Discriminative Reranking of Discourse Parses Using Tree Kernels

Shafiq Joty and **Alessandro Moschitti**
ALT Research Group
Qatar Computing Research Institute
{sjoty,amoschitti}@qf.org.qa

Abstract

In this paper, we present a discriminative approach for reranking discourse trees generated by an existing probabilistic discourse parser. The reranker relies on tree kernels (TKs) to capture the global dependencies between discourse units in a tree. In particular, we design new computational structures of discourse trees, which combined with standard TKs, originate novel discourse TKs. The empirical evaluation shows that our reranker can improve the state-of-the-art sentence-level parsing accuracy from 79.77% to 82.15%, a relative error reduction of 11.8%, which in turn pushes the state-of-the-art document-level accuracy from 55.8% to 57.3%.

1 Introduction

Clauses and sentences in a well-written text are interrelated and exhibit a coherence structure. Rhetorical Structure Theory (RST) (Mann and Thompson, 1988) represents the coherence structure of a text by a labeled tree, called discourse tree (DT) as shown in Figure 1. The leaves correspond to contiguous clause-like units called elementary discourse units (EDUs). Adjacent EDUs and larger discourse units are hierarchically connected by coherence relations (e.g., ELABORATION, CAUSE). Discourse units connected by a relation are further distinguished depending on their relative importance: *nuclei* are the core parts of the relation while *satellites* are the supportive ones.

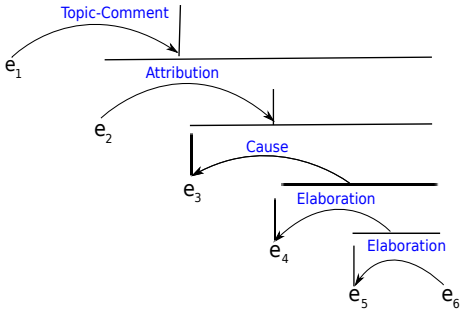
Conventionally, discourse analysis in RST involves two subtasks: (*i*) *discourse segmentation:* breaking the text into a sequence of EDUs, and (*ii*) *discourse parsing:* linking the discourse units to form a labeled tree. Despite the fact that discourse analysis is central to many NLP applications, the state-of-the-art document-level discourse parser (Joty et al., 2013) has an f-score of only 55.83% using manual discourse segmentation on the RST Discourse Treebank (RST-DT).

Although recent work has proposed rich linguistic features (Feng and Hirst, 2012) and powerful parsing models (Joty et al., 2012), discourse parsing remains a hard task, partly because these approaches do not consider global features and long range structural dependencies between DT constituents. For example, consider the human-annotated DT (Figure 1a) and the DT generated by the discourse parser of Joty et al. (2013) (Figure 1b) for the same text. The parser makes a mistake in finding the right structure: it considers only e_3 as the text to be attributed to e_2, where all the text spans from e_3 to e_6 (linked by CAUSE and ELABORATION) compose the statement to be attributed. Such errors occur because existing systems do not encode long range dependencies between DT constituents such as those between e_3 and e_{4-6}.

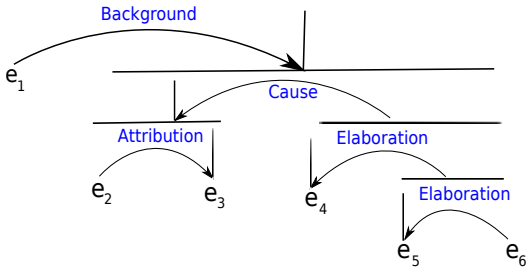
Reranking models can make the global structural information available to the system as follows: first, a base parser produces several DT hypotheses; and then a classifier exploits the entire information in each hypothesis, e.g., the complete DT with its dependencies, for selecting the best DT. Designing features capturing such global properties is however not trivial as it requires the selection of important DT fragments. This means selecting subtree patterns from an exponential feature space. An alternative approach is to implicitly generate the whole feature space using tree kernels (TKs) (Collins and Duffy, 2002; Moschitti, 2006).

In this paper, we present reranking models for discourse parsing based on Support Vector Machines (SVMs) and TKs. The latter allows us to represent structured data using the substructure space thus capturing structural dependencies between DT constituents, which is essential for effective discourse parsing. Specifically, we made the following contributions. First, we extend the

Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 2049–2060,
October 25-29, 2014, Doha, Qatar.



(a) A human-annotated discourse tree.



(b) A discourse tree generated by Joty et al. (2013).

Figure 1: Example of human-annotated and system-generated discourse trees for the text *[what's more,]$_{e_1}$ [he believes]$_{e_2}$ [seasonal swings in the auto industry this year aren't occurring at the same time in the past,]$_{e_3}$ [because of production and pricing differences]$_{e_4}$ [that are curbing the accuracy of seasonal adjustments]$_{e_5}$] [built into the employment data.]$_{e_6}$* Horizontal lines indicate text segments; satellites are connected to their nuclei by curved arrows.

existing discourse parser[1] (Joty et al., 2013) to produce a list of k most probable parses for each input text, with associated probabilities that define the initial ranking.

Second, we define a set of discourse tree kernels (DISCTK) based on the functional composition of standard TKs with structures representing the properties of DTs. DISCTK can be used for any classification task involving discourse trees.

Third, we use DISCTK to define kernels for reranking and use them in SVMs. Our rerankers can exploit the complete DT structure using TKs. They can ascertain if portions of a DT are compatible, incompatible or simply not likely to coexist, since each substructure is an exploitable feature. In other words, problematic DTs are expected to be ranked lower by our reranker.

Finally, we investigate the potential of our approach by computing the oracle f-scores for both document- and sentence-level discourse parsing. However, as demonstrated later in Section 6, for *document-level* parsing, the top k parses often miss the best parse. For example, the oracle f-scores for 5- and 20-best document-level parsing are only 56.91% and 57.65%, respectively. Thus the scope of improvement for the reranker is rather narrow at the document level. On the other hand, the oracle f-score for 5-best *sentence-level* discourse parsing is 88.09%, where the base parser (i.e., 1-best) has an oracle f-score of 79.77%. Therefore, in this paper we address the following two questions: (*i*) how far can a reranker improve the parsing accuracy at the sentence level? and (*ii*) how far can this improvement, if at all, push the (combined) document-level parsing accuracy?

To this end, our comparative experiments on RST-DT show that the sentence-level reranker can improve the f-score of the state-of-the-art from 79.77% to 82.15%, corresponding to a relative error reduction of 11.8%, which in turn pushes the state-of-the-art document-level f-score from 55.8% to 57.3%, an error reduction of 3.4%.

