Distributed Smoothed Tree Kernel

Lorenzo Ferrone
University of Rome "Tor Vergata"
Via del Politecnico 1
00133 Roma, Italy
lorenzo.ferrone@gmail.com

Fabio Massimo Zanzotto
University of Rome "Tor Vergata"
Via del Politecnico 1
00133 Roma, Italy
fabio.massimo.zanzotto@uniroma2.it

English. *In this paper we explore the possibility to merge the world of Compositional Distributional Semantic Models (CDSM) with Tree Kernels (TK). In particular, we will introduce a specific tree kernel (*smoothed tree kernel*, or STK) and then show that is possibile to approximate such kernel with the dot product of two vectors obtained compositionally from the sentences, creating in such a way a new CDSM.*

Italiano. *In questo paper vogliamo esplorare la possibilità di unire il mondo dei metodi di semantica distribuzione composizionale (CDSM) con quello dei tree Kernel (TK). In particolare introdurremo un particolare tree kernel e poi mostreremo che è possibile approssimare questo kernel tramite il prodotto scalare tra due vettori ottenuti composizionalmente a partire dalle frasi di partenza, creando così di fatto un nuovo modello di semantica distribuzionale composizionale.*

1. Introduction

Compositional distributional semantics is a flourishing research area that leverages distributional semantics (see (Baroni and Lenci 2010)) to produce meaning of simple phrases and full sentences (hereafter called *text fragments*). The aim is to scale up the success of word-level relatedness detection to longer fragments of text. Determining similarity or relatedness among sentences is useful for many applications, such as multi-document summarization, recognizing textual entailment (Dagan et al. 2013), and semantic textual similarity detection (Agirre et al. 2013). Compositional distributional semantics models (CDSMs) are functions mapping text fragments to vectors (or higher-order tensors). Functions for simple phrases directly map distributional vectors of words to distributional vectors for the phrases (Mitchell and Lapata 2008; Baroni and Zamparelli 2010; Zanzotto et al. 2010). Functions for full sentences are generally defined as recursive functions over the ones for phrases (Socher et al. 2011). Distributional vectors for text fragments are then used as input in larger machine learning algorithm, for example as layers in neural networks, or to compute similarity among text fragments directly via dot product or cosine similarity.

CDSMs generally exploit structured representations t^x of text fragments x to derive their meaning, in the form of a vector of real number $f(t^x)$. The structural information, although extremely important, is only used to guide the composition process, but it is obfuscated in the final vectors. Structure and meaning can interact in unexpected ways when computing cosine similarity (or dot product) between vectors of two text fragments, as shown for full additive models in (Ferrone and Zanzotto 2013).

Smoothed tree kernels (STK) are instead a family of kernels which realize a clearer interaction between structural information and distributional meaning (Croce, Mos-

chitti, and Basili 2011; Mehdad, Moschitti, and Zanzotto 2010). STKs are specific realizations of convolution kernels (Haussler 1999) where the similarity function is recursively (and, thus, compositionally) computed. Distributional vectors are used to represent word meaning in computing the similarity among nodes. STKs, however, are not considered part of the CDSMs family, in fact, as usual in kernel machines (Cristianini and Shawe-Taylor 2000), STKs directly compute the similarity between two text fragments x and y over their tree representations t^x and t^y, that is, $STK(t^x, t^y)$. Because STK is a valid kernel, there exist a function $f : T \rightarrow \mathbb{R}^n$ such that:

$$STK(t^x, t^y) = \langle f(t^x), f(t^y) \rangle$$

However, the function f that maps trees into vectors is never explicity used, and, thus, $STK(t^x, t^y)$ is not explicitly expressed as the dot product or the cosine between $f(t^x)$ and $f(t^y)$.

Such a function f, which is the underlying reproducing function of the kernel (Aronszajn 1950), would be a CDSM in its own right, since it maps trees to vectors, also including distributional meaning. However, the huge dimensionality of $\mathbb{R}^n$ (since it has to represent the set of all possible subtrees) prevents to actually compute the function $f(t)$, which thus can only remain *implicit*.

