Available online at www.sciencedirect.com

ScienceDirect

Computer Speech & Language 58 (2019) 51–75

www.elsevier.com/locate/csl

Making sense of kernel spaces in neural learning

D. Croce*, S. Filice, R. Basili

Department of Enterprise Engineering, University of Roma Tor Vergata, Via del Politecnico 1, Roma 00133, Italy

Received 20 April 2018; received in revised form 24 March 2019; accepted 24 March 2019
Available online 28 March 2019

Abstract

Kernel-based and Deep Learning methods are two of the most popular approaches in Computational Natural Language Learning. Although these models are rather different and characterized by distinct strong and weak aspects, they both had impressive impact on the accuracy of complex Natural Language Processing tasks. An advantage of kernel-based methods is their capability of exploiting structured information induced from examples. For instance, Sequence or Tree kernels operate over structures reflecting linguistic evidence, such as syntactic information encoded in syntactic parse trees. Deep Learning approaches are very effective as they can learn non-linear decision functions: however, general models require input instances to be explicitly modeled via vectors or tensors, and operating on structured data is made possible only by using ad-hoc architectures.

In this work, we discuss a novel architecture that efficiently combines kernel methods and neural networks, in the attempt at squeezing the best from the two paradigms. The so-called *Kernel-based Deep Architecture* (KDA) adopts a Nyström-based projection function to approximate any valid kernel function and convert any structure they operate on (for instance, linguistic structures, such as trees) into dense linear embeddings. These can be used as input of a Deep Feed-forward Neural Network that exploits such embeddings to learn non-linear classification functions. 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 *ad hoc* manual feature engineering or architectural design, (*ii*) achieves a drastic reduction of the computational cost w.r.t. pure kernel methods, and (*iii*) exploits the non-linearity of Deep Architectures to produce accurate models. We experimented the KDA in three rather different semantic inference tasks: Semantic Parsing, Question Classification, and Community Question Answering. Results show that the KDA achieves state-of-the-art accuracy, with a computational cost that is much lower than the one necessary to train and test a pure kernel-based method, such as the SVM algorithm.

Keywords: Kernel-based learning; Neural methods; Semantic spaces; Nystrom embeddings

1. Introduction

Nowadays, a very large portion of machine learning approaches to Natural Language Processing (NLP) are based on Deep Learning (Goldberg, 2016; Collobert et al., 2011). This widespread use of Deep Learning is supported by the impressive results such methods achieve, and their feature learning capability (Bengio et al., 2013; Turian et al., 2010; Kim, 2014): input words and sentences are modeled by means of dense embeddings (i.e., vectors or tensors)

* Corresponding author.
E-mail address: croce@info.uniroma2.it (D. Croce).

https://doi.org/10.1016/j.csl.2019.03.006
0885-2308/

whose dimensions correspond to latent semantic concepts which are automatically learned during the training phase. If this largely automatizes the feature engineering phase, on the other side, it has some inherent drawbacks.

In particular, injecting linguistic information into a NN is still an open problem. If pre-trained word embeddings are widely recognized as an effective approach for improving lexical generalization, there is no general agreement about how to provide syntactic information to the NN. Some structured NN models have been proposed (Hochreiter and Schmidhuber, 1997; Socher et al., 2013) although usually tailored to specific problems. 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 networks (Hochreiter and Schmidhuber, 1997) build intermediate representations of sequences, resulting in similarity estimates over sequences and their inner sub-sequences. However, the linguistic information that such models capture is never made explicit: it is embedded in a latent space whose dimensions cannot be easily interpreted. Therefore, understanding the linguistic aspects that are responsible for the network decision is not possible in very complex NN architectures. A few attempts to solve the interpretability problem of NNs have been proposed in computer vision (Erhan et al., 2010; Simonyan et al., 2013; Bach et al., 2015), but their extension to the Natural Language Processing scenario is not straightforward.

It is a strong and widely shared assumption that the target NN architecture should be made as much as possible sensitive to an explicit representation of the underlying linguistic structure: this is particularly useful to *methodologically unify the different NN paradigms* as well as for *augmenting the readability of a network output*. Portions of its architecture explicitly connected to the linguistic knowledge involved in the classification inference would allow one to better locate the local inferences justifying the global behavior. Making a NN sensitive to linguistic information may be seen as the problem of defining a modeling paradigm able to inject lexical, grammatical as well as semantic information directly in the network architecture and make a fruitful use of it during training. A general language aware NN paradigm should naturally allow to host different linguistic evidence for supporting different forms of inferences (such as disambiguation, entailment, paraphrasing or translation) in a homogeneous manner, i.e., through a unified model able to accommodate lexical, grammatical as well as semantic structures.

