# Deep Learning in Semantic Kernel Spaces

**Danilo Croce** **Simone Filice** **Giuseppe Castellucci** **Roberto Basili**
Department of Enterprise Engineering
University of Roma Tor Vergata,
Via del Politecnico 1, 00133, Rome, Italy
{croce,filice,basili}@info.uniroma2.it
castellucci@ing.uniroma2.it

## Abstract

Kernel methods enable the direct usage of structured representations of textual data during language learning and inference tasks. Expressive kernels, such as Tree Kernels, achieve excellent performance in NLP. On the other side, deep neural networks have been demonstrated effective in automatically learning feature representations during training. However, their input is tensor data, i.e., they cannot manage rich structured information. In this paper, we show that expressive kernels and deep neural networks can be combined in a common framework in order to (*i*) explicitly model structured information and (*ii*) learn non-linear decision functions. We show that the input layer of a deep architecture can be pre-trained through the application of the Nyström low-rank approximation of kernel spaces. The resulting "kernelized" neural network achieves state-of-the-art accuracy in three different tasks.

## 1 Introduction

Learning for Natural Language Processing (NLP) requires to more or less explicitly account for trees or graphs to express syntactic and semantic information. A straightforward modeling of such information has been obtained in statistical language learning with Tree Kernels (TKs) (Collins and Duffy, 2001), or by means of structured neural models (Hochreiter and Schmidhuber, 1997; Socher et al., 2013). In particular, kernel-based methods (Shawe-Taylor and Cristianini, 2004) have been largely applied in language processing for alleviating the need of complex activities of manual feature engineering (e.g., (Moschitti et al., 2008)). Although *ad-hoc* features are adopted by many successful approaches to language learning (e.g., (Gildea and Jurafsky, 2002)), kernels provide a natural way to capture textual generalizations directly operating over (possibly complex) linguistic structures. Sequence (Cancedda et al., 2003) or tree kernels (Collins and Duffy, 2001) are of particular interest as the feature space they implicitly generate reflects linguistic patterns. On the other hand, Recursive Neural Networks (Socher et al., 2013) have been shown to learn dense feature representations of the nodes in a structure, thus exploiting similarities between nodes and sub-trees. Also, Long-Short Term Memory (Hochreiter and Schmidhuber, 1997) networks build intermediate representations of sequences, resulting in similarity estimates over sequences and their inner sub-sequences.

While such methods are highly effective and reach state-of-the-art results in many tasks, their adoption can be problematic. In kernel-based Support Vector Machine (SVM) the classification model corresponds to the set of support vectors (SVs) and weights justifying the maximal margin hyperplane: the classification cost crucially depends on their number, as classifying a new instance requires a kernel computation against all SVs, making their adoption in large data settings prohibitive. This scalability issue is evident in many NLP and Information Retrieval applications, such as in answer re-ranking in question answering (Severyn et al., 2013; Filice et al., 2016), where the number of SVs is typically very large. Improving the efficiency of kernel-based methods is a largely studied topic. The reduction of computational costs has been early designed by imposing a budget (Dekel and Singer, 2006; Wang and Vucetic, 2010), that is limiting the maximum number of SVs in a model. However, in complex tasks, such methods still require large budgets to reach



*Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 345–354
Vancouver, Canada, July 30 - August 4, 2017. 
https://doi.org/10.18653/v1/P17-1032

adequate accuracies. On the other hand, training complex neural networks is also difficult as no common design practice is established against complex data structures. In Levy et al. (2015), a careful analysis of neural word embedding models is carried out and the role of the hyper-parameter estimation is outlined. Different neural architectures result in the same performances, whenever optimal hyper-parameter tuning is applied. In this latter case, no significant difference is observed across different architectures, making the choice between different neural architectures a complex and empirical task.

A general approach to the large scale modeling of complex structures is a critical and open problem. A viable and general solution to this scalability issue is provided by the Nyström method (Williams and Seeger, 2001); it allows to approximate the Gram matrix of a kernel function and support the embedding of future input examples into a low-dimensional space. For example, if used over TKs, the Nyström projection corresponds to the embedding of any tree into a low-dimensional vector.

In this paper, we show that the Nyström based low-rank embedding of input examples can be used as the early layer of a deep feed-forward neural network. A standard NN back-propagation training can thus be applied to induce non-linear functions in the kernel space. The resulting deep architecture, called *Kernel-based Deep Architecture* (KDA), is a mathematically justified integration of expressive kernel functions and deep neural architectures, with several advantages: it (*i*) directly operates over complex non-tensor structures, e.g., trees, without any manual feature or architectural engineering, (*ii*) achieves a drastic reduction of the computational cost w.r.t. pure kernel methods, and (*iii*) exploits the non-linearity of NNs to produce accurate models. The experimental evaluation shows that the proposed approach achieves state-of-the-art results in three semantic inference tasks: Semantic Parsing, Question Classification and Community Question Answering.

In the rest of the paper, Section 2 surveys some of the investigated kernels. In Section 3 the Nyström methodology and KDA are presented. Experimental evaluations are described in Section 4. Finally, Section 5 derives the conclusions.

