TITOLO 1: Distributed Convolution Kernels on Countable Sets
TITOLO 2: Learning with Structures without Kernels and with Simpler Machines
TITOLO 3: Learning with Structures, but without Kernels
TITOLO 4: Learning with Structures, but without Kernels and without Recurrent Neural Networks

Fabio Massimo Zanzotto fabio.massimo.zanzotto@uniroma2.it

Lorenzo Dell'Arciprete lorenzo.dellarciprete@gmail.com

Department of Enterprise Engineering
University of Rome "Tor Vergata"
Rome, Italy

Abstract

Contents

1. Introduction

[LD] Structured data may be used to interpret nearly everything – graphs are really powerful representations for natural and artificial objects. In fact, natural objects – bodies (?), proteins (?), natural language utterances (?) and, even, social networks (?) – are often represented with labeled nodes connected by labeled edges. Artificial objects are even linked more directly with structured data. These objects have often been dots and segments in some design before being tangible. Modeling them as structured data means only to rediscover what is already there. Consequently, machines that want to learn from observing objects are forced to deal with their structure.

[LD] *How to fit structured data in learning machines* has thus become a compelling challenge, which not surprisingly has attracted a lot of attention. Structures cannot fit naturally in *flat* vector spaces of features, which generally underly learning machines. Hence, many strategies have emerged to overcome this limitation.

[LD] The proposed strategies can be roughly grouped in three families: (1) *representing small sets of substructures*, (2) *defining graph kernels* and (3) *using recursive learning machines*. The first family defines small sets of relevant substructures – specific edges in graphs (Even-Zohar & Roth, 2000) or specific paths in trees (Gildea & Jurafsky, 2002) – that are the features of the vector spaces where structured objects are represented. Features are active if structured objects have the related substructures. The second family relies on the definition of *graph kernels* () that are similarity measures between structured objects. Convolution kernels (Haussler, 1999) are an important class of these graph kernels as they implicitly encode huge feature spaces of all possible substructures. Hence, convolution kernels have attracted a lot of research and specific convolution kernels have been defined for different structures – for example strings, sequences (Lodhi, Saunders, Shawe-Taylor, Cristianini, & Watkins, 2002) and trees (for example, (Collins & Duffy, 2001; Aiolli, Da San Martino, & Sperduti, 2009; Kimura, Kuboyama, Shibuya, & Kashima, 2011)). Differently, the third family uses a recursive application of learning machines on structured objects (). This strategy has been recently revitalized in the context of neural networks ().

[LD] The three families of strategies are not equivalent: computational complexity and awareness of structural information seem to be incompatible goals. Computationally-efficient linear learning machines (for example, (Joachims, 2006; Shalev-Shwartz, Singer, Srebro, & Cotter, 2011; ?, ?)) require explicit vector spaces. Hence, only the first family of strategies can be used. Unfortunately, as it represents structures with small sets of substructures, large part of structural information is lost. Thus, to fully exploit the information encoded in structures, one of the other two families is needed. But, consequently, computationally more complex learning machines are required: recursive learning machines and kernel machines, required by graph kernels, are more complex than linear machines. In fact, recursive learning machines require a recursive application to each structured object and kernel machines (see (Cristianini & Shawe-Taylor, 2000)) apply kernel functions thousands of times during learning and classification. Thus, the complexity of graph kernel functions is amplified. Although many approaches (Moschitti, 2006b; Rieck, Krueger, Brefeld, & Müller, 2010; Pighin & Moschitti, 2010; Shin, Cuturi, & Kuboyama, 2011) have drastically lowered the average execution time of kernel machines with graph kernels, their computational complexity in the learning phase has never been reduced.

[OCAR] In this paper, we challenged the dogma that low computational complexity and fully awareness of structural information are incompatible goals. We propose to encode structured data in *distributed convolution structures*, which are small vectors whose dot products approximate convolution kernels. Consequently, learning can be carried out with computationally-efficient linear machines. Our approximation is valid and effective for convolution kernels (Haussler, 1999) restricted on discrete sets[1]. In fact, these restricted convolution kernels $K(x, y)$ between structures x and y were rewritten as weighted sums of Kroneker's delta functions $\delta(a, b)$ over substructures a and b of x and y, respectively. By approximating this delta function $\delta(a, b)$ with dot products of small vectors $\Upsilon_\Sigma(a)$ and $\Upsilon_\Sigma(b)$ called *distributed substructures*, we showed that these convolution kernels $K(x, y)$ are approximated with dot products between *distributed structures* $D_\Sigma(x)$ and $D_\Sigma(y)$. Distributed structures $D_\Sigma(z)$ are weighted sums of distributed substructures of structures z. Our main result is that both distributed substructures $\Upsilon_\Sigma(x)$ and distributed structures $D_\Sigma(x)$ are computed efficiently with two recursive formulations: the *distributed convolution substructures* $\Upsilon(x)$ and the *distributed convolution structures* $D(x)$. Hence, convolution kernels on discrete sets can be efficiently approximated with dot products between *distributed convolution structures*. We then realized 5 different formulations of distributed convolution structures to approximate tree kernels (Collins & Duffy, 2002), subpath tree kernels (Kimura et al., 2011), route kernels (Aiolli et al., 2009), sequence kernels (Lodhi et al., 2002) and partial tree kernels (Kashima & Koyanagi, 2002; Moschitti, 2006a). Experiments show that distributed convolution structures approximate original kernels on simulated and real tasks with statistical significance. Thus, low computational complexity and fully awareness of structural information can coexist. Distributed convolution structures approximated convolution kernels and drastically reduce the complexity of learning and classification when applied to real kernels.

The rest of the paper is organized as follows. Section 2 introduces and proves the method in two steps. First, a preliminary model that show how to approximate convolution kernels with distributed substructures $\Upsilon_\Sigma(a)$ and distributed structures $D_\Sigma(z)$. Second, a recursive formulation – the distributed convolution substructures $\Upsilon(a)$ and the distributed convolution structures $D(z)$ – that makes the method computationally efficient. Section 3 describes the five realizations of the distributed convolution kernels: the distributed tree kernel (DTK), the distributed subpath tree kernel (DSTK), the distributed route tree kernels (DRTK), the distributed string or sequence kernel (DSK), and the distributed partial tree kernel (DPTK). Then, Section 4 formally investigates the complexity of the derived distributed convolution kernels along with linearized versions of kernel machines (Joachims, 2006; Shalev-Shwartz et al., 2011). Section 5 empirically determines how well distributed convolution kernels approximate the corresponding convolution kernels. Finally, Section 6 draws some conclusions and it sketches the future work.

2. The method: Distributed Convolution Kernels on Structured Objects

[OCAR] When building learning machines over structures, low computational complexity and fully awareness of structural information seem to be incompatible goals. A spark to accomplish both goals is hidden in one of the current strategies for learning with structures:

1. Sets endowed with the Kronecker's delta function

convolution kernels (Haussler, 1999). Stemming from this spark, we propose to approximate convolution kernels with dot products of *distributed convolution structures*, which are low dimensional vectors. By representing structures with low dimensional vectors, we enable learning over structured data with linear learning machines. Since linear learning machines have better complexity with respect to kernel machines, this section prove that these two goals can coexist.

The section introduces our method in two steps. First, Section 2.1 introduces the model and describes the preliminary result on approximating convolution kernels using *distributed substructures* $\Upsilon_\Sigma(a)$ and *distributed structures* $D_\Sigma(x)$. Second, Section 2.2 presents the recursive and efficient formulation – the *distributed convolution substructures* $\Upsilon(a)$ and the *distributed convolution structures* $D(x)$ – and formally proves that each convolution kernel has corresponding distributed convolution structure that approximate it.

2.1 Preliminaries and Direct Formulation

This section introduces the basic elements and the first formulation of our method: a recursive definition for structured objects, the result that convolution kernels on discrete set can be expressed as summations of Kronecker's delta on the substructures, a basic set of nearly-orthogonal unit vectors for defining *distributed substructures* $\Upsilon_\Sigma(a)$ to approximate Kronecker's delta between substructures and, finally, the proof that convolution kernels can be approximated with dot products between *distributed structures* $D_\Sigma(x)$. A full map of relevant symbols is instead given in Appendix A.

2.1.1 Structured objects

[LD] To propose an innovative way to treat structured objects in learning, we need a formal definition of what structured objects are.

[LD] Convolution kernels use a simple and effective definition for structured objects (Haussler, 1999) but this definition is not specific enough for actual convolution kernels. In fact, to prove general properties of convolution kernels, structured objects are decomposed only at the first level. Objects x are thus seen through their direct subparts. Unfortunately, many, if not all, actual convolution kernels rely on slightly more complex recursive definitions that are different case by case (Collins & Duffy, 2002; Kimura et al., 2011; Aiolli et al., 2009; Lodhi et al., 2002; Kashima & Koyanagi, 2002; Moschitti, 2006a; Croce, Moschitti, & Basili, 2011; Mehdad, Moschitti, & Zanzotto, 2010).

[LD] We thus proposed a *recursive definition of structured objects* that generalizes definitions given in actual convolution kernels. In our view, **structured objects** $x \in X$ are either terminal objects $x \in G_X$ that cannot be furthermore decomposed or non-terminal objects that can be decomposed into M parts $\overline{\mathbf{x}} = (x_1, ..., x_M) \in X_1 \times \ldots \times X_M$. Parts $x_i \in X_i$ are again structured objects. Thus, X is a set of objects, X_i are sets containing the i-th parts of objects $x \in X$, and G_X is a *ground set* that contains terminal objects including the empty object $\emptyset$. Finally, given a structured object x, we define $R(x)$ as the set of all possible decompositions into parts of x.

[LD] The recursive definition of structured objects adapts to specific convolution kernels by specializing the decomposition strategy and the sets X, X_i and $R(x)$. The following

examples of recursive structured objects aim to clarify the recursive definition, the related notation, and how it adapts to specific kernels:

- In sequence kernels (Lodhi et al., 2002), a sequence is $s = \mathtt{w}_1 \dots \mathtt{w}_k \in C^*$ and its decomposition into parts is $\bar{\mathbf{s}} = (\mathtt{w}_1, \mathtt{w}_2 \dots \mathtt{w}_k) \in C \times C^*$. For example, given the sequence $\mathtt{W}_1\mathtt{W}_2\mathtt{W}_3\mathtt{W}_4$, a decomposition into parts is $(\mathtt{W}_1, \mathtt{W}_2\mathtt{W}_3\mathtt{W}_4)$. The set $R(s)$ for the same sequence is:

$$R(\mathtt{W}_1\mathtt{W}_2\mathtt{W}_3\mathtt{W}_4) = \{(\mathtt{W}_1, \mathtt{W}_2\mathtt{W}_3\mathtt{W}_4), (\mathtt{W}_2, \mathtt{W}_3\mathtt{W}_4), (\mathtt{W}_3, \mathtt{W}_4)\}$$

- In tree kernels (Collins & Duffy, 2002), given a tree $t \in T$, a decomposition into parts is $\bar{\mathbf{t}} = (r, c_1, \dots, c_k) \in N \times T^k$ where N is the set of nodes, r is the root of the tree and $c_i \in T$ are the subtrees of t rooted in the i-th child of the root r. For example, the tree in Fig. **??** can be decomposed as:

```
(A,(B W1),(C (D W2)(E W3)))
```

where `A` is the root label, `(B W1)` and `(C (D W2)(E W3))` are the two children subtrees of `A`. A possible set $R(t)$ is the one used for the parse tree kernel (see Section 3.1 for details), that includes the decomposition of the subtree rooted in each non-terminal node of tree t:

$$R(t) = \{\texttt{(A,(B W1),(C (D W2)(E W3))),(B,W1),} \\ \texttt{(C,(D W2),(E W3)),(D,W2),(E,W3)}\}$$

2.1.2 Convolution Kernels as Dot Products on Spaces of Substructures

[OCAR] Clearly, the recursive definition of structured objects does not change a basic result of convolution kernels $K(x, y)$ (Haussler, 1999): $K(x, y)$ restricted to discrete sets are dot products of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ where base vectors $\phi(a) \in \mathbb{R}^m$ represent substructures a of x and y. Discrete sets are sets endowed with the Kronecker's delta function $\delta(a, b)$ that is one if $a = b$ and zero otherwise. Hence, $\langle \phi(a), \phi(b) \rangle = \delta(a, b)$. Thus , the basic result is more formally:

$$K(x, y) = \langle \mathbf{x}, \mathbf{y} \rangle = \langle \sum_{a \in S(x)} \omega_a \phi(a), \sum_{b \in S(y)} \omega_b \phi(b) \rangle \tag{1}$$

where $S(x)$ and $S(y)$ are discrete sets of all substructures of x and y that are relevant for the given kernel, ω_a and ω_b are the values (or weights) of structures a and b.

[LD] This section provides an alternative demonstration of the above result in our settings since this is an important step to demonstrate that convolution kernels can be efficiently approximated with dot products of low dimensional vectors. The purpose is to fix notation and to provide basic tools for our challenge.

[LD] The first step of the demonstration is to reformulate convolution kernels to make explicit their behavior on structured objects that are recursively defined and are restricted to discrete sets. Thus, the basic sets of structured objects – X, X_i and G_X – are discrete sets.

Definition 1 *A convolution kernel on discrete sets is defined as follows:*

$$K(x,y) = \begin{cases} \omega_x \omega_y \delta(x,y) & \text{if } x \text{ or } y \text{ belongs to } G_X \\ \sum_{\overline{\mathbf{x}} \in R(x), \overline{\mathbf{y}} \in R(y)} \prod_{i=1}^{M_{max}} K_i(x_i, y_i) & \text{otherwise} \end{cases}$$

where $K_i(x_i, y_i)$ are convolution kernels, possibly recursively defined, and $M_{max} = \max\{M_x, M_y\}$ being M_x and M_y the numbers of parts that, respectively, $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are decomposed in. Finally, $x_i = \emptyset$ if $i > M_x$, $y_i = \emptyset$ if $i > M_y$, $\omega_\emptyset = 0$, and $\delta(\emptyset, y) = \delta(x, \emptyset) = 0$ for any x and y.

