A Compositional Perspective in Convolution Kernels

Roberto Basili[1], Paolo Annesi[1], Giuseppe Castellucci[2], and Danilo Croce[1]

[1] Department of Enterprise Engineering,
University of Roma Tor Vergata, Roma Italy
{basili,annesi,croce}@info.uniroma2.it,
[2] Department of Electronic Engineering,
University of Roma Tor Vergata, Roma Italy
castellucci@ing.uniroma2.it

Abstract. Kernel-based learning has been largely adopted in many semantic textual inference tasks. In particular, Tree Kernels (TKs) have been successfully applied in the modeling of syntactic similarity between linguistic instances in Question Answering or Information Extraction tasks. At the same time, lexical semantic information has been studied through the adoption of the so-called Distributional Semantics (DS) paradigm, where lexical vectors are acquired automatically from large-scale corpora. Recently, Compositional Semantics phenomena arising in complex linguistic structures have been studied in an extended paradigm called Distributional Compositional Semantics (DCS), where, for example, algebraic operators on lexical vectors have been defined to account for grammatically typed bi-grams or complex verb or noun phrases.
In this paper, a novel kernel called Compositionally Smoothed Partial Tree Kernel is presented to integrate DCS operators into the tree kernel evaluation by also considering complex compositional nodes. Empirical results on well-known NLP tasks show that state-of-the-art performances can be achieved, without resorting to manual feature engineering, thus suggesting that a large set of Web and text mining tasks can be handled successfully by this kernel.

1 Introduction

Convolution Kernels [16] are well-known similarity functions among complex syntactic and semantic structures. They are largely used to solve natural language related tasks, such as Opinion Mining [18] in recommender systems, Semantic Textual Similarity [11] in retrieval, or Question Classification [22] in question answering systems. In particular, Tree Kernels (TKs) introduced in [8] are used for their ability in capturing syntactic information, directly from syntactic parse trees. When training supervised learning algorithms, they are a valid approach to avoid the difficulty of manually designing effective features from linguistic structures. They define a similarity measure between data points (i.e. trees) in terms of all possible shared substructures.

In order to take into account the lexical generalization provided from the analysis of large scale document collections, i.e. Distributional Semantic Models [19, 26, 25], a recent formulation of these kernel functions, called Smoothed Partial Tree Kernel (SPTK), has been introduced in [10]. The semantic information related to the lexical nodes of a parse tree is exploited for matching sub-structures characterized by different but (semantically) related lexical items: it means that the similarity function between sub-trees depends also on a distributional metric between vectors representing the semantic of words. The main limitations of this approach are that a) lexical semantic information only relies on the vector metrics applied to the nodes in a context free fashion and b) the semantic composition between words is neglected in the kernel computation, that only depends on their grammatical labels.

In our perspective, semantic information is expressed by all nodes of the parse tree where specific compositions between heads and modifiers are shown. For example in the sentence, "*What instrument does Hendrix play?*", the correct meaning of the verb *play* is fully captured if its modifiers, i.e. the noun *instrument*, is taken into account. This corresponds to model the contribution of the lexical item *instrument* as a function of the corresponding direct object relation (`dobj`) between *play* and *instrument*: we can say that *instrument* contributes to the sentence semantics by filling compositionally the slot `dobj` of its head *play*. In other words it seems that lexical semantic information should propagate over the entire parse tree, by making the compositionality phenomena of the sentence explicit. In recent years, Distributional Compositional Semantic (DCS) metrics have been proposed in order to combine lexical vector representations by algebraic operators defined in the corresponding distributional space [21, 13, 32, 4].

The purpose of this paper is to cast the use of words within the compositional relationships exposed by a tree in order to make it sensible to the underlying complex constituents. Even when the syntactic structure does not change, such as in "*play an instrument*" vs. "*play a match*", the contribution of the lexical information (e.g. the semantics of the verb "*play*") must be correspondingly different when estimating the kernel function with a phrase like "*win a game*". In traditional kernel computations, the similarity between *play* and *win* are context-free and the information of the direct object is neglected. The ideas underlying the **Compositionally Smoothed Partial Tree Kernel** (CSPTK), proposed in [3], are to i) use the SPTK formulation in order to exploit the lexical information encoded in a tree, ii) define a procedure to encode head-modifier information in the parse tree nodes in order to estimate the lexical similarity between sub-trees, iii) apply compositional distributional metrics to enforce a context-dependent estimation of the similarity of individual head-modifier information at the nodes.

In Section 2 a summary of the approaches for DCS and TKs is presented. In Section 3, a lexical mark-up method for parse trees is described to support the CSPTK formulation, presented in Section 4. Finally, Section 5 reports the empirical evaluation of the CSPTK over Question Classification and Paraphrase Identification tasks.

2 Related Work

This section reports a summary of the approaches for Distributional Compositional Semantics and Convolution Kernels.

