Towards Syntax-aware Compositional Distributional Semantic Models

Lorenzo Ferrone
Department of Enterprise Engineering
University of Rome "Tor Vergata"
Via del Politecnico, 1 00173 Roma
`lorenzo.ferrone@gmail.com`

Fabio Massimo Zanzotto
Department of Enterprise Engineering
University of Rome "Tor Vergata"
Via del Politecnico, 1 00173 Roma
`fabio.massimo.zanzotto@uniroma2.it`

Abstract

Compositional Distributional Semantics Models (CDSMs) are traditionally seen as an entire different world with respect to Tree Kernels (TKs). In this paper, we show that under a suitable regime these two approaches can be regarded as the same and, thus, structural information and distributional semantics can successfully cooperate in CSDMs for NLP tasks. Leveraging on distributed trees, we present a novel class of CDSMs that encode both structure and distributional meaning: the distributed smoothed trees (DSTs). By using DSTs to compute the similarity among sentences, we implicitly define the distributed smoothed tree kernels (DSTKs). Experiment with our DSTs show that DSTKs approximate the corresponding smoothed tree kernels (STKs). Thus, DSTs encode both structural and distributional semantics of text fragments as STKs do. Experiments on RTE and STS show that distributional semantics encoded in DSTKs increase performance over structure-only kernels.

1 Introduction

Compositional distributional semantics is a flourishing research area that leverages distributional semantics (see Turney and Pantel (2010), 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; Clark et al., 2008; Grefenstette and Sadrzadeh, 2011; Zanzotto et al., 2010). Functions for full sentences are generally defined as recursive functions over the ones for phrases (Socher et al., 2011; Socher et al., 2012; Kalchbrenner and Blunsom, 2013). Distributional vectors for text fragments are then used as inner layers in neural networks, or to compute similarity among text fragments via dot product.

CDSMs generally exploit structured representations t^x of text fragments x to derive their meaning $f(t^x)$, but the structural information, although extremely important, 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) (Mehdad et al., 2010; Croce et al., 2011) instead realize a clearer interaction between structural information and distributional meaning. 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. 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, $STK(t^x, t^y)$. The function f that maps trees into vectors is only implicitly 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), is a CDSM since it maps trees to vectors by using 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 (Aronszajn, 1950). Thus, these distributed trees are small vectors that encode structural information. In DTKs tree nodes u and v (and then also words) are represented by nearly orthonormal vectors, that is, vectors $\vec{u}$ and $\vec{v}$ such that $\langle \vec{u}, \vec{v}\rangle \approx \delta(\vec{u}, \vec{v})$ where δ is the Kroneker's delta. This is in contrast with distributional semantics vectors where $\langle \vec{u}, \vec{v}\rangle$ is allowed to be any value in $[0, 1]$ according to the similarity between the words v and u. Thus, early attempts to include distributional vectors in the DTs failed (Zanzotto and Dell'Arciprete, 2011).

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 carry structure and distributional meaning on a 2-dimensional 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 2-dimensional 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 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 4 reports on the experiments. Finally, Section 5 draws some conclusions.

2 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, while w_n is the semantic headword of the tree headed by n, along with its part-of-speech tag. For example, the root node of the tree in Fig.1 has the label S:*booked::v* where S is the syntactic information and *booked::v* is the semantic head of the whole tree. 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 (see Fig. 1). The structure of a tree is represented as follows: Given a tree t, $\mathsf{h}(t)$ is its root node and $\mathsf{s}(t)$ is the tree formed from t but considering only the syntactic structure (that is, only the s_n part of the labels), $c_i(n)$ denotes 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, $\vec{s_n} \in \mathbb{R}^m$ and $\vec{w_n} \in \mathbb{R}^k$ represent respectively *distributed* vectors for node labels s_n and *distributional* vectors for words w_n, whereas $\mathbf{T}$ represents the matrix of a tree t encoding structure and distributional meaning. The difference between distributed and distributional vectors is described in the next section.

