1$	0.384	
	$[P]_{m,l} = e^{-\frac{(m-l)^2}{2\theta^2}}$	$\lambda = 2^4, w = 1, \theta = 0.3$	0.423	0.447
	$P = I$	$\lambda = 2^{-3}, w = 1$	0.406	
Locality-improved (16)	$P = D$	$\lambda = 2^6, w = 1, d_1 = 2, d_2 = 1$	0.387	0.432
Spectrum (19)		$\lambda = 2^4, h = 2$	0.378	0.410

is divided into two datasets containing 500 and 600 sentences. The first dataset is used for the parameter estimation and the second one is reserved for the final validation. For each sentence, there is a certain number 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 are available for a sentence, we sampled randomly the necessary amount; when fewer are available, all parses are used. We used the Kendall's correlation coefficient τ_b defined in (7) as a performance measure in all experiments. The parse goodness function that determines the true ranks of the parses and the ranking procedure is described in Sect. 3.

7.1 Parameter estimation

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 the θ parameter that determines the width of the position sensitive function and w, the size of the locality window, constructed around the matching features. Finally, the locality-improved and spectrum kernel have parameters described in Sect. 6.3. The d_2 parameter of the locality-improved kernel is set to 1 as suggested by [13], and $D_{m,l} = 1 + w - |m|$ when $m = l$, and $D_{m,l} = 0$ otherwise. The appropriate values of all other parameters are determined by grid search with 10-fold cross-validation.

The evaluation of the kernel functions (11), (13), (16) and (19) is conducted on the dataset consisting of 500 sentences. Observed performance of the RLS ranker is reported in Table 1. The positional weight matrices P are presented in Sect. 6.2. The results show that when order of the features is ignored (i.e. $P = A$), the performance of kernels (11) and (13) is clearly worse compared to the case when the order is taken into account. When considering the kernels (11) and (13), we observe that the best values for θ are small indicating that the positional matching of the features inside the locality window (i.e. $P \neq A$) is an important contributor to the ranking performance. The results of Table 1 show that the optimal size of the window is small. To find out whether some of the feature types would prefer longer windows, we also tried different window sizes for different features. There was, however, no notable improvement in performance.

7.2 Final validation

We compare the LC kernel (13) with the locality-improved kernel (16) and the spectrum kernel (19). In order to test the statistical significance of the ranking performance difference between the baseline methods, we conduct Wilcoxon signed-ranks test. The RLS rankers are trained on the parameter estimation data using parameter set found during training. The 600 sentences reserved for the final validation are considered as independent trials. The performance of the RLS ranker with the LC kernel on the validation data is 0.447 and the improvement is statistically significant ($p < 0.05$) when compared to 0.410 and 0.432 correlation obtained using the RLS ranker with the spectrum and the locality-improved kernel, respectively.

8 Conclusions

In this paper, we propose a framework for constructing locality kernels that take advantage of the correlations in sequential data. The kernels designed using the framework are applied to the task of dependency parse ranking with regularized least-squares algorithm. These kernels use feature sequences extracted in the order of the appearance from the parse, construct local windows around matching features in the sequences in order to capture local correlations between the parses, and perform a position sensitive (or insensitive) evaluation of the features within the window. The usage of the proposed locality kernels is not restricted for the parse ranking tasks, but can be applied to sequential data where positional matching, or local correlations play an important role.

We evaluate the kernels against other kernel functions applicable to sequential data, namely the locality-improved and the spectrum. Final validation results demonstrate that the performance gain is statistically significant. We have also conducted experiments with gap-weighted subsequences kernel (see e.g., [1]). However, we were not able to perform a parameter selection due to the computational complexity of the kernel.

We have already proposed graph kernels and graph representations for dependency parses that can be used when addressing the ranking task [23]. The obtained results are encouraging and the study provides possible further directions. In the future, we also plan to investigate the task of dependency parse ranking by learning the ranks directly instead of regressing the parse goodness function. Moreover, we plan to use the locality kernels for biomedical problems.

Acknowledgements 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 Funding Agency for Technology and Innovation.