In the rest of the paper, after introducing the TK technology in Section 2, we illustrate our novel structures, and how they lead to the design of novel DISCTKs in Section 3. We present the k-best discourse parser in Section 4. In Section 5, we describe our reranking approach using DISCTKs. We report our experiments in Section 6. We briefly overview the related work in Section 7, and finally, we summarize our contributions in Section 8.

2 Kernels for Structural Representation

Tree kernels (Collins and Duffy, 2002; Shawe-Taylor and Cristianini, 2004; Moschitti, 2006) are a viable alternative for representing arbitrary subtree structures in learning algorithms. Their basic idea is that kernel-based learning algorithms, e.g., SVMs or perceptron, only need the scalar product between the feature vectors representing the data instances to learn and classify; and kernel functions compute such scalar products in an efficient way. In the following subsections, we briefly describe the kernel machines and three types of tree kernels (TKs), which efficiently compute the scalar product in the subtree space, where the vector components are all possible substructures of the corresponding trees.

2.1 Kernel Machines

Kernel Machines (Cortes and Vapnik, 1995), e.g., SVMs, perform binary classification by learning a hyperplane $H(\vec{x}) = \vec{w} \cdot \vec{x} + b = 0$, where

[1] Available from http://alt.qcri.org/tools/

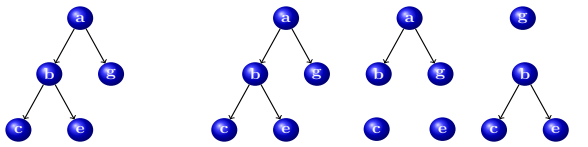


Figure 2: A tree with its STK subtrees; STK$_b$ also includes leaves as features.

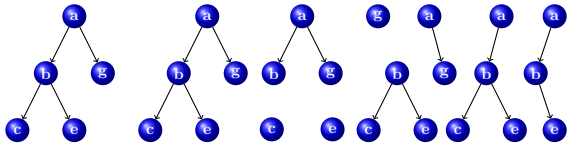


Figure 3: A tree with its PTK fragments.

$\vec{x} \in \mathbb{R}^n$ is the feature vector representation of an object $o \in \mathcal{O}$ to be classified and $\vec{w} \in \mathbb{R}^n$ and $b \in \mathbb{R}$ are parameters learned from the training data. One can train such machines in the dual space by rewriting the model parameter $\vec{w}$ as a linear combination of training examples, i.e., $\vec{w} = \sum_{i=1..l} y_i \alpha_i \vec{x}_i$, where y_i is equal to 1 for positive examples and -1 for negative examples, $\alpha_i \in \mathbb{R}_+$ and $\vec{x}_i \forall i \in \{1, .., l\}$ are the training instances. Then, we can use the data object $o_i \in \mathcal{O}$ directly in the hyperplane equation considering their mapping function $\phi : \mathcal{O} \rightarrow \mathbb{R}^n$, as follows: $H(o) = \sum_{i=1..l} y_i \alpha_i \vec{x}_i \cdot \vec{x} + b = \sum_{i=1..l} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b = \sum_{i=1..l} y_i \alpha_i K(o_i, o) + b$, where the product $K(o_i, o) = \langle \phi(o_i) \cdot \phi(o) \rangle$ is the kernel function (e.g., TK) associated with the mapping ϕ.

2.2 Tree Kernels

Convolution TKs compute the number of common tree fragments between two trees T_1 and T_2 without explicitly considering the whole fragment space. A TK function over T_1 and T_2 is defined as: $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the nodes of T_1 and T_2, respectively, and $\Delta(n_1, n_2)$ is equal to the number of common fragments rooted in the n_1 and n_2 nodes.[2] The computation of Δ function depends on the shape of fragments, conversely, a different Δ determines the richness of the kernel space and thus different tree kernels. In the following, we briefly describe two existing and well-known tree kernels. Please see several tutorials on kernels (Moschitti, 2013; Moschitti, 2012; Moschitti, 2010) for more details.[3]

Syntactic Tree Kernels (STK) produce fragments such that each of their nodes includes all or none of its children. Figure 2 shows a tree T and its three fragments (do not consider the single nodes) in the STK space on the left and right of the arrow, respectively. STK(T,T) counts the number of common fragments, which in this case is the number of subtrees of T, i.e., three. In the figure, we also show three single nodes, c, e, and g, i.e., the leaves of T, which are computed by a variant of the kernel, that we call STK$_b$. The computational complexity of STK is $O(|N_{T_1}||N_{T_2}|)$, but the average running time tends to be linear (i.e. $O(|N_{T_1}| + |N_{T_2}|)$) for syntactic trees (Moschitti, 2006).

Partial Tree Kernel (PTK) generates a richer set of tree fragments. Given a target tree T, PTK can generate any subset of connected nodes of T, whose edges are in T. For example, Figure 3 shows a tree with its nine fragments including all single nodes (i.e., the leaves of T). PTK is more general than STK as its fragments can include any subsequence of children of a target node. The time complexity of PTK is $O(p\rho^2|N_{T_1}||N_{T_2}|)$, where p is the largest subsequence of children that one wants to consider and ρ is the maximal out-degree observed in the two trees. However, the average running time again tends to be linear for syntactic trees (Moschitti, 2006).

3 Discourse Tree Kernels (DISCTK)

Engineering features that can capture the dependencies between DT constituents is a difficult task. In principle, any dependency between words, relations and structures (see Figure 1) can be an important feature for discourse parsing. This may lead to an exponential number of features, which makes the feature engineering process very hard.

The standard TKs described in the previous section serve as a viable option to get useful subtree features automatically. However, the definition of the input to a TK, i.e., the tree representing a training instance, is extremely important as it implicitly affects the subtree space generated by the TK, where the target learning task is carried out. This can be shown as follows. Let $\phi_M()$ be a mapping from linguistic objects o_i, e.g., a discourse parse, to a *meaningful* tree T_i, and let $\phi_{TK}()$ be a mapping into a tree kernel space us-

[2] To get a similarity score between 0 and 1, it is common to apply a normalization in the kernel space, i.e. $\frac{TK(T_1,T_2)}{\sqrt{TK(T_1,T_1) \times TK(T_2,T_2)}}$.