## 2 Kernel-based Semantic Inference

In almost all NLP tasks, explicit models of complex syntactic and semantic structures are required, such as in Paraphrase Detection: deciding whether two sentences are valid paraphrases involves learning grammatical rewriting rules, such as semantics preserving mappings among subtrees. Also in Question Answering, the syntactic information about input questions is crucial. While manual feature engineering is always possible, kernel methods on structured representations of data objects, e.g., sentences, have been largely applied. Since Collins and Duffy (2001), sentences can be modeled through their corresponding parse tree, and Tree Kernels (TKs) result in similarity metrics directly operating over tree fragments. Such kernels corresponds to dot products in the (implicit) feature space made of all possible tree fragments (Haussler, 1999). Notice that the number of tree fragments in a tree bank is combinatorial with the number of tree nodes and gives rise to billions of features, i.e., dimensions. In this high-dimensional space, kernel-based algorithms, such as SVMs, can implicitly learn robust prediction models (Shawe-Taylor and Cristianini, 2004), resulting in state-of-the-art approaches in several NLP tasks, e.g., Semantic Role Labeling (Moschitti et al., 2008), Question Classification (Croce et al., 2011) or Paraphrase Identification (Filice et al., 2015). As the feature space generated by the structural kernels depends on the input structures, different tree representations can be adopted to reflect more or less expressive syntactic/semantic feature spaces. While constituency parse trees have been early used (e.g., (Collins and Duffy, 2001)), dependency parse trees correspond to graph structures. TKs usually rely on their tree conversions, where grammatical edge labels corresponds to nodes. An expressive tree representation of dependency graphs is the Grammatical Relation Centered Tree (GRCT). As illustrated in Figure 1, PoS-Tags and grammatical functions correspond to nodes, dominating their associated lexicals.

**Types of tree kernels**. While a variety of TK functions have been studied, e.g., the *Partial Tree Kernel* (PTK) (Moschitti, 2006), the kernels used in this work model grammatical and semantic information, as triggered respectively by the dependency edge labels and lexical nodes. The latter is exploited through recent results in distributional models of lexical semantics, as proposed in

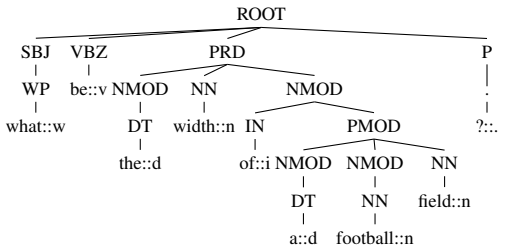


Figure 1: Grammatical Relation Centered Tree (GRCT) of *"What is the width of a football field?"*

word embedding methods (e.g., (Mikolov et al., 2013; Sahlgren, 2006). In particular, we adopt the *Smoothed Partial Tree Kernel* (SPTK) described in Croce et al. (2011): it extends the PTK formulation with a similarity function between lexical nodes in a GRCT, i.e., the cosine similarity between word vector representations based on word embeddings. We also use a further extension of the SPTK, called *Compositionally Smoothed Partial Tree Kernel* (CSPTK) (as in Annesi et al. (2014)). In CSPTK, the lexical information provided by the sentence words is propagated along the non-terminal nodes representing head-modifier dependencies. Figure 2 shows a compositionally-labeled tree, where the similarity function at the nodes can model lexical composition, i.e., capturing contextual information. For example, in the sentence, "*What instrument does Hendrix play?*", the role of the word *instrument* can be fully captured only if its composition with the verb *play* is considered. The CSPTK applies a composition function between nodes: while several algebraic functions can be adopted to compose two word vectors representing a head/modifier pair, here we refer to a simple additive function that assigns to each $(h, m)$ pair the linear combination of the involved vectors, i.e., $(\mathbf{h}, \mathbf{m}) = A\mathbf{h} + B\mathbf{m}$: although simple and efficient, it actually produces very effective CSPTK functions.

**Complexity**. The training phase of an optimal maximum margin algorithm (such as SVM) requires a number of kernel operations that is more than linear (almost $\mathcal{O}(n^2)$) with respect to the number of training examples $n$, as discussed in Chang and Lin (2011). Also the classification phase depends on the size of the input dataset and the intrinsic complexity of the targeted task: classifying a new instance requires to evaluate the kernel function with respect to each support vector. For complex tasks, the number of selected support vectors tends to be very large, and using the resulting model can be impractical. This cost is also problematic as single kernel operations can be very expensive: the cost of evaluating the PTK on a single tree pair is almost linear in the number of nodes in the input trees, as shown in Moschitti (2006). When lexical semantics is considered, as in SPTKs and CSPTKs, it is more than linear in the number of nodes (Croce et al., 2011).

## 3 Deep Learning in Kernel Spaces

### 3.1 The Nyström method

Given an input training dataset $D$, a kernel $K(o_i, o_j)$ is a similarity function over $\mathcal{D}^2$ that corresponds to a dot product in the implicit kernel space, i.e., $K(o_i, o_j) = \Phi(o_i) \cdot \Phi(o_j)$. The advantage of kernels is that the projection function $\Phi(o) = \mathbf{x} \in \mathbb{R}^n$ is never explicitly computed (Shawe-Taylor and Cristianini, 2004). In fact, this operation may be prohibitive when the dimensionality $n$ of the underlying kernel space is extremely large, as for Tree Kernels (Collins and Duffy, 2001). Kernel functions are used by learning algorithms, such as SVM, to operate only implicitly on instances in the kernel space, by never accessing their explicit definition. Let us apply the projection function $\Phi$ over all examples from $\mathcal{D}$ to derive representations, $\mathbf{x}$ denoting the rows of the matrix $X$. The Gram matrix can always be computed as $G = XX^\top$, with each single element corresponding to $G_{ij} = \Phi(o_i)\Phi(o_j) = K(o_i, o_j)$. The aim of the Nyström method is to derive a new low-dimensional embedding $\tilde{x}$ in a $l$-dimensional space, with $l \ll n$ so that $\tilde{G} = \tilde{X}\tilde{X}^\top$ and $\tilde{G} \approx G$. This is obtained by generating an approximation $\tilde{G}$ of $G$ using a subset of $l$ columns of the matrix, i.e., a selection of a subset $L \subset \mathcal{D}$ of the available examples, called *landmarks*. Suppose we randomly sample $l$ columns of $G$, and let $C \in \mathbb{R}^{|D| \times l}$ be the matrix of these sampled columns. Then, we can rearrange the columns and rows of $G$ and define $X = [X_1 \;\; X_2]$ such that:

$$G = XX^\top = \begin{bmatrix} W & X_1^\top X_2 \\ X_2^\top X_1 & X_2^\top X_2 \end{bmatrix}$$
$$\text{and} \quad C = \begin{bmatrix} W \\ X_2^\top X_1 \end{bmatrix} \tag{1}$$

where $W = X_1^\top X_1$, i.e., the subset of $G$ that contains only landmarks. The Nyström approximation can be defined as:

$$G \approx \tilde{G} = CW^\dagger C^\top \tag{2}$$

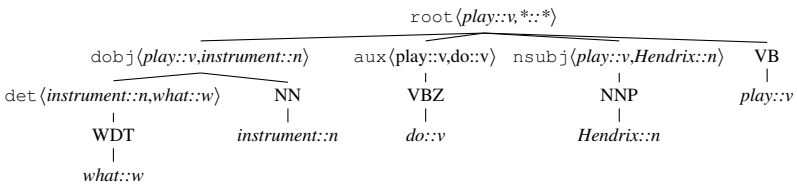


Figure 2: Compositional Grammatical Relation Centered Tree (CGRCT) of "*What instrument does Hendrix play?*"

where $W^\dagger$ denotes the Moore-Penrose inverse of $W$. The Singular Value Decomposition (SVD) is used to obtain $W^\dagger$ as it follows. First, $W$ is decomposed so that $W = USV^\top$, where $U$ and $V$ are both orthogonal matrices, and $S$ is a diagonal matrix containing the (non-zero) singular values of $W$ on its diagonal. Since $W$ is symmetric and positive definite $W = USU^\top$. Then $W^\dagger = US^{-1}U^\top = US^{-\frac{1}{2}}S^{-\frac{1}{2}}U^\top$ and the Equation 2 can be rewritten as

$$G \approx \tilde{G} = CUS^{-\frac{1}{2}}S^{-\frac{1}{2}}U^\top C^\top = (CUS^{-\frac{1}{2}})(CUS^{-\frac{1}{2}})^\top = \tilde{X}\tilde{X}^\top \quad (3)$$

Given an input example $o \in \mathcal{D}$, a new low-dimensional representation $\tilde{\mathbf{x}}$ can be thus determined by considering the corresponding item of $C$ as

$$\tilde{\mathbf{x}} = \mathbf{c}US^{-\frac{1}{2}} \quad (4)$$

where $\mathbf{c}$ is the vector whose dimensions contain the evaluations of the kernel function between $o$ and each landmark $o_j \in L$. Therefore, the method produces $l$-dimensional vectors. If $k$ is the average number of basic operations required during a single kernel computation, the overall cost of a single projection is $\mathcal{O}(kl + l^2)$, where the first term corresponds to the cost of generating the vector $\mathbf{c}$, while the second term is needed for the matrix multiplications in Equation 4. Typically, the number of landmarks $l$ ranges from hundreds to few thousands and, for complex kernels (such as Tree Kernels), the projection cost can be reduced to $\mathcal{O}(kl)$. Several policies have been defined to determine the best selection of landmarks to reduce the Gram Matrix approximation error. In this work the uniform sampling without replacement is adopted, as suggested by Kumar et al. (2012), where this policy has been theoretically and empirically shown to achieve results comparable with other (more complex) selection policies.

### 3.2 A Kernel-based Deep Architecture

The above introduced Nyström representation $\tilde{\mathbf{x}}$ of any input example $o$ is linear and can be adopted to feed a neural network architecture. We assume a labeled dataset $\mathcal{L} = \{(o, y) \mid o \in \mathcal{D},\ y \in Y\}$ being available, where $o$ refers to a generic instance and $y$ is its associated class. In this Section, we define a Multi-Layer Perceptron (MLP) architecture, with a specific Nyström layer based on the Nyström embeddings of Eq. 4. We will refer to this architecture as Kernel-based Deep Architecture (KDA). KDA has an *input layer*, a *Nyström layer*, a possibly empty sequence of non-linear *hidden layers* and a final *classification layer*, which produces the output.

The *input* layer corresponds to the input vector $\mathbf{c}$, i.e., the row of the $C$ matrix associated to an example $o$. Notice that, for adopting the KDA, the values of the matrix $C$ should be all available. In the training stage, these values are in general cached. During the classification stage, the $\mathbf{c}$ vector corresponding to an example $o$ is directly computed by $l$ kernel computations between $o$ and each one of the $l$ landmarks.