Most of the recent literature usually addresses *ad-hoc* architectures, defined to solve specific problems and their ability to generalize across tasks is still to be fully demonstrated. In general, training complex neural networks is 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 architectures result in the same performance, whenever optimal hyper-parameter tuning is applied. In this latter case, no significant difference is observed across different architectures: in some sense, it seems that parameter optimization is more important than the architectural design of the network. This makes the generalization of different paradigms a still open issue, where largely empirical solutions are still common practice.

A natural way to *explicitly* encode lexical, syntactic and semantic information is by means of linguistic structures, such as dependency graphs or constituency trees. Kernel methods (Shawe-Taylor and Cristianini, 2004) can directly operate on such structures. Their combination with linear learning algorithms, such as the Support Vector Machines (SVM) (Vapnik, 1998), allowed these methods to achieve very good performance in several NLP tasks, as summarized in Moschitti (2012).

Sequence (Cancedda et al., 2003) or tree kernels (Collins and Duffy, 2001) are of particular interest as the feature space they capture reflects linguistic patterns. For instance, Chali et al. (2009) used tree kernels for question answering, while (Katrenko et al., 2010) defined a local alignment kernel for relation recognition. Moreover, tree-kernel based models are more readable. In fact, kernel-based Support Vector Machines models correspond to sets of support vectors (SVs), whose weights characterize the classification function, i.e., the maximal margin hyperplane. The semantic association between such influential examples and a test instance is precisely expressed by the kernel: the higher the similarity the higher the influence on the final decision. Therefore, in kernels the decision quantitatively depends on structural properties, and this supports a readable, although qualitative, interpretation.

It must be said that the adoption of kernel-based methods can be problematic in many NLP tasks. The classification cost usually depends on the model complexity. In SVMs, it corresponds to the number of support vectors, 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 initially tackled 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 adequate accuracy.

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 to project future input examples into low-dimensional embeddings, in a vector space defined by a set of selected instances called *landmarks*. For example, if used over Tree Kernels (TKs), the Nyström projection corresponds to the embedding of trees into low-dimensional vectors, where each vector dimension reflects the tree kernel similarity between the input tree and a landmark.

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 an 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, (*iii*) exploits the non-linearity of NNs to produce accurate models, and (*iv*) facilitates the model interpretability, as the dimensions of the input embeddings are not latent linguistic properties as in standard NNs, but directly correspond to the structured similarity between the new example and the Nyström landmarks.

A preliminary introduction of the KDA was provided in Croce et al. (2017). In this paper, we significantly extend that work by adding:

- detailed introductory material about the semantics expressed via semantic kernels that a KDA proposes to inject in the target deep learning reference methodology;
- a specific empirical investigation aiming at demonstrating that the low-dimensional embeddings generated using the Nyström method provide a very accurate approximation of the kernel functions;
- a thoughtful analysis where we will show that such embeddings are more expressive than the standard vectorial representations used in NLP, i.e., Bag-of-Words and Word Embedding based representations. This motivates our work, as the KDA is highly competitive in comparison with recently proposed Deep Networks which are generally recognized as state-of-the-art machine learning models w.r.t. language learning tasks;
- additional experiments regarding a variety of tree kernels aimed at better assessing the proposed model and studying the impact of the different kernels on the reachable accuracy.

In the rest of the paper, Section 2 surveys some of the investigated kernels. Section 3 describes the Nyström methodology and justifies its adoption by showing the rich semantic expressiveness of the produced spaces. Then the Section focuses on how to incorporate such Nyström spaces into the proposed KDA, which is an efficient and sound method for combining kernel methods and Neural models. Section 4 describes the experimental evaluation: three rather different NLP tasks are explored. Furthermore, the paper will investigate the impact of linguistic information on the performance reachable by a KDA by studying the benefits that different kernels can bring to the inference quality. Finally, Section 5 discusses some related works and derives the conclusions.

2. Kernel-based semantic inference

Several NLP tasks require the explorations of complex semantic and syntactic phenomena. For instance, in Paraphrase Detection, verifying whether two sentences are valid paraphrases involves the recognition of correspondences between sentence fragments in which syntax plays a fundamental role. In Question Answering, the syntactic information is crucial, as largely demonstrated in Croce et al. (2011). Similar needs are applicable to the Semantic Role Labeling task, that consists in the automatic discovery of linguistic predicates (together with their corresponding arguments) in texts, as discussed in Section 4.3.