Distributional Compositional Semantics. Vector-based models typically represent isolated words and ignore grammatical structure [29] (a noticeable exception is [23]). They have thus a limited capability to model compositional operations over phrases and sentences. Distributional methods have been recently extended to better account for compositionality, in the so-called Distributional Compositional Semantics (DCS) approaches.

Existing models are still controversial and provide general algebraic operators over lexical vectors and sophisticated composition methods. In [27] compositionality of two vector $\boldsymbol{u}$ and $\boldsymbol{v}$ is accounted by the tensor product $\boldsymbol{u} \otimes \boldsymbol{v}$, while in [14] lexical vectors are summed, keeping the resulting vector with the same dimension of the latter. In [21] a semantic composition is seen as a function $\boldsymbol{p} = f(\boldsymbol{u}, \boldsymbol{v}, R, K)$ of two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$, their syntactic relation R and the factor K, defined as any additional knowledge or information which is needed to construct the semantics of their composition. This perspective clearly leads to a variety of efficient yet shallow models of compositional semantics. Two simplified models are derived from these general forms: i) the additive model $\boldsymbol{p}_+ = \alpha\boldsymbol{u} + \beta\boldsymbol{v}$, where the weights α and β are applied the lexical vectors; ii) the multiplicative model $\boldsymbol{p}_\cdot = \boldsymbol{u} \odot \boldsymbol{v}$, where the symbol $\odot$ represents the point-wise multiplication, i.e. $p_i = u_i \cdot v_i$. In [21] a more complex asymmetric type of function called *dilation*, i.e. $\boldsymbol{p}_d = (\boldsymbol{u} \cdot \boldsymbol{u})\boldsymbol{v} + (\lambda - 1)(\boldsymbol{u} \cdot \boldsymbol{v})\boldsymbol{u}$, is introduced.

In [13], the concept of a *structured vector space* is introduced, where each word is associated to a set of vectors corresponding to different syntactic dependencies. Every word is expressed by a tensor, and tensor operations are imposed. In [28] a similar geometric approach over a space of second-order distributional vectors is presented. Vectors represent the words typically co-occurring with the contexts in which the target word appears. A parallel strand of research also seeks to represent the meaning of larger compositional structures using matrix and tensor algebra, like in [27] and in [24]. In [5] and [7] composition is characterized from formal semantics in terms of a function application, where the distributional representation of one element in a composition (the *functor*) is not a vector but a function. Moreover, a hybrid Logic-Distributional Model is presented in [15], whilst an approach based on vector permutation and Random Indexing technique, i.e. [25], is presented in [6]. In [2] composition is expressed via a projection operation into a *subspace*, i.e. a subset of the original features of a syntactically motivated compound. A projection is a mapping (a selection function) over the set of features more representative of the compound: a subspace local to the (*play*,*guitar*) phrase can be found such that only the features specific to its meaning are selected. The resulting subspace of a compound should thus select the most appropriate word senses, neglecting the irrelevant ones and preserving the compositional semantics of the phrase.

Convolution Tree Kernels. Convolution Kernels, introduced in [16], determine a similarity function among discrete structures, such as sequences, strings or trees. They are representationally efficient ways to encode similarity metrics able to support supervised learning algorithms for complex textual inferences (e.g. semantic role classification). In particular Tree Kernels (TKs) allow estimating the similarity among texts, directly from the sentence syntactic structures, represented by trees, e.g. the dependency parse trees. A TK computes the number of substructures (as well as their partial fragments) shared by two parse trees T_1 and T_2. For this purpose, let the set $\mathcal{F} = \{f_1, f_2, \dots, f_{|\mathcal{F}|}\}$ be a space of tree fragments and $\chi_i(n)$ be an indicator function: it is 1 if the target f_i is rooted at node n and 0 otherwise. A tree-kernel is a function $TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2)$, where N_{T_1} and N_{T_2} are the sets of the T_1's and T_2's nodes and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1)\chi_i(n_2)$. The Δ function recursively compute the amount of similarity derived from the number of common substructures. The type of considered fragments determines the expressiveness of the kernel space and different tree kernels are characterized by different choices. In particular, the **Smoothed Partial Tree Kernels**, as discussed in [10], measure the similarity between syntactic tree structures, which are semantically related, i.e. partially similar, even when nodes, e.g. words at the leaves, differ. This is achieved by the following formulation of the function Δ over nodes $n_i \in T_i$:

$$\Delta_\sigma(n_1, n_2) = \mu\lambda\sigma(n_1, n_2), \text{ if } n_1 \text{ and } n_2 \text{ are leaves, else}$$

$$\Delta_\sigma(n_1, n_2) = \mu\sigma(n_1, n_2)\Big(\lambda^2 + \sum_{\boldsymbol{I}_1, \boldsymbol{I}_2, l(\boldsymbol{I}_1) = l(\boldsymbol{I}_2)} \lambda^{d(\boldsymbol{I}_1) + d(\boldsymbol{I}_2)} \prod_{j=1}^{l(\boldsymbol{I}_1)} \Delta_\sigma(c_{n_1}(\boldsymbol{I}_{1j}), c_{n_2}(\boldsymbol{I}_{2j}))\Big) \qquad (1)$$