Distributed tree kernels (DTK) (Zanzotto and Dell'Arciprete 2012) partially solve the last problem. DTKs approximate standard tree kernels (such as (Collins and Duffy 2002)) by defining an *explicit* function DT that maps trees to vectors in $\mathbb{R}^m$ where $m \ll n$ and $\mathbb{R}^n$ is the explicit space for tree kernels. DTKs approximate standard tree kernels (TK), that is,

$$\langle DT(t^x), DT(t^y) \rangle \approx TK(t^x, t^y)$$

by approximating the corresponding reproducing function. In this sense distributed trees are low-dimensional vectors that encode structural information. In DTKs tree nodes u and v are represented by nearly orthonormal vectors, that is, vectors $\mathbf{u}$ and $\mathbf{v}$ such that: $\langle \mathbf{u}, \mathbf{v} \rangle \approx \delta(\mathbf{u}, \mathbf{v})$ where δ is the Kroneker's delta function, defined as:

$$\delta(\mathbf{u}, \mathbf{v}) = \begin{cases} 1 \text{ if } \mathbf{u} = \mathbf{v} \\ 0 \text{ if } \mathbf{u} \neq \mathbf{v} \end{cases}$$

This is in contrast with distributional semantics vectors where the dot product $\langle \mathbf{u}, \mathbf{v} \rangle$ is allowed to take on any value in $[0, 1]$ according to the semantic similarity between the words u and v.

In this paper, leveraging on distributed trees, we present a novel class of CDSMs that encode both structure and distributional meaning: the distributed smoothed trees (DST). DSTs encode both structure and distributional meaning in a rank-2 tensor (a matrix): one dimension encodes the structure and one dimension encodes the meaning. By using DSTs to compute the similarity among sentences with a generalized dot product (or cosine), we implicitly define the distributed smoothed tree kernels (DSTK) which approximate the corresponding STKs.

We present two DSTs along with the two smoothed tree kernels (STKs) that they approximate.

We experiment with our DSTs to show that their generalized dot products approximate STKs by directly comparing the produced similarities and by comparing

their performances on two tasks: recognizing textual entailment (RTE) and semantic similarity detection (STS). Both experiments show that the dot product on DSTs approximates STKs and, thus, DSTs encode both structural and distributional semantics of text fragments in tractable rank-2 tensors. Experiments on STS and RTE show that distributional semantics encoded in DSTs increases performance over structure-only kernels.

DSTs are the first positive way of taking into account both structure and distributional meaning in CDSMs.

The rest of the paper is organized as follows. Section 2 introduces the necessary background on distributed trees (Zanzotto and Dell'Arciprete 2012) used in the rest of the paper, 3.1 introduces the basic notation used in the paper. Section 3 describe our distributed smoothed trees as compositional distributional semantic models that can represent both structural and semantic information. Section 5 reports on the experiments. Finally, Section 6 draws some conclusions and possibilities for future works.

2. Background: DTK

Encoding Structures with Distributed Trees (Zanzotto and Dell'Arciprete 2012) (DT) is a technique to embed the structural information of a syntactic tree into a dense, low-dimensional vector of real numbers. DT were introduced in order to allow one to exploit the modelling capacity of tree kernels (Collins and Duffy 2001) but without their computational complexity. More specifically for each tree kernel TK (Aiolli, Da San Martino, and Sperduti 2009; Collins and Duffy 2002; Vishwanathan and Smola 2002; Kimura et al. 2011) there is a corresponding distributed tree function (Zanzotto and Dell'Arciprete 2012) which maps from trees to vectors:

$$\begin{aligned} \mathrm{DT}\colon T &\to \mathbb{R}^d \\ t &\mapsto \mathrm{DT}(t) = \mathbf{t} \end{aligned}$$

such that:

$$\langle \mathrm{DT}(t_1), \mathrm{DT}(t_2) \rangle \approx \mathrm{TK}(t_1, t_2) \tag{1}$$

where $t \in T$ is a tree, $\langle \cdot, \cdot \rangle$ indicates the standard inner product in $\mathbb{R}^d$ and $\mathrm{TK}(\cdot, \cdot)$ represents the original tree kernel. It has been shown that the quality of the approximation depends on the dimension d of the embedding space $\mathbb{R}^d$.

To approximate tree kernels, distributed trees use the following property and intuition. It is possible to represent subtrees $\tau \in S(t)$ of a given tree t in distributed tree fragments $\mathrm{DTF}(\tau) \in \mathbb{R}^d$ such that:

$$\langle \mathrm{DTF}(\tau_1), \mathrm{DTF}(\tau_2) \rangle \approx \delta(\tau_1, \tau_2) \tag{2}$$

Where δ is the Kronecker's delta function. With this definition we can define the distributed tree of a given tree t as a summation over all of its subtrees, that is:

$$\mathrm{DT}(t) = \sum_{\tau \in S(t)} \sqrt{\lambda}^{|\mathcal{N}(\tau)|} \mathrm{DTF}(\tau)$$

where λ is the classical decaying factor in tree kernels (Collins and Duffy 2002), used to penalize the importance given to longer tree, and $|\mathcal{N}(\tau)|$ is the cardinality of the set of the nodes of the subtree τ. With this definition in place one can show that the property in Equation 1 holds.