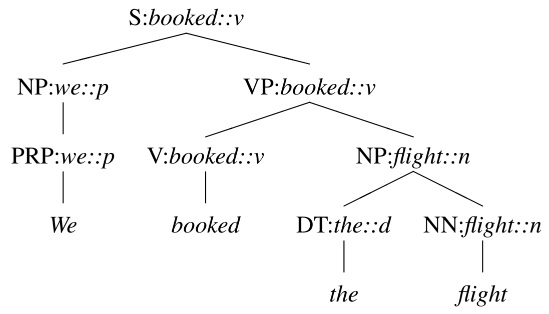

Figure 1: A lexicalized trees

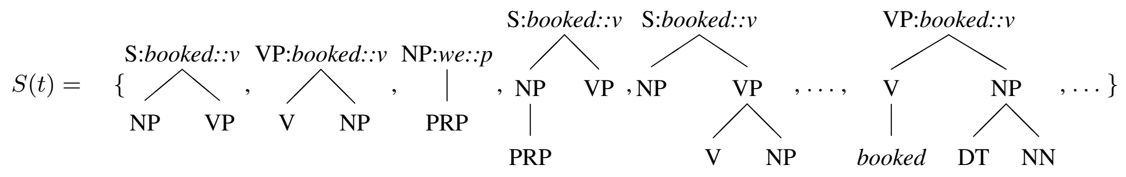

Figure 2: Subtrees of the tree t in Figure 1 (a non-exhaustive list)

3 Distributed Smoothed Trees as Compositional Distributional Semantic Models

We define Distributed Smoothed Trees as recursive functions DST mapping lexicalized trees t to $\mathbb{R}^{m\times k}$ where matrices $\mathbf{T} = DST(t)$ encode both syntactic structures and distributional vectors. DSTs are thus compositional distributional models, as they map lexicalized trees to matrices, and they are defined recursively on distributed vectors for syntactic node labels and distributional vectors for words. In the following we introduce DSTs: Section 3.1 gives a rough idea of the method, Section 3.2 describes how to recursively encode structures in vectors by means of distributed trees (Zanzotto and Dell'Arciprete, 2012), and finally Section 3.3 merges distributed trees and distributional semantic vectors in matrices.

3.1 The method in 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. Figure 2 reports a sample for the subtrees of the tree in Fig. 1 The recursive functions DSTs we define compute the following:

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

where $\mathbf{T}_i$ is the matrix associated to each subtree t_i. The similarity between two text fragments a and b represented as lexicalized trees t^a and t^b can be computed using the Frobenius product between the two matrices $\mathbf{T}^a$ and $\mathbf{T}^b$, that is:

$$\langle \mathbf{T}^a, \mathbf{T}^b \rangle_F = \sum_{\substack{t_i^a \in S(t^a) \\ t_j^b \in S(t^b)}} \langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \tag{1}$$

We want to obtain that the product $\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F$ approximates the dot product between the distributional vectors of the head words ($\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \approx \langle \overrightarrow{\mathsf{h}(t_i^a)}, \overrightarrow{\mathsf{h}(t_j^b)} \rangle$) whenever the syntactic structure of the subtrees is the same (that is $\mathsf{s}(t_i^a) = \mathsf{s}(t_j^b)$), and $\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \approx 0$ otherwise. This property is expressed as:

$$\langle \mathbf{T}_i^a, \mathbf{T}_j^b \rangle_F \approx \delta(\mathsf{s}(t_i^a), \mathsf{s}(t_j^b)) \cdot \langle \overrightarrow{\mathsf{h}(t_i^a)}, \overrightarrow{\mathsf{h}(t_j^b)} \rangle \tag{2}$$