Here, $\boldsymbol{I}_{1j}$ represents the sequence of subtrees, dominated by node n_1, that are shared with the children of n_2 (i.e. $\boldsymbol{I}_{2j}$): all other non-matching substructures are neglected[3]. The novelty of SPTK is represented by the introduction of a similarity function σ between nodes, which are typed according to syntactic categories, pos-tags or lexical entries. A boolean measure of similarity is applied between non lexical nodes by assigning 1 for identical syntactic or POS nodes, 0 otherwise. For lexical nodes the cosine similarity is applied between words sharing the same pos-tag. One main limitation of SPTK is that lexical similarity does not consider compositional interaction between words. Given the phrase pairs (*np* (*nn river*)(*nn bank*)) and (*np* (*nn savings*)(*nn bank*)), the SPTK would estimate the similarity by relying on a unique meaning for *bank*, that is wrong whenever one considers the compositional role of modifiers, *river* vs. *savings*.

3 Making lexical composition explicit in parse trees

In this Section, a way of representing compositional information is discussed as a labeling of individual nodes in the tree. The result is an enriched representation, on which Distributional Compositional Operators can be applied while evaluating the SPTK function, i.e. resulting in a Compositionally Smoothed Partial Tree Kernel (CSPTK).

Compositional semantic constraints over a tree kernel computation can be applied only when individual syntagms corresponding to nodes are made explicit. The CSPTK takes as input two trees where nodes express a lexical composition: this is used in the recursive compositional matching foreseen by the underlying convolution model, i.e. the same as in SPTK. Given the question "*What instrument does Hendrix play?*" and its dependency structure, the relative syntactic structure is shown in Figure 1 in terms of a Lexically Centered Tree (LCT), as proposed in [10].

Nodes in LCTs can be partitioned into: **lexical nodes**, i.e. nodes $n \in \mathcal{NT}$, representing one lexical item, in terms of $\langle l_n\text{::}pos_n \rangle$, such as *instrument::n* or *play::v*, where *l* is the lemma of the token and pos its part-of-speech[4]; **terminal nodes**, $n \in \mathcal{T}$, that are the children of lexical nodes which encode a dependency function $d \in \mathcal{D}$ (e.g. nsubj) or the pos-tag of the node father (e.g. NN).

An additional tree can be derived by emphasizing the importance of syntactic information in the so-called Grammatical Relation Centered Tree (GRCT) in Figure 3,

[3] Two decay factors, μ and λ, are introduced, respectively for the height of the tree and for the length of the child sequences.

[4] General POS tags are obtained from the PennTreebank standard by truncating at their first char (as in *instrument::n*).

where nodes are partitioned into: **syntactic nodes**, i.e. nodes $n \in \mathcal{NT}$, that encode dependency functions $d \in \mathcal{D}$ (e.g. `nsubj`); **pre-terminal nodes**, $n \in \mathcal{PT}$, that encode `pos`-tag of each lexical item that is assigned to a leaf; **lexical nodes**, i.e. nodes $n \in \mathcal{T}$, representing one lexical item, in terms of $\langle l_n\text{::}pos_n\rangle$.

We aim at introducing lexical compositionality in these syntactic representation, through the explicit mark-up of every compositional (sub)structure. Notice how grammatical dependencies in a LCT representation are encoded as direct descendants of a modifier non terminal. For example, the dependency `dobj` is a direct descendant of the lexical node *instrument::n* and expresses its grammatical role in the relationship with its parent node *play::v*. We propose to mark up such lexical nodes with the full description of the corresponding head/modifier relationship, denoted by (h, m). In order to emphasize the semantic contribution of this relationship, the lexical information about the involved head (l_h) and modifier (l_m) must be represented: we denote it through a 4-tuple $\langle \mathrm{l}_h\text{::}\mathrm{pos}_h,\mathrm{l}_m\text{::}\mathrm{pos}_m\rangle$. Therefore, in an extended (compositional representation for LCTs) every non terminal $n \in \mathcal{NT}$ is marked as

$$\langle d_{h,m}, \langle l_h :: pos_h, l_m :: pos_m\rangle\rangle \tag{2}$$

where $d_{h,m} \in \mathcal{D}$ is the dependency function between h and m and l_i and pos_i are lexical entries, and `pos-tags`. Moreover, one lexical node is also added to represent the simple lexical information of the involved modifier, in order to limit data sparseness. Notice that this mark-up resembles closely the representation of immediate dominance rules for grammatical phrases in Attribute Value Matrices (AVM) expressing feature structures in HPSG-like formalisms [9].