[LD] The second step is to define sets $S(x)$ – that contain substructures for structures x – for a specific convolution kernel and the subsequent set S collecting all sets S(x) for structures in X. Elements of these sets are substructures. A substructure of x is a structured object that is a subpart of x and it is considered relevant from the specific convolution kernel. For example, in sequence kernels, a relevant subsequence of a sequence s is a sequence with a subset of the elements of s but in the same order. Thus, for each convolution kernel, we introduced the related sets $S(x)$ recursively defined as follows:

Definition 2 *A convolution kernel K has an associated substructure function $S(x)$, defined as:*

$$S(x) = \begin{cases} \{x\} & \text{if } x \in G_X \\ \bigcup_{\overline{\mathbf{x}} \in R(x)} \mathbb{S}(\overline{\mathbf{x}}) & \text{otherwise} \end{cases} \tag{2}$$

where $\mathbb{S}(\overline{\mathbf{x}}) = S_1(x_1) \times \ldots \times S_M(x_M)$. By recursion, functions $S_i(x_i)$ are the substructure functions associated to convolution kernels K_i decomposing the convolution kernel K (see Definiton 1).

Then, the set S was derived by collecting all valid substructures for structured objects in X, that is, $S = \bigcup_{x \in X} S(x)$. The set S is possibly infinite and it is in principle different from the set X of the structures although these two sets can share the same definition of structured object. For example, in tree kernels (Collins & Duffy, 2002) applied to parse trees of natural language utterances, the set X contains full parse trees whereas the set S contains fragments of these parse trees. The two sets shares the same structured data type, but the nature of the two sets is different.

[LD] Thus, we proved that that convolution kernels $K(x, y)$ restricted to discrete sets are dot products of vectors $\mathbf{x}, \mathbf{y} \in \mathbb{R}^m$ where base vectors $\phi(a) \in \mathbb{R}^m$ represent substructures $a \in S$. Thus, if the basic sets X, X_i and G_X are endowed with Kroneker's delta, also the set S is endowed with it.

Lemma 1 *A convolution kernel on discrete sets can be expressed in the form:*

$$K(x,y) = \langle \mathbf{x}, \mathbf{y} \rangle = \langle \sum_{a \in S(x)} \omega_a \phi(a), \sum_{b \in S(y)} \omega_b \phi(b) \rangle \tag{3}$$

Proof: The proof is by structural induction.

Basic step Let x and y be structures that are not decomposable into parts, i.e. $S(x) = \{x\}$ and $S(y) = \{y\}$. Equation (3) becomes $K(x, y) = \omega_x \omega_y \delta(x, y) = \langle \omega_a \phi(a), \omega_b \phi(b) \rangle$.

Induction step By induction, we have that Equation (3) holds for each $K_i(x_i, y_i)$ of the definition of the convolution kernel:

$$K_i(x_i, y_i) = \langle \sum_{a_i \in S(x_i)} \omega_{a_i} \phi(a_i), \sum_{b_i \in S(y_i)} \omega_{b_i} \phi(b_i) \rangle = \sum_{a_i \in S(x_i), b_i \in S(y_i)} \omega_{a_i} \omega_{b_i} \delta(a_i, b_i)$$

Thus, we can express the convolution kernel $K(x, y)$:

$$K(x, y) = \sum_{\overline{\mathbf{x}} \in R(x), \overline{\mathbf{y}} \in R(y)} \prod_{i=1}^{M_{max}} \sum_{a_i \in S_i(x_i), b_i \in S_i(y_i)} \omega_{a_i} \omega_{b_i} \delta(a_i, b_i)$$

where $M_{max} = \max\{M_x, M_y\}$ being M_x and M_y the numbers of parts that, respectively, $\overline{\mathbf{x}}$ and $\overline{\mathbf{y}}$ are decomposed in. Finally, $x_i = \emptyset$ if $i > M_x$, $y_i = \emptyset$ if $i > M_y$, and $S_i(\emptyset) = \{\emptyset\}$. Hence:

$$K(x, y) = \sum_{\overline{\mathbf{x}} \in R(x), \overline{\mathbf{y}} \in R(y)} \sum_{\overline{\mathbf{a}} \in \mathbb{S}(\overline{\mathbf{x}}), \overline{\mathbf{b}} \in \mathbb{S}(\overline{\mathbf{y}})} \prod_{i=1}^{M_{max}} \omega_{a_i} \omega_{b_i} \prod_{j=1}^{M_{max}} \delta(a_j, b_j)$$

according to Definition 2.

Finally, since $\delta(\overline{\mathbf{a}}, \overline{\mathbf{b}}) = \prod_{i=1}^{M_{max}} \delta(a_i, b_i)$ and defining $\omega_{\overline{\mathbf{a}}} = \prod_{i=1}^{M_{max}} \omega_{a_i}$ and $\omega_{\overline{\mathbf{b}}} = \prod_{i=1}^{M_{max}} \omega_{b_i}$, we obtain:

$$K(x, y) = \sum_{a \in S(x), b \in S(y)} \omega_a \omega_b \delta(a, b) = \langle \sum_{a \in S(x)} \omega_a \phi(a), \sum_{b \in S(y)} \omega_b \phi(b) \rangle$$

□

[LD] The above result is the avenue to approximate convolution kernels on discrete sets.

2.1.3 Direct Embedding Spaces of Substructures in Reduced Spaces: Potential and Limits

[LD] Given that convolution kernels are dot product in spaces $\mathbb{R}^m$ of substructures and using the theory on the Johnson&Lindestrauss Tranform (Johnson & Lindenstrauss, 1984), it is possible to derive a first conclusion: convolution kernels can be approximated with dot product of *distributed structures* $D_\Sigma(z)$ that are vectors in $\mathbb{R}^d$ with $d << m$. This section shows this first conclusion and analyzes its limits that Section 2.2 resolves.

[LD] *Distributed structures* D_Σ are functions mapping structures in small vectors in $\mathbb{R}^d$:

$$D_\Sigma(z) = \sum_{c \in S(z)} \omega_c \Upsilon_\Sigma(c)$$

where Υ_Σ is an injective function between substructures S and vectors in $\mathbb{R}^d$. The aim is to build these functions such that:

$$\mathbb{P}(K(x, y) - \varepsilon \vartheta(x, y) < \langle D_\Sigma(x), D_\Sigma(y) \rangle < K(x, y) + \varepsilon \vartheta(x, y) > 1 - \theta$$

where ε and θ are small and $\vartheta(x, y)$ is an approximation factor depending on x and y.

[LD] To approximate convolution kernels, the injective function Υ_Σ can be derived from practices in embedding with Johnson-Lindestrauss Transform (JLT) (Johnson & Lindenstrauss, 1984) as adapted by random indexing (?) or random projection (?). The general idea is that vectors $\Upsilon_\Sigma(a)$ for substructures a are drawn out of a d-dimensional Gaussian distribution $\mathbf{v} \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ (Indyk & Motwani, 1998; Dasgupta & Gupta, 1999). We showed that given $\mathbf{a}, \mathbf{b} \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$, ε and d, it is possible to compute $\overline{\theta}$ such that:

$$\mathbb{P}(\delta(\mathbf{a}, \mathbf{b}) - \varepsilon \leq \langle \mathbf{a}, \mathbf{b} \rangle \leq \delta(\mathbf{a}, \mathbf{b}) + \varepsilon) \geq 1 - \overline{\theta} \tag{4}$$

(see Appendix E). For example, $\overline{\theta} \propto 10^{-xxx}$ with $\varepsilon = 0.1$ and $d = yyy$. Hence, there is a very high probability that $\delta(\mathbf{a}, \mathbf{b}) - \varepsilon \leq \langle \mathbf{a}, \mathbf{b} \rangle \leq \delta(\mathbf{a}, \mathbf{b}) + \varepsilon$.

We refer to Equation 4 as the property of vectors of being nearly orthonormal.

We thus demonstrated the following lemma:

Lemma 2 *For each convolution kernel on discrete sets $K(x, y)$, given an injective function $\Upsilon : S \to \mathbb{R}^d$ that maps substructures $a \in S$ to vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ and given the previously defined function D_Σ, the following property holds:*

$$\mathbb{P}(K(x, y) - \varepsilon\vartheta(x, y) < \langle D_\Sigma(x), D_\Sigma(y) \rangle < K(x, y) + \varepsilon\vartheta(x, y)) > 1 - \theta$$

where $\vartheta(x, y) = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + K(x, y)$.

Proof: We first observe that

$$\sum_{a \in S(x)} \omega_a^2 = \|\mathbf{x}\|^2 \text{ and } \sum_{b \in S(y)} \omega_b^2 = \|\mathbf{y}\|^2$$

given the definition of x and y (see equation 3). Hence, distributed structures $D_\Sigma(x)$, $D_\Sigma(y)$ and $D_\Sigma(x) + D_\Sigma(y)$ behave as random vectors drawn from $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d\|\mathbf{x}\|^2)$, $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d\|\mathbf{y}\|^2)$ and $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d\|\mathbf{x} + \mathbf{y}\|^2)$ as $D_\Sigma(\cdot)$ are linear combinations of independent normal random vectors $\Upsilon_\Sigma(\cdot)$.

From the property in equation (4), the following probabilities hold:

$$\mathbb{P}(\|D_\Sigma(x)\|^2 > (1 + \varepsilon)\|\mathbf{x}\|^2, \|D_\Sigma(x)\|^2 < (1 - \varepsilon)\|\mathbf{x}\|^2) < \overline{\theta} \tag{5}$$

$$\mathbb{P}(\|D_\Sigma(y)\|^2 > (1 + \varepsilon)\|\mathbf{y}\|^2, \|D_\Sigma(y)\|^2 < (1 - \varepsilon)\|\mathbf{y}\|^2) < \overline{\theta} \tag{6}$$

$$\mathbb{P}(\|D_\Sigma(x) + D_\Sigma(y)\|^2 > (1+\varepsilon)\|\mathbf{x}+\mathbf{y}\|^2, \|D_\Sigma(x) + D_\Sigma(y)\|^2 < (1-\varepsilon)\|D_\Sigma(x) + D_\Sigma(y)\|^2) < \overline{\theta}$$

as vectors $\mathbf{v} \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d B^2)$ behave as vectors $B\mathbf{v}'$ where $\mathbf{v}' \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$.

The union of the above events has a probability $p < 3\overline{\theta} = \theta$.

As

$$\|D_\Sigma(x) + D_\Sigma(y)\|^2 = \|D_\Sigma(x)\|^2 + \|D_\Sigma(y)\|^2 + 2\langle D_\Sigma(x), D_\Sigma(y) \rangle$$

and

$$\|\mathbf{x} + \mathbf{y}\|^2 = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2\langle \mathbf{x}, \mathbf{y} \rangle$$

we rewrite:

$$\|D_\Sigma(x) + D_\Sigma(y)\|^2 > (1 + \varepsilon)\|\mathbf{x} + \mathbf{y}\|^2$$

as:

$$\|D_\Sigma(x)\|^2 + \|D_\Sigma(y)\|^2 + 2\langle D_\Sigma(x), D_\Sigma(y)\rangle > (1+\varepsilon)(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + 2\langle \mathbf{x}, \mathbf{y}\rangle)$$

From equations (5) and (6) , we can derive the following:

$$-\|D_\Sigma(x)\|^2 - \|D_\Sigma(y)\|^2 > -\|\mathbf{x}\|^2 - \|\mathbf{y}\|^2 + \varepsilon\|\mathbf{x}\|^2 + \varepsilon\|\mathbf{y}\|^2$$

Hence, the event:

$$\langle D_\Sigma(x), D_\Sigma(y)\rangle > \langle \mathbf{x}, \mathbf{y}\rangle + \varepsilon(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + \langle \mathbf{x}, \mathbf{y}\rangle)$$

holds with a probability 3θ. In the same way, it is possible to derive that:

$$\langle D_\Sigma(x), D_\Sigma(y)\rangle > \langle \mathbf{x}, \mathbf{y}\rangle - \varepsilon(\|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + \langle \mathbf{x}, \mathbf{y}\rangle)$$

Finally, since the two equations hold with a probability 3θ, the following probability holds:

$$\mathbb{P}(K(x,y) - \varepsilon\vartheta(x,y) < \langle D_\Sigma(x), D_\Sigma(y)\rangle < K(x,y) + \varepsilon\vartheta(x,y)) > 1 - \theta$$

where $K(x,y) = \langle \mathbf{x}, \mathbf{y}\rangle$ and $\vartheta(x,y) = \|\mathbf{x}\|^2 + \|\mathbf{y}\|^2 + K(x,y)$. □

[LD] Unfortunately, this first solution is impractical as it is not trivial to directly build bijective mapping functions Υ_Σ and, thus, the function D_Σ. In fact, structures x have exponentially large sets of substructures $S(x)$. Thus, even if we solve the problem of directly defining function Υ, the direct computation of distributed structures $D_\Sigma(x)$ is implausible.

The next section describes our model on how to efficiently build both a bijective function Υ and a function D with a basic vector composition operation $\odot$ and two recursive definition: one for the distributed substructures and one for the distributed substructures that will be named distributed convolution structures.

2.2 From Distributed Structures to Distributed *Convolution* Structures

We showed that convolution kernels on discrete sets are approximated with dot products between small vectors called distributed structures. This is a positive result although a direct mapping between structures and these vectors in impossible.

A theory to efficiently compute *distributed structures* is the major result of the paper. To this aim, we introduced the *distributed convolution structures* $D(x)$ that compute *distributed structures* by recursively composing vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ that represent basic elements G_X of structured objects. We then showed that each convolution kernel $K(x,y)$ on discrete sets has a related distributed convolution structure function D such that the dot products between *distributed convolution structures* approximate convolution kernels.