The input layer is mapped to the *Nyström* layer, through the projection in Equation 4. Notice that the embedding provides also the proper weights, defined by $US^{-\frac{1}{2}}$, so that the mapping can be expressed through the Nyström matrix $H_{Ny} = US^{-\frac{1}{2}}$: it corresponds to a pre-trained stage derived through SVD, as discussed in Section 3.1. Equation 4 provides a static definition for $H_{Ny}$ whose weights can be left invariant during the neural network training. However, the values of $H_{Ny}$ can be made available for the standard back-propagation adjustments applied for training[1]. Formally, the low-dimensional embedding of an input example $o$, is $\tilde{\mathbf{x}} = \mathbf{c}\, H_{Ny} = \mathbf{c}\, US^{-\frac{1}{2}}$.

The resulting outcome $\tilde{\mathbf{x}}$ is the input to one or more non-linear *hidden* layers. Each $t$-th hidden layer is realized through a matrix $H_t \in \mathbb{R}^{h_{t-1} \times h_t}$ and a bias vector $\mathbf{b}_t \in \mathbb{R}^{1 \times h_t}$, whereas $h_t$ denotes

[1] In our preliminary experiments, adjustments to the $H_{Ny}$ matrix have been tested, but no significant effect was observed. Therefore, no adjustment has been used in any reported experiment, although more in depth exploration is needed on this aspect.

the desired hidden layer dimensionality. Clearly, given that $H_{Ny} \in \mathbb{R}^{l \times l}$, $h_0 = l$. The first hidden layer in fact receives in input $\tilde{\mathbf{x}} = \mathbf{c}H_{Ny}$, that corresponds to $t = 0$ layer input $\mathbf{x}_0 = \tilde{\mathbf{x}}$ and its computation is formally expressed by $\mathbf{x}_1 = f(\mathbf{x}_0 H_1 + \mathbf{b}_1)$, where $f$ is a non-linear activation function. In general, the generic $t$-th layer is modeled as:

$$\mathbf{x}_t = f(\mathbf{x}_{t-1} H_t + \mathbf{b}_t) \quad (5)$$

The final layer of KDA is the *classification layer*, realized through the output matrix $H_O$ and the output bias vector $\mathbf{b}_O$. Their dimensionality depends on the dimensionality of the last hidden layer (called $O_{-1}$) and the number $|Y|$ of different classes, i.e., $H_O \in \mathbb{R}^{h_{O_{-1}} \times |Y|}$ and $\mathbf{b}_O \in \mathbb{R}^{1 \times |Y|}$, respectively. In particular, this layer computes a linear classification function with a softmax operator so that $\hat{y} = softmax(\mathbf{x}_{O_{-1}} H_O + \mathbf{b}_O)$.

In order to avoid over-fitting, two different regularization schemes are applied. First, the dropout is applied to the input $\mathbf{x}_t$ of each hidden layer ($t \geq 1$) and to the input $\mathbf{x}_{O_{-1}}$ of the final classifier. Second, a $L_2$ regularization is applied to the norm of each layer[2] $H_t$ and $H_O$.

Finally, the KDA is trained by optimizing a loss function made of the sum of two factors: first, the cross-entropy function between the gold classes and the predicted ones; second the $L_2$ regularization, whose importance is regulated by a meta-parameter $\lambda$. The final loss function is thus

$$L(y, \hat{y}) = \sum_{(o,y) \in \mathcal{L}} y \, log(\hat{y}) + \lambda \sum_{H \in \{H_t\} \cup \{H_O\}} ||H||^2$$

where $\hat{y}$ are the softmax values computed by the network and $y$ are the true one-hot encoding values associated with the example from the labeled training dataset $\mathcal{L}$.

## 4 Empirical Investigation

The proposed KDA has been applied adopting the same architecture but with different kernels to three NLP tasks, i.e., Question Classification, Community Question Answering, and Automatic Boundary Detection in Semantic Role Labeling. The Nyström projector has been implemented in the KeLP framework[3]. The neural network has been implemented in Tensorflow[4], with 2 hidden layers whose dimensionality corresponds to the number of involved Nyström landmarks. The *rectified linear unit* is the non-linear activation function in each layer. The dropout has been applied in each hidden layer and in the final classification layer. The values of the dropout parameter and the $\lambda$ parameter of the $L_2$-regularization have been selected from a set of values via grid-search. The Adam optimizer with a learning rate of $0.001$ has been applied to minimize the loss function, with a multi-epoch ($500$) training, each fed with batches of size $256$. We adopted an early stop strategy, where the best model was selected according to the performance over the development set. Every performance measure is obtained against a specific sampling of the Nyström landmarks. Results averaged against $5$ such samplings are always hereafter reported.

### 4.1 Question Classification

Question Classification (QC) is the task of mapping a question into a closed set of answer types in a Question Answering system. We used the UIUC dataset (Li and Roth, 2006), including a training and test set of $5,452$ and $500$ questions, respectively, organized in 6 classes (like `ENTITY` or `HUMAN`). TKs resulted very effective, as shown in Croce et al. (2011); Annesi et al. (2014). In Annesi et al. (2014), QC is mapped into a One-vs-All multi-classification schema, where the CSPTK achieves state-of-the-art results of 95%: it acts directly over compositionally labeled trees without relying on any manually designed feature.