Figure 2 shows a fully compositional labeled tree for the sentence whose unlabeled version has been shown in Figure 1. Now, non terminal nodes represent compositional lexical pairs of type head/modifier marked as in Eq. 2. For example, the node `dobj`$\langle$*play::v*,*instrument::n*$\rangle$ express the direct object relation between the verb *to play* and the object *instrument*, as shown in Fig. 2. Dependency functions (`dobj`) and POS-Tags (VBZ) are encoded in terminal nodes as in the LCT before. Eventually, simple lexical nodes, e.g. *play::v*, are added as terminal nodes.

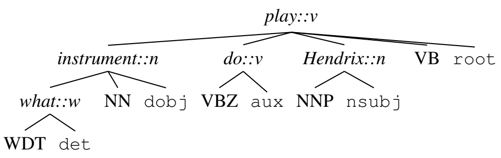

Fig. 1. Lexically centered tree of the sentence "*What instrument does Hendrix play?*"

In a similar fashion, the GRCT in Figure 3 can be extended in the Compositional counterpart, i.e. the CGRCT in Figure 4. In this case non terminal nodes represent compositional lexical pairs of type head/modifier marked as in Eq. 2; `pos-tags` are encoded in pre-terminal nodes as in the GRCT before; finally, simple lexical nodes, e.g. *play::v*, are preserved as terminal nodes. Every compositional node supports the application of a compositional distributional semantic model, where lexical entries for

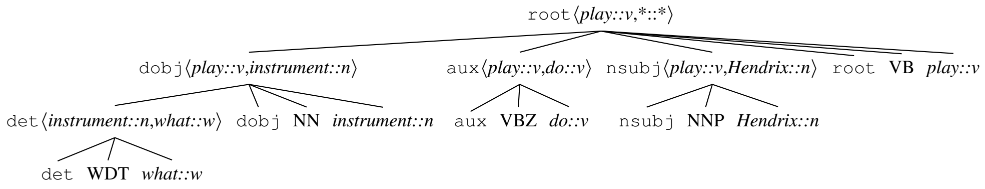

Fig. 2. Compositional Lexical Centered Tree (CLCT) of "*What instrument does Hendrix play?*"

heads ($l_h :: pos_h$) and modifiers ($l_m :: pos_m$) correspond to unique vectors. Given two subtrees T_1 and T_2, rooted at nodes n_1 and n_2 and the corresponding head-modifier pairs $p_1 = (h_1, m_1)$ and $p_2 = (h_2, m_2)$, a shallow compositional function, independent from any dependency relation $d_{h,m}$, is defined over non terminal nodes, by adopting one of the DCS models discussed in Section 2 so that:

$$\sigma_{Comp}\big((h_1, m_1), (h_2, m_2)\big) = \Phi^{\circ}\big((h_1, m_1), (h_2, m_2)\big) \tag{3}$$

Hereafter, the application of the DCS models to the SPTK kernel computation is discussed.

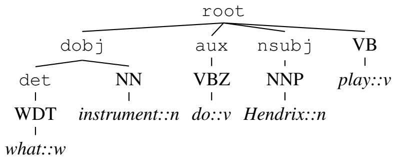

Fig. 3. Grammatical Relation Centered Tree (GRCT) of "*What instrument does Hendrix play?*"

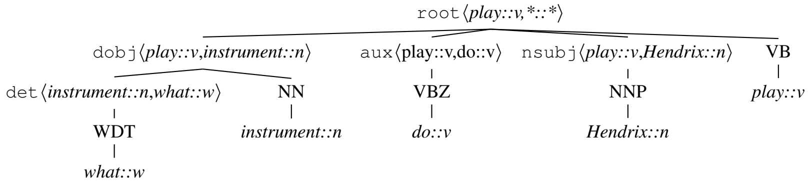

Fig. 4. Compositional Grammatical Relation Centered Tree (CGRCT) of "*What instrument does Hendrix play?*"

4 The Compositionally Smoothed Partial Tree Kernel

When the compositionality of individual lexical constituents is made explicit at the nodes, a compositional measure of similarity between entire parse trees can be computed through a Tree Kernel. We define here such a similarity function as the Compositionally Smoothed Partial Tree Kernel (CSPTK), by extending the SPTK formulation. Let us consider the application of a SPTK to a tree pair represented in their CLCT form: for example, let us consider the trees derived from sentences such as "*Hendrix plays guitar*" and "*Hendrix plays football*". Although the kernel recursively estimates the similarity among all nodes through the σ function in Equation 1, it would not be able to distinguish different senses for the verb *to play*, as it is applied on a unique distributional vector. The contribution of the lexical information (e.g. the vector for the verb *to play*) must be correspondingly dependent from the other words in the sentence.