Distributed tree fragments are defined as follows. To each node label n we associate a random vector $\mathbf{n}$ drawn randomly from the d-dimensional hypersphere. Random vectors of high dimensionality have the property of being quasi-orthonormal (that is, they obey a relationship similar to equation (2)). The following functions are then defined:

$$\text{DTF}(\tau) = \bigodot_{n \in \mathcal{N}(\tau)} \mathbf{n}$$

where $\odot$ indicates the shuffled circular convolution operation [1], which has the property of preserving quasi-orthonormality between vectors.

To actually compute distributed trees in an efficient manner however, a different (equivalent) formulation is used. Firstly we define a function SN(n) for each node n in a tree t that collects all the distributed tree fragments of t, where n is its head:

$$\text{SN}(n) = \begin{cases} \mathbf{0} \text{ if } n \text{ is terminal} \\ \mathbf{n} \odot \bigodot_i \sqrt{\lambda}\,[\mathbf{n_i} + SN(n_i)] \text{ otherwise} \end{cases} \tag{3}$$

where n_i are the direct children of n in the tree t. Given S(n), distributed trees can be efficiently computed as:

$$\text{DT}(t) = \sum_{n \in \mathcal{N}} \text{SN}(n)$$

In the next section we will finally generalize the ideas of DTK in order to also include semantic information.

3. Distributed Smoothed Tree Kernel

We here propose a model that can be considered a compositional distributional semantic model as it transforms sentences into matrices (which can also be seen as vectors, once they have been "flattened") that can then used by the learner as feature vectors. Our model is called *Distributed Smoothed Tree Kernel* (Ferrone and Zanzotto 2014) as it mixes the distributed trees which we introduced in the previous section (Zanzotto and Dell'Arciprete 2012) representing syntactic information with distributional semantic vectors representing semantic information, as used in the smoothed tree kernels (Croce, Moschitti, and Basili 2011).

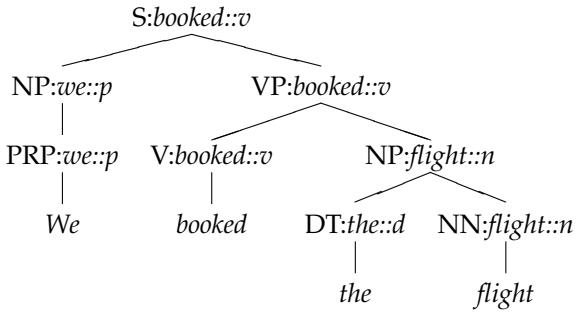


Figure 1
A lexicalized tree

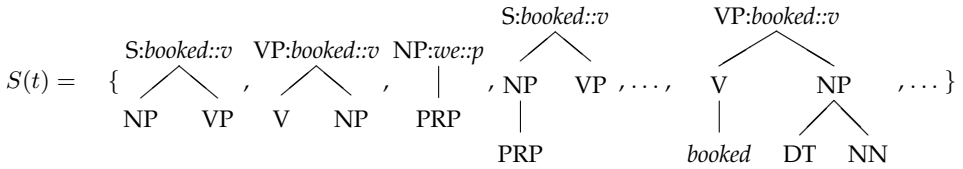


Figure 2
Subtrees of the tree t in figure (1) (a non-exhaustive list)

3.1 Notation

Before describing the *distributed smoothed trees* (DST) we introduce a formal way to denote constituency-based *lexicalized parse trees*, as DSTs exploit this kind of data structures.

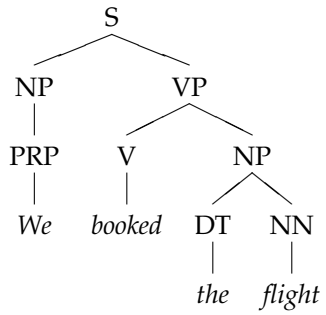
Lexicalized trees are denoted with the letter t and $N(t)$ denotes the set of non terminal nodes of tree t. Each non-terminal node $n \in N(t)$ has a label l_n composed of two parts $l_n = (s_n, w_n)$: s_n is the syntactic label, (for example NP, VP, S, and so forth) while w_n is the semantic headword of the tree headed by n, along with its part-of-speech tag. The semantic headwords are derived with the Stanford Parser implementation of Collins' rules (Collins 1999).

Terminal nodes of trees are treated differently, these nodes represent only words w_n without any additional information, and their labels thus only consist of the word itself. An example of such a structure can be seen in figure (1).