In order to proof the benefits of the KDA architecture, we generated Nyström representation of the CSPTK kernel function[5] with default parameters (i.e., $\mu = \lambda = 0.4$). The SVM formulation by Chang and Lin (2011), fed with the CSPTK (hereafter KSVM), is here adopted to determine the reachable upper bound in classification quality, i.e., a 95% of accuracy, at higher computational costs. It establishes the state-of-the-art over the UIUC dataset. The resulting model includes 3,873 support vectors: this corresponds to the number of kernel operations required to classify any input test question. The Nyström method based on a number of landmarks ranging from 100 to 1,000 is adopted for modeling input vectors in

[2] The input layer and the Nyström layer are not modified during the learning process, and they are not regularized.

[3] http://www.kelp-ml.org

[4] https://www.tensorflow.org/

[5] The lexical vectors used in the CSPTK are generated again using the Word2vec tool with a Skip-gram model.

the CSPTK kernel space. Results are reported in Table 1: computational saving refers to the percentage of avoided kernel computations with respect to the application of the KSVM to each test instance. To justify the need of the Neural Network, we compared the proposed KDA to an efficient linear SVM that is directly trained over the Nyström embeddings. This SVM implements the Dual Coordinate Descent method (Hsieh et al., 2008) and will be referred as $SVM_{DCD}$. We also measured the state-of-the-art Convolutional Neural Network[6] (CNN) of Kim (2014), achieving the remarkable accuracy of 93.6%. Notice that the linear classifier $SVM_{DCD}$ operating over the approximated kernel space achieves the same classification quality of the CNN when just 1,000 landmarks are considered. KDA improves this results, achieving 94.3% accuracy even with fewer landmarks (only 600), showing the effectiveness of non-linear learning over the Nyström input. Although KSVM improves to 95%, KDA provides a saving of more than 84% kernel computations at classification time. This result is straightforward as it confirms that **linguistic information encoded in a tree is important in the analysis of questions** and can be used as a pre-training strategy. Figure 3 shows the accuracy curves according to various approximations of the kernel space, i.e., number of landmarks.

Table 1: Results in terms of Accuracy and saving in the Question Classification task

| Model | #Land. | Accuracy | Saving |
|---|---|---|---|
| CNN | - | 93.6% | - |
| KSVM | - | 95.0% | 0.0% |
| KDA ($SVM_{DCD}$) | 100 | 88.5% (84.1%) | 97.4% |
| | 200 | 92.2% (88.7%) | 94.8% |
| | 400 | 93.7% (91.6%) | 89.7% |
| | 600 | **94.3%** (92.8%) | **84.5%** |
| | 800 | 94.3% (93.0%) | 79.3% |
| | 1,000 | 94.2% (93.6%) | 74.2% |

## 4.2 Community Question-Answering

In the SemEval-2016 task 3, participants were asked to automatically provide good answers in a community question answering setting (Nakov et al., 2016). We focused on the subtask A: given a question and a large collection of question-comment threads created by a user community, the task consists in (re-)ranking the comments w.r.t. their utility in answering the question. Subtask A can be modeled as a binary classification problem, where instances are (question, comment) pairs. Each pair generates an example for a binary SVM, where the positive label is associated to a *good* comment and the negative label refers to *potentially useful* and *bad* comments. The classification score achieved over different (question, comment) pairs is used to sort instances and produce the final ranking over comments. The above setting results in a train and test dataset made of 20,340 and 3,270 examples, respectively. In Filice et al. (2016), a Kernel-based SVM classifier (KSVM) achieved state-of-the-art results by adopting a kernel combination that exploited (*i*) feature vectors containing linguistic similarities between the texts in a pair; (*ii*) shallow syntactic trees that encode the lexical and morpho-syntactic information shared between text pairs; (*iii*) feature vectors capturing task-specific information.

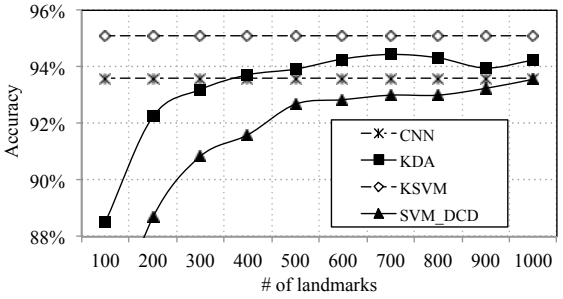


Figure 3: QC task - accuracy curves w.r.t. the number of landmarks.

Table 2: Results in terms of $F_1$ and savings in the Community Question Answering task

| Model | #Land. | $F_1$ | Saving |
|---|---|---|---|
| KSVM | - | 0.644 | 0.0% |
| ConvKN | - | 0.662 | - |
| KDA ($SVM_{DCD}$) | 100 | 0.638 (0.596) | 99.1% |
| | 200 | 0.635 (0.627) | 98.2% |
| | 400 | 0.657 (0.637) | 96.5% |
| | 600 | 0.669 (0.645) | 94.7% |
| | 800 | **0.680** (0.653) | **92.9%** |
| | 1,000 | 0.674 (0.644) | 91.2% |

Such model includes 11,322 support vectors. We investigated the KDA architecture, trained by maximizing the $F_1$ measure, based on a Nyström layer initialized using the same kernel functions as KSVM. We varied the Nyström dimensions from 100 to 1,000 landmarks, i.e., a much lower number than the support vectors of KSVM.