Algorithm 1 $\sigma_\tau(n_x, n_y, lw)$ Compositional estimation of the lexical contribution in the CSPTK

```
$\sigma_\tau \leftarrow 0$,
/*Matching between simple lexical nodes*/
if $n_x = \langle lex_x{::}pos \rangle$ and $n_y = \langle lex_y{::}pos \rangle$ then
    $\sigma_\tau \leftarrow \sigma_{LEX}(n_x, n_y)$
end if
/*Matching between identical grammatical nodes,
e.g. POS tags*/
if ($n_x = pos$ or $n_x = dep$) and $n_x = n_y$ then
    $\sigma_\tau \leftarrow lw$
end if
if $n_x = \langle d_{h,m}, \langle li_x \rangle \rangle$ and $n_y = \langle d_{h,m}, \langle li_y \rangle \rangle$ then
    /*both modifiers are missing*/
    if $li_x = \langle h_x{::}pos \rangle$ and $li_y = \langle h_y{::}pos \rangle$ then
        $\sigma_\tau \leftarrow \sigma_{Comp}\big((h_x), (h_y)\big) = \sigma_{LEX}(n_x, n_y)$
    end if
    /*one modifier is missing*/
    if $li_x = \langle h_x{::}pos_h \rangle$ and $li_y = \langle h_y{::}pos_h, m_y{::}pos_m \rangle$ then
        $\sigma_\tau \leftarrow \sigma_{Comp}\big((h_x, h_x), (h_y, m_y)\big)$
    end if
    /*the general case*/
    if $li_x = \langle h_x{::}pos_h, m_x{::}pos_m \rangle$ and
    $li_y = \langle h_y{::}pos_h, m_y{::}pos_m \rangle$ then
        $\sigma_\tau \leftarrow \sigma_{Comp}\big((h_x, m_x), (h_y, m_y)\big)$
    end if
end if
return $\sigma_\tau$
```

The core novelty of the CSPTK is thus a new estimation of σ_τ as described in Algorithm 1. Notice that for simple lexical nodes, a lexical kernel σ_{LEX}, such as the cosine similarity between vectors of words sharing the same POS-tag, is applied. Moreover, the other non lexical nodes contribute according to a strict matching policy: they provide full similarity only when the same POS, or dependency, is matched and 0 otherwise. The novel part of Algorithm 1 correspond to the treatment of compositional nodes. The similarity function σ_{Comp} between such nodes is computed only when they exhibit the same $d_{h,m}$ and the respective heads and modifiers share the POS label: then, a compositional metric is applied over the two involved (h, m) compounds. The metric depends on the involved compounds that can be only partially instantiated, so that different strategies must be applied:

- **General case**. When fully instantiated compounds (h_1, m_1) and (h_2, m_2) are available, the similarity function of Equation 3 is applied as usual. Notice that the *POS-tags* of heads and modifiers must be pairwise identical.
- **Both modifiers are missing**. When modifiers m_i are unexpressed in both trees, no composition is observed: a lexical kernel σ_{LEX}, i.e. the cosine similarity between unique word vectors $\boldsymbol{h}_1$ and $\boldsymbol{h}_2$, is applied.

- **One modifier is missing**. Let be $(h_1, *)$ and (h_2, m_2) the lexical information of the two nodes n_1 (that lacks of the modifier) and n_2. Here, the general similarity function is applied to the pair (h_1, h_1) and the pair (h_2, m_2), so that no specific lexical semantic constraint is applied to the head h_1.

As formally described in Algorithm 1, other cases are discarded, e.g. heads are unknown or different POS tags are observed for the heads h_i of the n_1 and n_2 nodes. The factor lw is here adopted to reduce the contribution of non lexical nodes.

5 Experimental Evaluations

In order to study the behavior of the proposed compositional kernels onto the automation of semantic inference over texts, we carried out experiments over two fine grained linguistic tasks, i.e. Question Classification (QC) and Semantic Text Similarity (STS). In all experiments, texts are processed with the Stanford CoreNLP[5] and the resulting dependency trees are extended with compositional information as discussed in Section 3. The lexical similarity function σ_{LEX} exploited in SPTK and CSPTK through the Algorithm 1 was based on a distributional analysis of the UkWaC corpus, that gives rise to a word vector representation for all words occurring more than 100 times (i.e. the *targets*), as discussed in [12]. The vector dimensions correspond to co-occurrences within a left or right window of 3 tokens between the targets and the set of the 20,000 most frequent words (i.e. *features*) in UkWaC. Scores correspond to point-wise Mutual Information values between each feature and a target[6] across the entire corpus. The Singular Value Decomposition is then applied to the input matrix with a space dimensionality cut at $k = 250$. The similarity σ_{LEX} required by Algorithm 1 corresponds to the cosine similarity in the resulting space, as in [10].

5.1 Question Classification

Question Classification (QC) is the task of assigning one (or more) class label(s) to a given question written in natural language. Question classification plays an important role in Question Answering (QA) [31], as it has been shown that its performance significantly influences the overall performance of a QA system, e.g. [17]. Thanks to availability of our kernel similarity models, the QC task has been modeled directly in terms of the parse trees representing questions, i.e. the classification objects. The reference corpus is the UIUC dataset [20], including 5,452 questions for training and 500 questions for test[7], organized in six coarse-grained classes, i.e. ABBREVIATION, ENTITY, DESCRIPTION, HUMAN, LOCATION and NUMBER. SVM training has been carried out by applying (i) the PTK and SPTK kernels over the LCT and GRCT representations and (ii) the compositional tree kernels (CSPTKs), according to different compositional similarity metrics σ_{LEX}, to the CLCT and CGRCT formats. For learning our models, we used the Kernel-based Learning Platform[8], which includes structural kernels, i.e.,

[5] http://nlp.stanford.edu/software/corenlp.shtml

[6] Left contexts of targets are treated separately from the right ones so that 40,000 features are used derived for each target.