[3] Tutorials notes available at `http://disi.unitn.it/moschitti/`

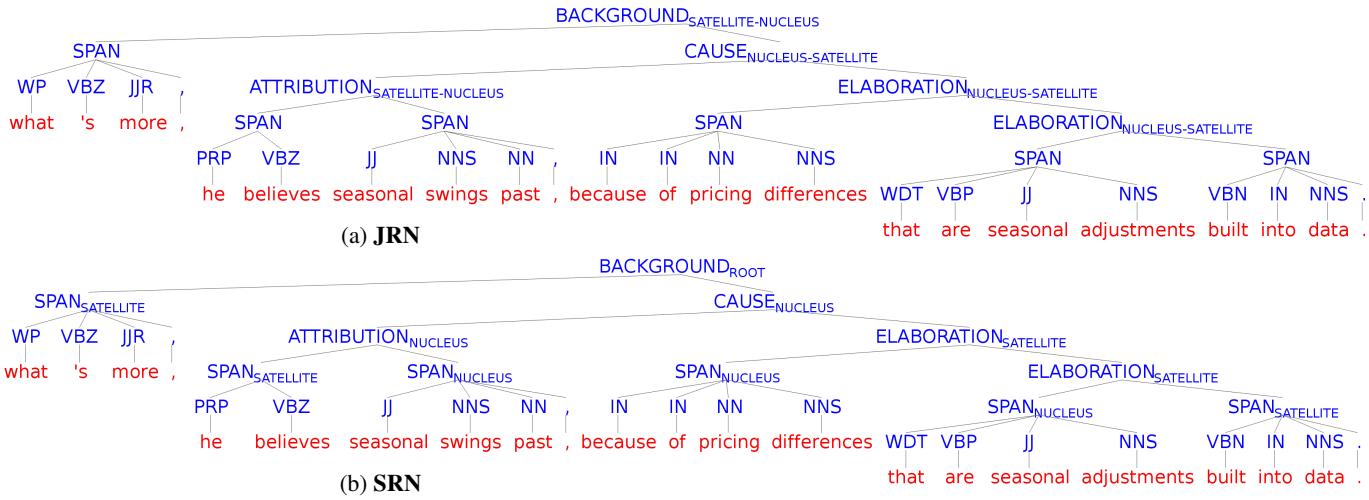


Figure 4: DISCTK trees: (a) Joint Relation-Nucleus (JRN), and (b) Split Relation Nucleus (SRN).

ing one of the TKs described in Section 2.2, i.e., $TK(T_1, T_2) = \phi_{TK}(T_1) \cdot \phi_{TK}(T_2)$. If we apply TK to the objects o_i transformed by $\phi_M()$, we obtain $TK(\phi_M(o_1), \phi_M(o_2)) = \phi_{TK}(\phi_M(o_1)) \cdot \phi_{TK}(\phi_M(o_2)) = \big(\phi_{TK} \circ \phi_M\big)(o_1) \cdot \big(\phi_{TK} \circ \phi_M\big)(o_2) = DiscTK(o_1, o_2)$, which is a new kernel[4] induced by the mapping $\phi_{DiscTK} = \big(\phi_{TK} \circ \phi_M\big)$.

We define two different mappings ϕ_M to transform the discourse parses generated by the base parser into two different tree structures: (*i*) the Joint Relation-Nucleus tree (JRN), and (*ii*) the Split Relation Nucleus tree (SRN).

3.1 Joint Relation-Nucleus Tree (JRN)

As shown in Figure 4a, JRN is a direct mapping of the parser output, where the nuclearity statuses (i.e., satellite or nucleus) of the connecting nodes are attached to the relation labels.[5] For example, the root BACKGROUND$_{Satellite-Nucleus}$ in Figure 4a denotes a *Background* relation between a *satellite* discourse unit on the left and a *nucleus* unit on the right. Text spans (i.e., EDUs) are represented as sequences of Part-of-Speech (POS) tags connected to the associated words, and are grouped under dummy SPAN nodes. We experiment with two lexical variations of the trees: (*i*) *All* includes all the words in the EDU, and (*ii*) *Bigram* includes only the first and last two words in the EDU.

When JRN is used with the STK kernel, an exponential number of fragments are generated. For example, the upper row of Figure 5 shows two smallest (atomic) fragments and one subtree composed of two atomic fragments. Note that much larger structures encoding long range dependencies are also part of the feature space. These fragments can reveal if the discourse units are organized in a compatible way, and help the reranker to detect the kind of errors shown earlier in Figure 1b. However, one problem with JRN representation is that since the relation nodes are composed of three different labels, the generated subtrees tend to be sparse. In the following, we describe SRN that attempts to solve this issue.

3.2 Split Relation Nucleus Tree (SRN)

SRN is not very different from JRN as shown in Figure 4b. The only difference is that instead of attaching the nuclearity statuses to the relation labels, in this representation we assign them to their respective discourse units. When STK kernel is applied to SRN it again produces an exponential number of fragments. For example, the lower row of Figure 5 shows two atomic fragments and one subtree composed of two atomic fragments. Comparing the two examples in Figure 5, it is easy to understand that the space of subtrees extracted from SRN is less sparse than that of JRN.

Note that, as described in Secion 2.2, when the PTK kernel is applied to JRN and SRN trees, it can generate a richer feature space, e.g., features that are paths containing relation labels (e.g., BACKGROUND - CAUSE - ELABORATION or ATTRIBUTION - CAUSE - ELABORATION).

4 Generation of k-best Discourse Parses

In this section we describe the 1-best discourse parser of Joty et al. (2013), and how we extend

[4] People interested in algorithms may like it more designing a complex algorithm to compute $\big(\phi_{TK} \circ \phi_M\big)$. However, the design of ϕ_M is conceptually equivalent and more effective from an engineering viewpoint.

[5] This is a common standard followed by the parsers.

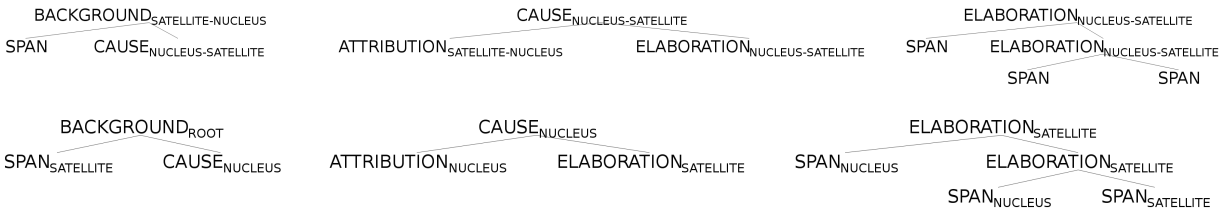


Figure 5: Fragments from JRN in Figure 4a (upper row) and SRN in Figure 4b (lower row).

it to k-best discourse parsing.