3.2 Representing Syntactic Structures with Distributed Trees

Distributed trees (Zanzotto and Dell'Arciprete, 2012) recursively encode syntactic trees t in small vectors by means of a recursive function DT. These DTs preserve structural information as the dot product between the DTs of two trees approximates the classical tree kernels TK as defined by Collins and Duffy (2002), that is, $TK(t^a, t^b) \approx \langle DT(t^a), DT(t^b)\rangle$. To obtain this result, distributed trees $DT(t)$ are defined as follows:

$$DT(t) = \sum_{t_i \in S(t)} \sqrt{\lambda^{|N(t_i)|}}\ \vec{\mathsf{s}(t_i)} \tag{3}$$

where $S(t)$ is again the set of the subtrees of t, $\vec{\mathsf{s}(t_i)}$ are vectors in $\mathbb{R}^m$ corresponding to tree fragment t_i and $\sqrt{\lambda^{|N(t_i)|}}$ is the weight of subtree t_i in the final feature space, with λ being the traditional parameter used to penalize large subtrees and $|N(t_i)|$ being the number of nodes in t_i. The approximation of tree kernels is then given by the fact that $\langle \vec{\mathsf{s}(t_i)}, \vec{\mathsf{s}(t_j)}\rangle \approx \delta(\mathsf{s}(t_i), \mathsf{s}(t_j))$. Vectors with this property are called *distributed vectors*. A key feature of the distributed vectors of subtrees $\vec{\mathsf{s}(t_i)}$ is that these vectors are built compositionally from a set $\mathcal{N}$ of *nearly orthonormal random vectors* $\vec{s_n}$, that are associated to each node label. Given a subtree $\mathsf{s}(t_i)$, the related vector is obtained as:

$$\vec{\mathsf{s}(t_i)} = \vec{s_{n_1}} \odot \vec{s_{n_2}} \odot \ldots \odot \vec{s_{n_k}} = \bigodot_{(s_n, w_n) \in N(t_i)} \vec{s_n}$$

where node vectors $\vec{s_{n_i}}$ are ordered according to a depth-first visit of subtree t_i and $\odot$ is a vector composition operation, specifically the *shuffled circular convolution*[1]. This function guarantees that two different subtrees have nearly orthonormal vectors (see (Zanzotto and Dell'Arciprete, 2012) for more details). For example, the fifth tree t_5 of set $S(t)$ in Figure 2 is $\vec{\mathsf{s}(t_5)} = \vec{S} \odot (\vec{NP} \odot (\vec{VP} \odot (\vec{V} \odot \vec{NP})))$. Thus, DTs in Equation 3 can be recursively defined as:

$$DT(t) = \sum_{n \in N(t)} \sigma(n) \tag{4}$$

where $\sigma(n)$ is recursively defined as follows:

$$\sigma(n) = \begin{cases} \sqrt{\lambda}\ (\vec{s_n} \odot \vec{w}) & \text{if } n \text{ is a pre-terminal node} \\ \sqrt{\lambda}\ \vec{s_n} \odot (\bigodot_i (\vec{s_{c_i(n)}} + \sigma(c_i(n)))) & \text{if } n \text{ is an internal node} \end{cases} \tag{5}$$

The vector $\sigma(n)$ encodes all the subtrees that have root in n along with their penalizing weight $\sqrt{\lambda^{|N(t_i)|}}$, that is:

$$\sigma(n) = \sum_{t_i \in S(t) \wedge \mathsf{h}(t_i) = n} \sqrt{\lambda^{|N(t_i)|}}\ \vec{\mathsf{s}(t_i)}$$

This is what we need in order to define our *distributed smoothed trees*.

3.3 Representing distributional meaning and distributed structure with matrices

We now move from distributed trees (encoded as small vectors) to distributed smoothed trees (DST) represented as matrices. DST is a function that maps trees t to matrices $\mathbf{T}$. In analogy with Equation 4, DST is defined as:

$$DST(t) = \sum_{n \in N(t)} S(n)$$

where $S(n)$ is now defined as:

$$S(n) = \sigma(n) \vec{w_n}^{\top}$$

[1]The *shuffled circular convolution* $\odot$ is defined as $\vec{a} \odot \vec{b} = s_1(\vec{a}) * s_2(\vec{b})$ where $*$ is the circular convolution and s_1 and s_2 are two different (but fixed) random permutations of vector elements.