This section describes how we obtained the above result. We first introduce the *shuffled circular convolution* as basic vector composition function and we show its properties. We then describe how to produce bijective Υ using this composition function to produce *distributed convolution substructures*. Finally, we introduce *distributed convolution structures* D associated to convolution kernels and we show that their dot product approximate the kernels.

2.2.1 A Composition Function preserving Vector Near-orthogonality

[LD] In this section, we introduce the *shuffled circular convolution* $\odot$, which is a binary function with *two important features*: (1) it guarantees that different expressions with the same vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ have different results; (2) it preserves multivariate normality and near orthogonality when composing up to m vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ where m depends on d. Hence, this is the basic function to build the recursive definition for the distributed substructures Υ and the distributed convolution structures D.

[LD] The *shuffled circular convolution* $\odot$ is defined as follows:

$$\mathbf{u} \odot \mathbf{v} = \Phi_1\mathbf{u} * \Phi_2\mathbf{v}$$

We combined circular convolution $*$ and vector shuffling that uses two permutation matrices Φ_1 and Φ_2. Circular convolution is a popular operation in signal processing and has been used for purposes similar to ours – encoding flat structures – for distributed representations (Plate, 1995). Unfortunately, circular convolution is commutative and associative. Consequently, different sequences of applications of the circular convolution to the same vectors yield the same result. Order is not encoded. We used vector shuffling to take order into consideration as done in random indexing models to encode word order (Sahlgren, Holst, & Kanerva, 2008). This shuffling is realized with two non-circular permutation matrices Φ_1 and Φ_2.

[LD] The properties of the shuffled circular convolution $\odot$ are the following:

Property 1 *Given* $\mathbf{a}$, $\mathbf{b}$, $\mathbf{c}$, $\mathbf{d} \in \mathbb{R}^d$, $\mathbf{e} = (1, 0, ..., 0) \in \mathbb{R}^d$ *and a scalar* $s \in \mathbb{R}$, *the following properties hold:*

1.1 Non-associativity: $\mathbf{a} \odot (\mathbf{b} \odot \mathbf{c}) \neq (\mathbf{a} \odot \mathbf{b}) \odot \mathbf{c}$

1.2 Bilinear form: *I)* $(\mathbf{a}+\mathbf{b}) \odot \mathbf{c} = \mathbf{a} \odot \mathbf{c} + \mathbf{b} \odot \mathbf{c}$; *II)* $\mathbf{c} \odot (\mathbf{a}+\mathbf{b}) = \mathbf{c} \odot \mathbf{a} + \mathbf{c} \odot \mathbf{b}$; *III)* $(s\mathbf{a}) \odot \mathbf{b} = \mathbf{a} \odot (s\mathbf{b}) = s(\mathbf{a} \odot \mathbf{b})$

1.3 Non-commutativity with a degree k. The degree of non-commutativity k is the lowest value such that for any $j < k$ *and any permutation* $[p_1, p_2, \ldots, p_j]$ *of* $[1, 2, \ldots, j]$, $\mathbf{c}_1 \odot \mathbf{c}_2 \ldots \odot \mathbf{c}_j \neq \mathbf{c}_{p_1} \odot \mathbf{c}_{p_2} \ldots \odot \mathbf{c}_{p_j}$ *if* $c_i \neq c_{p_i}$.

1.4 Identity elements: $\Phi_1^T\mathbf{e}$ *and* $\Phi_2^T\mathbf{e}$ *are the left and right identity elements, respectively*

[LD] These properties can be proven. Matrices Φ_1 and Φ_2 guarantee Property 1.1 since $\mathbf{a} \odot (\mathbf{b} \odot \mathbf{c}) = \Phi_1\mathbf{a} * \Phi_2(\Phi_1\mathbf{b} * \Phi_2\mathbf{c}) \neq \Phi_1(\Phi_1\mathbf{a} * \Phi_2\mathbf{b}) * \Phi_2\mathbf{c}) = (\mathbf{a} \odot \mathbf{b}) \odot \mathbf{c}$. As a consequence, the symbol $\bigodot_{i=1}^{k} \mathbf{x}_i$ is intended as follows $\bigodot_{i=1}^{k} \mathbf{x}_i = \mathbf{x}_1 \odot (\bigodot_{i=2}^{k} \mathbf{x}_i)$. The bilinear form of $\odot$ (Properties 1.2) is inherited from circular convolution $*$. In fact, circular convolution can be expressed as matrix multiplication. Hence, Properties 1.2 hold as $\odot$ is a matrix multiplication of permutation and circular convolution matrices. Although we could not formally prove the non-commutativity of $\odot$, extensive experimental validation shows that the Property 1.3 holds for very high values of k. We took k up to 100 with 10^6 permutations $[p_1, p_2, \ldots, p_j]$ and the property was not falsified provided that Φ_1 and Φ_2 were not circulant matrices. The necessity of this last condition can be formally proven[2].

2. If Φ_1 and Φ_2 are circulant matrices, $\Phi_i(\mathbf{a} * \mathbf{b}) = (\mathbf{a} * \Phi_i\mathbf{b}) = (\Phi_i\mathbf{a} * \mathbf{b})$. In fact, $\mathbf{a} * \mathbf{b} = A\mathbf{b}$ where A is a circulant matrix with a as first column. Thus, if Φ_i is a circulant matrix, $\Phi_i A = A\Phi_i$ from where

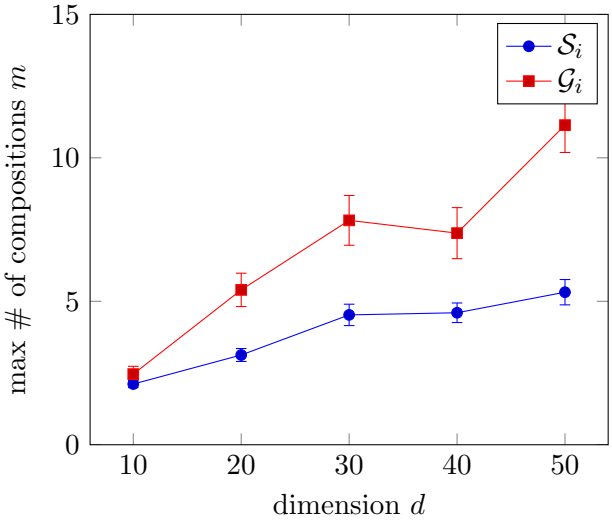

Figure 1: Average number of $\odot$ compositions preserving multivariate normality $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ for $\mathcal{G}_i$ or $\mathcal{S}_i$: d is the dimensionality of the space $\mathbb{R}^d$ of the multivariate normal and m is the maximum number of compositions such that $\mathcal{G}_i$ or $\mathcal{S}_i$ are multivariate normal for $i < m$

[LD] Non-associativity (Prop. 1.1) and non-commutativity (Prop. 1.3) plus an additional lemma hereafter introduced guarantee the first important feature of $\odot$: different $\odot$-expressions e combing vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ with $\odot$ produce different vectors. In fact, a valid $\odot$-expression e is either a symbol that represent a vector drawn from $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ or $e = e_1 \odot e_2$ where e_1 and e_2 are valid expressions. Consequently, given the $\odot$-expressions e_1, e_2 and e_3, $e_1 \odot (e_2 \odot e_3) \neq (e_1 \odot e_2) \odot e_3$ as $\odot$ is non-associative and $e_1 \odot (e_2 \odot e_3) \neq e_3 \odot (e_2 \odot e_1)$ as $\odot$ is non-commutative. Moreover, it is possible to show the additional lemma:

Lemma 3 *Given* $\mathbf{a}$ *as the vector derived from an* $\odot$*-expression with more than one symbol and* $\mathbf{b}$ *directly drawn from* $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$, $\mathbf{a} \neq \mathbf{b}$

by using the non-commutative property and the fact that it is very unlike to obtain identity elements (Prop. 1.4) by combining vectors directly drawn from $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$. Hence, different expressions e produce different computations and, thus, different vectors. We used $\odot$ to define the bijective function Υ between structures and vectors as shown in Sec. 2.2.2.

[LD] With an empirical investigation, we show that $\odot$ has a second important feature: it preserves (1) multivariate normality and (2) near orthogonality when composing up to m vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ where m depends on d.

For the set up of the empirical investigation, we generated sets $\mathcal{S}_n$ containing vectors obtained combining up to n vectors $v \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ for different d. $\mathcal{S}_n$ is built as follows: we first generated a set $\mathcal{G}_1 = \mathcal{S}_1$ of vectors $\mathbf{v} \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$. We then generated sets $\mathcal{G}_i$ of vectors $\mathbf{v} = \mathbf{v}_j \odot \mathbf{v}_{i-j}$ where j is randomly selected in $\{1, \ldots, i-1\}$ and $\mathbf{v}_k$ are randomly picked in $\mathcal{G}_k$. Finally, we set $\mathcal{S}_n = \bigcup_{i=1}^{n} \mathcal{G}_i$.

$\Phi_i(\mathbf{a} * \mathbf{b}) = (\mathbf{a} * \Phi_i \mathbf{b})$. The same applies for $\mathbf{b}$ as $*$ is commutative. Hence, $\Phi_i(\mathbf{a} \odot \mathbf{b}) = (\Phi_1 \mathbf{a} * \Phi_i \Phi_2 \mathbf{b})$ and,

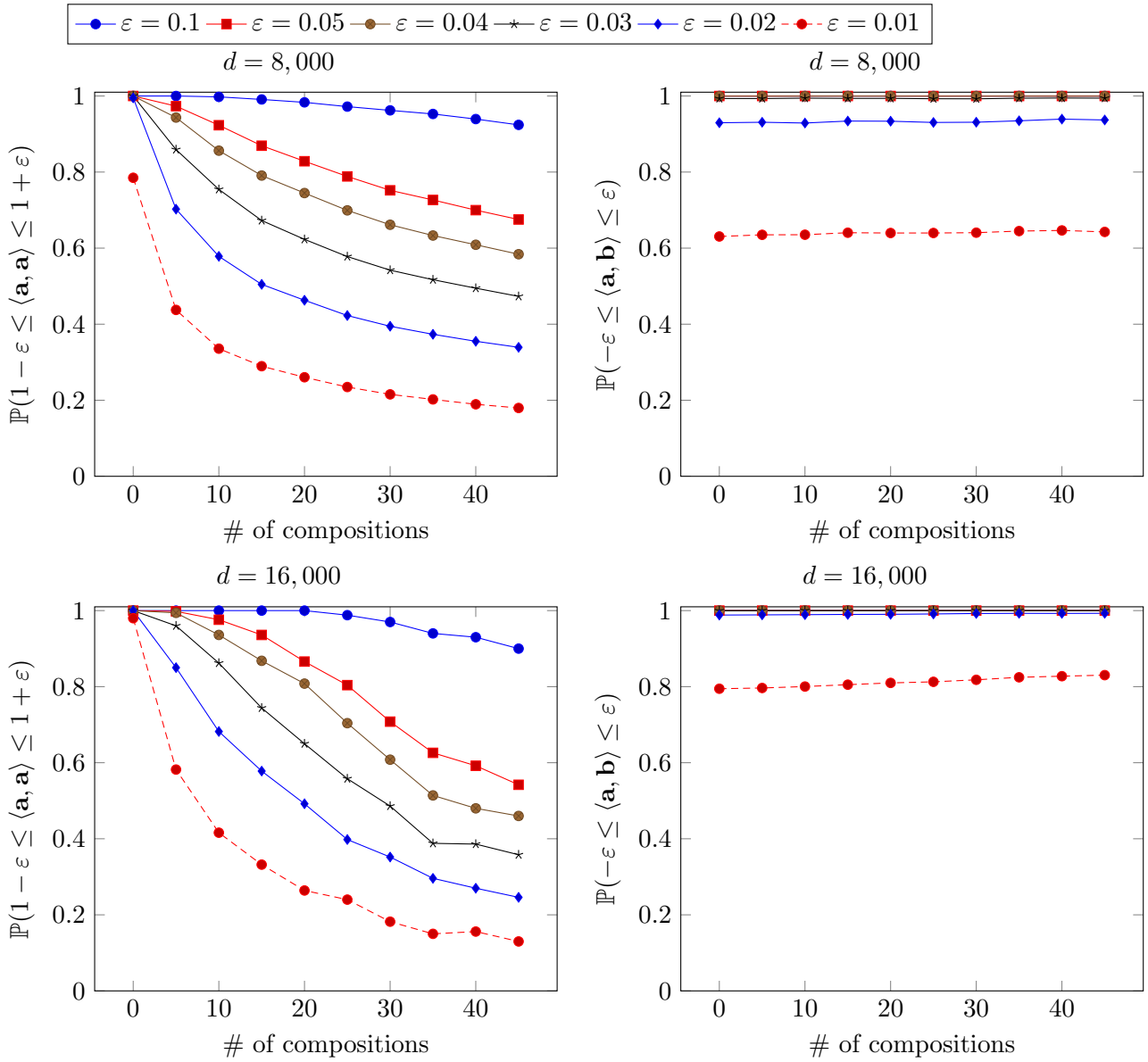

Figure 2: Probability $\mathbb{P}(\delta(\mathbf{a}, \mathbf{b}) - \varepsilon \leq \langle \mathbf{a}, \mathbf{b} \rangle \leq \delta(\mathbf{a}, \mathbf{b}) + \varepsilon) \geq 1 - \overline{\theta}$ of the sets $\mathcal{S}_i$ for different values of ε and two different values d (8,000 and 16,000). Plots are divided for $\delta(\mathbf{a}, \mathbf{b}) = 0$ and $\delta(\mathbf{a}, \mathbf{b}) = 1$.

[LD] In a first experiment, we showed that $\odot$ preserves multivariate normality for m compositions where m depends on d. The experiment wants to empirically answer to the question: does m increase if d does? To this extent, we used the Royston's Test (Royston,

with recursive applications of $\odot$, permutations will accumulate to the last vector. Thus, commutative properties hold.