In these scenarios, a possible solution is represented by the manual definition of an artificial feature set that is able to capture the syntactic and semantic aspects useful to solve a target problem.

Determining how to exploit these features to generate robust predictive models is left to the learning algorithm. However, the definition of meaningful features is still a rather expensive and complicated process that requires a

domain expert; moreover, every task has specific patterns that must be considered, making the underlying manual feature engineering an extremely complex and not portable process.

Instead of trying to design a synthetic feature space, a more natural approach consists in applying kernel methods (Robert Müller et al., 2001; Shawe-Taylor and Cristianini, 2004) on structured representations of data objects, e.g., documents. A sentence s can be represented as a parse tree[1] that expresses the grammatical relations implied by s. Tree kernels (Collins and Duffy, 2001) can be employed to directly operate on pairs of parse trees, evaluating the tree fragments shared by the two involved trees. This operation corresponds to a dot product in the implicit feature space of all possible tree fragments (which, of course, include most of the artificial features that can be engineered, such as the predicate-argument path). The dimensionality of such space is extremely large and an explicit representation is not viable due to computational and memory usage requirements: the number of different sub-fragments in a tree is combinatorial with the number of tree nodes, therefore, even a small tree can generate up to billions of features.

Whenever the dot product is available in the implicit feature space, kernel-based learning algorithms, such as SVMs (Cortes and Vapnik, 1995), can operate in order to automatically generate robust prediction models. This approach achieves state-of-the-art results in several NLP tasks, such as Semantic Role Labeling (Moschitti et al., 2008), Question Classification (Croce et al., 2011), Paraphrase Identification (Filice et al., 2015b), Recognizing Textual Entailment (Zanzotto et al., 2009; Filice et al., 2015b), and Community Question Answering (Filice et al., 2016).

Notice that given the nature of SVM-like learning algorithms, kernels allow to motivate the classifier decisions, as the mostly important (positive or negative) examples are the support vectors with the highest weights, whose inner structure is entirely captured by the kernel itself. Explaining the SVM decisions can be traced back to determining the closest (most influential) examples through the kernel. Kernels that make semantic information about a sentence explicit are good candidates as similarity metrics suitable for training complex NLP classifiers.

2.1. Semantic tree kernels

Tree Kernels (TKs) allow to estimate the similarity among texts, directly from sentence syntactic structures, that can be represented by parse trees.

The underlying basic idea is that the similarity between two trees T_1 and T_2 can be derived from the number of shared tree fragments. Let the set $\mathcal{T} = \{t_1, t_2, \ldots, t_{|\mathcal{T}|}\}$ be the space of all the possible substructures and $\chi_i(n_2)$ be an indicator function that is equal to 1 if the target t_i is rooted at the node n_2 and 0 otherwise. A tree-kernel function over T_1 and T_2 is defined as follows:

$$TK(T_1, T_2) = \sum_{n_1 \in N_{T_1}} \sum_{n_2 \in N_{T_2}} \Delta(n_1, n_2), \tag{1}$$

where N_{T_1} and N_{T_2} are the sets of nodes of T_1 and T_2, respectively, and

$$\Delta(n_1, n_2) = \sum_{k=1}^{|\mathcal{T}|} \chi_k(n_1)\chi_k(n_2) \tag{2}$$

which computes the number of common fragments between trees rooted at nodes n_1 and n_2.

The feature space generated by the structural kernels obviously depends on the input structures. Notice that different tree representations embody different linguistic theories and may produce more or less effective syntactic/semantic feature spaces for a given task. According to the phrase structure paradigm, the sentence *"What is the width of a football field"* is represented by the constituency tree shown in Fig. 1.

Notice that lexical leaves are used to represent lemmas and their coarse grained POS tags, e.g., noun (n::), verb (v::), etc., in order to distinguish between lexical items in different grammatical categories.

Dependency grammars produce a significantly different representation which is exemplified in Fig. 2. Since tree kernels are not tailored to model the labeled edges that are typical of dependency graphs, these latter are rewritten into explicit hierarchical representations. Different rewriting strategies are possible, as discussed in Croce et al. (2011). A representation that is shown to be effective in several tasks is the Grammatical Relation Centered Tree (GRCT) illustrated in Fig. 3: the PoS-Tags are children of grammatical function nodes and direct ancestors of their

[1] Parse trees can be extracted using automatic parsers. In our experiments, we used the Stanford Parser https://nlp.stanford.edu/software/lex-parser.shtml.