The structure of a DST is represented as follows: Given a tree t, we will use $\mathsf{h}(t)$ to indicate its root node and $\mathsf{s}(t)$ to indicate its syntactic part. That is, $\mathsf{s}(t)$ is the tree derived from t but considering only the syntactic structure (that is, only the s_n part of the labels).

1 The circular convolution between $\mathbf{a}$ and $\mathbf{b}$ is defined as the vector $\mathbf{c}$ with component $c_i = \sum_j a_j b_{i-j \bmod d}$. The shuffled circular convolution is the circular convolution after the vectors have been randomly shuffled.

For example the tree in figure (1) is mapped to the tree:



We will also use $c_i(n)$ to denote i-th child of a node n. As usual for constituency-based parse trees, pre-terminal nodes are nodes that have a single terminal node as child. Finally, we use $\mathbf{w_n} \in \mathbb{R}^k$ to denote the *distributional* vector for word w_n.

3.2 The method at a glance

We describe here the approach in a few sentences. In line with tree kernels over structures (Collins and Duffy 2002), we introduce the set $S(t)$ of the subtrees t_i of a given lexicalized tree t. A subtree t_i is in the set $S(t)$ if $\mathsf{s}(t_i)$ is a subtree of $\mathsf{s}(t)$ and, if n is a node in t_i, all the siblings of n in t are in t_i. For each node of t_i we only consider its syntactic label s_n, except for the head $\mathsf{h}(t_i)$ for which we also consider its semantic component w_n (see Fig. 2).

In analogy with equation (2) the functions DSTs we define compute the following sum:

$$\text{DST}(t) = \mathbf{T} = \sum_{t_i \in S(t)} \mathbf{T}_i$$

where $\mathbf{T}_i$ is the matrix associated to each subtree t_i (how this matrix is computed will be explained in the following).

The similarity between two text fragments a and b represented as lexicalized trees t^a and t^b can be then computed using the Frobenius product between the two matrices $\mathbf{T}^a$ and $\mathbf{T}^b$, that is:

$$\text{DSTK}(t_a, t_b)) = \langle \mathbf{T}^a, \mathbf{T}^b \rangle_F = \sum_{\substack{t_i^a \in S(t^a) \\ t_j^b \in S(t^b)}} \langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \tag{4}$$

This is nothing more than the usual dot product between two vectors, if we flatten the two $m \times k$ matrices into two vectors, each with mk components.

We want to generalize equation (2), and obtain that the product $\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F$ approximates the following similarity between lexicalized trees:

$$\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \approx \begin{cases} \langle \mathbf{w}_{\mathsf{h}(t_i^a)}, \mathbf{w}_{\mathsf{h}(t_j^b)} \rangle \text{ if } \mathsf{s}(t_i^a) = \mathsf{s}(t_j^b) \\ 0 \text{ otherwise} \end{cases}$$

In other words, whenever two subtrees have the same syntactic structure, we define their similarity as the semantic similarity of their heads (as computed via dot product of the corresponding distributional vectors), when their syntactic structure is different we instead define their similarity to be 0.

This definition can also be written as:

$$\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \approx \delta(\mathsf{s}(t_i^a), \mathsf{s}(t_j^b)) \cdot \langle \mathbf{w}_{\mathsf{h}(t_i^a)}, \mathbf{w}_{\mathsf{h}(t_j^b)} \rangle \tag{5}$$

In order to obtain the above approximation property, we define:

$$\mathbf{T}_i = \mathsf{s}(\mathbf{t_i}) \otimes \mathbf{w}_{\mathsf{h}(t_i)}$$

where $\mathsf{s}(\mathbf{t_i})$ are distributed tree fragment (Zanzotto and Dell'Arciprete 2012) for the subtree t, $\mathbf{w}_{\mathsf{h}(t_i)}$ is the distributional vector of the head of the subtree t and $\otimes$ denotes the tensor product. In this particular case, the tensor product is equivalent to the matrix $\mathsf{s}(\mathbf{t_i})\mathbf{w}_{\mathsf{h}(t_i)}^\top$, between a column vector and a row vector.