Joty et al. (2013) decompose the problem of document-level discourse parsing into two stages as shown in Figure 6. In the first stage, the *intra-sentential* discourse parser produces discourse subtrees for the individual sentences in a document. Then the *multi-sentential* parser combines the sentence-level subtrees and produces a DT for the document. Both parsers have the same two components: a *parsing model* and a *parsing algorithm*. The parsing model explores the search space of possible DTs and assigns a probability to every possible DT. Then the parsing algorithm selects the most probable DT(s). While two separate parsing models are employed for intra- and multi-sentential parsing, the same parsing algorithm is used in both parsing conditions. The two-stage parsing exploits the fact that sentence boundaries correlate very well with discourse boundaries. For example, more than 95% of the sentences in RST-DT have a well-formed discourse subtree in the full document-level discourse tree.

The choice of using two separate models for intra- and multi-sentential parsing is well justified for the following two reasons: (*i*) it has been observed that discourse relations have different distributions in the two parsing scenarios, and (*ii*) the models could independently pick their own informative feature sets. The parsing model used for intra-sentential parsing is a Dynamic Conditional Random Field (DCRF) (Sutton et al., 2007) shown in Figure 7. The observed nodes U_j at the bottom layer represent the discourse units at a certain level of the DT; the binary nodes S_j at the middle layer predict whether two adjacent units U_{j-1} and U_j should be connected or not; and the multi-class nodes R_j at the top layer predict the discourse relation between U_{j-1} and U_j. Notice that the model represents the structure and the label of a DT constituent jointly, and captures the sequential dependencies between the DT constituents. Since the chain-structured DCRF model does not scale up to multi-sentential parsing of long documents,

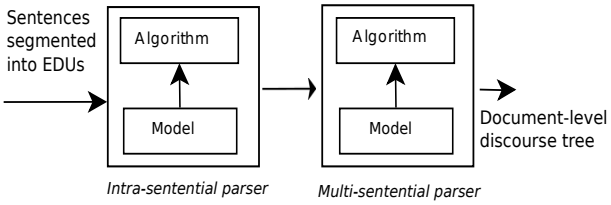


Figure 6: The two-stage document-level discourse parser proposed by Joty et al. (2013).

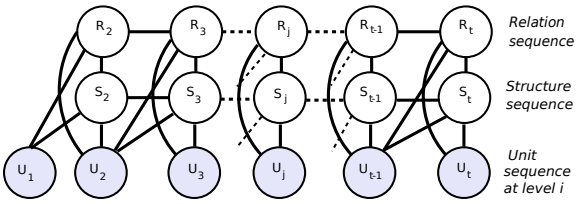


Figure 7: The intra-sentential parsing model.

the multi-sentential parsing model is a CRF which breaks the chain structure of the DCRF model.

The parsing models are applied recursively at different levels of the DT in their respective parsing scenarios (i.e., intra- and multi-sentential), and the probabilities of all possible DT constituents are obtained by computing the posterior marginals over the relation-structure pairs (i.e., $P(R_j, S_j{=}1|U_1, \cdots, U_t, \Theta)$, where Θ are model parameters). These probabilities are then used in a CKY-like probabilistic parsing algorithm to find the globally optimal DT for the given text.

Let U_x^b and U_x^e denote the beginning and end EDU Ids of a discourse unit U_x, and $R[U_i^b, U_m^e, U_j^e]$ refers to a coherence relation R that holds between the discourse unit containing EDUs U_i^b through U_m^e and the unit containing EDUs $U_m^e{+}1$ through U_j^e. Given n discourse units, the parsing algorithm uses the upper-triangular portion of the $n{\times}n$ dynamic programming table A, where cell $A[i,j]$ (for $i < j$) stores:

$$A[i,j] = P(r^*[U_i^b, U_{m^*}^e, U_j^e]), \text{ where}$$

$$(m^*, r^*) = \underset{i \le m < j \ ; \ R}{\operatorname{argmax}} P(R[U_i^b, U_m^e, U_j^e]) \times A[i,m] \times A[m+1,j] \quad (1)$$

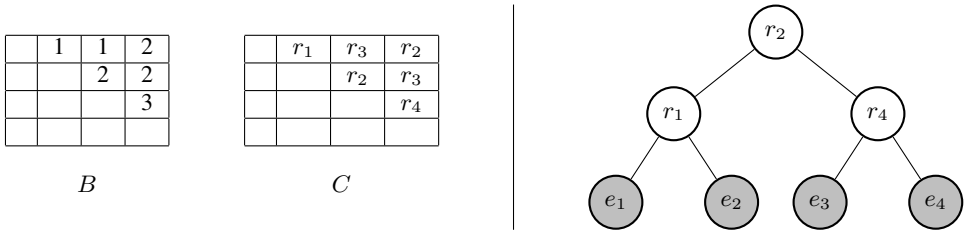


Figure 8: The B and C dynamic programming tables (left), and the corresponding discourse tree (right).

In addition to A, which stores the *probability* of the most probable constituents of a DT, the parsing algorithm also simultaneously maintains two other tables B and C for storing the best structure (i.e., $U^e_{m^*}$) and the relations (i.e., r^*) of the corresponding DT constituents, respectively. For example, given 4 EDUs $e_1 \cdots e_4$, the B and C tables at the left side in Figure 8 together represent the DT shown at the right. More specifically, to generate the DT, we first look at the top-right entries in the two tables, and find $B[1,4] = 2$ and $C[1,4] = r_2$, which specify that the two discourse units $e_{1:2}$ and $e_{3:4}$ should be connected by the relation r_2 (the root in the DT). Then, we see how EDUs e_1 and e_2 should be connected by looking at the entries $B[1,2]$ and $C[1,2]$, and find $B[1,2] = 1$ and $C[1,2] = r_1$, which indicates that these two units should be connected by the relation r_1 (the left pre-terminal). Finally, to see how EDUs e_3 and e_4 should be linked, we look at the entries $B[3,4]$ and $C[3,4]$, which tell us that they should be linked by the relation r_4 (the right pre-terminal).

It is straight-forward to generalize the above algorithm to produce k most probable DTs. When filling up the dynamic programming tables, rather than storing a single best parse for each subtree, we store and keep track of k-best candidates simultaneously. More specifically, each cell in the dynamic programming tables (i.e., A, B and C) should now contain k entries (sorted by their probabilities), and for each such entry there should be a back-pointer that keeps track of the decoding path.