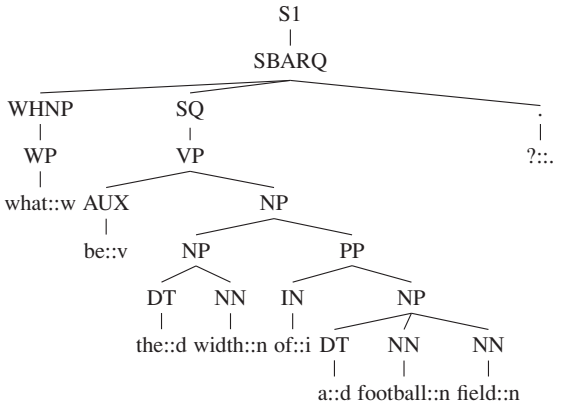

Fig. 1. Constituency Tree (CT) of the sentence *"What is the width of a football field?"*.

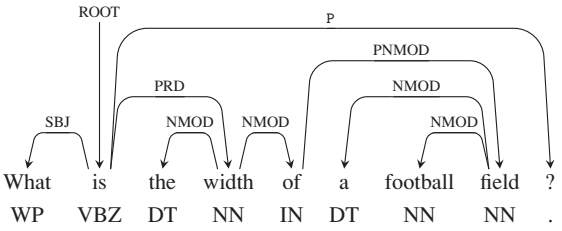

Fig. 2. Dependency parse tree of the sentence *"What is the width of a football field?"*.

associated lexical items. Another possible representation is the Lexical Only Centered Tree (LOCT) shown in Fig. 4, which contains only lexical nodes and the edges reflect some dependency relations.

Different tree kernels can be defined according to the types of tree fragments considered in the evaluation of the matching structures.

In the *Subtree Kernel* (Vishwanathan and Smola, 2002), valid fragments are only the grammatically well formed and complete subtrees: every node in a subtree corresponds to a context free rule whose left hand side is the node label and the right hand side is completely described by the node descendants. Subset trees are exploited by the *Subset Tree Kernel* (Collins and Duffy, 2001), which is usually referred to as Syntactic Tree Kernel (STK); they are more general structures since their leaves can be non-terminal symbols. The subset trees satisfy the constraint that grammatical rules cannot be broken and every tree exhaustively represents a CFG rule. For example, the constituency tree in Fig. 5a has [VP [VBD PP]] as a subset tree. Instead [VP [VBD]] is not a valid subset tree because the rule VP->VBD PP cannot be split. *Partial Tree Kernel* (PTK) (Moschitti, 2006) relaxes this constraint considering partial trees, i.e., fragments generated by the application of partial production rules (e.g., sequences of non-terminals with gaps). Examples of different kinds of tree fragments are shown in Fig. 5.

The strict constraint imposed by the STK may be problematic especially when the training dataset is small and only few syntactic tree configurations can be observed. The Partial Tree Kernel (PTK) overcomes this limitation, and usually leads to higher accuracy, as shown in Moschitti (2006).

Given a sentence s with $|s|$ words, $s[\overrightarrow{I_s}]$ defines the subsequence of s that includes the words corresponding to the indices $\overrightarrow{I_s} = \{i^s_1, \ldots, i^s_{\underset{l(I)}{\rightarrow}}\}$, where $1 \leq i^s_1 < i^s_2 < \cdots < i^s_{\underset{i(I)}{\rightarrow}} \leq |s|$, and $l(\overrightarrow{I_s})$ is the number of indices in $\overrightarrow{I_s}$.

We define $d(\overrightarrow{I}) = i_{\underset{l(I)}{\rightarrow}} - i_1$ as the length of the subsequence in the original sentence (i.e., gaps are included in this count). The PTK computation (Moschitti, 2006), is carried out by the following Δ_{PTK} function:

$$\Delta_{PTK}(n_1, n_2) = 0, \quad \text{if the labels of } n_1 \text{ and } n_2 \text{ differ}$$

$$\Delta_{PTK}(n_1, n_2) = \mu\left(\lambda^2 + \sum_{\mathbf{I}_1, \mathbf{I}_2 : l(\mathbf{I}_1) = l(\mathbf{I}_2)} \lambda^{d(\mathbf{I}_1) + d(\mathbf{I}_2)} \prod_{k=1}^{l(\mathbf{I}_1)} \Delta_{PTK}\left(c_{n_1}(i^1_k), c_{n_2}(i^2_k)\right)\right) \tag{3}$$

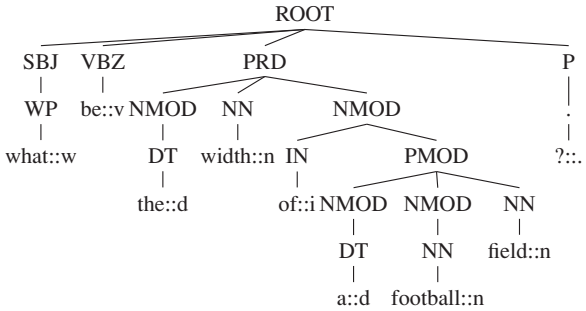

Fig. 3. Grammatical Relation Centered Tree (GRCT) of the sentence *"What is the width of a football field?"*.