where $\sigma(n)$ is the one defined in Equation 5 and $(\cdot)^\top$ is vector transposition. By combining the two equations, $DST(t)$ is the sum of the matrices described in Equation 1:

$$DST(t) = \sum_{n \in N(t)} \sum_{t_i \in S(t) \wedge \mathsf{h}(t_i)=n} \sqrt{\lambda^{|N(t_i)|}}\ \overrightarrow{\mathsf{s}(t_i)} \vec{w_n}^\top = \sum_{t_i \in S(t)} \overrightarrow{\mathsf{s}(t_i)} \vec{w_n}^\top$$

where n is $\mathsf{h}(t_i)$ and $\mathbf{T}_i = \overrightarrow{\mathsf{s}(t_i)} \overrightarrow{w_{\mathsf{h}(t_i)}}^\top$ is the outer product between the distributed vector $\overrightarrow{\mathsf{s}(t_i)}$ and the distributional vector $\overrightarrow{w_{\mathsf{h}(t_i)}}$. There is an important property of the outer product that applies to the Frobenius product: $\langle \vec{a}\vec{w}^\top, \vec{b}\vec{v}^\top \rangle_F = \langle \vec{a}, \vec{b} \rangle \cdot \langle \vec{w}, \vec{v} \rangle$. Using this property, we have that Equation 2 is satisfied as:

$$\langle \mathbf{T}_i, \mathbf{T}_j \rangle_F = \langle \overrightarrow{\mathsf{s}(t_i)}, \overrightarrow{\mathsf{s}(t_j)} \rangle \cdot \langle \overrightarrow{w_{h(t_i)}}, \overrightarrow{w_{h(t_j)}} \rangle \approx \delta(\mathsf{s}(t_i), \mathsf{s}(t_j)) \cdot \langle \overrightarrow{w_{h(t_i)}}, \overrightarrow{w_{h(t_j)}} \rangle$$

We refer to the Frobenius product of two distributed smoothed trees as *distributed smoothed tree kernel* (DSTK). These DSTKs are approximating the smoothed tree kernels described in the next section. We propose two versions of our DSTKs according to how we produce distributional vectors for words. We have a plain version $DSTK_0$ when we use distributional vectors $\vec{w_n}$ as they are, and a slightly modified version $DSTK_{+1}$ when we use as distributional vectors $\vec{w_n}' = \begin{pmatrix} 1 & \vec{w_n} \end{pmatrix}$.

3.4 The Approximated Smoothed Tree Kernels

The two CDSMs we proposed, that is, the two distributed smoothed tree kernels $DSTK_0$ and $DSTK_{+1}$, are approximating two specific tree kernels belonging to the smoothed tree kernels class (e.g., (Mehdad et al., 2010; Croce et al., 2011)). These two specific smoothed tree kernels recursively compute (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 STK_0 and STK_{+1} where the weights are defined as follows:

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

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

STK_{+1} is actually computing a sum between STK_0 and the tree kernel (Collins and Duffy, 2002).

4 Experimental investigation

4.1 Experimental set-up

Generic settings We experimented with two datasets: the Recognizing Textual Entailment datasets (RTE) (Dagan et al., 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 et al., 2013) from a corpus obtained by the concatenation of ukWaC (wacky.sslmit.unibo.it), a mid-2009 dump of

		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

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. 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 two classes of kernels: (1) the syntactic/semantic class (SS) with the final kernel defined as $K(p_1, p_2) = (\kappa(t_1^a, t_1^b) \cdot \kappa(t_2^a, t_2^b) + 1)^2$; and, (2) the SS class along with token-based similarity (SSTS) where the final kernel is $K(p_1, p_2) = (\kappa(t_1^a, t_1^b) \cdot \kappa(t_2^a, t_2^b) + TS(a_1, b_1) \cdot TS(a_2, b_2) + 1)^2$ where $TS(a, b)$ counts the percent of the common content tokens in a and b.

In RTE, we followed standard approaches (Dagan et al., 2013; Zanzotto et al., 2009), that is, we exploited two models: a model with only a rewrite rule feature space (RR) and a model with the previous space along with a token-level similarity feature (RRTWS). The two models use our DSTs and the standard STKs in the following way as kernel functions: (1) $RR(p_1, p_2) = \kappa(t_1^a, t_2^a) + \kappa(t_1^b, t_2^b)$; (2) $RRTS(p_1, p_2) = \kappa(t_1^a, t_2^a) + \kappa(t_1^b, t_2^b) + (TWS(a_1, b_1) \cdot TS(a_2, b_2) + 1)^2$ where TWS is a weighted token similarity as in Corley and Mihalcea (2005).