1983) to assess whether vectors in $\mathcal{S}_i$ are distributed as $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$. For each d, we generated $\mathcal{S}_i$ for $i = 1 \dots 20$, where the base sets $\mathcal{G}_i$ contain 100 vectors. Then, we determined the maximum m such that $\mathcal{S}_i$ with $i \leq m$ satisfies the Royston's Test. Finally, we repeated this procedure 40 times and averaged the results. Due to the computational limits of the Royston's Test, we tested for $d = 10, 20, 30, 40$ and 50. Results confirm that, increasing d, $\odot$ preserves the distribution $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ for the sets $\mathcal{G}_i$ and $\mathcal{S}_i$ for increasingly more compositions (see Fig. 1). This is an extremely important property of $\odot$.

[LD] In a second experiment, we showed that $\odot$ preserves nearly orthogonality of unit vectors, that is, $\mathbb{P}(\delta(\mathbf{a}, \mathbf{b}) - \varepsilon \leq \langle \mathbf{a}, \mathbf{b} \rangle \leq \delta(\mathbf{a}, \mathbf{b}) + \varepsilon) \geq 1 - \overline{\theta}$ (Equation 4). In this experiments, $\mathcal{S}_i$ are sets with 1,000 vectors and we considered two values 8,000 and 16,000 as dimension d. Results show that $\odot$ easily preserves nearly orthogonality of vectors in $\mathcal{S}_i$ (see Fig. 2). $\mathbb{P}(-\varepsilon \leq \langle \mathbf{a}, \mathbf{b} \rangle \leq \varepsilon)$ is high for a large number of compositions. Moreover, it also preserves the unit norm of vectors although the probability $\mathbb{P}(1 - \varepsilon \leq \langle \mathbf{a}, \mathbf{a} \rangle \leq 1 + \varepsilon)$ decreases as ε decreases and *# of compositions* increases. Fortunately, increasing d improves these probabilities.

[LD] Since the function $\odot$ guarantees bijectivity, preserves multivariate normality and nearly orthonormality up to a given number of compositions, it is a good composing function for defining Υ that build recursively distributed substructures.

FLUSSO: *NOT REVISED FROM HERE*

2.2.2 Distributed Convolution Substructures: Small Vectors for Substructures

[OCAR] Using the properties of the shuffled circular convolution, we showed that it is possible to recursively define a bijective function Υ that builds vectors $\Upsilon(x)$ using basic vectors for terminal objects. Starting from a bijective function $\upsilon : G_X \to \mathcal{G}_\mathcal{X}$, we recursively defined Υ and, then, we proved that it is a bijective function between sets of substructures and vectors in $NOV(\varepsilon, \theta)$.

[LD] The function $\Upsilon(x)$ is defined as follows:

Definition 3 *Let $x \in S$ be a structured object, G_X the ground set for X, $\mathcal{G}_\mathcal{X} = NOV(\varepsilon, \theta)$, $\mathbf{q} \in G_X$ and $\upsilon : G_X \to \mathcal{G}_\mathcal{X}$ a bijective function. We recursively define $\Upsilon(x)$ as:*

$$\Upsilon(x) = \upsilon(x) = \mathbf{x} \text{ if } x \text{ is a terminal object, i.e. } x \in G_X$$

$$\Upsilon(\overline{\mathbf{x}}) = \Upsilon(x_1) \odot \bigodot_{i=2}^{M} (\mathbf{q} \odot \Upsilon(x_i)) \text{ otherwise}$$

For example, the tree τ in Fig. **??** is represented by the vector obtained as:

$$\tau = (\mathbf{A} \odot (\mathbf{q} \odot (\mathbf{B} \odot (\mathbf{q} \odot \mathbf{W1})))) \odot (\mathbf{q} \odot (\mathbf{q} \odot (\mathbf{C} \odot (\mathbf{q} \odot (\mathbf{D} \odot (\mathbf{q} \odot \mathbf{W2}))))) \odot (\mathbf{q} \odot (\mathbf{E} \odot (\mathbf{q} \odot \mathbf{W3})))))$$

whereas the sequence $s =$ `W1 W2 W3` is represented by the following vector:

$$\mathbf{s} = (\mathbf{W1} \odot (\mathbf{q} \odot (\mathbf{W2} \odot (\mathbf{q} \odot \mathbf{W3}))))$$

[LD] We then proved that the function Υ is bijective with basic vectors in $NOV(\varepsilon, \theta)$.

Lemma 4 *Given two structures x and y in S and the function Υ, the following property holds:*

$$\delta(x, y) = 1 \iff \delta(\Upsilon(x), \Upsilon(y)) = 1$$

Proof: This part of the statement is obvious: if $x = y$ then $\Upsilon(x) = \Upsilon(y)$. We need to formally show that: if $x \neq y$ then $\Upsilon(x) \neq \Upsilon(y)$.

Basic step Let x be a terminal object, i.e. $x \in G_X$. Two cases are possible:
Case 1 y is a terminal object too. As $x \neq y$, $\upsilon(x) \neq \upsilon(y)$ by construction of $\upsilon(\cdot)$.
Case 2 y is not a terminal object. Hence, $\upsilon(x) \neq \Upsilon(y)$ for Lemma 3.

Induction step Let $\overline{\mathbf{x}} = (x_1, \ldots, x_h)$ and $\overline{\mathbf{y}} = (y_1, \ldots, y_k)$ be two different non-terminal structured objects. We have two cases:
Case 1 It exists a $z \leq k$ and $z \leq h$ such that $x_z \neq y_z$. By inductive hypothesis, $\Upsilon(x_z) \neq \Upsilon(y_z)$. Thus, $\bigodot_{i=z}^{k} \Upsilon(x_i) \neq \bigodot_{j=z}^{h} \Upsilon(y_j)$ as the first vectors are different $\Upsilon(x_z) \neq \Upsilon(y_z)$. Hence, as $\odot$ is non-associative, $\bigodot_{i=1}^{k} \Upsilon(x_i) \neq \bigodot_{j=1}^{h} \Upsilon(y_j)$ by recursively applying $\odot$ as the right vectors are always different.
Case 2 Otherwise, without loss of generality, we assume that $h < k$. Thus, $\Upsilon(x_h) \neq \bigodot_{j=h}^{k} (\mathbf{q} \odot \Upsilon(y_j))$ as $\Upsilon(x_h)$ does not start with the vector $\mathbf{q}$ □

[LD] Hence, distributed substructures $\Upsilon(x)$ are nearly orthogonal unit vectors in $NOV(\varepsilon, \theta)$ whenever their degree $d(\Upsilon(x))$ is lower than n as Υ is a bijective function mapping substructures to vectors in $NOV(\varepsilon, \theta)$.

2.2.3 Distributed Convolution Structures for Distributed Convolution Kernels

We have now nearly all the elements in place to realize the goal of efficiently mapping structures in small vectors and, thus, having the possibility of using linear learning machines. The last element is the definition of *distributed convolution structures* $D(x)$ and to show that each convolution kernel on discrete sets has a related D function. We then showed that $D(x)$ computes sums of *distributed substructures* related to a particular convolution kernel $K(x, y)$. Hence, by reformulating Lemma 2, convolution kernels are approximated with dot products of *distributed convolution structures*.

Distributed convolution structure functions are defined as follows.

Definition 4 ***Distributed Convolution Structure*** *A function $D : X \to \mathbb{R}^d$ is a Distributed Convolution Structure function of a convolution kernel $K(x, y)$ if*

$$D(x) = \begin{cases} \omega_x \upsilon(x) & \text{if } x \in G_X \\ \sum\limits_{\overline{\mathbf{x}} \in R(x)} \bigodot\limits_{i=1}^{M} D_i(x_i) & \text{otherwise} \end{cases}$$

where $D_i(\cdot)$ are distributed convolution structure functions.

Hence, we can prove the following lemma.

Lemma 5 *If the ground set G_X is endowed with a basic function $\upsilon(x) = \mathbf{x}$, that maps terminal objects to nearly orthonormal vectors in a set $\mathcal{G}_X$, that is, $\upsilon : G_X \to \mathcal{G}_X$, a convolution distributed representation:*

$$D(x) = \sum_{\overline{\mathbf{x}} \in R(x)} \bigodot_{i=1}^{M} D_i(x_i)$$

computes:

$$D(x) = \sum_{\overline{\mathbf{a}} \in S(x)} \omega_{\overline{\mathbf{a}}} \Upsilon(\overline{\mathbf{a}})$$

Proof: The theorem is proved by structural induction.
Basic step Let x be a terminal object, we have that $D(x) = \omega_x \upsilon(x) = \omega_x \mathbf{x}$.
Induction step Applying the inductive hypothesis, we have

$$D_i(x_i) = \sum_{a_i \in S(x_i)} \omega_{a_i} \Upsilon(a_i)$$

Hence:

$$D(x) = \sum_{\overline{\mathbf{x}} \in R(x)} \bigodot_{i=1}^{M} \sum_{a_i \in S_i(x_i)} \omega_{a_i} \Upsilon(a_i) =$$

(for the bilinear properties of $\odot$)

$$= \sum_{\overline{\mathbf{x}} \in R(x)} \sum_{\overline{\mathbf{a}} \in \mathbb{S}(\overline{\mathbf{x}})} \prod_{i=1}^{M} \omega_{a_i} \bigodot_{j=1}^{M} \Upsilon(a_j) =$$

(according to the definitions of $S(x)$ and $\mathbb{S}(\overline{\mathbf{x}})$ in Definition 2))

$$= \sum_{\overline{\mathbf{a}} \in S(x)} \prod_{i=1}^{M} \omega_{a_i} \Upsilon(\overline{\mathbf{a}})$$

Posing $\omega_{\overline{\mathbf{a}}} = \prod_{i=1}^{M} \omega_{a_i}$, the lemma is shown. □

Finally, the main theorem of the paper is the following:

Theorem 1 *Convolution kernels over discrete sets:*

$$K(x, y) = \sum_{\overline{\mathbf{x}} \in R(x), \overline{\mathbf{y}} \in R(y)} \prod_{k=1}^{D} K_k(x_k, y_k)$$

have associated distributed convolution structures:

$$D(z) = \sum_{\overline{\mathbf{z}} \in R(z)} \bigodot_{k=1}^{M} D_k(z_k)$$

where $D_k(z_k)$ are distributed convolution structures for $K_k(x_k, y_k)$ and the following property holds:

$$\mathbb{P}(K(x, y) - \varepsilon\vartheta(x, y) < \langle D(x), D(y) \rangle < K(x, y) + \varepsilon\vartheta(x, y)) > 1 - |S(x)||S(y)|\theta$$

with $\vartheta(x, y) = \sum_{a \in S(x), b \in S(y)} \omega_a \omega_b$

We omit the proof that at this point of the paper is trivial. It can be obtained by using Lemmas 1, 2 and 5 and by observing that $R(\cdot)$ and G_Z are the same sets both for D and K.

We then obtained the promised result: *distributed convolution structures* represent structures in small vectors and, thus, linear learning machines can be used when learning with structured data. Therefore, *distributed convolution structures* are the avenue to crack the dogma that low computational complexity and fully awareness of structural information are incompatible goals.

3. Reformulating Existing Convolution Kernels as Distributed Convolution Structures

Distributed convolution kernels are a general framework where specific kernels can be defined. We then apply the idea to three existing kernels: the tree kernels (Collins & Duffy, 2002), the string or sequence kernels (Lodhi et al., 2002), and, finally, the partial tree kernels (Kashima & Koyanagi, 2002; Moschitti, 2006a). For the distributed version of each kernel, we introduce the corresponding *distributed convolution structure*. Each section contains: the description of the feature space of the substructures along with the definition of related distributed substructures and, finally, the definition of the distributed convolution structures that computes the distributed substructures in $\mathbb{R}^d$ without enumerating all the substructures (as it happens for convolution kernels).

3.1 Distributed Tree Kernels

In (Zanzotto & Dell'Arciprete, 2012), we already approached tree kernels using the theory generalized in this paper. Thus, the distributed tree kernels are a good starting point to describe the application of the distributed convolution kernels.

The feature space of the tree fragments used in (Collins & Duffy, 2002) can be easily described in this way. Given a context-free grammar $G = (N, \Sigma, P, S)$ where N is the set of non-terminal symbols, Σ is the set of terminal symbols, P is the set of production rules, and S is the start symbol, the valid tree fragments for the feature space are any tree obtained by any derivation starting from any non-terminal symbol in N.

The above definition is impractical as it describes a possibly infinite set of features. Thus, a description of a function $S(t)$ that extracts the active subtrees from a given tree t is generally preferred. Given a tree t, $S_{TK}(t)$ contains any subtree of t which includes more than one node, with the restriction that entire (not partial) rule productions must be included. In other words, if node n in the original tree has children $c_1, \ldots, c_m$, every subtree containing n must include either the whole set of children $c_1, \ldots, c_m$, or none of them (i.e. having n as terminal node). Figure 3 gives an example.