Exploiting the following properties of the tensor and Frobenius product:

$$\langle \mathbf{a} \otimes \mathbf{w}, \mathbf{b} \otimes \mathbf{v} \rangle_F = \langle \mathbf{a}, \mathbf{b} \rangle \cdot \langle \mathbf{w}, \mathbf{v} \rangle$$

we have that Equation (5) is satisfied as:

$$\begin{aligned} \langle \mathbf{T}_i, \mathbf{T}_j \rangle_F &= \langle \mathsf{s}(\mathbf{t_i}), \mathsf{s}(\mathbf{t_j}) \rangle \cdot \langle \mathbf{w}_{\mathsf{h}(t_i)}, \mathbf{w}_{\mathsf{h}(t_j)} \rangle \\ &\approx \delta(\mathsf{s}(t_i), \mathsf{s}(t_j)) \cdot \langle \mathbf{w}_{\mathsf{h}(t_i)}, \mathbf{w}_{\mathsf{h}(t_j)} \rangle \end{aligned}$$

As in the distributed trees, it is possible to introduce a different formulation to compute $\mathrm{DST}(t)$. Such formulation has the advantage of being more computationally efficient, and also makes it clear that the process is compositional in nature, because it composes distributional and distributed vector of each node.

More specifically, it can be shown that:

$$\mathrm{DST}(t) = \sum_{n \in \mathcal{N}} \mathrm{SN^*}(n)$$

where $\mathrm{SN}*$ is defined as:

$$\mathrm{SN^*}(n) = \begin{cases} \mathbf{0} \text{ if } n \text{ is terminal} \\ \mathrm{SN}(n) \otimes \mathbf{w_n} \text{ otherwise} \end{cases}$$

and $\mathrm{S}(n)$ is the same as in equation (3).

It is possible to show that the overall compositional distributional model $\mathrm{DST(t)}$ can be obtained with a recursive algorithm that exploits vectors of the nodes of the tree.

We actually propose two slightly different versions of our DSTs according to how we produce distributional vectors for words. We have a plain version DST_0 when we use distributional vectors $\mathbf{w_n}$ as they are, and a slightly modified version DST_{+1} when we use as distributional vectors $\mathbf{w_n}' = \begin{pmatrix} 1 & \mathbf{w_n} \end{pmatrix}$.

4. The Approximated Smoothed Tree Kernels

The two CDSM we propose approximate two specific tree kernels belonging to the smoothed tree kernels class. These recursively computes (but, the recursive formulation is not given here) the following general equation:

$$STK(t^a, t^b) = \sum_{\substack{t_i \in S(t^a) \\ t_j \in S(t^b)}} \omega(t_i, t_j)$$

where $\omega(t_i, t_j)$ is the similarity weight between two subtrees t_i and t_j. $DTSK_0$ and $DSTK_{+1}$ approximate respectively the kernels STK_0 and STK_{+1} defined respectively by the following equations for the weights:

$$\omega_0(t_i, t_j) = \langle \mathbf{w}_{\mathsf{h}(\mathbf{t_i})}, \mathbf{w}_{\mathsf{h}(\mathbf{t_j})} \rangle \cdot \delta(\mathsf{s}(t_i), \mathsf{s}(t_j)) \cdot \sqrt{\lambda^{|N(t_i)|+|N(t_j)|}}$$

$$\omega_{+1}(t_i, t_j) = (\langle \mathbf{w}_{\mathsf{h}(\mathbf{t_i})}, \mathbf{w}_{\mathsf{h}(\mathbf{t_j})} \rangle + 1) \cdot \delta(\mathsf{s}(t_i), \mathsf{s}(t_j)) \cdot \sqrt{\lambda^{|N(t_i)|+|N(t_j)|}}$$

5. Experimental investigation

5.1 Experimental set-up

Generic settings. We experimented with two datasets: the Recognizing Textual Entailment datasets (RTE) (Dagan, Glickman, and Magnini 2006) and the the Semantic Textual Similarity 2013 datasets (STS) (Agirre et al. 2013). The STS task consists of determining the degree of similarity (ranging from 0 to 5) between two sentences. We used the data for core task of the 2013 challenge data. The STS datasets contains 5 datasets: headlines, OnWN, FNWN, SMT and MSRpar, which contains respectively 750, 561, 189, 750 and 1500 pairs. The first four datasets were used for testing, while all the training has been done on the fifth. RTE is instead the task of deciding whether a long text T entails a shorter text, typically a single sentence, called hypothesis H. It has been often seen as a classification task (see (Dagan et al. 2013)). We used four datasets: RTE1, RTE2, RTE3, and RTE5, with the standard split between training and testing. The dev/test distribution for RTE1-3, and RTE5 is respectively 567/800, 800/800, 800/800, and 600/600 T-H pairs.

Distributional vectors are derived with DISSECT (Dinu, The Pham, and Baroni 2013) from a corpus obtained by the concatenation of ukWaC (wacky.sslmit.unibo.it), a mid-2009 dump of the English Wikipedia (en.wikipedia.org) and the British National Corpus (www.natcorp.ox.ac.uk), for a total of about 2.8 billion words. We collected a 35K-by-35K matrix by counting co-occurrence of the 30K most frequent content lemmas in the corpus (nouns, adjectives and verbs) and all the content lemmas occurring in the datasets within a 3 word window. The raw count vectors were transformed into positive Pointwise Mutual Information scores and reduced to 300 dimensions by Singular Value Decomposition. This setup was picked without tuning, as we found it effective in previous, unrelated experiments.