The algorithm works in polynomial time. For n discourse units and M number of relations, the 1-best parsing algorithm has a time complexity of $O(n^3M)$ and a space complexity of $O(n^2)$, where the k-best version has a time and space complexities of $O(n^3Mk^2 \log k)$ and $O(n^2k)$, respectively. There are cleverer ways to reduce the complexity (e.g., see (Huang and Chiang, 2005) for three such ways). However, since the efficiency of the algorithm did not limit us to produce k-best parses for larger k, it was not a priority in this work.

5 Kernels for Reranking Discourse Trees

In Section 3, we described DISCTK, which essentially can be used for any classification task involving discourse trees. For example, given a DT, we can use DISCTK to classify it as correct vs. incorrect. However, such classification is not completely aligned to our purpose, since our goal is to select the best (i.e., the most correct) DT from k candidate DTs; i.e., a ranking task. We adopt a preference reranking technique as described in (Moschitti et al., 2006; Dinarelli et al., 2011).

5.1 Preference Reranker

Preference reranking (PR) uses a classifier $\mathcal{C}$ of pairs of hypotheses $\langle h_i, h_j \rangle$, which decides if h_i (i.e., a candidate DT in our case) is better than h_j. We generate positive and negative examples to train the classifier using the following approach. The pairs $\langle h_1, h_i \rangle$ constitute positive examples, where h_1 has the highest f-score accuracy on the Relation metric (to be described in Section 6) with respect to the gold standard among the candidate hypotheses, and vice versa, $\langle h_i, h_1 \rangle$ are considered as negative examples. At test time, $\mathcal{C}$ classifies all pairs $\langle h_i, h_j \rangle$ generated from the k-best hypotheses. A positive decision is a vote for h_i, and a negative decision is a vote for h_j. Also, the classifier score can be used as a weighted vote. Hypotheses are then ranked according to the number (sum) of the (weighted) votes they get.[6]

We build our reranker using simple SVMs.[7]

[6] As shown by Collins and Duffy (2002), only the classification of k hypotheses (paired with the empty one) is needed in practice, thus the complexity is only $O(k)$.

[7] Structural kernels, e.g., TKs, cannot be used in more advanced algorithms working in structured output spaces, e.g., SVM^{struct}. Indeed, to our knowledge, no one could successfully find a general and exact solution for the $argmax$ equation, typically part of such advanced models, when structural kernels are used. Some approximate solutions for simple kernels, e.g., polynomial or gaussian kernels, are given in (Joachims and Yu, 2009), whereas (Severyn and Moschitti, 2011; Severyn and Moschitti, 2012) provide solutions for using the cutting-plane algorithm (which requires $argmax$ computation) with structural kernels but in binary SVMs.

Since in our problem a pair of hypotheses $\langle h_i, h_j \rangle$ constitutes a data instance, we now need to define the kernel between the pairs. However, notice that DISCTK only works on a single pair.

Considering that our task is to decide whether h_i is better than h_j, it can be convenient to represent the pairs in terms of differences between the vectors of the two hypotheses, i.e., $\phi_K(h_i) - \phi_K(h_j)$, where K (i.e., DISCTK) is defined between two hypotheses (not on two pairs of hypotheses). More specifically, to compute this difference implicitly, we can use the following kernel summation: $PK(\langle h_1, h_2 \rangle, \langle h'_1, h'_2 \rangle) = (\phi_K(h_1) - \phi_K(h_2)) \circ (\phi_K(h'_1) - \phi_K(h'_2)) = K(h_1, h'_1) + K(h_2, h'_2) - K(h_1, h'_2) - K(h_2, h'_1)$.

In general, Preference Kernel (PK) works well because it removes many identical features by taking differences between two huge implicit TK-vectors. In our reranking framework, we also include traditional feature vectors in addition to the trees. Therefore, each hypothesis h is represented as a tuple $\langle T, \vec{v} \rangle$ composed of a tree T and a feature vector $\vec{v}$. We then define a structural kernel (i.e., similarity) between two hypotheses h and h' as follows: $K(h, h') = DiscTK(T, T') + FV(\vec{v}, \vec{v}')$, where DISCTK maps the DTs T and T' to JRN or SRN and then applies STK, STK$_b$ or PTK defined in Sections 2.2 and 3, and FV is a standard kernel, e.g., linear, polynomial, gaussian, etc., over feature vectors (see next section).

5.2 Feature Vectors

We also investigate the impact of traditional (i.e., not subtree) features for reranking discourse parses. Our feature vector comprises two types of features that capture global properties of the DTs.

Basic Features. This set includes eight global features. The first two are the probability and the (inverse) rank of the DT given by the base parser. These two features are expected to help the reranker to perform at least as good as the base parser. The other six features encode the structural properties of the DT, which include depth of the DT, number of nodes connecting two EDUs (i.e., SPANs in Figure 4), number of nodes connecting two relational nodes, number of nodes connecting a relational node and an EDU, number of nodes that connects a relational node as left child and an EDU as right child, and vice versa.

Relation Features. We encode the relations in the DT as bag-of-relations (i.e., frequency count). This will allow us to assess the impact of a flat representation of the DT. Note that more important relational features would be the subtree patterns extracted from the DT. However, they are already generated by TKs in a simpler way. See (Pighin and Moschitti, 2009; Pighin and Moschitti, 2010) for a way to extract the most relevant features from a model learned in the kernel space.

6 Experiments

Our experiments aim to show that reranking of discourse parses is a promising research direction, which can improve the state-of-the-art. To achieve this, we (*i*) compute the oracle accuracy of the k-best parser, (*ii*) test different kernels for reranking discourse parses by applying standard kernels to our new structures, (*iii*) show the reranking performance using the best kernel for different number of hypotheses, and (*iv*) show the relative importance of features coming from different sources.

6.1 Experimental Setup

Data. We use the standard RST-DT corpus (Carlson et al., 2002), which comes with discourse annotations for 385 articles (347 for training and 38 for testing) from the Wall Street Journal. We extracted sentence-level DTs from a document-level DT by finding the subtrees that exactly span over the sentences. This gives 7321 and 951 sentences in the training and test sets, respectively. Following previous work, we use the same 18 coarser relations defined by Carlson and Marcu (2001).