For the formulation of the tree kernels described in (Zanzotto & Dell'Arciprete, 2012), the weight ω_τ for the feature representing the tree fragment $\tau \in S_{TK}(t)$ for a given tree t is:

$$\omega_\tau = f_{\tau,t}\sqrt{\lambda^{|N(\tau)|-1}}$$

where $|N(\tau)|$ is the number of nodes of τ. In the original formulation (Collins & Duffy, 2001), the contribution of the tree fragment τ to the TK is $\lambda^{|N(\tau)|}$ giving a $\omega = \sqrt{\lambda^{|N(\tau)|}}$

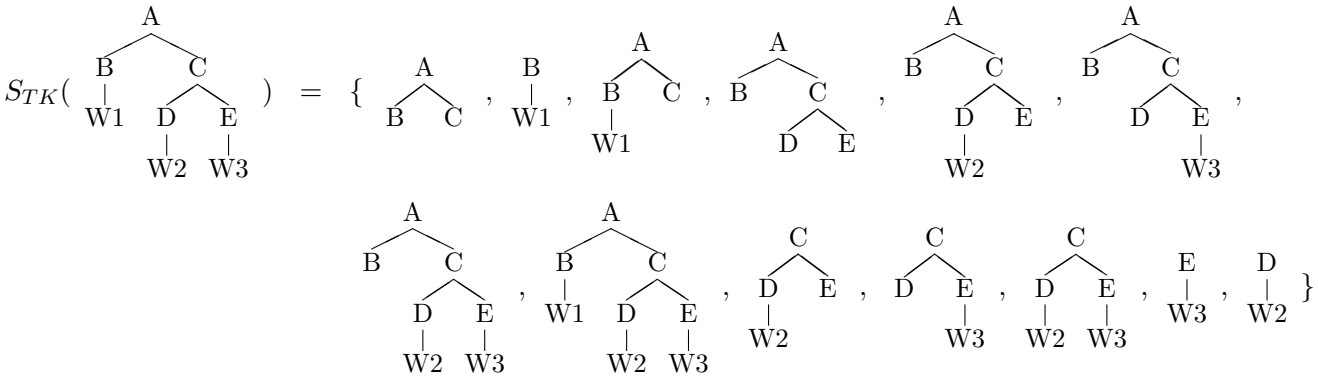

Figure 3: Tree Fragments for the tree kernel

(see also (Pighin & Moschitti, 2010)). This difference is not relevant in the overall theory of the distributed tree kernels.

The distributed tree fragments for the reduced version of this space are the depth-first visits of the above subtrees. For example, given the tree fragments in Figure 3, the corresponding distributed tree fragments for the first, the second, and the third tree are: $(\mathbf{A} \odot (\mathbf{B} \odot \mathbf{C}))$, $(\mathbf{B} \odot \mathbf{W1})$ and $(\mathbf{A} \odot ((\mathbf{B} \odot \mathbf{W1}) \odot \mathbf{C}))$.

The distributed convolution structure for the computation of distributed trees $DT(t) = \mathbf{t}$ is the following:

$$DT(t) = \sum_{\overline{\mathbf{c}} \in R_N(t)} s(\overline{\mathbf{c}}) \tag{7}$$

where $s(n)$ represents the sum of distributed vectors for the subtrees of t rooted in node n. Function $s(n)$ is recursively defined as follows:

$$s(\overline{\mathbf{c}}) = \begin{cases} \mathbf{0} & \text{if } c \text{ is a terminal node} \\ \upsilon(c_0) \odot \bigodot_{i=1}^{|\overline{\mathbf{c}}|-1} (\upsilon(r(c_i)) + \sqrt{\lambda} s(\overline{\mathbf{c_i}})) & \text{otherwise} \end{cases}$$

The distributed convolution structures DT follows Lemma 5 with $S(t) = S_{TK}(t)$ and with the weight ω_τ of each distributed tree fragment $\tau \in S_{TK}(t)$. With dynamic programming, the time complexity of this function is linear $O(|N(t)|)$ in terms of application of $\odot$ and the space complexity is d (where d is the size of the vectors in $\mathbb{R}^d$).

3.2 Distributed Sequence Kernels

Given two sequences of symbols (strings), String or Sequence Kernels (SK) (Lodhi et al., 2002) perform a weighted count of the common subsequences of symbols, possibly including gaps. Sequence kernels are convolution kernels on countable sets. Thus, this is the fourth kernel function family to which we can apply the method described in this paper, by defining the Distributed Sequences (DS).

To describe the feature space of the subsequences and the actual distributed sequences, we need to introduce a specific set of symbols. In this context, the set of basic symbols is C. A sequence s is a juxtaposition of symbols, that is, $s \in C^*$. By $C(s)$, we denote the characters of the sequence s. Each symbol of a sequence s will be denoted by s_i, with $1 \leq i \leq |C(s)|$. Given a vector $\mathbf{i}$ of ordered indexes, $s[\mathbf{i}]$ denotes the subsequence $s_{i_1} s_{i_2} ... s_{i_n}$. By $l(\mathbf{i})$ we denote the length spanned by $s[\mathbf{i}]$ in s, i.e.: $l(\mathbf{i}) = i_n - i_1 + 1$. Finally, by $s[n, m]$ we denote the subsequence $s_n s_{n+1} ... s_{m-1} s_m$. Most of this notation is used only in this section.

Without loss of generality, we focus here on the p-bounded sequence kernel (as also defined in (Lodhi et al., 2002)). For this kernel, a subsequence ss is an active feature in the set $S_p^{DS}(s)$ for a sequence s if ss is a subsequence of s, that is $ss = s[\mathbf{i}]$ for some $\mathbf{i}$, and $|ss| \leq p$. For example, given the sequence $s = \mathtt{w}_1\mathtt{w}_2\mathtt{w}_3\mathtt{w}_4$ where $\mathtt{w}_i \in C$, the set $S_2^{DS}(\mathtt{w}_1\mathtt{w}_2\mathtt{w}_3\mathtt{w}_4)$ is:

$$S_2^{DS}(\mathtt{w}_1\mathtt{w}_2\mathtt{w}_3\mathtt{w}_4) = \{\mathtt{w}_1, \mathtt{w}_2, \mathtt{w}_3, \mathtt{w}_4, \mathtt{w}_1\mathtt{w}_2, \mathtt{w}_1\mathtt{w}_3, \mathtt{w}_1\mathtt{w}_4, \mathtt{w}_2\mathtt{w}_3, \mathtt{w}_2\mathtt{w}_4, \mathtt{w}_3\mathtt{w}_4\}$$

The weight ω_{ss} of each subsequence ss is:

$$\omega_{ss} = f_{ss,s} \lambda^{l(\mathbf{i})}$$

where $f_{ss,s}$ counts the occurrences of the subsequence ss in s. The weight depends on $l(\mathbf{i})$ that represents the span of ss in s.

The definition of the p-bounded distributed sequence $DS_p(s)$ as a distributed convolution structure follows:

$$DS_p(s) = \sum_{i=1}^{|s|} \sum_{n=1}^{p} d_n(s[i, |C(s)|])$$

with:

$$d_n(xs) = \begin{cases} 0 & \text{if } |C(xs)| < n \\ \lambda \upsilon(x) & \text{if } n = 1 \\ \upsilon(x) \odot v_n(xs) & \text{otherwise} \end{cases}$$

$$v_n(xs) = \begin{cases} 0 & \text{if } |C(xs)| < n \text{ or } n = 1 \\ \lambda \left(d_{n-1}(s) + v_n(s) \right) & \text{otherwise} \end{cases}$$

In this formulation, $d_n(s)$ are all the nearly orthonormal vectors for the substrings $s[\mathbf{i}]$ of s of length n and starting at s_1, that is, $|s[\mathbf{i}]| = n$ and $i_1 = 1$.

The time computational complexity of function DS_p is $O(|C(s)|p)$ in terms of both vector sums and $\odot$ compositions.

3.3 Distributed Partial Tree Kernels

Partial Tree Kernels (PTK) (Kashima & Koyanagi, 2002; Moschitti, 2006a) is a variant of Tree Kernels (Collins & Duffy, 2002). In this variant, a larger feature space is obtained by relaxing the constraint on the integrity of the production rules appearing in a subtree. Thus, partial production rules may be included in a subtree, i.e. a subtree may contain any subset of the original children for each one of its nodes. Moreover, even single nodes are included as subtrees. Figure 4 gives an example of the corresponding function $S_{PTK}(t)$.

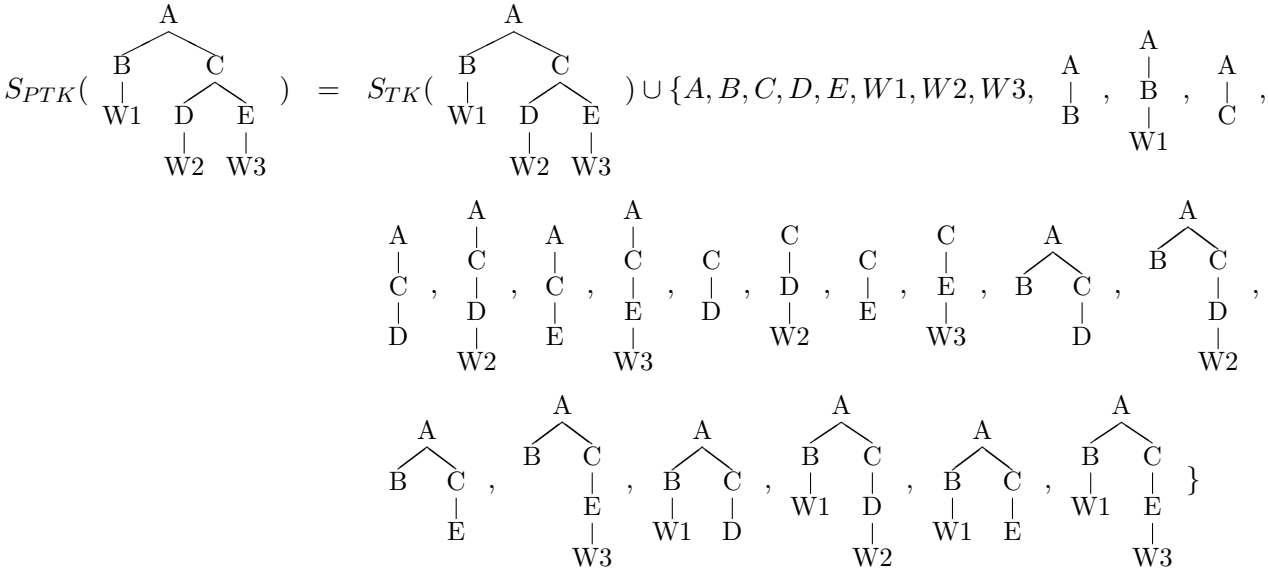

Figure 4: Tree Fragments for the Partial Tree Kernel

The PTK weights tree fragments according to their size and to the number of gaps in their productions, with respect to the original tree. Thus, the weight ω_τ for the feature representing the tree fragment $\tau \in S_{PTK}(t)$ for a given tree t is:

$$\omega_\tau = \mu^{\bar{\tau}} \sqrt{\lambda^{|N(\tau)|}}$$

where $|N(\tau)|$ is the number of nodes of τ and $\bar{\tau}$ is the number of children missing in the productions of τ, with respect to those of t.

The definition of the distributed partial trees (DPT) as a distributed convolution structure follows:

$$DPT(t) = \sum_{\overline{\mathbf{c}} \in R_N(t)} s(\overline{\mathbf{c}}) \tag{8}$$

where:

$$\begin{aligned} s(\overline{\mathbf{c}}) &= \begin{cases} \sqrt{\lambda}\mu\upsilon(c) & \text{if } c \text{ is terminal} \\ \sqrt{\lambda}\left(\mu\upsilon(c_0) + \upsilon(c_0) \odot \sum_{k=1}^{|\overline{\mathbf{c}}|-1} d_k(\overline{\mathbf{c}})\right) & \text{otherwise} \end{cases} \\ d_k(\overline{\mathbf{c}}) &= \begin{cases} s(\overline{\mathbf{c}}_k) & \text{if } k = |\overline{\mathbf{c}}| - 1 \\ s(\overline{\mathbf{c}}_k) + s(\overline{\mathbf{c}}_k) \odot \sum_{i=k+1}^{|\overline{\mathbf{c}}|-1} \mu^{i-k-1} d_i(\overline{\mathbf{c}}) & \text{if } k < |\overline{\mathbf{c}}| - 1 \end{cases} \end{aligned} \tag{9}$$

The time computational complexity of function DPT is $0(\rho|N(t)|)$ in terms of $\odot$ compositions and $O(\rho^2|N(t)|)$ in terms of vector sums, where ρ is the maximum branching factor of t.

4. Computational complexity of the Distributed Convolution Kernels in Linear SVMs

We have introduced the distributed convolution kernels as a different approach to representing structured data in machine learning. In the previous sections, we have shown their properties and their computational complexities that are strictly connected to the cost of the real used basic binary operations. This section investigates whether this approach is computationally convenient. We perform the worst case analysis of the linear SVMs (Joachims, 2006; Shalev-Shwartz et al., 2011) with the DCKs against a kernelized SVM with convolution kernels. The kernels are analyzed in class of equivalence of kernels and distributed kernels grouped with respect to the computational complexity: (1) the tree, the subpath, and the route kernels; (2) the string or sequence kernels; (3) and, the partial tree kernels.

The time complexity of the learning phase of the kernelized verions of SVM is $O(nn_{SV} + n_{SV}^3)$ in terms of application of the kernel functions where n is the number of training examples and n_{SV} is the number of support vectors. (Steinwart, 2004) shows that n_{SV} grows linearly with n, thus, the complexity of the learning phase is $O(n^3)$. The time complexity of a single classification step is $O(n_{SV})$, that, for the same reasons, become $O(n)$. The learning and classification time compexities when using the different classes of convolution kernels are then the following: (1) $O(n^3|N(t)|^2)$ and $O(n|N(t)|^2)$ for TK, SPK, and RK as computational complexities of these kernels is $O(|N(t)|^2)$ where $|N(t)|$ is the number of nodes of a generic tree t (see (Collins & Duffy, 2002) for tree kernels); (2) $O(n^3p|C(s)|^2)$ and $O(np|C(s)|^2)$ for the SK as the kernel cost is $O(p|C(s)|^2)$ where $|C(s)|$ is the number of characters in the generic sequence s and p is the selected max length of the substrings (Lodhi et al., 2002); (3) $O(n^3\rho^2|N(t)|^2)$ and $O(n\rho^2|N(t)|^2)$ for the PTK as the kernel cost is $O(\rho^2|N(t)|^2)$ where ρ is the max branching factor of the generic tree t (Moschitti, 2006a).