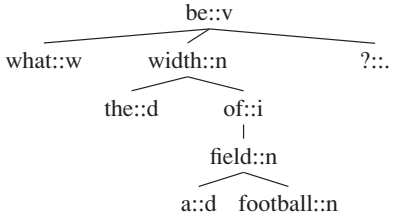

Fig. 4. Lexical Only Centered Tree (LOCT) of the sentence *"What is the width of a football field?"*.

where $c_n(k)$ is the kth child of the node n, and λ and μ are decay factors in (0,1] that penalize large child subsequences (that can include gaps) and deep partial trees, respectively.

The computational complexity of a PTK computation is $\mathcal{O}(\eta\delta^2|N_{T_1}||N_{T_2}|)$ (Moschitti, 2006), where η is the largest subsequence of children to be considered, and δ is the maximal number of children observed in the two trees. However, the average running time tends to be linear in the number of nodes for natural language syntactic trees (Moschitti, 2006). It is interesting to note that the kernel computation is bounded by the number of substructures in the parse trees of a pair of sentences that is much lower with respect to the combinatorial number of all possible tree fragments.

2.2. *Capitalizing on lexical information in convolution kernels*

The tree kernels introduced in previous section perform a hard match between nodes when comparing two substructures, i.e., n_1 and n_2 in Eq. (3). In NLP tasks, when nodes are words, this requirement results in a too strict lexical constraint, that poorly reflects semantic phenomena, such as the synonymy of different words or the polysemy of a lexical entry.

To overcome this limitation, we adopt Distributional models of Lexical Semantics (Sahlgren, 2006; Pado and Lapata, 2007; Mikolov et al., 2013; Turney and Pantel, 2010) to generalize the meaning of individual words by replacing them with geometrical representations (also called Word Embeddings) that are automatically derived from the analysis of large-scale corpora. These representations are based on the idea, introduced in the work (Harris, 1964), that words occurring in the same contexts tend to have similar meaning: the adopted distributional models generate vectors that are similar when the associated words exhibit a similar usage in large-scale document collections. Under this perspective, the distance between vectors reflects semantic relations between the represented words, such as paradigmatic relations, e.g., quasi-synonymy[2]. These word spaces allow to define meaningful soft matching between lexical nodes, in terms of the distance between their representative vectors. As a result, it is possible to obtain more informative kernel functions which are able to capture syntactic and semantic phenomena through grammatical and lexical constraints. Moreover, the supervised setting of a learning algorithm (such as SVM), operating

[2] As an example, in such spaces the vectors representing the nouns *football* and *soccer* will be near (as they are synonyms according to one of their senses) while *football* and *dog* are far