Table 2 reports the results: very high $F_1$ scores

[6]The deep architecture presented in Kim (2014) outperforms several NN models, including the Recursive Neural Tensor Network or Tree-LSTM presented in (Socher et al., 2013; Tai et al., 2015) which presents a semantic compositionality model that exploits parse trees.

are observed with impressive savings in terms of kernel computations (between 91.2% and 99%). Also on the cQA task, the $F_1$ obtained by the $SVM_{DCD}$ is significantly lower than the KDA one. Moreover, with 800 landmarks KDA achieves the remarkable results of 0.68 of $F_1$, that is the state-of-the-art against other convolutional systems, e.g., ConvKN (Barrón-Cedeño et al., 2016): this latter combines convolutional tree kernels with kernels operating on sentence embeddings generated by a convolutional neural network.

### 4.3 Argument Boundary Detection

Semantic Role Labeling (SRL) consists of the detection of the semantic arguments associated with the predicate of a sentence (called Lexical Unit) and their classification into their specific roles (Fillmore, 1985). For example, given the sentence "*Bootleggers then copy the film onto hundreds of tapes*" the task would be to recognize the verb *copy* as representing the DUPLICATION frame with roles, CREATOR for *Bootleggers*, ORIGINAL for *the film* and GOAL for *hundreds of tapes*.

Argument Boundary Detection (ABD) corresponds to the SRL subtask of detecting the sentence fragments spanning individual roles. In the previous example the phrase "*the film*" represents a role (i.e., ORIGINAL), while "*of tapes*" or "*film onto hundreds*" do not, as they just partially cover one or multiple roles, respectively. The ABD task has been successfully tackled using TKs since Moschitti et al. (2008). It can be modeled as a binary classification task over each parse tree node $n$, where the argument span reflects words covered by the sub-tree rooted at $n$. In our experiments, Grammatical Relation Centered Tree (GRCT) derived from dependency grammar (Fig. 4) are employed, as shown in Fig. 5. Each node is considered as a candidate in covering a possible *argument*. In particular, the structure in Fig. 5a is a positive example. On the contrary, in Fig. 5b the NMOD node only covers the phrase "*of tapes*", i.e., a subset of the correct role, and it represents a negative example[7].

We selected all the sentences whose predicate word (lexical unit) is a verb (they are about

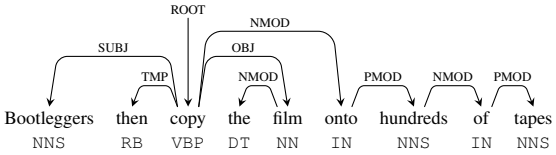


Figure 4: Example of dependency parse tree

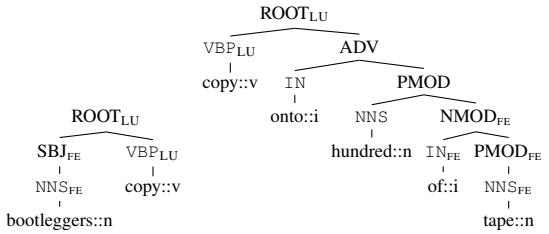


Figure 5: Conversion from dependency graph to GRCT. Tree in Fig. 5a is a positive example, while in Fig. 5b a negative one.

60,000), from the 1.3 version of the Framenet dataset (Baker et al., 1998). This gives rise to about 1,400,000 sub-trees, i.e., the positive and negative instances. The dataset is split in train and test according to the 90/10 proportion (as in (Johansson and Nugues, 2008)). This size makes the application of a traditional kernel-based method unfeasible, unless a significant instance sub-sampling is performed.

We firstly experimented standard SVM learning over a sampled training set of 10,000 examples, a typical size for annotated datasets in computational linguistics tasks. We adopted the Smoothed Partial Tree Kernel (Croce et al., 2011) with standard parameters (i.e., $\mu = \lambda = 0.4$) and lexical nodes expressed through 250-dimensional vectors obtained by applying Word2Vec (Mikolov et al., 2013) to the entire Wikipedia. When trained over this 10k instances dataset, the kernel-based SVM (KSVM) achieves an $F_1$ of 70.2%, over the same test set used in Croce and Basili (2016) that includes 146,399 examples. The KSVM learning produces a model including $2,994$ support vectors, i.e., the number of kernel operations required to classify each new test instance. We then apply the Nyström linearization to a larger dataset made of 100k examples, and trained a classifier using both the Dual Coordinate Descent method (Hsieh et al., 2008), $SVM_{DCD}$, and the KDA proposed in this work. Table 3 presents the results in terms of $F_1$ and saved kernel operation. Although $SVM_{DCD}$ with 500 landmarks already achieves 0.713 $F_1$, a score higher than KSVM, it is signif-

[7]The nodes of the subtree covering the words to be verified as possible argument are marked with a FE tag. The word evoking the frame and its ancestor nodes are also marked with the LU tag. The other nodes are pruned out, except the ones connecting the LU nodes to the FE ones.

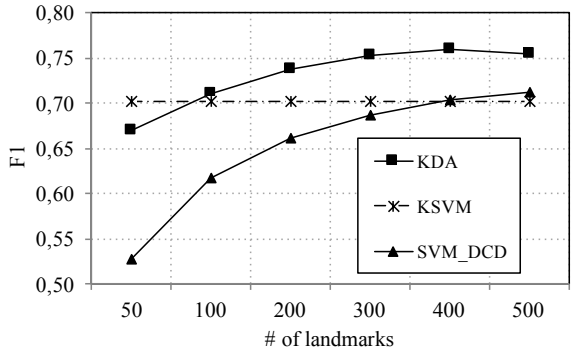


Figure 6: ABD task: $F_1$ measure curves w.r.t. the number of landmarks.

icantly improved by the KDA. KDA achieves up to 0.76 $F_1$ with only 400 landmarks, resulting in a huge step forward w.r.t. the KSVM. This result is straightforward considering (*i*) the reduction of required kernel operations, i.e., more than 86% are saved and (*ii*) the quality achieved since 100 landmarks (i.e., 0.711, higher than the KSVM).