Kernel Class		SVM+CK	LinearSVM+DCK
Tree	Train	$O(n^3\|N(t)\|^2)$	$O(n\|N(t)\|d\log d)$
	Classif.	$O(n\|N(t)\|^2)$	$O(\|N(t)\|d\log d)$
String or Sequence	Train	$O(n^3p\|C(s)\|^2)$	$O(np\|C(s)\|d\log d)$
	Classif.	$O(np\|C(s)\|^2)$	$O(p\|C(s)\|d\log d)$
Partial Tree	Train	$O(n^3\rho^2\|N(t)\|^2)$	$O(n\rho\|N(t)\|d\log d)$
	Classif.	$O(n\rho^2\|N(t)\|^2)$	$O(\rho\|N(t)\|d\log d)$

Table 1: Time computational complexities: Linear SVMs along the different distributed convolution kernels (LinearSVM+DCK) compared with kernelized SVMs with the convolution kernels (SVM+CK). n is the number of training examples, d is the dimension of the space $\mathbb{R}^d$ for the DCSs, t is a generic tree, s is a generic string, $|N(t)|$ is the number of nodes of t, $|C(s)|$ is the number of characters in the sequence s, p is the max length of the considered substring, and ρ is the max branching factor of the trees t. The cost unit of SVM+CK is an access to a Δ function and a product whereas the cost unit of LinearSVM+DCK is a product.

Linear versions of SVM (Joachims, 2006; Shalev-Shwartz et al., 2011) have a time complexity of $O(gn)$ for the learning phase where n is still the number of training examples and g is the max number of non-zero dimensions in a generic vector. The complexity is expressed in terms of products in the target space. The one-step classification complexity is $O(g)$. As the distributed convolution structures have dense vectors, $g = d$ for the DCKs where d is the dimension of the space $\mathbb{R}^d$ of the DCSs. The learning and classification costs are then $O(dn)$ and $O(d)$. To exploit DCKs in a linear SVM, we have to sum the previous costs with the cost of converting the structure to a vector in $\mathbb{R}^d$. As this conversion cost is bigger that d, the conversion is the dominant term in the sum of complexities. The time complexities of the computation of the distributed convolution structures are expressed with respect to the basic binary function. We use here the two operations selected in the previous section: the shuffled γ-product $\otimes$ with a complexity of d and the shuffled circular convolution $\circledast$ with a complexity of $d \log d$. The complexity of the computation of the different distributed convolution structures is reported at the end of each corresponding sub-section in Section 3.

The analysis of the time computational complexity is summarized in Table 1. These complexities show that linear SVMs with distributed convolution kernels are computationally much better than kernelized SVMs with convolution kernels. The dependence with respect to the number of training examples becomes the dominant factor as n increases as we can obtain relevant results in the final task with d below 10,000 (see Section 5).

The space complexity comparison is even more promising. The space complexity of the kernelized SVM with convolution kernels that is $O(n^2)$. Even the space complexity of a linear SVM with an explicit space for the convolution kernel is prohibitive. Using the possibly sparse vectors in learning or classification requires a space complexity of $O(gn)$ or $O(G)$ where G is actual dimension of the explicit feature space. A linear SVMs with distributed convolution kernels has instead only a space complexity of $O(d)$ (since in that case $g = G = d$) that is extremely smaller than the other two.

5. Experiments and Evaluation

Distributed convolution kernels are an attractive counterpart of the traditional convolution kernels. But, as these kernels are approximations of the original ones, we want to investigate how well they approximate the original kernels. The rest of the section is organized as follows. Section 5.1 describes the test-bed and Section 5.2 then proposes a direct and task-based evaluation on how DCKs approximate CKs.

5.1 Test-bed and Parameters

We experimented with 3 linguistic tasks: question classification (QC) (Li & Roth, 2002), recognizing textual entailment (RTE) (Dagan, Glickman, & Magnini, 2006) and paraphrasing ().

These 3 tasks provided structured data, that is, sequences and trees

For the experiments, we used standard datasets for the two NLP tasks of QC and RTE.

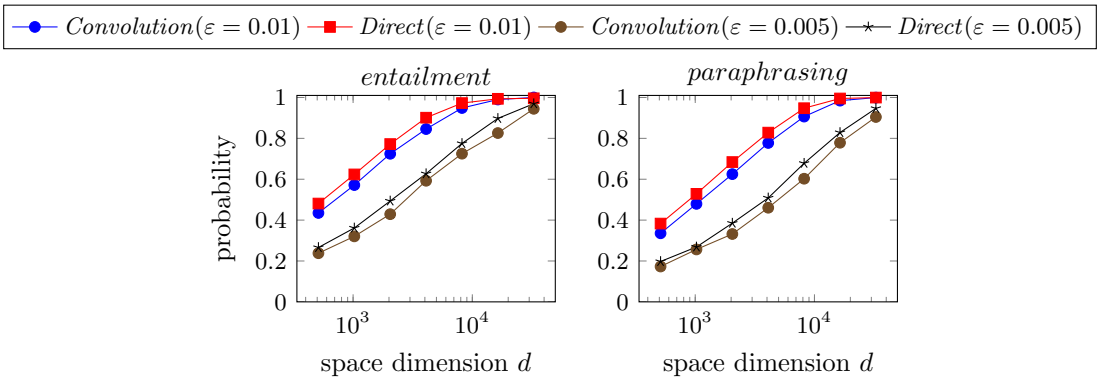

Figure 5: Comparison between direct embedding of fragment spaces (*Direct*) and *convolution distributed structures* (*Convolution*): Estimated probabilities for Lemma 2 ($\varepsilon = 0.01$ and $\varepsilon = 0.005$) with respect to different sizes of the embedding space $\mathbb{R}^d$. Considered distributed structures: Distributed Trees (DT) with $\lambda = 0.4$. Datasets: entailment and paraphrasing. Estimation done over 2,000 pairs of trees.

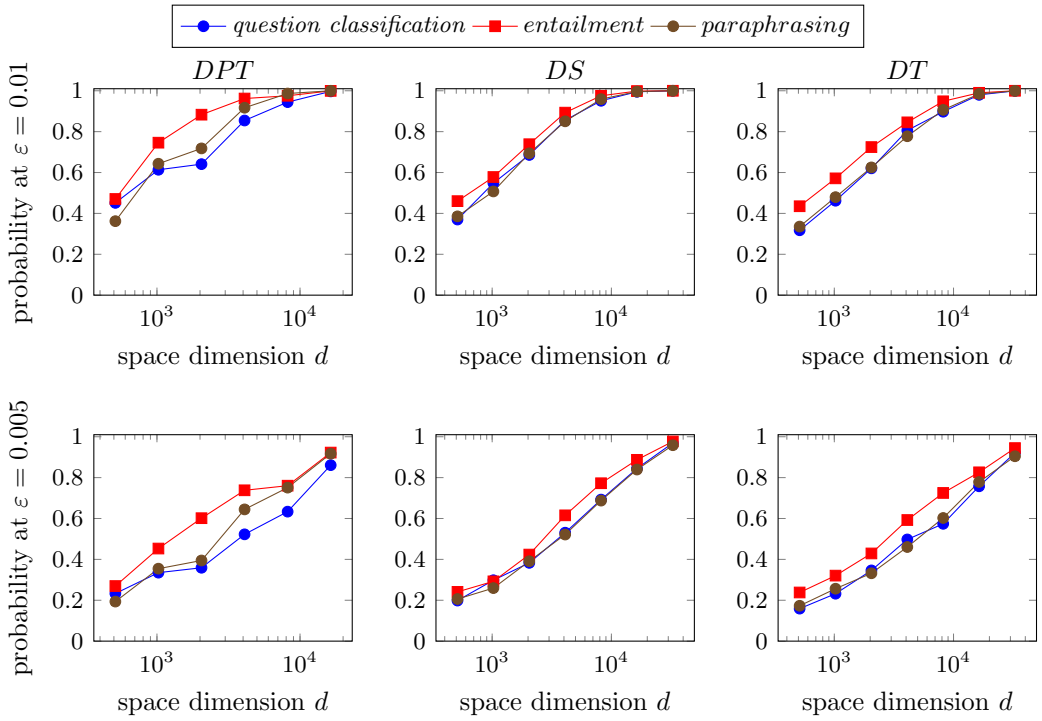

Figure 6: Estimated probabilities for Lemma 2 ($\varepsilon = 0.01$ and $\varepsilon = 0.005$) with respect to different sizes of the embedding space $\mathbb{R}^d$. Considered distributed structures: Distributed Trees (DT) with $\lambda = 0.4$, Distributed Sequences (DS) with $\lambda = 0.4$ and sequence boud $= 4$ and Distributed Partial Trees (DPT) with $\lambda = 0.4$ and $\mu = 0.4$. Datasets: question classification, entailment and paraphrasing. Estimation done over 2,000 pairs of structures.

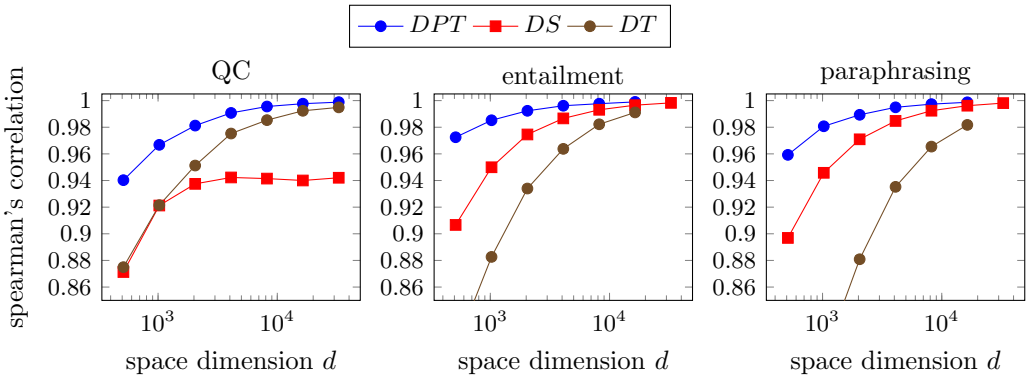

Figure 7: Spearman's correlations between distributed convolution kernels and convolution kernels in the configurations applied for the 3 tasks: question classification, entailment and paraphrasing. Compared kernels: PTK vs. DPTK, SK vs. DSK, TK vs. DTK.

	entailment			*paraphrasing*			*question classification*		
	DC-ll	*DC-pa*	*CK-SVM*	*DC-ll*	*DC-pa*	*CK-SVM*	*DC-ll*	*DC-pa*	*CK-SVM*
S	0.529	0.517	0.515	0.663	0.639	0.712	0.858	0.868	0.878
T	0.601	0.594	0.65	0.687	0.673	0.733	0.868	0.846	0.908
PT	0.642	0.632	0.661	0.738	0.724	0.726	0.866	0.89	0.9

Table 2: Accuracies of SVM with convolution kernels and linear machines with distributed convolution structures in $\mathbb{R}^d$ with $d = 16384$. Used linear machines: LIBLINEAR (*ll*) and Passive-Aggressive (*pa*). Used distributed convolution structures: *DS*, *DT* and *DPT*. Used datasets: question classification, entailment and paraphrasing. Dimension of Distributed Representation is $d = 16384$

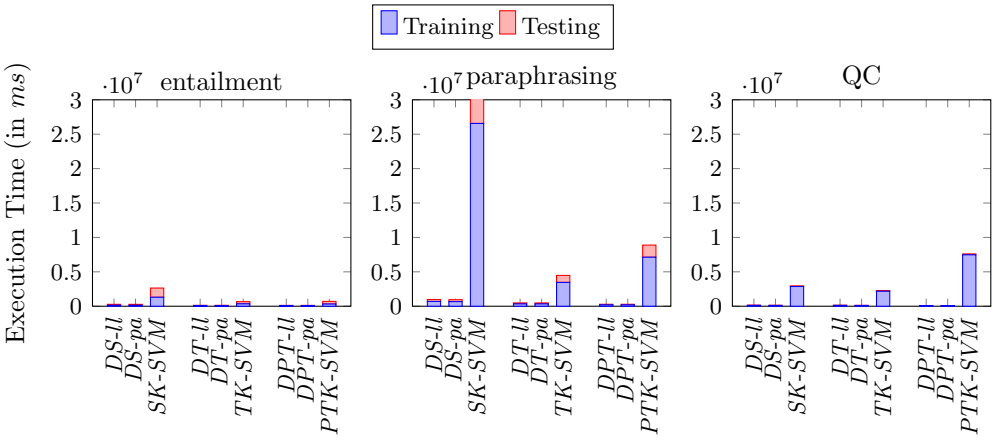

Figure 8: Execution times in *ms* for SVM with convolution kernels and linear machines with distributed convolution structures in $\mathbb{R}^d$ with $d = 16384$. Used linear machines: LIBLINEAR (*ll*) and Passive-Aggressive (*pa*). Used distributed convolution structures: *DS*, *DT* and *DPT*. Used datasets: question classification, entailment and paraphrasing.

For QC, we used a standard question classification training and test set[3] (Li & Roth, 2002), where the test set is the 500 TREC 2001 test questions. To measure the task performance, we used a question multi-classifier by combining n binary SVMs according to the ONE-vs-ALL scheme, where the final output class is the one associated with the most probable prediction. We used the classification task based on the coarse-grain labels. The trees in the set contain 9817 node labels, including 67 non-terminal labels (POS-tags) and 9750 terminal labels (words). The trees have an average number of 30 nodes, with up to 115 nodes for the largest ones. The maximum branching factor in the trees is 18, and the maximum tree depth is 20. Sentences in the set are 10 tokens long on average, with a maximum of 37.

For RTE we considered the corpora ranging from the first challenge to the fifth (Dagan et al., 2006), except for the fourth, which has no training set. These sets are referred to as RTE1-5. The dev/test distribution for RTE1-3, and RTE5 is respectively 567/800, 800/800, 800/800, and 600/600 T-H pairs. We used these sets for the traditional task of pair-based entailment recognition, where a pair of text-hypothesis $p = (t, h)$ is assigned a positive or negative entailment class. The trees in the set contain 30965 node labels, including 74 non-terminal labels and 31039 terminal labels. The trees are composed of 79 nodes on average, with up to 736 nodes for the largest ones. The maximum branching factor in the trees is 47, and the maximum tree depth is 60. Sentences in the set are 28 tokens long on average, with a maximum of 260.