[7] http://cogcomp.cs.illinois.edu/Data/QA/QC/

[8] http://sag.art.uniroma2.it/demo-software/kelp/

STK and PTK: i) with the smooth match between tree leaves, i.e. the SPTK defined in [10] and ii) with the smooth match between compositionally labeled nodes of the tree, as in the CSPTK. Different tree representations are denoted in the subscripts, so that CSPTK$_{CLCT}$ refers to the use of a CSPTK over the compositionally labeled LCT (as in Fig. 2). As the compositional kernel has to make use of a compositional similarity metrics, the notation of different kernels is specialized according to the adopted metrics. Additive (with $\alpha = \beta$), dilation (with $\lambda = 1$) and point-wise multiplication models have been adopted and denoted by $^{+}$, ^{d} and $^{\cdot}$ superscripts respectively.

The accuracy achieved by the different systems is the percentage of sentences that are correctly assigned to the proper question class, and it is reported in Table 1. The parameter of the SVM has been estimated in a held-out subset sampled from the training material. As a baseline, a simple Linear kernel (LIN) over a bag-of-word model (denoted by BoW) is reported (row 1) representing questions as binary word vectors and resulting in a lexical overlap kernel. A first observation is that the introduction of lexical semantic information in tree kernel operators, such as in SPTK vs. PTK, is beneficial, confirming the outcomes of previous studies, [10]. Moreover, the compositional kernels proposed (CSPTK) seem to make an effective use of the compositional distributional semantic information as they all outperform their non compositional counterpart. Among the compositional kernels the ones adopting the simple sum operator, i.e. CSPTK$^{+}_{CLCT}$, seem to outperform all the other compositional operators. This remains true independently from the type of tree adopted, i.e. LCT vs. GRCT. The Lexically Centered Tree seems to better exploit compositional information, as it achieves, on average, higher accuracy than GRCT, given that it emphasizes lexical nodes that occur closer to the tree root an thus contribute mostly to the kernel recursive computation.

Kernel	**Accuracy**
LIN$_{BOW}$	86,8%
PTK$_{LCT}$	90,4%
PTK$_{GRCT}$	89,8%
SPTK$_{LCT}$	94,0%
SPTK$_{GRCT}$	91,4%
CSPTK$^{+}_{CLCT}$	**94,8%**
CSPTK$^{+}_{CGRCT}$	94,0%
CSPTK$^{\cdot}_{CLCT}$	93,6%
CSPTK$^{\cdot}_{CGRCT}$	93,6%
CSPTK$^{d}_{CLCT}$	93,6%
CSPTK$^{d}_{CGRCT}$	92,8%

Table 1. Results in the QC task

	Kernel	**Pearson**
Unsupervised	PTK$_{LCT}$	0.291
	PTK$_{GRCT}$	0.118
	SPTK$_{LCT}$	0.270
	SPTK$_{GRCT}$	0.102
	CSPTK$^{+}_{CLCT}$	0.370
	CSPTK$^{+}_{CGRCT}$	0.400
	CSPTK$^{\cdot}_{CLCT}$	0.316
	CSPTK$^{\cdot}_{CGRCT}$	0.341
	CSPTK$^{d}_{CLCT}$	0.432
	CSPTK$^{d}_{CGRCT}$	0.454
Regression	LIN$_{BOW}$	0.537
	LIN$_{BOW}$ + PTK$_{LCT}$	0.578
	LIN$_{BOW}$ + PTK$_{GRCT}$	0.630
	LIN$_{BOW}$ + PTK$_{LCT}$ + SPTK$_{LCT}$	0.577
	LIN$_{BOW}$ + PTK$_{GRCT}$ + SPTK$_{LCT}$	0.636
	LIN$_{BOW}$ + PTK$_{LCT}$ + CSPTK$^{d}_{CLCT}$	0.590
	LIN$_{BOW}$ + PTK$_{GRCT}$ + CSPTK$^{d}_{CGRCT}$	**0.660**

Table 2. Results in the STS task: unsupervised and supervised settings.