		RTE1	RTE2	RTE3	RTE5	headl	FNWN	OnWN	SMT
STK_0 vs $DSTK_0$	1024	0.86	0.84	0.90	0.84	0.87	0.65	0.95	0.77
	2048	0.87	0.84	0.91	0.84	0.90	0.65	0.96	0.77
STK_{+1} vs $DSTK_{+1}$	1024	0.81	0.77	0.83	0.72	0.88	0.53	0.93	0.66
	2048	0.82	0.78	0.84	0.74	0.91	0.56	0.94	0.67

Table 1
Spearman's correlation between Distributed Smoothed Tree Kernels and Smoothed Tree Kernels

To build our DTSKs and for the two baseline kernels TK and DTK, we used the implementation of the distributed tree kernels[2]. We used: 1024 and 2048 as the dimension of the distributed vectors, the weight λ is set to 0.4 as it is a value generally considered optimal for many applications (see also (Zanzotto and Dell'Arciprete 2012)).

The statistical significance, where reported, is computed according to the sign test.

Direct correlation settings. For the *direct correlation* experiments, we used the RTE data sets and the testing sets of the STS dataset (that is, *headlines*, *OnWN*, *FNWN*, *SMT*). We computed the Spearman's correlation between values produced by our $DSTK_0$ and $DSTK_{+1}$ and produced by the standard versions of the smoothed tree kernel, that is, respectively, STK_0 and STK_{+1}. We obtained text fragment pairs by randomly sampling two text fragments in the selected set. For each set, we produced exactly the number of examples in the set, e.g., we produced 567 pairs for RTE1 dev, etc..

Task-based settings. For the *task-based* experiments, we compared systems using the standard evaluation measure and the standard split in the respective challenges. As usual in RTE challenges the measure used is the accuracy, as testing sets have the same number of entailment and non-entailment pairs. For STS, we used MSRpar as training, and we used the 4 test sets as testing. We compared systems using the Pearson's correlation as the standard evaluation measure for the challenge[3]. Thus, results can be compared with the results of the challenge.

As classifier and regression learner, we used the java version of LIBSVM (Chang and Lin 2011). In the two tasks we used in a different way our DSTs (and the related STKs) within the learners. In the following, we refer to instances in RTE or STS as pairs $p = (t^a, t^b)$ where t^a and t^b are the two parse trees for the two sentences a and b for STS and for the text a and the hypothesis b in RTE.

We will indicate with $K(p_1, p_2)$ the final kernel used in the learning algorithm, which takes as input two training instances, while we will use κ to denote either any of our DSTK (that is, $\kappa(x, y) = \langle DST(x), DST(y) \rangle$) or any of the standard smoothed tree kernels (that is, $\kappa(x, y) = STK(x, y)$).

In STS, we encoded only similarity feature between the two sentences. Thus, we used the kernel defined as:

$$K(p_1, p_2) = (\kappa(t_1^a, t_1^b) \cdot \kappa(t_2^a, t_2^b) + 1)^2$$

2 http://code.google.com/p/distributed-tree-kernels/
3 Correlations are obtained with the organizers' script

	STS				
	headl	**FNWN**	**OnWN**	**SMT**	**Average**
DTK	0.448	0.118	0.162	0.301	0.257
TK	0.456	0.145	0.158	0.303	0.265*
$DSTK_0$	0.491	0.155	0.358	0.305	**0.327**†
STK_0	0.490	0.159	0.349	0.305	0.325*
$DSTK_{+1}$	0.475	0.138	0.266	0.304	0.295
STK_{+1}	0.478	0.156	0.259	0.305	0.299*

Table 2
Task-based analysis: Correlation on Semantic Textual Similarity († is different from DTK, TK, $DSTK_{+1}$, and STK_{+1} with a stat.sig. of $p > 0.1$; $*$ the difference between the kernel and its distributed version is not stat.sig.)

In RTE, we followed standard approaches (Dagan et al. 2013; Zanzotto, Pennacchiotti, and Moschitti 2009), that is, we exploited a model with only a rewrite rule feature space (RR). The model use our DSTs and the standard STKs in the following way as kernel function:

$$RR(p_1, p_2) = \kappa(t_1^a, t_2^a) + \kappa(t_1^b, t_2^b)$$

Finally, to investigate whether our DSTKs behave better than purely structural models, we experimented with the classical tree kernel (TK) (Collins and Duffy 2002) and the distributed tree kernel (DTK) (Zanzotto and Dell'Arciprete 2012). Again, these kernels are used in the above models as $\kappa(t_a, t_b)$.