References

1. Shawe-Taylor J, Cristianini N (2004) Kernel methods for pattern analysis. Cambridge University Press, New York
2. Scholkopf B, Smola AJ (2001) Learning with kernels: Support vector machines, regularization, optimization, and beyond. MIT Press, Cambridge
3. Herbrich R (2002) Learning kernel classifiers: theory and algorithms. MIT Press, Cambridge
4. Collins M, Duffy N (2001) Convolution kernels for natural language. In: Dietterich TG, Becker S, Ghahramani Z (eds) NIPS. MIT Press, Cambridge, pp 625–632
5. Sleator DD, Temperley D (1991) Parsing English with a link grammar. Technical Report CMU-CS-91-196, Department of Computer Science, Carnegie Mellon University, Pittsburgh, PA
6. Pyysalo S, Ginter F, Heimonen J, Björne J, Boberg J, Järvinen J, Salakoski T (2007) BioInfer: A corpus for information extraction in the biomedical domain. BMC Bioinformatics 8:50. The corpus is available at http://www.it.utu.fi/BioInfer
7. Pyysalo S, Ginter F, Pahikkala T, Boberg J, Järvinen J, Salakoski T, Koivula J (2004) 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, pp 15–21
8. Tsivtsivadze E, Pahikkala T, Pyysalo S, Boberg J, Mylläri A, Salakoski T (2005) Regularized least-squares for parse ranking. In: Proceedings of the 6th international symposium on intelligent data analysis. Springer, Berlin, pp 464–474
9. Poggio T, Smale S (2003) The mathematics of learning: Dealing with data. Am Math Soc Not 50:537–544
10. Rifkin R (2002) Everything old is new again: A fresh look at historical approaches in machine learning. PhD thesis, MIT
11. Pahikkala T, Boberg J, Salakoski T (2006) Fast n-fold cross-validation for regularized least-squares. In: Honkela T, Raiko T, Kortela J, Valpola H (eds) Proceedings of the 9th Scandinavian conference on artificial intelligence (SCAI 2006), Espoo, Finland, Otamedia Oy, pp 83–90
12. Tsivtsivadze E, Pahikkala T, Boberg J, Salakoski T (2006) Locality-convolution kernel and its application to dependency parse ranking. In: Ali M, Dapoigny R (eds) IEA/AIE. Lecture notes in computer science, vol 4031. Springer, Berlin, pp 610–618
13. Zien A, Ratsch G, Mika S, Scholkopf B, Lengauer T, Muller KR (2000) Engineering support vector machine kernels that recognize translation initiation sites. Bioinformatics 16:799–807
14. Leslie CS, Eskin E, Noble WS (2002) The spectrum kernel: A string kernel for svm protein classification. In: Pacific symposium on biocomputing, pp 566–575
15. Kendall MG (1970) Rank correlation methods, 4th edn. Griffin, London
16. Haussler D (1999) Convolution kernels on discrete structures. Technical Report UCSC-CRL-99-10, UC Santa Cruz
17. Schölkopf B, Herbrich R, Smola AJ (2001) A generalized representer theorem. In: Helmbold D, Williamson R (eds) Proceedings of the 14th annual conference on computational learning theory and 5th European conference on computational learning theory. Springer, Berlin, pp 416–426
18. Lodhi H, Saunders C, Shawe-Taylor J, Cristianini N, Watkins CJCH (2002) Text classification using string kernels. J Mach Learn Res 2:419–444
19. Cancedda N, Gaussier E, Goutte C, Renders JM (2003) Word-sequence kernels. J Mach Learn Res 3:1059–1082
20. Moschitti A (2006) Making tree kernels practical for natural language learning. In: 11st Conference of the European chapter of the association for computational linguistics. The Association for Computer Linguistics
21. Gärtner T, Flach PA, Wrobel S (2003) On graph kernels: Hardness results and efficient alternatives. In: Schölkopf B, Warmuth MK (eds) 16th annual conference on computational learning theory and 7th kernel workshop (COLT-2003). Lecture notes in computer science, vol 2777. Springer, Berlin, pp 129–143
22. Suzuki J, Isozaki H, Maeda E (2004) Convolution kernels with feature selection for natural language processing tasks. In: ACL, pp 119–126
23. Pahikkala T, Tsivtsivadze E, Boberg J, Salakoski T (2006) Graph kernels versus graph representations: a case study in parse ranking. In: Gärtner T, Garriga GC, Meinl T (eds) Proceedings of the ECML/PKDD'06 workshop on mining and learning with graphs (MLG'06)
24. Pahikkala T, Pyysalo S, Ginter F, Boberg J, Järvinen J, Salakoski T (2005) Kernels incorporating word positional information in natural language disambiguation tasks. In: Russell I, Markov Z (eds) Proceedings of the 18th international Florida artificial intelligence research society conference, Menlo Park, CA. AAAI Press, Menlo Park, pp 442–447
25. Pahikkala T, Pyysalo S, Boberg J, Mylläri A, Salakoski T (2005) 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, pp 90–97
26. Pahikkala T, Boberg J, Mylläri A, Salakoski T (2006) Incorporating external information in Bayesian classifiers via linear feature transformations. In: Salakoski T, Ginter F, Pyysalo S, Pahikkala T (eds) Proceedings of the 5th international conference on natural language processing FinTAL 06, Turku, Finland. Lecture notes in artificial intelligence, vol 4139. Springer, Heidelberg, pp 399–410