5.2 The Semantic Text Similarity task

The second experiment aims at evaluating the contribution of the different kernels in estimating a semantic similarity between text pairs. A meaningful similarity function between texts is crucial in many NLP and Retrieval tasks such as Textual Entailment, Paraphrasing, Search Re-ranking, Machine Translation, Text Summarization or Deep Question Answering. Kernel functions are evaluated against the dataset of the Semantic Text Similarity (STS) task built in SemEval 2012 [1], in particular the MSR Video Paraphrase (MSRvid) dataset . This dataset is composed of 1,500 sentence pairs (split in 750 train and 750 test pairs). Each sentence is a short description of a video and the pairs have been manually labeled through Amazon Mechanical Turk with a numerical score (between 0 and 5) reflecting the similarity between the actions and actors involved in the video. As an example the sentences "*A woman is slicing an onion*" and "*A woman is cutting an onion*" have a similarity of 4.2, while "*A man is talking*" and "*A man is walking*" have a similarity of 1. The task consists in assigning a similarity score to each test pair and the performance is measured in terms of the Pearson Correlation between the obtained scores and the gold-standard ones, as reported in Table 2.

The same kernels and syntactic structures of the previous experiments have been considered. The kernel functions have been applied to the STS task both in an unsupervised as well as a supervised fashion. In the first case, as shown in the top rows of Table 2, each kernel has been evaluated by measuring the Pearson correlation between the result of the kernel function between each text pair and the human score. In the supervised setting, shown in the bottom rows of Table 2, the kernel scores are used to populate a feature vector used to train a SVM based regressor [30], which determines the best kernel combination approximating the human scores over the test material.

The results achieved by the single kernels suggests that the semantic similarity in the MSRVid dataset strongly depends on the lexical overlap between sentence pairs, and a regressor trained with a linear kernel over the Bag-of-Word representations (Lin$_{BoW}$) achieve a correlation of 0.537. This result is higher with respect to the unsupervised application of the PTK where only full syntactic information is used; this is mainly due to the fact that most sentences are in the form `Subject-Verb-Object` and the syntactic information is not discriminative. The simple smoothing over lexical nodes in the SPTK is still not very informative, as the smoothing over the pure lexical information may be misleading. As an example, the sentence pair "*A man plays a guitar*" has a low similarity (i.e. 2.0) w.r.t. "*A boy plays a piano*" as they evoke different actions: while generalizing is useful in comparing *boy* and *man*, it can be misleading in comparing *guitar* and *piano*. Such smoothing is more controlled in the CSPTK that achieves better results. The above difference is mainly due to the increasing sensitivity of PTK, SPTK and CSPTK to the incrementally rich lexical information. This is especially evident in sentence pairs with very similar syntactic structure. For example a sentence pair is given by *The man are playing soccer* and *A man is riding a motorcycle*, that are strictly syntactically correlated. In fact, PTK provide a similarity score of 0.647 between the to sentences as differences between tree structures is confined to the leaves. By scoring 0.461, SPTK introduces an improvement as the distributional similarity that acts as a smoothing factor between lexical nodes better discriminates uncorrelated words, like *motorcycle* and *soccer*. However, ambiguous words such verbs *ride* and *play* are still promoting a similarity that is locally misleading. Notice that both PTK and SPTK receive a strong contribution in the recursive computation of the kernels by the left branching of the tree, as the subject is the same, i.e. *man*. Compositional information about direct objects (*soccer* vs. *motorcycle*) is better propagated by the CSPTK^{dilation} operator. Its final scores for the pair is 0.36, as semantic differences between the sentences are emphasized. Even if grammatical types strongly contribute to the final score (as in PTK or SPTK), now the DCS computation over intermediate nodes (i.e. the VPs (*ride, motorcycle*) and (*play, soccer*) is faced with less ambiguity with corresponding lower scores. This improvement is even more noticeable in a supervised setting where the feature vector derived with the Lin$_{BoW}$ kernel is first enriched with the PTK scores, and then with the score of the SPTK and CSPTK. Correlation of the latter kernel is valuable with respect to the pure lexical overlap, resulting in the best results of 0.66.

6 Conclusions

In this work, a model for compositionality in natural language within a kernel formulation has been introduced. It accounts for structural (i.e. grammatical) and lexical properties by adopting vector representations as models for distributional semantics and extending previously proposed tree kernels. As an improvement to previous extensions of tree kernels (e.g. [10]), the smoothing here proposed applies in a context-aware fashion throughout the parse tree, and is not only focusing on lexical nodes. The Compositionally Smoothed Partial Tree Kernel thus corresponds to a Convolution Kernel that measures the semantic similarity between complex linguistic structures, where local matching at sub-trees is sensible to head-modifier compositional properties. The results obtained by applying CSPTK across different NLP tasks, such as Question Classification and Semantic Text Similarity, confirm the robustness and the generalization capability of this kernel. Comparison against simpler lexical models or pure grammatical kernels shows that systematically CSPTK is the best model. CSPTK can be also easily designed against new domains (i.e. corpora and annotated datasets), but mainly new compositional operators will be investigated, such as the Support Subspace [2].