[2]http://code.google.com/p/distributed-tree-kernels/

[3]Correlations are obtained with the organizers' script

	SS					SSTS				
	headl	**FNWN**	**OnWN**	**SMT**	Average	**headl**	**FNWN**	**OnWN**	**SMT**	Average
TS	—	—	—	—	—	0.701	0.311	0.515	0.323	0.462
Add	—	—	—	—	—	0.691	0.268	0.511	0.317	0.446
Mult	—	—	—	—	—	0.291	−0.03	0.228	0.291	0.201
DTK	0.448	0.118	0.162	0.301	0.257	0.698	0.311	0.510	0.329	0.462
TK	0.456	0.145	0.158	0.303	0.265^*	0.699	0.316	0.511	0.329	0.463^*
DSTK$_0$	0.491	0.155	0.358	0.305	$\mathbf{0.327}^\dagger$	0.700	0.314	0.519	0.327	0.465
STK$_0$	0.490	0.159	0.349	0.305	0.325^*	0.700	0.314	0.519	0.327	0.465^*
DSTK$_{+1}$	0.475	0.138	0.266	0.304	0.295	0.700	0.314	0.519	0.327	$\mathbf{0.465}$
STK$_{+1}$	0.478	0.156	0.259	0.305	0.299^*	0.700	0.314	0.519	0.327	0.465^*

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.)

We also used two standard and simple CDSMs to compare with: the Additive model (Add) and the Multiplicative model (Mult) as firstly discussed in Mitchell and Lapata (2008). The Additive Model performs a sum of all the distributional vectors of the content words in the text fragment and the Multiplicative model performs an element-wise product among all the content vectors. These are used in the above models as $\kappa(a, b)$.

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 $\kappa(t_a, t_b)$.

4.2 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 baseline systems (Add, $Mult$, TK, and 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 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 (SS) and columns 7-11 report the accuracies of SS systems along with token-based similarity (SSTS). The first observation for this task is that baseline systems based only on the token similarity (first row) behave extremely well. These results are above many models presented in the 2013 Shared Task (see (Agirre et al., 2013)). This can be disappointing as we cannot appreciate differences among methods in the columns SSTS. But, focusing on the results without this important token-based similarity, we can better understand if our model is capturing both structural and semantic information, that is, if DSTKs behave similarly to STKs. It is also useless to compare results of DSTKs and STKs to the *Add* baseline model as *Add* is basically doing a weighted count of the common words

	RR					RRTWS				
	RTE1	**RTE2**	**RTE3**	**RTE5**	**Average**	**RTE1**	**RTE2**	**RTE3**	**RTE5**	**Average**
Add	0.541	0.496	0.507	0.520	0.516	0.560	0.538	0.643	0.578	0.579
Mult	0.495	0.481	0.497	0.528	0.500	0.533	0.563	0.642	0.586	0.581
DTK	0.533	0.515	0.516	0.530	0.523	0.583	0.601	0.643	0.621	0.612
TK	0.561	0.552	0.531	0.54	0.546	0.608	0.627	0.648	0.630	0.628
DSTK$_0$	0.571	0.551	0.547	0.531	$0.550^\dagger$	0.628	0.616	0.650	0.625	$0.629^\dagger$
STK$_0$	0.586	0.563	0.538	0.545	0.558^*	0.638	0.618	0.648	0.636	0.635^*
DSTK$_{+1}$	0.588	0.562	0.555	0.541	$\mathbf{0.561}^\dagger$	0.638	0.621	0.646	0.652	$\mathbf{0.639}^\dagger$
STK$_{+1}$	0.586	0.562	0.542	0.546	0.559^*	0.638	0.618	0.650	0.636	0.635^*

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.)