As final specification of the experiment setting, we give the list of the external resources and tools used. For the syntactic interpretation we used the Charniak's parser (Charniak, 2000).

5.2 Experimental Evaluation

This section reports the results. The first set of experiments is the direct evaluation: we compared the similarity produced with the DCKs with those produced with the CKs. The second set of experiments is a task-based evaluation where we investigated the behavior of the DCKs in the two NLP tasks.

6. Discussion and Conclusions

Distributed convolution kernels on countable sets are an innovative way of representing structured data in machine learning as these kernels well approximate convolution kernels on countable sets. The major computational benefit is obtained when using these kernels in combination with linearized versions of kernel machines such as linear SVM (Joachims, 2006; Shalev-Shwartz et al., 2011). We empirically prove that DCKs well approximate CKs with the λ penalizing factor in the range $\lambda \leq 0.4$. These values of λ are those generally used when exploiting convolution kernels in tasks.

We presented the distributed convolution kernels only from a point of view, that is a way to approximate the original convolution kernel in order to reduce the computational complexity of learning and classification. As we will see later in this section, this is not the only important characteristic of DCKs but, as such, DCKs are in competition with two other

3. The QC set is available at `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

families of approaches working on reducing the computational complexity. A *first family* of approaches is the one that aims to reduce or to control the computational complexity of kernel computation. For example in the case of the tree kernels (Collins & Duffy, 2002), there is a line of research that aims to build more efficient methods by exploiting characteristics of trees in specific applications (Moschitti, 2006b) or by performing a feature selection (in the kernel computation (Rieck et al., 2010) or directly on the trees (Pighin & Moschitti, 2010)). But, these approaches do not reduce the n^3 factor in the overall complexities (where n is the number of training examples). Thus, for all these cases, DCKs with linear SVMs are still better than the others. A *second family* instead works on the approximation of the Gram matrix by using methods like the Nyström method (Williams & Seeger, 2000) or the incomplete Cholesky decomposition (Fine & Scheinberg, 2002; Bach & Jordan, 2003) and those proposed in this line of research. By working on reduced-rank approximations of the Gram matrix, these models bound the learning complexity to $O(nm^2)$ where m is the rank of the approximation. These methods are competitive with DCKs. For example, in the case of the tree kernels, the two complexities of SVM with TKs and the approximation of thee Gram matrix vs. the linear SVM with DTKs are comparable, that is, respectively, $O(nm^2|N(t)|^2)$ vs. $O(n|N(t)|d\log d)$. With respect to this second family, DCKs seem to be only another, although different, way of approximating kernel functions for the purpose of reducing the computational complexity.

Distributed convolution kernels also have another important characteristics that is not treated in this paper that makes them totally different from approaches approximating the Gram matrix (Williams & Seeger, 2000; Fine & Scheinberg, 2002; Bach & Jordan, 2003). DCKs with their distributed convolution structures give also the possibility of encoding structures in small vectors. This is extremely relevant in an emerging field in natural language processing, namely, the compositional distributional semantics (Mitchell & Lapata, 2008; Baroni & Zamparelli, 2010; Zanzotto, Korkontzelos, Fallucchi, & Manandhar, 2010) and the strictly related work on recursive autoencoders (Socher, Huang, Pennington, Ng, & Manning, 2011; Socher, Huval, Manning, & Ng, 2012). In this area, distributional (and not distributed) vectors representing the meaning of words are composed to obtain the meaning of word sequences or full sentences. Except for those methods encoding distributional semantics in tensors (Clark, Coecke, & Sadrzadeh, 2008; Grefenstette & Sadrzadeh, 2011), compositional distributional semantics methods use structural information to derive the meaning of word sequences but the structural information is lost in the final vector. Distributed convolution structures represent a way to fill this gap as showing a way to encode the lost structural information. How to encode structures along with distributional meaning is still matter for research. Simple attempts fail (see (Zanzotto & Dell'Arciprete, 2011)). The real research issue is how to scale from *distributed convolution kernels on countable sets* to *distributed convolution kernels* by removing the restriction *on countable sets*.

As a final remark, DCKs on countable sets are not bounded to support vector machines and not even to kernel machines. DCKs could be used in other kernel machines such as on-line learning models by helping in using unbounded on-line learning models (Cavallanti, Cesa-Bianchi, & Gentile, 2007) even when dealing with structured data without any need to go for bounded on-line learning models that select and discard vectors for memory constraint or time complexity (Cavallanti et al., 2007; Dekel, Shalev-Shwartz, & Singer, 2005; Orabona, Keshet, & Caputo, 2008). DCKs with the associated distributed convolution structures

also give the possibility to fully use structured data in non kernel machines, for example, probabilistic classifiers such as naive bayes or maximun entropy classifiers as well as decision tree learners (Quinlan, 1993). Results on pilot experiments show that this is a viable possibility.

Appendix A. A Map of the Symbols

Symbol type	*Description of its use*
X, Y, A, B, S, ...	sets containing structures
x, y, a, b, ...	a complete structure (eventually a substructure of a bigger structure)
$x_i \in X_i$	the i-th part of the structure x and X_i is the related set of structures
$\overline{\mathbf{x}} = (x_1, ..., x_M) \in X_1 \times \ldots \times X_M$	a decomposition into parts of a structure x
$\lvert\overline{\mathbf{x}}\rvert = M$	it is the number of parts the structure x is decomposed into
$R(x)$	the set of all the possible decomposition into parts of x
$R_i(x)$	as above, but it denotes a specific way for extracting the possible decomposition into parts
$S(x)$	the set of all the possible substructures of x
$S_i(x)$	as above, but it denotes a specific way for extracting the possible substructures of x
$\mathbb{S}(\overline{\mathbf{x}})$	$S_1(x_1) \times \ldots \times S_M(x_M)$
$S(x) = \{a \lvert \overline{\mathbf{x}} \in R(x), \overline{\mathbf{a}} \in \mathbb{S}(\overline{\mathbf{x}})\}$	
$K(x, y)$	a kernel function between x and y
$K_i(x, y)$	a specific kernel function between x and y
$D(x)$	a distributed convolution function on a structure x
$D_i(x)$	a specific distributed convolution function on a structure x
$\phi(x)$	the base vector in $\mathbb{R}^m$ representing the substructure x
$\delta(x, y)$	the Kroneker delta between x and y
ω_x	the weight assigned to a structure x or the value assigned to the feature x
$\mathbf{x} \in \mathbb{R}^d$	the vector representing its distributed representation of a structure x in the reduced space $\mathbb{R}^d$
$\mathcal{S}, \mathcal{S}', \ldots$	discrete subsets of $\mathbb{R}^d$
$MOV(\varepsilon, \theta)$	a sets of nearly orthorormal vectors with the Properties 2 and 3 defined in Section 2.2.2
$\Upsilon(x)$	the function that maps a structure x in the vector $\mathbf{x}$ representing its distributed representation
$\upsilon(x)$	the basic random indexing function that maps a final object in its random vector
$N(t)$	the set of nodes of a given tree t
$C(s)$	the set of characters of a given sequence s
$f_{s,t}$	the frequency of the substructure s in the structure t
$\mathbf{x} \odot \mathbf{y}$	Ideal basic binary operator on vectors
$t_a(\mathbf{x})$	a transformation function that permutes the vector components where a is a parameter of the permutation
$\mathbf{x} \circ \mathbf{y}$	Real (but generic) basic binary operator without permutation
$\mathbf{x} \circledcirc \mathbf{y} = t_a(x) \circ t_b(y)$	Real (but generic) basic binary operator with permutation
$\mathbf{x} \times \mathbf{y}$	γ-element-wise Shur product
$\mathbf{x} \otimes \mathbf{y} = t_a(x) \times t_b(y)$	transformed γ-element-wise Shur product (the selected transformation is the random shuffling after Section **??**)
$\mathbf{x} * \mathbf{y}$	circular convolution
$\mathbf{x} \circledast \mathbf{y} = t_a(x) * t_b(y)$	transformed circular convolution (the selected transformation is the random shuffling after Section **??**)

Appendix B.

Lemma 6 *Given the subset* $V_1 = \{v | v \in \{0,1\}^m$ *and* $||v|| = 1\}$ *such that* $|V_1| = m$*, there exists a function* $f : \mathbb{R}^m \to \mathbb{R}^d$ *such that, for any* v_a *and* v_b *in* V_1*, these two properties hold:*

$$P(1-\epsilon < ||f(v_a)||^2 < 1-\epsilon) > 1-\theta$$
$$P(|f(v_a) \cdot f(v_b)| < 1.5\epsilon) > (1-\theta)^3$$

and the lower-bound of the dimension d *of the space* $\mathbb{R}^d$ *is* $\Omega(\epsilon^{-2} \log m \log 1/\theta)$*.*

Proof: Theorem 4.3 in (Jayram & Woodruff, 2011) proves that there exists a function $f : \mathbb{R}^m \to \mathbb{R}^d$ such that, for any vectors v' and v'' in $\{0,1\}^m$, these two properties hold:

$$P(||v'||^2 - \epsilon < ||f(v')||^2 < ||v'||^2 + \epsilon) > 1-\theta$$
$$P(||v'-v''||^2 - \epsilon < ||f(v') - f(v'')||^2 < ||v'-v''||^2 + \epsilon) > 1-\theta$$

and the lower-bound of the dimension d of the space $\mathbb{R}^d$ is $\Omega(\epsilon^{-2} \log m \log 1/\theta)$. Thus, for vectors $v_a, v_b \in V_1 \subset \{0,1\}^m$, the above properties hold. v_a and v_b are unit vectors and orthogonal. Thus:

$$P(1-\epsilon < ||f(v_a)||^2 < 1+\epsilon) > 1-\theta$$
$$P(1-\epsilon < ||f(v_b)||^2 < 1+\epsilon) > 1-\theta$$

Observing that $||v_a - v_b||^2 = ||v_a||^2 + ||v_b||^2 - 2v_a \cdot v_b = 2$ and that $||f(v_a) - f(v_b)||^2 = ||f(v_a)||^2 + ||f(v_b)||^2 - 2f(v_a) \cdot f(v_b)$, we have that:

$$P(||v_a - v_b||^2 - \epsilon < ||f(v_a) - f(v_b)||^2 < ||v_a - v_b||^2 + \epsilon) > 1-\theta$$

becomes:

$$P(2-\epsilon < ||f(v_a)||^2 + ||f(v_b)||^2 - 2f(v_a) \cdot f(v_b) < 2+\epsilon) > 1-\theta$$

By combining the three events in a joint event and by considering these events as independent events, we have:

$$P(|f(v_a) \cdot f(v_b)| < 3/2\epsilon) > (1-\theta)^3$$

□

$$\Omega(\overline{\epsilon}^{-2} \log m \log 1/\overline{\theta})$$

where: $\varepsilon = 1.5\overline{\epsilon}$ and $\theta = \overline{\theta}^3 - 3\overline{\theta}^2 + 3\overline{\theta}$. that is:

$$\overline{\theta} = 1 + \sqrt[3]{\theta - 1}$$

Appendix C. Theorems 4.2 and 4.3 of (Jayram & Woodruff, 2011)

Theorem 2 (Theorem 4.2.) *The* ***one-way communication complexity*** *of the problem of approximating the* $\|\cdot\|_p$ *difference of two vectors of length* n *to within a factor* $1+\varepsilon$ *with failure probability at most* δ *is* $\Omega(\varepsilon^{-2} \log n \log(1/\delta))$

The **one-way communication complexity** of Q is the minimum **communication cost** of a protocol for Q with failure probability at most δ.

Communication cost of the protocol P: The maximum length of Alice's message (in bits) over all inputs.

Appendix D. Counting nearly-orthogonal unit vectors in $\mathbb{R}^d$

Theorem 3 (Johnson-Lindenstrauss Lemma) *For any $0 < \epsilon < 1$ and any integer m. Let d be a positive integer such that*

$$d = O(\epsilon^{-2} \log m)$$

Then for any set V of m points in $\mathbb{R}^k$, there is a map $f : \mathbb{R}^k \to \mathbb{R}^d$ such that for all $\mathbf{u}, \mathbf{v} \in V$,

$$(1 - \epsilon)\|\mathbf{u} - \mathbf{v}\|_2^2 \leq \|f(\mathbf{u}) - f(\mathbf{v})\|_2^2 \leq (1 + \epsilon)\|\mathbf{u} - \mathbf{v}\|_2^2.$$

The theorem can be derived using the following lemma:

Lemma 7 *For any $\epsilon > 0$, $\tau < 1/2$ and positive integer d, there exists a distribution $\mathcal{D}$ over $\mathbb{R}^{d \times k}$ for $d = O(\epsilon^{-2} \log 1/\tau)$ such that, for any $\mathbf{x} \in \mathbb{R}^k$ with $||\mathbf{x}||_2 = 1$,*

$$\mathbb{P}(|\|A\mathbf{x}\|_2^2 - 1| > \epsilon) < \tau$$

by choosing $\tau = 1/m^2$ and by applying the union bound on the vectors $(\mathbf{u} - \mathbf{v})/\|\mathbf{u} - \mathbf{v}\|_2$ for all vectors $\mathbf{u}$ and $\mathbf{v}$ in V. It is possible to demonstrate that there is a probability strictly greater than 0 that a function f exists.

Now we can demonstrate that the following lemma holds:

Corollary 4 *For any $0 < \epsilon < 1$ and any integer m. Let d be a positive integer such that*

$$d = O(\epsilon^{-2} \log m)$$

Then given the standard basis E of $\mathbb{R}^m$, there is a map $f : \mathbb{R}^m \to \mathbb{R}^d$ such that for all $\mathbf{e}_i, \mathbf{e}_j \in E$,

$$\mathbb{P}(1 - \epsilon < \|f(\mathbf{e}_i)\|_2^2 < 1 + \epsilon) > 1 - \tau = 1 - 1/m^2 \tag{10}$$

and

$$\mathbb{P}(|f(\mathbf{e}_i)f(\mathbf{e}_j)| < 2\epsilon) > (1 - \tau)^2 = (1 - 1/m^2)^2 \tag{11}$$

Proof: Equation (10) derives from lemma 7 as $\mathbf{e}_i \in E$ are unitary, that is, $\|\mathbf{e}_i\|_2 = 1$ as $\tau = 1/m^2$.