Table 3: Results in terms of $F_1$ and saving in the Argument Boundary Detection task.

| Model | Land. | Tr.Size | $F_1$ | Saving |
|---|---|---|---|---|
| KSVM | - | 10k | 0.702 | 0.0% |
| KDA ($SVM_{DCD}$) | 100 | 100k | 0.711 (0.618) | 96.7% |
| | 200 | 100k | 0.737 (0.661) | 93.3% |
| | 300 | 100k | 0.753 (0.686) | 90.0% |
| | 400 | 100k | **0.760** (0.704) | **86.6%** |
| | 500 | 100k | 0.754 (0.713) | 83.3% |

## 5 Discussion and Conclusions

In this work, we promoted a methodology to embed structured linguistic information within NNs, according to mathematically rich semantic similarity models, based on kernel functions. Structured data, such as trees, are transformed into dense vectors according to the Nyström methodology, and the NN is effective in capturing non-linearities in these representations, but still improving generalization at a reasonable complexity.

At the best our knowledge, this work is one of the few attempts to systematically integrate linguistic kernels within a deep neural network architecture. The problem of combining such methodologies has been studied in specific works, such as (Baldi et al., 2011; Cho and Saul, 2009; Yu et al., 2009). In Baldi et al. (2011) the authors propose a hybrid classifier, for bridging kernel methods and neural networks. In particular, they use the output of a kernelized k-nearest neighbors algorithm as input to a neural network. Cho and Saul (2009) introduced a family of kernel functions that mimic the computation of large multilayer neural networks. However, such kernels can be applied only on vector inputs. In Yu et al. (2009), deep neural networks for rapid visual recognition are trained with a novel regularization method taking advantage of kernels as an oracle representing prior knowledge. The authors transform the kernel regularizer into a loss function and carry out the neural network training by gradient descent. In Zhuang et al. (2011) a different approach has been promoted: a multiple (two) layer architecture of kernel functions, inspired by neural networks, is studied to find the best kernel combination in a Multiple Kernel Learning setting. In Mairal et al. (2014) the invariance properties of convolutional neural networks (LeCun et al., 1998) are modeled through kernel functions, resulting in a Convolutional Kernel Network. Other effort for combining NNs and kernel methods is described in Tymoshenko et al. (2016), where a SVM adopts a tree kernels combinations with embeddings learned through a CNN.

The approach here discussed departs from previous approaches in different aspects. First, a general framework is promoted: it is largely applicable to any complex kernel, e.g., structural kernels or combinations of them. The efficiency of the Nyström methodology encourages its adoption, especially when complex kernel computations are required. Notice that other low-dimensional approximations of kernel functions have been studied, as for example the randomized feature mappings proposed in Rahimi and Recht (2008). However, these assume that (*i*) instances have vectorial form and (*ii*) shift-invariant kernels are adopted. The Nyström method adopted here does not suffer of such limitations: as our target is the application to structured (linguistic) data, more general kernels, i.e., non-shift-invariant convolution kernels are needed.

Given the Nyström approximation, the learning setting corresponds to a general well-known neural network architecture, i.e., a multilayer perceptron, and does not require any manual feature engineering or the design of ad-hoc network architectures. The success in three different tasks confirms its large applicability without major changes or adaptations. Second, we propose a novel learning strategy, as the capability of kernel methods to represent complex search spaces is combined with the ability of neural networks to find non-linear so-

lutions to complex tasks. Last, the suggested KDA framework is *fully scalable*, as (*i*) the network can be parallelized on multiple machines, and (*ii*) the computation of the Nyström reconstruction vector $\mathbf{c}$ can be easily parallelized on multiple processing units, ideally $l$, as each unit can compute one $c_i$ value. Future work will address experimentations with larger scale datasets; moreover, it is interesting to experiment with more landmarks in order to better understand the trade-off between the representation capacity of the Nyström approximation of the kernel functions and the over-fitting that can be introduced in a neural network architecture. Finally, the optimization of the KDA methodology through the suitable parallelization on multicore architectures, as well as the exploration of mechanisms for the dynamic reconstruction of kernel spaces (e.g., operating over $H_{Ny}$) also constitute interesting future research directions on this topic.

## References


Paolo Annesi, Danilo Croce, and Roberto Basili. 2014. Semantic compositionality in tree kernels. In *Proceedings of CIKM 2014*. ACM.

Collin F. Baker, Charles J. Fillmore, and John B. Lowe. 1998. The Berkeley FrameNet project. In *Proc. of COLING-ACL*. Montreal, Canada.

Pierre Baldi, Chloe Azencott, and S. Joshua Swamidass. 2011. Bridging the gap between neural network and kernel methods: Applications to drug discovery. In *Proceedings of the 20th Italian Workshop on Neural Nets*. http://dl.acm.org/citation.cfm?id=1940632.1940635.

Alberto Barrón-Cedeño, Giovanni Da San Martino, Shafiq Joty, Alessandro Moschitti, Fahad Al-Obaidli, Salvatore Romeo, Kateryna Tymoshenko, and Antonio Uva. 2016. ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on arabic and english fora. In *Proceedings of SemEval-2016*.