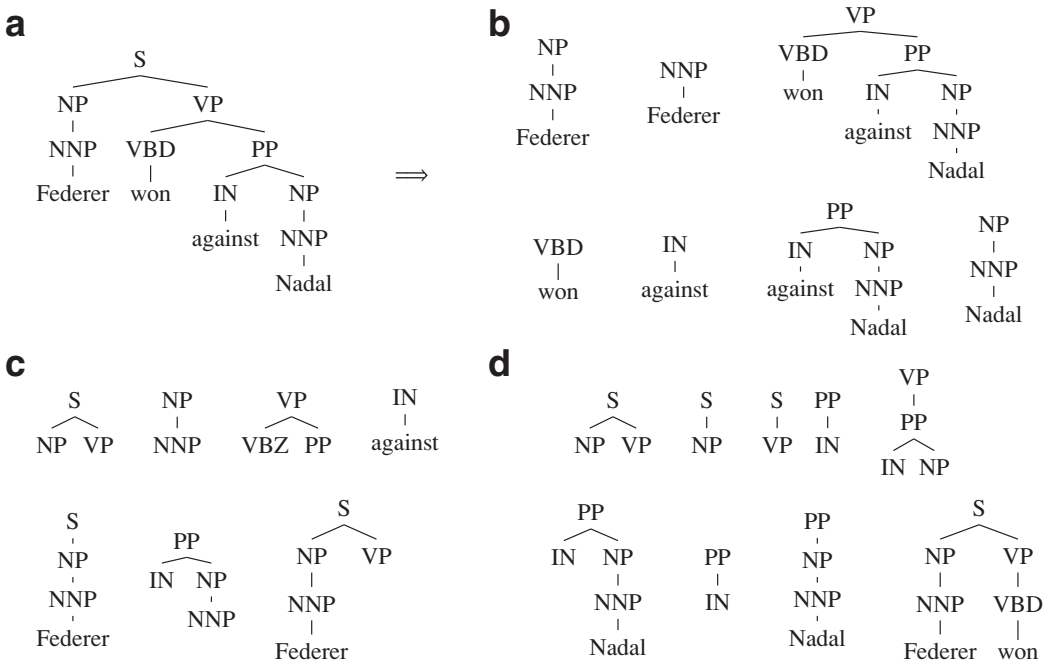

Fig. 5. (a) Constituency parse tree for the sentence " *Federer won against Nadal*", where (b) shows some valid subtrees (Vishwanathan and Smola, 2002), (c) some subset trees (Collins and Duffy, 2001) and the examples of partially specified trees, as used in the PTK formulation (Moschitti, 2006), are depicted in (d).

over the resulting kernel, is augmented with the word representations generated by the unsupervised distributional methods, thus characterizing a cost-effective semi-supervised paradigm.

The *Smoothed Partial Tree Kernel* (SPTK) described in Croce et al. (2011) exploits this idea extending the PTK formulation with a similarity function σ between nodes:

$$\begin{aligned}
&\Delta_{SPTK}(n_1, n_2) = \mu\lambda\sigma(n_1, n_2), \quad \text{if } n_1 \text{ and } n_2 \text{ are leaves} \\
&\Delta_{SPTK}(n_1, n_2) = \mu\sigma(n_1, n_2)\left(\lambda^2 + \sum_{\mathbf{I}_1, \mathbf{I}_2 : l(\mathbf{I}_1) = l(\mathbf{I}_2)} \lambda^{d(\mathbf{I}_1) + d(\mathbf{I}_2)} \prod_{k=1}^{l(\mathbf{I}_1)} \Delta_{SPTK}\left(c_{n_1}(i_k^1), c_{n_2}(i_k^2)\right)\right)
\end{aligned} \tag{4}$$

In the SPTK formulation, the similarity function $\sigma(n_1, n_2)$ between two nodes n_1 and n_2 can be defined as follows:

- if n_1 and n_2 are both lexical nodes, then $\sigma(n_1, n_2) = \sigma_{LEX}(n_1, n_2) = \tau \frac{\mathbf{v}_{n_1} \cdot \mathbf{v}_{n_2}}{\|\mathbf{v}_{n_1}\| \, \|\mathbf{v}_{n_2}\|}$. It is the cosine similarity between the word vectors $\mathbf{v}_{n_1}$ and $\mathbf{v}_{n_2}$ associated with the labels of n_1 and n_2, respectively. τ is called *terminal factor* and weights the contribution of the lexical similarity to the overall kernel computation.
- else if n_1 and n_2 are syntactic nodes sharing the same label, then $\sigma(n_1, n_2) = 1$.
- else $\sigma(n_1, n_2) = 0$.

Dealing with compositionality in tree kernels. The main limitations of the SPTK are that (*i*) lexical semantic information only relies on the vector metrics applied to the leaves in a context free fashion and (*ii*) the semantic compositions between words are neglected in the kernel computation, that only depends on their grammatical labels.

In Annesi et al. (2014) a solution for overcoming these issues is proposed. The pursued idea is that the semantics of a specific word depends on its context. For example, in the sentence, "*What instrument does Hendrix play?*", the role of the word *instrument* is fully captured if its composition with the verb *play* is taken into account. Such combination of lexical semantic information can be directly expressed into the tree structures, as shown in Fig. 6. The resulting representation is a compositional extension of a GRCT structure, where the original label d_n of grammatical function nodes n (i.e., dependency relations in the tree) is augmented by also denoting their corresponding head/modifier pairs (h_n, m_n).