We create the training data for the reranker in a 5-fold cross-validation fashion.[8] Specifically, we split the training set into 5 equal-sized folds, and train the parsing model on 4 folds and apply to the rest to produce k most probable DTs for each text. Then we generate and label the pairs (by comparing with the gold) from the k most probable trees as described in Section 5.1. Finally, we merge the 5 labeled folds to create the full training data.

SVM Reranker. We use SVM-light-TK to train our reranking models,[9] which enables the use of tree kernels (Moschitti, 2006) in SVM-light (Joachims, 1999). We build our new kernels for reranking exploiting the standard built-in TK functions, such as STK, STK$_b$ and PTK. We applied

[8] Note that our earlier experiments with a 2-fold cross validation process yielded only 50% of our current improvement.

[9] http://disi.unitn.it/moschitti/Tree-Kernel.htm

a linear kernel to standard feature vectors as it showed to be the best on our development set.

Metrics. The standard procedure to evaluate discourse parsing performance is to compute Precision, Recall and f-score of the unlabeled and labeled metrics proposed by Marcu (2000b).[10] Specifically, the unlabeled metric *Span* measures how accurate the parser is in finding the right structure (i.e., skeleton) of the DT, while the labeled metrics *Nuclearity* and *Relation* measure the parser's ability to find the right labels (nuclearity and relation) in addition to the right structure. Optimization of the Relation metric is considered to be the hardest and the most desirable goal in discourse parsing since it gives aggregated evaluation on tree structure and relation labels. Therefore, we measure the *oracle* accuracy of the k-best discourse parser based on the f-scores of the *Relation* metric, and our reranking framework aims to optimize the Relation metric.[11] Specifically, the oracle accuracy for k-best parsing is measured as follows: $\text{ORACLE} = \frac{\sum_{i=1}^{N} \max_{j=1}^{k} \text{f-score}_r(g_i, h_i^j)}{N}$, where N is the total number of texts (sentences or documents) evaluated, g_i is the gold DT annotation for text i, h_i^j is the j^{th} parse hypothesis generated by the k-best parser for text i, and $f\text{-}score_r(g_i, h_i^j)$ is the f-score accuracy of hypothesis h_i^j on the Relation metric. In all our experiments we report the f-scores of the Relation metric.

6.2 Oracle Accuracy

Table 1 presents the oracle scores of the k-best intra-sentential parser PAR-S on the standard RST-DT test set. The 1-best result corresponds to the accuracy of the base parser (i.e., 79.77%). The 2-best shows dramatic oracle-rate improvement (i.e., 4.65% absolute), suggesting that the base parser often generates the best tree in its top 2 outputs. 5-best increases the oracle score to 88.09%. Afterwards, the increase in accuracy slows down, achieving, e.g., 90.37% and 92.57% at 10-best and 20-best, respectively.

The results are quite different at the document level as Table 2 shows the oracle scores of the k-best document-level parser PAR-D.[12] The results suggest that the best tree is often missing in the top k parses, and the improvement in oracle-rate is very little as compared to the sentence-level parsing. The 2-best and the 5-best improve over the base accuracy by only 0.7% and 1.0%, respectively. The improvement becomes even lower for larger k. For example, the gain from 20-best to 30-best parsing is only 0.09%. This is not surprising because generally document-level DTs are big with many constituents, and only a very few of them change from k-best to k+1-best parsing. These small changes do not contribute much to the overall f-score accuracy.[13] In summary, the results in Tables 1 and 2 demonstrate that a k-best reranker can potentially improve the parsing accuracy at the sentence level, but may not be a suitable option for improving parsing at the document level. In the following, we report our results for reranking sentence-level discourse parses.

k	1	2	5	10	15	20
PAR-S	79.77	84.42	88.09	90.37	91.74	92.57

Table 1: Oracle scores as a function of k of k-best sentence-level parses on RST-DT test set.

k	1	2	5	10	15	20
PAR-D	55.83	56.52	56.91	57.23	57.54	57.65

Table 2: Oracle scores as a function of k of k-best document-level parses on RST-DT test set.

6.3 Performance of Different DISCTKs

Section 3 has pointed out that different DISCTKs can be obtained by specifying the TK type (e.g., STK, STK$_b$, PTK) and the mapping ϕ_M (i.e., JRN, SRN) in the overall kernel function $\big(\phi_{TK} \circ \phi_M\big)(o_1) \cdot \big(\phi_{TK} \circ \phi_M\big)(o_2)$. Table 3 reports the performance of such model compositions using the 5-best hypotheses on the RST-DT test set. Additionally, it also reports the accuracy for the two versions of JRN and SRN, i.e., Bigram and All. From these results, we can note the following.

Firstly, the kernels generally perform better on Bigram than All lexicalization. This suggests that using all the words from the text spans (i.e., EDUs) produces sparse models.

[10] Precision, Recall and f-score are the same when the discourse parser uses manual discourse segmentation. Since all our experiments in this paper are based on manual discourse segmentation, we only report the f-scores.

[11] It is important to note that optimizing Relation metric may also result in improved Nuclearity scores.

[12] For document-level parsing, Joty et al. (2013) propose two approaches to combine intra- and multi-sentential parsers, namely 1S-1S (1 Sentence-1 Subtree) and Sliding window. In this work we extend 1S-1S to k-best document-level parser PAR-D since it is not only time efficient but it also achieves better results on the Relation metric.

[13] Note that Joty et al. (2012; 2013) report lower f-scores both at the sentence level (i.e., 77.1% as opposed to our 79.77%) and at the document level (i.e., 55.73% as opposed to our 55.83%). We fixed a crucial bug in their (1-best) parsing algorithm, which accounts for the improved performance.

$\phi_{TK} \circ \phi_M$	JRN		SRN	
	Bigram	All	Bigram	All
STK	81.28	80.04	82.15	80.04
STK_b	81.35	80.28	82.18	80.25
PTK	81.63	78.50	81.42	78.25

Table 3: Reranking performance of different discourse tree kernels on different representations.

Secondly, while the tree kernels perform similarly on the JRN representation, STK performs significantly better (p-value < 0.01) than PTK on SRN.[14] This result is interesting as it provides indications of the type of DT fragments useful for improving parsing accuracy. As pointed out in Section 2.2, PTK includes all features generated by STK, and additionally, it includes fragments whose nodes can have any subsets of the children they have in the original DT. Since this does not improve the accuracy, we speculate that complete fragments, e.g., [CAUSE [ATTRIBUTION][ELABORATION]] are more meaningful than the partial ones, e.g., [CAUSE [ATTRIBUTION]] and [CAUSE [ELABORATION]], which may add too much uncertainty on the signature of the relations contained in the DT. We verified this hypothesis by running an experiment with PTK constraining it to only generate fragments whose nodes preserve all or none of their children. The accuracy of such fragments approached the ones of STK, suggesting that relation information should be used as a whole for engineering features.