Nicola Cancedda, Éric Gaussier, Cyril Goutte, and Jean-Michel Renders. 2003. Word-sequence kernels. *Journal of Machine Learning Research* 3:1059–1082.

Chih-Chung Chang and Chih-Jen Lin. 2011. Libsvm: A library for support vector machines. *ACM Trans. Intell. Syst. Technol.* 2(3):27:1–27:27. https://doi.org/10.1145/1961189.1961199.

Youngmin Cho and Lawrence K. Saul. 2009. Kernel methods for deep learning. In Y. Bengio, D. Schuurmans, J. D. Lafferty, C. K. I. Williams, and A. Culotta, editors, *Advances in Neural Information Processing Systems 22*, Curran Associates, Inc., pages 342–350. http://papers.nips.cc/paper/3628-kernel-methods-for-deep-learning.pdf.

Michael Collins and Nigel Duffy. 2001. Convolution kernels for natural language. In *Proceedings of Neural Information Processing Systems (NIPS'2001)*. pages 625–632.

Danilo Croce and Roberto Basili. 2016. Large-scale kernel-based language learning through the ensemble nystrom methods. In *Proceedings of ECIR 2016*.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of EMNLP '11*. pages 1034–1046.

Ofer Dekel and Yoram Singer. 2006. Support vector machines on a budget. In *NIPS*. MIT Press, pages 345–352.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 task 3: Learning semantic relations between questions and comments. In *Proceedings of SemEval '16*.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural representations for learning relations between pairs of texts. In *Proceedings of ACL 2015*. Beijing, China, pages 1003–1013. http://www.aclweb.org/anthology/P15-1097.

Charles J. Fillmore. 1985. Frames and the semantics of understanding. *Quaderni di Semantica* 6(2):222–254.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic Labeling of Semantic Roles. *Computational Linguistics* 28(3):245–288.

David Haussler. 1999. Convolution kernels on discrete structures. In *Technical Report UCS-CRL-99-10*. University of California, Santa Cruz.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.* 9(8):1735–1780.

Cho-Jui Hsieh, Kai-Wei Chang, Chih-Jen Lin, S. Sathiya Keerthi, and S. Sundararajan. 2008. A dual coordinate descent method for large-scale linear svm. In *Proceedings of the ICML 2008*. ACM, pages 408–415.

Richard Johansson and Pierre Nugues. 2008. The effect of syntactic representation on semantic role labeling. In *Proceedings of COLING*.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *Proceedings EMNLP 2014*. Doha, Qatar, pages 1746–1751.

Sanjiv Kumar, Mehryar Mohri, and Ameet Talwalkar. 2012. Sampling methods for the nyström method. *J. Mach. Learn. Res.* 13:981–1006.

Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. 1998. Gradient-based learning applied to document recognition. *Proc. of the IEEE* 86(11).

Omer Levy, Yoav Goldberg, and Ido Dagan. 2015. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics* 3:211–225. https://transacl.org/ojs/index.php/tacl/article/view/570.

Xin Li and Dan Roth. 2006. Learning question classifiers: the role of semantic information. *Natural Language Engineering* 12(3):229–249.

Julien Mairal, Piotr Koniusz, Zaid Harchaoui, and Cordelia Schmid. 2014. Convolutional kernel networks. In *Advances in Neural Information Processing Systems*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *CoRR* abs/1301.3781. http://arxiv.org/abs/1301.3781.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *ECML*. Berlin, Germany.

Alessandro Moschitti, Daniele Pighin, and Robert Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics* 34.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, Abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 task 3: Community question answering. In *Proceedings of SemEval-2016*.

Ali Rahimi and Benjamin Recht. 2008. Random features for large-scale kernel machines. In J. C. Platt, D. Koller, Y. Singer, and S. T. Roweis, editors, *Advances in Neural Information Processing Systems 20*, Curran Associates, Inc., pages 1177–1184. http://papers.nips.cc/paper/3182-random-features-for-large-scale-kernel-machines.pdf.

Magnus Sahlgren. 2006. *The Word-Space Model*. Ph.D. thesis, Stockholm University.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Building structures from classifiers for passage reranking. ACM, New York, NY, USA, CIKM '13, pages 969–978.

John Shawe-Taylor and Nello Cristianini. 2004. *Kernel Methods for Pattern Analysis*. Cambridge University Press, New York, NY, USA.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of EMNLP '13*.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*. pages 1556–1566. http://aclweb.org/anthology/P/P15/P15-1150.pdf.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2016. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *Proceedings of NAACL 2016*. http://www.aclweb.org/anthology/N16-1152.

Zhuang Wang and Slobodan Vucetic. 2010. Online passive-aggressive algorithms on a budget. *Journal of Machine Learning Research - Proceedings Track* 9:908–915.

Christopher K. I. Williams and Matthias Seeger. 2001. Using the nyström method to speed up kernel machines. In *Proceedings of NIPS 2000*.

Kai Yu, Wei Xu, and Yihong Gong. 2009. Deep learning with kernel regularization for visual recognition. In *Advances in Neural Information Processing Systems 21*, Curran Associates, Inc., pages 1889–1896.

Jinfeng Zhuang, Ivor W. Tsang, and Steven C. H. Hoi. 2011. Two-layer multiple kernel learning. In *AISTATS*. JMLR.org, volume 15 of *JMLR Proceedings*, pages 909–917.