5.2 Results

Table 1 reports the results for the correlation experiments. We report the Spearman's correlations over the different sets (and different dimensions of distributed vectors) between our $DSTK_0$ and the STK_0 (first two rows) and between our $DSTK_{+1}$ and the corresponding STK_{+1} (second two rows) . The correlation is above 0.80 in average for both RTE and STS datasets in the case of $DSTK_0$ and the STK_0. The correlation between $DSTK_{+1}$ and the corresponding STK_{+1} is instead a little bit lower. This depends on the fact that $DSTK_{+1}$ is approximating the sum of two kernels the TK and the STK_0 (as STK_{+1} is the sum of the two kernels). Then, the underlying feature space is bigger with respect to the one of STK_0 and, thus, approximating it is more difficult. The approximation also depends on the size of the distributed vectors. Higher dimensions yield to better approximation: if we increase the distributed vectors dimension from 1024 to 2048 the correlation between $DSTK_{+1}$ and STK_{+1} increases up to 0.80 on RTE and up to 0.77 on STS. This direct analysis of the correlation shows that our CDSM are approximating the corresponding kernel function and there is room of improvement by increasing the size of distributed vectors.

Task-based experiments confirm the above trend. Table 2 and Table 3, respectively, report the correlation of different systems on STS and the accuracies of the different systems on RTE. Our CDSMs are compared against a baseline system (DTK) in order to understand whether in the specific tasks our more complex model is interesting, and against, again, the systems with the corresponding smoothed tree kernels in order to

	RTE				
	RTE1	**RTE2**	**RTE3**	**RTE5**	**Average**
DTK	0.533	0.515	0.516	0.530	0.523
TK	0.561	0.552	0.531	0.54	0.546
$DSTK_0$	0.571	0.551	0.547	0.531	$0.550^{\dagger}$
STK_0	0.586	0.563	0.538	0.545	0.558^{*}
$DSTK_{+1}$	0.588	0.562	0.555	0.541	$\mathbf{0.561}^{\dagger}$
STK_{+1}	0.586	0.562	0.542	0.546	0.559^{*}

Table 3
Task-based analysis: Accuracy on Recognizing Textual Entailment († is different from DTK and TK wiht a stat.sig. of $p > 0.1$; $*$ the difference between the kernel and its distributed counterpart is not statistically significant.)

explore whether our DSTKs approximate systems based on STKs. For all this set of experiment we fixed the dimension of the distributed vectors to 1024.

Table 2 is organized as follows: columns 2-6 report the correlation of the STS systems based on syntactic/semantic similarity. Comparing rows in this columns, we can discover that $DSTK_0$ and $DSTK_{+1}$ behave significantly better than DTK and that $DSTK_0$ behave better than the standard TK. Thus, our DSTKs are positively exploiting distributional semantic information along with structural information. Moreover, both $DSTK_0$ and $DSTK_{+1}$ behave similarly to the corresponding models with standard kernels STKs. Results in this task confirm that structural and semantic information are both captured by CDSMs based on DSTs.

Table 3 is organized as follows: columns 2-6 report the accuracy of the RTE systems based on rewrite rules (RR).

Results on RTE are extremely promising as all the models including structural information and distributional semantics have better results than the baseline models with a statistical significance of 93.7%. As expected (Mehdad, Moschitti, and Zanzotto 2010), STKs behave also better than tree kernels exploiting only syntactic information. But, more importantly, our CDSMs based on the DSTs are behaving similarly to these smoothed tree kernels, in contrast to what reported in (Zanzotto and Dell'Arciprete 2011). In (Polajnar, Rimell, and Kiela 2013), it appears that results of the (Zanzotto and Dell'Arciprete 2011)'s method are comparable to the results of STKs for STS, but this is mainly due to the flattening of the performance given by the lexical token similarity feature which is extremely relevant in STS. Even if distributed tree kernels do not approximate well tree kernels with distributed vectors dimension of 1024, our smoothed versions of the distributed tree kernels approximate correctly the corresponding smoothed tree kernels. Their small difference is not statistically significant (less than 70%). The fact that our DSTKs behave significantly better than baseline models in RTE and they approximate the corresponding STKs shows that it is possible to positively exploit structural information in CDSMs.