Finally, STK_b is slightly (but not significantly) better than STK suggesting that the lexical information is already captured by the base parser.

Note that the results in Table 3 confirms many other experiments we carried out on several development sets. For any run: (*i*) STK always performs as well as STK_b, (*ii*) STK is always better than PTK, and (*iii*) SRN is always better than JRN. In what follows, we show the reranking performance based on STK applied to SRN with Bigram.

6.4 Insights on DISCTK-based Reranking

Table 4 reports the performance of our reranker (RR) in comparison with the oracle (OR) accuracy for different values of k, where we also show the corresponding relative error rate reduction (ERR) with respect to the baseline. To assess the generality of our approach, we evaluated our reranker on both the standard test set and the entire training set using 5-fold cross validation.[15]

[14] Statistical significance is verified using paired t-test.

[15] The reranker was trained on 4 folds and tested on the rest

Baseline	Basic feat.	+ Rel. feat.	+ Tree
79.77	79.84	79.81	82.15

Table 5: Comparison of features from different sources for 5-best discourse reranking.

	(Joty et al., 2013)	With Reranker
PAR-D	55.8	**57.3**

Table 6: Document-level parsing results with 5-best sentence-level discourse reranker.

We note that: (*i*) the best result on the standard test set is 82.15% for $k = 4$ and 5, which gives an ERR of 11.76%, and significantly (p-value < 0.01) outperforms the baseline, (*ii*) the improvement is consistent when we move from standard test set to 5-folds, (*iii*) the best result on the 5-folds is 80.86 for $k = 6$, which is significantly (p-value < 0.01) better than the baseline 78.57, and gives an ERR of 11.32%. We also experimented with other values of k in both training and test sets (also increasing k only in the test set), but we could not improve over our best result. This suggests that outperforming the baseline (which in our case is the state of the art) is rather difficult.[16]

In this respect, we also investigated the impact of traditional ranking methods based on feature vectors, and compared it with our TK-based model. Table 5 shows the 5-best reranking accuracy for different feature subsets. The *Basic features* (Section 5.2) alone do not significantly improve over the *Baseline*. The only relevant features are the probability and the rank of each hypothesis, which condense all the information of the local model (TKs models always used them).

Similarly, adding the relations as bag-of-relations in the vector (*Rel. feat.*) does not provide any gain, whereas the relations encoded in the tree fragments (Tree) gives improvement. This shows the importance of using structural dependencies for reranking discourse parses.

Finally, Table 6 shows that if we use our sentence-level reranker in the document-level parser of Joty et al. (2013), the accuracy of the latter increases from 55.8% to 57.3%, which is a significant improvement ($p < 0.01$), and establishes a new state-of-the-art for document-level parsing.

6.5 Error Analysis

We looked at some examples where our reranker failed to identify the best DT. Unsurprisingly, it

[16] The human agreement on sentence-level parsing is 83%.

	Standard test set						5-folds (average)					
	k=1	k=2	k=3	k=4	k=5	k=6	k=1	k=2	k=3	k=4	k=5	k=6
RR	79.77	81.08	81.56	**82.15**	**82.15**	82.11	78.57	79.76	80.28	80.68	80.80	**80.86**
ERR	-	6.48	8.85	**11.76**	**11.76**	11.57	-	5.88	8.45	10.43	11.02	**11.32**
OR	79.77	84.42	86.55	87.68	88.09	88.75	78.57	83.20	85.13	86.49	87.35	88.03

Table 4: Reranking performance (RR) in comparison with oracle (OR) accuracy for different values of k on the standard testset and 5-folds of RST-DT. Second row shows the relative error rate reduction (ERR).

happens many times for small DTs containing only two or three EDUs, especially when the relations are semantically similar. Figure 9 presents such a case, where the reranker fails to rank the DT with *Summary* ahead of the DT with *Elaboration*. Although we understand that the reranker lacks enough structural context to distinguish the two relations in this example, we expected that including the lexical items (e.g., *(CFD)*) in our DT representation could help. However, similar short parenthesized texts are also used to elaborate as in *Senate Majority Leader George Mitchell (D., Maine)*, where the text *(D., Maine)* (i.e., Democrat from state Maine) elaborates its preceding text. This confuses our reranker. We also found error examples where the reranker failed to distinguish between *Background* and *Elaboration*, and between *Cause* and *Elaboration*. This suggests that we need rich semantic representation of the text to improve our reranker further.

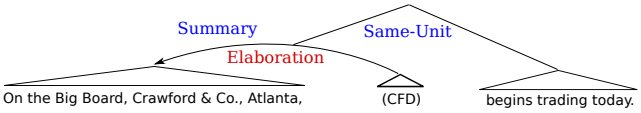


Figure 9: An error made by our reranker.

7 Related Work

Early work on discourse parsing applied hand-coded rules based on discourse cues and surface patterns (Marcu, 2000a). Supervised learning was first attempted by Marcu (2000b) to build a shift-reduce discourse parser. This work was then considerably improved by Soricut and Marcu (2003). They presented probabilistic generative models for *sentence-level* discourse parsing based on lexico-syntactic patterns. Sporleder and Lapata (2005) investigated the necessity of syntax in discourse analysis. More recently, Hernault et al. (2010) presented the HILDA discourse parser that iteratively employs two SVM classifiers in pipeline to build a DT in a greedy way. Feng and Hirst (2012) improved the HILDA parser by incorporating rich linguistic features, which include lexical semantics and discourse production rules.

Joty et al. (2013) achieved the best prior results by (*i*) jointly modeling the structure and the label of a DT constituent, (*ii*) performing optimal rather than greedy decoding, and (*iii*) discriminating between intra- and multi-sentential discourse parsing. However, their model does not consider long range dependencies between DT constituents, which are encoded by our kernels. Regarding the latter, our work is surely inspired by (Collins and Duffy, 2002), which uses TK for syntactic parsing reranking or in general discriminative reranking, e.g., (Collins and Koo, 2005; Charniak and Johnson, 2005; Dinarelli et al., 2011). However, such excellent studies do not regard discourse parsing, and in absolute they achieved lower improvements than our methods.