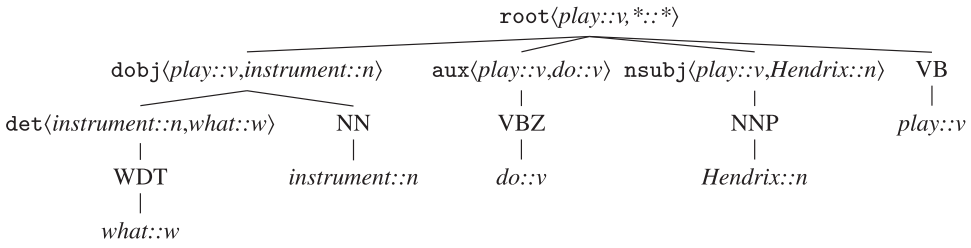

Fig. 6. Compositional grammatical relation centered tree (CGRCT) of the sentence "*What instrument does Hendrix play?*".

In CGRCTs, (sub)trees rooted at dependency nodes can be used to provide a contribution to the kernel that is a function of the composition of vectors, $\mathbf{h}$ and $\mathbf{m}$, expressing the lexical semantics of the head h and modifier m, respectively. Several algebraic functions have been proposed in Annesi et al. (2014) to compose the vectors of h=l^h::pos^h and m=l^m::pos^m into a vector $\mathbf{c}^{h,m}$ representing the head modifier pair $c = \langle$ l^h::pos^h, l^m::pos$^m\rangle$, in line with the research on Compositional Distributional Semantics (e.g., Mitchell and Lapata (2010)). In this work, we investigated the additive function (according to the notation proposed in Mitchell and Lapata, 2010) that assigns to a head/modifier pair c the vector resulting from the linear combination of the vectors representing the head and the modifier, i.e., $\mathbf{c}^{h,m} = \alpha\mathbf{h} + \beta\mathbf{m}$. Although this composition method is very simple and efficient, it actually produces very effective kernel functions, as demonstrated in Annesi et al. (2014); Filice et al. (2015a).

According to the CGRCT structures, Annesi et al. (2014) defines the Compositionally Smoothed Partial Tree Kernel (CSPTK). The core novelty of the CSPTK is the compositionally enriched estimation of the function σ, as described in Algorithm 1. The function σ can be applied to lexical nodes, to POS tag nodes as well as to augmented dependency nodes. In the algorithm the three cases are defined. For simple lexical nodes, σ consists of a lexical kernel σ_{LEX}, such as the cosine similarity between word vectors (sharing the same POS-tag): this is equivalent to Croce et al. (2011). For POS nodes σ consists of the identity function that is 1 only when the same POS is matched and it is 0 elsewhere.

Algorithm 1. $\sigma(n_1, n_2)$ Compositional estimation of the lexical contribution to semantic tree kernels.

Algorithm 1: $\sigma(n_1, n_2)$ Compositional estimation of the lexical contribution to semantic tree kernels

$\sigma \leftarrow 0$,
/*Matching between simple lexical nodes*/
if n_1 and n_2 are lexical nodes **then**
  $\sigma \leftarrow \sigma_{LEX}(n_1, n_2) = \tau \frac{\mathbf{v}_{n_1} \cdot \mathbf{v}_{n_2}}{||\mathbf{v}_{n_1}|| ||\mathbf{v}_{n_2}||}$
end if
/*Matching between identical POS tag nodes*/
if (n_1 is a POS tag) **and** ($n_1 = n_2$) **then**
  $\sigma \leftarrow 1$
end if
/*Matching between fully instantiated dependency nodes*/
if $n_1 = \langle d, h_1, m_1\rangle$ **and** $n_2 = \langle d, h_2, m_2\rangle$ **then**
  $\sigma \leftarrow \sigma_{Comp}((h_1, m_1), (h_2, m_2))$
end if
/*Matching between dependency nodes with missing modifiers*/
if $n_1 = \langle d, h_1, *\rangle$ **and** $n_2 = \langle d, h_2, *\rangle$ **then**
  $\sigma \leftarrow \sigma_{LEX}(h_1, h_2)$
end if
/*Matching between a fully instantiated dependency node
and a dependency node with missing modifier*/
if $n_1 = \langle d, h_1, m_1\rangle$ **and** $n_2 = \langle d, h_2, *\rangle$ **then**
  $\sigma \leftarrow \sigma_{Comp}((h_1, m_1), (h_2, h_2))$
end if
return σ