6. Conclusions and future work

Distributed Smoothed Trees (DST) are a novel class of Compositional Distributional Semantics Models (CDSM) that effectively encode structural information and distributional semantics in tractable rank-2 tensors, as experiments show. The paper shows

that DSTs contribute to close the gap between two apparently different approaches: CDSMs and convolution kernels. This contribute to start a discussion on a deeper understanding of the representation power of structural information of existing CDSMs.

References

Agirre, Eneko, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, and Weiwei Guo. 2013. *sem 2013 shared task: Semantic textual similarity. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 32–43, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Aiolli, Fabio, Giovanni Da San Martino, and Alessandro Sperduti. 2009. Route kernels for trees. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 17–24, New York, NY, USA. ACM.

Aronszajn, N. 1950. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

Baroni, Marco and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 36(4):673–721, December.

Baroni, Marco and Roberto Zamparelli. 2010. Nouns are vectors, adjectives are matrices: Representing adjective-noun constructions in semantic space. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 1183–1193, Cambridge, MA, October. Association for Computational Linguistics.

Chang, Chih-Chung and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Collins, Michael. 1999. *Head-driven Statistical Models for Natural Language Processing*. Ph.D. thesis, University of Pennsylvania.

Collins, Michael and Nigel Duffy. 2001. Convolution kernels for natural language. In *NIPS*, pages 625–632.

Collins, Michael and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*.

Cristianini, Nello and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March.

Croce, Danilo, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1034–1046, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dagan, Ido, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In Quiñonero-Candela et al., editor, *LNAI 3944: MLCW 2005*. Springer-Verlag, Milan, Italy, pages 177–190.

Dagan, Ido, Dan Roth, Mark Sammons, and Fabio Massimo Zanzotto. 2013. *Recognizing Textual Entailment: Models and Applications*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Dinu, Georgiana, Nghia The Pham, and Marco Baroni. 2013. DISSECT: DIStributional SEmantics Composition Toolkit. In *Proceedings of ACL (System Demonstrations)*, pages 31–36, Sofia, Bulgaria.

Ferrone, Lorenzo and Fabio Massimo Zanzotto. 2013. Linear compositional distributional semantics and structural kernels. In *Proceedings of the Joint Symposium of Semantic Processing (JSSP)*.

Ferrone, Lorenzo and Fabio Massimo Zanzotto. 2014. Towards syntax-aware compositional distributional semantic models. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 721–730, Dublin, Ireland, August. Dublin City University and Association for Computational Linguistics.

Haussler, David. 1999. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz.

Kimura, Daisuke, Tetsuji Kuboyama, Tetsuo Shibuya, and Hisashi Kashima. 2011. A subpath kernel for rooted unordered trees. In *Proceedings of the 15th Pacific-Asia conference on Advances in knowledge discovery and data mining - Volume Part I*, PAKDD'11, pages 62–74, Berlin, Heidelberg. Springer-Verlag.

Mehdad, Yashar, Alessandro Moschitti, and Fabio Massimo Zanzotto. 2010. Syntactic/semantic structures for textual entailment recognition. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, HLT '10, pages 1020–1028, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell, Jeff and Mirella Lapata. 2008. Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pages 236–244, Columbus, Ohio, June. Association for Computational Linguistics.

Polajnar, Tamara, Laura Rimell, and Douwe Kiela. 2013. Ucam-core: Incorporating structured distributional similarity into sts. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 1: Proceedings of the Main Conference and the Shared Task: Semantic Textual Similarity*, pages 85–89, Atlanta, Georgia, USA, June. Association for Computational Linguistics.

Socher, Richard, Eric H. Huang, Jeffrey Pennington, Andrew Y. Ng, and Christopher D. Manning. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*.

Vishwanathan, S. V. N. and Alexander J. Smola. 2002. Fast kernels for string and tree matching. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *NIPS*, pages 569–576. MIT Press.

Zanzotto, Fabio Massimo and Lorenzo Dell'Arciprete. 2011. Distributed structures and distributional meaning. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pages 10–15, Portland, Oregon, USA, June. Association for Computational Linguistics.

Zanzotto, Fabio Massimo and Lorenzo Dell'Arciprete. 2012. Distributed tree kernels. In *Proceedings of International Conference on Machine Learning*, pages –, June 26–July 1,.

Zanzotto, Fabio Massimo, Ioannis Korkontzelos, Francesca Fallucchi, and Suresh Manandhar. 2010. Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*, August,.

Zanzotto, Fabio Massimo, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *NATURAL LANGUAGE ENGINEERING*, 15-04:551–582.

Zanzotto, F.M. and L. Dell'Arciprete. 2012. Distributed tree kernels. In *Proceedings of International Conference on Machine Learning*, pages 193–200.