8 Conclusions and Future Work

In this paper, we have presented a discriminative approach for reranking discourse trees generated by an existing discourse parser. Our reranker uses tree kernels in SVM preference ranking framework to effectively capture the long range structural dependencies between the constituents of a discourse tree. We have shown the reranking performance for sentence-level discourse parsing using the standard tree kernels (i.e., STK and PTK) on two different representations (i.e., JRN and SRN) of the discourse tree, and compare it with the traditional feature vector-based approach. Our results show that: (*i*) the reranker improves only when it considers subtree features computed by the tree kernels, (*ii*) SRN is a better representation than JRN, (*iii*) STK performs better than PTK for reranking discourse trees, and (*iv*) our best result outperforms the state-of-the-art significantly.

In the future, we would like to apply our reranker to the document-level parses. However, this will require a better hypotheses generator.

Acknowledgments

This research is part of the Interactive sYstems for Answer Search (Iyas) project, conducted by the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Qatar Foundation.

References

Lynn Carlson and Daniel Marcu. 2001. Discourse Tagging Reference Manual. Technical Report ISI-TR-545, University of Southern California Information Sciences Institute.

Lynn Carlson, Daniel Marcu, and Mary Ellen Okurowski. 2002. RST Discourse Treebank (RST-DT) LDC2002T07. *Linguistic Data Consortium, Philadelphia*.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-Fine n-Best Parsing and MaxEnt Discriminative Reranking. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics*, ACL'05, pages 173–180, NJ, USA. ACL.

Michael Collins and Nigel Duffy. 2002. New Ranking Algorithms for Parsing and Tagging: Kernels over Discrete Structures, and the Voted Perceptron. In *ACL*.

Michael Collins and Terry Koo. 2005. Discriminative Reranking for Natural Language Parsing. *Comput. Linguist.*, 31(1):25–70, March.

Corinna Cortes and Vladimir Vapnik. 1995. Support Vector Networks. *Machine Learning*, 20:273–297.

Marco Dinarelli, Alessandro Moschitti, and Giuseppe Riccardi. 2011. Discriminative Reranking for Spoken Language Understanding. *IEEE Transactions on Audio, Speech and Language Processing (TASLP)*, 20:526539.

Vanessa Feng and Graeme Hirst. 2012. Text-level Discourse Parsing with Rich Linguistic Features. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics*, ACL '12, pages 60–68, Jeju Island, Korea. ACL.

Hugo Hernault, Helmut Prendinger, David A. duVerle, and Mitsuru Ishizuka. 2010. HILDA: A Discourse Parser Using Support Vector Machine Classification. *Dialogue and Discourse*, 1(3):1—33.

Liang Huang and David Chiang. 2005. Better K-best Parsing. In *Proceedings of the Ninth International Workshop on Parsing Technology*, Parsing '05, pages 53–64, Stroudsburg, PA, USA. Association for Computational Linguistics.

Thorsten Joachims and Chun-Nam John Yu. 2009. Sparse Kernel SVMs via Cutting-Plane Training. *Machine Learning*, 76(2-3):179–193. ECML.

Thorsten Joachims. 1999. Making large-Scale SVM Learning Practical. In *Advances in Kernel Methods - Support Vector Learning*.

Shafiq Joty, Giuseppe Carenini, and Raymond T. Ng. 2012. A Novel Discriminative Framework for Sentence-Level Discourse Analysis. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, EMNLP-CoNLL '12, pages 904–915, Jeju Island, Korea. ACL.

Shafiq Joty, Giuseppe Carenini, Raymond T. Ng, and Yashar Mehdad. 2013. Combining Intra- and Multi-sentential Rhetorical Parsing for Document-level Discourse Analysis. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics*, ACL '13, Sofia, Bulgaria. ACL.

William Mann and Sandra Thompson. 1988. Rhetorical Structure Theory: Toward a Functional Theory of Text Organization. *Text*, 8(3):243–281.

Daniel Marcu. 2000a. The Rhetorical Parsing of Unrestricted Texts: A Surface-based Approach. *Computational Linguistics*, 26:395–448.

Daniel Marcu. 2000b. *The Theory and Practice of Discourse Parsing and Summarization*. MIT Press, Cambridge, MA, USA.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2006. Semantic Role Labeling via Tree Kernel Joint Inference. In *Proceedings of the Tenth Conference on Computational Natural Language Learning (CoNLL-X)*, pages 61–68, New York City, June. Association for Computational Linguistics.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *17th European Conference on Machine Learning*, pages 318–329. Springer.

Alessandro Moschitti. 2010. Kernel Engineering for Fast and Easy Design of Natural Language Applications. In *COLING (Tutorials)*, pages 1–91.

Alessandro Moschitti. 2012. State-of-the-Art Kernels for Natural Language Processing. In *Tutorial Abstracts of ACL 2012*, page 2, Jeju Island, Korea, July. Association for Computational Linguistics.

Alessandro Moschitti. 2013. Kernel-based Learning to Rank with Syntactic and Semantic Structures. In *SIGIR*, page 1128.

Daniele Pighin and Alessandro Moschitti. 2009. Reverse Engineering of Tree Kernel Feature Spaces. In *EMNLP*, pages 111–120.

Daniele Pighin and Alessandro Moschitti. 2010. On Reverse Feature Engineering of Syntactic Tree Kernels. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 223–233, Uppsala, Sweden, July. Association for Computational Linguistics.

Aliaksei Severyn and Alessandro Moschitti. 2011. Fast Support Vector Machines for Structural Kernels. In *ECML/PKDD (3)*, pages 175–190.

Aliaksei Severyn and Alessandro Moschitti. 2012. Fast Support Vector Machines for Convolution Tree Kernels. *Data Min. Knowl. Discov.*, 25(2):325–357.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press.

Radu Soricut and Daniel Marcu. 2003. Sentence Level Discourse Parsing Using Syntactic and Lexical Information. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL'03, pages 149–156, Edmonton, Canada. ACL.

Caroline Sporleder and Mirella Lapata. 2005. Discourse Chunking and its Application to Sentence Compression. In *Proceedings of the conference on Human Language Technology and Empirical Methods in Natural Language Processing*, HLT-EMNLP'05, pages 257–264, Vancouver, British Columbia, Canada. ACL.

Charles Sutton, Andrew McCallum, and Khashayar Rohanimanesh. 2007. Dynamic Conditional Random Fields: Factorized Probabilistic Models for Labeling and Segmenting Sequence Data. *Journal of Machine Learning Research (JMLR)*, 8:693–723.