that is exactly what the token-based similarity is doing. Add slightly decreases the performance of the token-based similarity. The $Mult$ model instead behaves very poorly. Comparing rows in the SS 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 exploitng 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) and columns 7-11 report the accuracies of RR systems along with token similarity (RRTS). Results on RTE are extremely promising as all the models including structural information and distributional semantics have better results than the two baseline models with a statistical significance of 93.7%. For RR models $DTSK_0$, STK_0, $DSTK_{+1}$, and STK_{+1} have an average accuracy 7.9% higher than Add and 11.4% higher than $Mult$ model. For RRTS, the same happens with an average accuracy 9.58% higher than Add and 9.2% higher than the $Mult$. This task is more sensible to syntactic information than STS. As expected (Mehdad et al., 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 et al., 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.

5 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 2-dimensional tensors, as experiments show. The paper shows that DSTs contribute to close the gap between two apparently different approaches: CDSMs and convolution kernels (Haussler, 1999). This contribute to start a discussion on a deeper understanding of the representation power of structural information of existing CDSMs.

References

[Agirre et al.2013] Eneko Agirre, 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.

[Aronszajn1950] N. Aronszajn. 1950. Theory of reproducing kernels. *Transactions of the American Mathematical Society*, 68(3):337–404.

[Baroni and Lenci2010] Marco Baroni and Alessandro Lenci. 2010. Distributional memory: A general framework for corpus-based semantics. *Comput. Linguist.*, 36(4):673–721, December.

[Baroni and Zamparelli2010] Marco Baroni 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 and Lin2011] Chih-Chung Chang 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`.

[Clark et al.2008] Stephen Clark, Bob Coecke, and Mehrnoosh Sadrzadeh. 2008. A compositional distributional model of meaning. *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, pages 133–140.

[Collins and Duffy2002] Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*.

[Corley and Mihalcea2005] Courtney Corley and Rada Mihalcea. 2005. Measuring the semantic similarity of texts. In *Proc. of the ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment*, pages 13–18. Association for Computational Linguistics, Ann Arbor, Michigan, June.

[Cristianini and Shawe-Taylor2000] Nello Cristianini and John Shawe-Taylor. 2000. *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press, March.

[Croce et al.2011] Danilo Croce, 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 et al.2006] Ido Dagan, Oren Glickman, and Bernardo Magnini. 2006. The pascal recognising textual entailment challenge. In Quionero-Candela et al., editor, *LNAI 3944: MLCW 2005*, pages 177–190. Springer-Verlag, Milan, Italy.

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

[Ferrone and Zanzotto2013] Lorenzo Ferrone and Fabio Massimo Zanzotto. 2013. Linear compositional distributional semantics and structural kernels. In *Proceedings of the Joint Symposium of Semantic Processing (JSSP)*, pages –.

[Grefenstette and Sadrzadeh2011] Edward Grefenstette and Mehrnoosh Sadrzadeh. 2011. Experimental support for a categorical compositional distributional model of meaning. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1394–1404, Stroudsburg, PA, USA. Association for Computational Linguistics.

[Haussler1999] David Haussler. 1999. Convolution kernels on discrete structures. Technical report, University of California at Santa Cruz.

[Kalchbrenner and Blunsom2013] Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent convolutional neural networks for discourse compositionality. *Proceedings of the 2013 Workshop on Continuous Vector Space Models and their Compositionality*.

[Mehdad et al.2010] Yashar Mehdad, 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 and Lapata2008] Jeff Mitchell 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 et al.2013] Tamara Polajnar, 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 et al.2011] Richard Socher, 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*.

[Socher et al.2012] Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

[Turney and Pantel2010] Peter D. Turney and Patrick Pantel. 2010. From frequency to meaning: Vector space models of semantics. *J. Artif. Intell. Res. (JAIR)*, 37:141–188.

[Zanzotto and Dell'Arciprete2011] Fabio Massimo Zanzotto 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 and Dell'Arciprete2012] F.M. Zanzotto and L. Dell'Arciprete. 2012. Distributed tree kernels. In *Proceedings of International Conference on Machine Learning*, pages 193–200.

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

[Zanzotto et al.2010] Fabio Massimo Zanzotto, 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,.