References

1. Agirre, E., Diab, M., Cer, D., Gonzalez-Agirre, A.: Semeval-2012 task 6: A pilot on semantic textual similarity. In: Proceedings SemEval 2012. Stroudsburg, PA, USA (2012)
2. Annesi, P., Storch, V., Basili, R.: Space projections as distributional models for semantic composition. In: In Proceedings of CICLing 2012. vol. 7181, pp. 323–335 (2012)
3. Annesi, P., Croce, D., Basili, R.: Semantic compositionality in tree kernels. In: Li, J., Wang, X.S., Garofalakis, M.N., Soboroff, I., Suel, T., Wang, M. (eds.) Proc. of CIKM 2014 (2014)
4. Baroni, M., Lenci, A.: Distributional memory: A general framework for corpus-based semantics. Computational Linguistics 36(4), 673–721 (2010)
5. Baroni, M., Zamparelli, R.: Nouns are vectors, adjectives are matrices: representing adjective-noun constructions in semantic space. In: Proceedings of the EMNLP 2010 (2010)
6. Basile, P., Caputo, A., Semeraro, G.: Encoding syntactic dependencies by vector permutation. In: Proc. of the EMNLP 2011 Workshop on GEMS. pp. 43–51. Edinburgh, UK (2011)
7. Coecke, B., Sadrzadeh, M., Clark, S.: Mathematical foundations for a compositional distributional model of meaning. CoRR abs/1003.4394 (2010)

8. Collins, M., Duffy, N.: Convolution kernels for natural language. In: Proceedings of Neural Information Processing Systems (NIPS). pp. 625–632 (2001)
9. Copestake, A., Flickinger, D., Pollard, C., Sag, I.: Minimal Recursion Semantics: An Introduction. Research on Language &amp; Computation 3(2-3), 281–332 (Dec 2005)
10. Croce, D., Moschitti, A., Basili, R.: Structured lexical similarity via convolution kernels on dependency trees. In: Proceedings of EMNLP. Edinburgh, Scotland, UK. (2011)
11. Croce, D., Annesi, P., Storch, V., Basili, R.: Unitor: Combining semantic text similarity functions through sv regression. In: Proceedings of SemEval 2012. pp. 597–602 (2012)
12. Croce, D., Previtali, D.: Manifold learning for the semi-supervised induction of framenet predicates: An empirical investigation. In: Proceedings of GEMS 2010 (2010)
13. Erk, K., Pado, S.: A structured vector space model for word meaning in context. In: Proceedings of EMNLP 2008. pp. 897–906 (2008)
14. Foltz, P., Kintsch, W., Landauer, T.: The measurement of textual coherence with latent semantic analysis. Discourse Processes 25 (1998)
15. Grefenstette, E., Sadrzadeh, M.: Experimental support for a categorical compositional distributional model of meaning. CoRR abs/1106.4058 (2011)
16. Haussler, D.: Convolution kernels on discrete structures. Tech. rep., Dept. of Computer Science, University of California at Santa Cruz (1999)
17. Hovy, E., Gerber, L., Hermjakob, U., Lin, C.Y., Ravichandran, D.: Toward semantics-based answer pinpointing. In: Proc. of HLTR. pp. 1–7 (2001)
18. Jiang, P., Zhang, C., Fu, H., Niu, Z., Yang, Q.: An approach based on tree kernels for opinion mining of online product reviews. In: Proceedings of ICDM 2010. pp. 256–265 (Dec 2010)
19. Landauer, T., Dumais, S.: A solution to platos problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. Psychological review pp. 211–240 (1997)
20. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of ACL '02. pp. 1–7. COLING '02, Stroudsburg, PA, USA (2002)
21. Mitchell, J., Lapata, M.: Vector-based models of semantic composition. In: In Proceedings of ACL/HLT 2008. pp. 236–244 (2008)
22. Moschitti, A., Quarteroni, S., Basili, R., Manandhar, S.: Exploiting syntactic and shallow semantic kernels for question answer classification. In: ACL (2007)
23. Pado, S., Lapata, M.: Dependency-based construction of semantic space models. Computational Linguistics 33(2) (2007)
24. Rudolph, S., Giesbrecht, E.: Compositional matrix-space models of language. In: Proceedings of ACL 2010. pp. 907–916. Stroudsburg, PA, USA (2010)
25. Sahlgren, M.: The Word-Space Model. Ph.D. thesis, Stockholm University (2006)
26. Schütze, H.: Automatic word sense discrimination. Comput. Linguist. 24(1), 97–123 (1998)
27. Smolensky, P.: Tensor product variable binding and the representation of symbolic structures in connectionist systems. Artif. Intell. 46, 159–216 (November 1990)
28. Thater, S., Fürstenau, H., Pinkal, M.: Contextualizing semantic representations using syntactically enriched vector models. In: Proceedings of ACL 2010. pp. 948–957 (2010)
29. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. Journal of artificial intelligence research 37, 141 (2010)
30. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience (1998)
31. Voorhees, E.M.: Overview of the trec 2001 question answering track. In: TREC (2001)
32. Zanzotto, F.M., Korkontzelos, I., Fallucchi, F., Manandhar, S.: Estimating linear models for compositional distributional semantics. In: Proceedings of COLING 2010 (2010)