To prove Equation (11), first, we can observe that $||\mathbf{e}_i - \mathbf{e}_j||^2 = ||\mathbf{e}_i||^2 + ||\mathbf{e}_j||^2 - 2\mathbf{e}_i\mathbf{e}_j = 2$ as $\mathbf{e}_i$ and $\mathbf{e}_j$ are unitary and orthogonal. Then, we can see that $||f(\mathbf{e}_i) - f(\mathbf{e}_j)||^2 = ||f(\mathbf{e}_i)||^2 + ||f(\mathbf{e}_j)||^2 - 2f(\mathbf{e}_i)f(\mathbf{e}_j)$. With Theorem 3, the following holds:

$$2(1 - \epsilon) \leq ||f(\mathbf{e}_i)||^2 + ||f(\mathbf{e}_j)||^2 - 2f(\mathbf{e}_i)f(\mathbf{e}_j) \leq 2(1 + \epsilon)$$

Hence:

$$||f(\mathbf{e}_i)||^2 + ||f(\mathbf{e}_j)||^2 - 2 - 2\epsilon \leq 2f(\mathbf{e}_i)f(\mathbf{e}_j) \leq ||f(\mathbf{e}_i)||^2 + ||f(\mathbf{e}_j)||^2 - 2 + 2\epsilon$$

Thus, using Equation (10) on the two independent events $f(\mathbf{e}_i)$ and $f(\mathbf{e}_j)$:

$$\mathbb{P}(2 - 2\epsilon - 2 - 2\epsilon \leq 2f(\mathbf{e}_i)f(\mathbf{e}_j) \leq 2 + 2\epsilon - 2 + 2\epsilon) = \mathbb{P}(|f(\mathbf{e}_i)f(\mathbf{e}_j| < 2\epsilon) > (1 - \tau)^2$$

□

Putting together Equation (10) and Equation (11), it is possible to derive a set $NOV(\varepsilon, \theta)$ of m nearly-orthogonal unit vectors such that for each $\mathbf{a}, \mathbf{b} \in NOV(\varepsilon, \theta)$:

$$\mathbb{P}(\delta(\mathbf{a}, \mathbf{b}) - \varepsilon \leq \langle \mathbf{a}, \mathbf{b} \rangle \leq \delta(\mathbf{a}, \mathbf{b}) + \varepsilon) > 1 - \theta$$

by choosing $\varepsilon = 2\epsilon$, a space $\mathbb{R}^d$ with $d = O(\varepsilon^{-2} \log m)$ and $\theta = 2/m^2 - 1/m^4$.

Appendix E. Properties of d-dimensional Gaussian Vectors $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$

In the rest of the paper, the property for vectors to belong to $NOV(\varepsilon, \theta)$ (Equation (**??**)) is also referred as the following two properties:

Property 2 (Nearly Unit Vectors) *A vector* $\mathbf{a} \in NOV(\varepsilon, \theta)$ *is a nearly unit vector, that is:*

$$\mathbb{P}(1 - \epsilon \leq ||\mathbf{a}||_2^2 \leq 1 + \epsilon) \geq 1 - \theta$$

where $||\mathbf{a}||_2$ *is the* ℓ^2 *norm (Euclidean norm)*

Property 3 (Nearly Orthogonal Vectors) *Two different vectors* $\mathbf{a}, \mathbf{b} \in NOV(\varepsilon, \theta)$ *are nearly orthogonal if:*

$$\mathbb{P}(|\langle \mathbf{a}, \mathbf{b} \rangle| \leq \epsilon) \geq 1 - \theta$$

Taking into account the potentially infinite dimensional spaces $\mathcal{H}$ of substructures, our method uses directly $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ as $NOV(\varepsilon, \theta)$. In this way, we can extract potentially infinite vectors that pairwise respect Equation **??**. We showed (see Appendix E) that vectors $\mathbf{a}, \mathbf{b} \sim \mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ have the following distributions for Properties 2 and 3 splitting Equation **??**:

For Property 2, $\mathbb{P}(1 - \varepsilon < \|\mathbf{a}\|^2 < 1 + \varepsilon) = \frac{1}{\Gamma\left(\frac{d}{2}\right)}\gamma\left(\frac{d}{2}, \frac{d(1+\varepsilon)}{2}\right) - \frac{1}{\Gamma\left(\frac{d}{2}\right)}\gamma\left(\frac{d}{2}, \frac{d(1-\varepsilon)}{2}\right)$ where $\Gamma(\cdot)$ is the gamma function and $\gamma(\cdot, \cdot)$ is the lower incomplete Gamma function (Arfken, 1985)

For Property 3, $\mathbb{P}(|\langle \mathbf{a}, \mathbf{b} \rangle| \leq \varepsilon) = \int_{-\varepsilon}^{\varepsilon} p_Z\left(x; d, \sigma\right) \mathrm{d}x$ with: $p_Z(t; d) = \frac{2^{\frac{1}{2}-\frac{d}{2}} d^{\frac{d+1}{4}} |x|^{\frac{d-1}{2}} K_{\frac{d-1}{2}}\left(\sqrt{d}|x|\right)}{\sqrt{\pi}\Gamma\left(\frac{d}{2}\right)}$ where $K_\alpha(x)$ is the modified Bessel function of the second kind.

These distributions show that vectors drawn from $\mathcal{N}(0, \frac{1}{d}\mathbb{I}_d)$ with $d = 4096$ and $d = 8192$ have a very high probability of respecting the two properties with $\varepsilon = 0.1$, that is, $\theta \propto 10^{-xxx}$ and $\theta \propto 10^{-yyy}$, respectively. With these θ, it is possible to approximate convolution kernels with dot products of distributed convolution structures (cf. Equation **??**).

Appendix F. Code

The code of the distributed convolution kernels is available at:

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

References

Aiolli, F., Da San Martino, G., & Sperduti, A. (2009). Route kernels for trees. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pp. 17–24, New York, NY, USA. ACM.

Arfken, G. (1985). The incomplete gamma function and related functions. *Mathematical Methods for Physicists.*

Bach, F. R., & Jordan, M. I. (2003). Kernel independent component analysis. *J. Mach. Learn. Res.*, *3*, 1–48.

Baroni, M., & Zamparelli, R. (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*, pp. 1183–1193, Cambridge, MA. Association for Computational Linguistics.

Cavallanti, G., Cesa-Bianchi, N., & Gentile, C. (2007). Tracking the best hyperplane with a simple budget perceptron. *Mach. Learn.*, *69*(2-3), 143–167.

Charniak, E. (2000). A maximum-entropy-inspired parser. In *Proc. of the 1st NAACL*, pp. 132–139. Seattle, Washington.

Clark, S., Coecke, B., & Sadrzadeh, M. (2008). A compositional distributional model of meaning. *Proceedings of the Second Symposium on Quantum Interaction (QI-2008)*, 133–140.

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In *NIPS*, pp. 625–632.

Collins, M., & Duffy, N. (2002). New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of ACL02*.

Cristianini, N., & Shawe-Taylor, J. (2000). *An Introduction to Support Vector Machines and Other Kernel-based Learning Methods*. Cambridge University Press.

Croce, D., Moschitti, A., & Basili, R. (2011). Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pp. 1034–1046, Stroudsburg, PA, USA. Association for Computational Linguistics.

Dagan, I., Glickman, O., & Magnini, B. (2006). The pascal recognising textual entailment challenge. In et al., Q.-C. (Ed.), *LNAI 3944: MLCW 2005*, pp. 177–190. Springer-Verlag, Milan, Italy.

Dasgupta, S., & Gupta, A. (1999). An elementary proof of the johnson-linderstrauss lemma. Tech. rep. TR-99-006, ICSI, Berkeley, California.

Dekel, O., Shalev-Shwartz, S., & Singer, Y. (2005). The forgetron: A kernel-based perceptron on a fixed budget. In *NIPS*.

Even-Zohar, Y., & Roth, D. (2000). A classification approach to word prediction. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, NAACL 2000, pp. 124–131, Stroudsburg, PA, USA. Association for Computational Linguistics.

Fine, S., & Scheinberg, K. (2002). Efficient svm training using low-rank kernel representations. *J. Mach. Learn. Res.*, *2*, 243–264.

Gildea, D., & Jurafsky, D. (2002). Automatic Labeling of Semantic Roles. *Computational Linguistics*, *28*(3), 245–288.

Grefenstette, E., & Sadrzadeh, M. (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, pp. 1394–1404, Stroudsburg, PA, USA. Association for Computational Linguistics.

Haussler, D. (1999). Convolution kernels on discrete structures. Tech. rep., University of California at Santa Cruz.

Indyk, P., & Motwani, R. (1998). Approximate nearest neighbors: Towards removing the curse of dimensionality. In *Proceedings of the Thirtieth Annual ACM Symposium on Theory of Computing*, STOC '98, pp. 604–613, New York, NY, USA. ACM.

Jayram, T. S., & Woodruff, D. (2011). Optimal bounds for johnson-lindenstrauss transforms and streaming problems with sub-constant error. In *Proceedings of the Twenty-Second Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '11, pp. 1–10. SIAM.

Joachims, T. (2006). Training linear svms in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, KDD '06, pp. 217–226, New York, NY, USA. ACM.

Johnson, W., & Lindenstrauss, J. (1984). Extensions of lipschitz mappings into a hilbert space. *Contemp. Math.*, *26*, 189–206.

Kashima, H., & Koyanagi, T. (2002). Kernels for semi-structured data. In *Proceedings of the Nineteenth International Conference on Machine Learning*, ICML '02, pp. 291–298, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Kimura, D., Kuboyama, T., Shibuya, T., & Kashima, H. (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, pp. 62–74, Berlin, Heidelberg. Springer-Verlag.

Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics - Volume 1*, COLING '02, pp. 1–7, Stroudsburg, PA, USA. Association for Computational Linguistics.

Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. *J. Mach. Learn. Res.*, *2*, 419–444.

Mehdad, Y., Moschitti, A., & Zanzotto, F. M. (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, pp. 1020–1028, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mitchell, J., & Lapata, M. (2008). Vector-based models of semantic composition. In *Proceedings of ACL-08: HLT*, pp. 236–244, Columbus, Ohio. Association for Computational Linguistics.

Moschitti, A. (2006a). Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of The 17th European Conference on Machine Learning*, Berlin, Germany.

Moschitti, A. (2006b). Making tree kernels practical for natural language learning. In *Proceedings of EACL'06*, Trento, Italy.

Orabona, F., Keshet, J., & Caputo, B. (2008). The projectron: a bounded kernel-based perceptron. In *Proceedings of the 25th international conference on Machine learning*, ICML '08, pp. 720–727, New York, NY, USA. ACM.

Pighin, D., & Moschitti, A. (2010). On reverse feature engineering of syntactic tree kernels. In *Conference on Natural Language Learning (CoNLL-2010)*, Uppsala, Sweden.

Plate, T. A. (1995). Holographic reduced representations. *IEEE Transactions on Neural Networks*, *6*(3), 623–641.

Quinlan, J. (1993). *C4:5:programs for Machine Learning*. Morgan Kaufmann, San Mateo.

Rieck, K., Krueger, T., Brefeld, U., & Müller, K.-R. (2010). Approximate tree kernels. *J. Mach. Learn. Res.*, *11*, 555–580.

Royston, J. P. (1983). Some techniques for assessing multivarate normality based on the shapiro- wilk w. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, *32*(2), 121–133.

Sahlgren, M., Holst, A., & Kanerva, P. (2008). Permutations as a means to encode order in word space. In Sloutsky, V., Love, B., & Mcrae, K. (Eds.), *Proceedings of the 30th Annual Conference of the Cognitive Science Society*, pp. 1300–1305. Cognitive Science Society, Austin, TX.

Shalev-Shwartz, S., Singer, Y., Srebro, N., & Cotter, A. (2011). Pegasos: primal estimated sub-gradient solver for svm. *Math. Program.*, *127*(1), 3–30.

Shin, K., Cuturi, M., & Kuboyama, T. (2011). Mapping kernels for trees. In Getoor, L., & Scheffer, T. (Eds.), *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, ICML '11, pp. 961–968, New York, NY, USA. ACM.

Socher, R., Huang, E. H., Pennington, J., Ng, A. Y., & Manning, C. D. (2011). Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems 24*.

Socher, R., Huval, B., Manning, C. D., & Ng, A. Y. (2012). Semantic Compositionality Through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Steinwart, I. (2004). Sparseness of support vector machines—some asymptotically sharp bounds. In Thrun, S., Saul, L., & Schölkopf, B. (Eds.), *Advances in Neural Information Processing Systems 16*. MIT Press, Cambridge, MA.

Williams, C. K. I., & Seeger, M. (2000). Using the nyström method to speed up kernel machines. In Leen, T. K., Dietterich, T. G., & Tresp, V. (Eds.), *NIPS*, pp. 682–688. MIT Press.

Zanzotto, F. M., & Dell'Arciprete, L. (2011). Distributed structures and distributional meaning. In *Proceedings of the Workshop on Distributional Semantics and Compositionality*, pp. 10–15, Portland, Oregon, USA. Association for Computational Linguistics.

Zanzotto, F. M., Korkontzelos, I., Fallucchi, F., & Manandhar, S. (2010). Estimating linear models for compositional distributional semantics. In *Proceedings of the 23rd International Conference on Computational Linguistics (COLING)*.

Zanzotto, F., & Dell'Arciprete, L. (2012). Distributed tree kernels. In *Proceedings of International Conference on Machine Learning*, pp. 193–200.