The novel part of Algorithm 1 corresponds to the compositional treatment of two dependency nodes, $n_1 = \langle d_1, h_1, m_1 \rangle$ and $n_2 = \langle d_2, h_2, m_2 \rangle$. The similarity function σ in this case corresponds to a compositional function σ_{Comp} between the two nodes. σ_{Comp} is not null only when the two nodes exhibit the same dependency relation, i.e., $d = d_1 = d_2$, so that also the respective heads and modifiers share the same POS labels. In all these cases a compositional metric is applied over the two involved (h_i, m_i) compounds. In the simple case, the cosine similarity between the two vectors $\mathbf{c_i}^{h_i,m_i} = \alpha\mathbf{h_i} + \beta\mathbf{m_i}$, i=1,2, is applied. Other metrics correspond to more complex compositions $\Psi((\mathbf{h_1}, \mathbf{m_1}), (\mathbf{h_2}, \mathbf{m_2}))$ that account for linear algebra operators among the four vectors. Notice that two trees may exhibit equivalent dependencies of different complexity so that Algorithm 1 also accounts for partial matches. The full details on the definition of σ_{Comp} are in Annesi et al. (2014).

It is worth emphasizing that semantic kernels, such as SPTK and CSPTK, depend explicitly on syntactic and semantic information. They can be thus used to connect the classifier output to an input, e.g. through the involved support vectors whose parse tree nodes are providing explicit linguistic information, in order to semantically justify the classifier decisions. The most important (positive or negative) examples correspond to maxima of the kernel function and are related to the support vectors semantics. In this perspective, providing a semantic explanation of the SVM decision corresponds to tracing back the closest support vectors, whose semantics mostly contributed to the classifier outcome, via the kernel function.

2.3. Computational drawbacks in adopting complex kernels on large datasets

As discussed in Section 1, the complexity of kernel-based methods may prevent their adoption in real applications, especially when a significant amount of data is involved. First of all, the training phase of an optimal maximum margin learning algorithm (such as SVM) requires a number of kernel operations that is more than linear (almost $\mathcal{O}(|\mathcal{D}|^2)$) with respect to the cardinality of training examples $\mathcal{D}$, as discussed in Chang and Lin (2011). Also complexity of the classification phase is related to the size of the input dataset and the intrinsic complexity of the targeted task. Although a formal bound to the number of kernel operations required in the classification phase cannot be easily defined, classifying a new instance requires evaluating the kernel function with respect to each support vector, i.e., the examples selected by the learning algorithm to be representative of the final decision function (Vapnik, 1998). This number depends on the difficulty of the targeted task, but the average number of support vectors tends to grow linearly with the number of examples observed in the training phase.

The final decision is the weighted sum of these kernel operations. Even if it can be easy parallelized, a model made of thousands of support vectors may be quite challenging. This cost is even more problematic if considering the cost of a single kernel operation. While custom kernels operating on feature vectors, such as the RBF kernel, are basically equivalent to a dot product, structural kernels have a cost that depends on the size and dimension of the involved discrete structures. Considering the kernels described in the previous sections, the cost of evaluating a single Subtree Kernel or a Partial Tree Kernel is almost linear in the number of nodes of the input trees, as discussed in Moschitti (2006). This cost is higher when considering the Smoothed Partial Tree Kernel, that is more than linear in the number of nodes, as empirically measured in Croce et al. (2011).

The aim of the Nyström method presented in the following section is thus to improve the efficiency of both training and testing phases, through (*i*) a drastic reduction of the required kernel operations, and (*ii*) the adoption of linear learning approaches, such as SVMs. In particular, SVM solvers, such as the Dual Coordinate Descent (DCD) (Hsieh et al., 2008), are well-known for being much faster than their kernel-based counterparts. As a result, the proposed method enables interesting learning possibilities, including scaling to larger datasets with simpler learning algorithms as well as using more complex kernel functions.

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 Φ over all examples from $\mathcal{D}$ to derive representations, $\mathbf{x}$ denoting the rows of the matrix $\mathbf{X}$. The Gram matrix can always be computed as $\mathbf{G} = \mathbf{X}\mathbf{X}^\top$, with each single element corresponding to $\mathbf{G}_{ij} = \Phi(o_i)\Phi(o_j) = K(o_i, o_j)$. The aim of the Nyström method (Drineas and Mahoney, 2005) is to derive a new low-dimensional embedding $\widetilde{\mathbf{x}}$ in a l-dimensional space, with $l \ll n$ so that $\widetilde{\mathbf{G}} = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^\top$ and $\widetilde{\mathbf{G}} \approx \mathbf{G}$. This is obtained by generating an approximation $\widetilde{\mathbf{G}}$ of $\mathbf{G}$ 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 $\mathbf{G}$, and let $\mathbf{C} \in \mathbb{R}^{|D| \times l}$ be the matrix of these sampled columns. Then, we can rearrange the columns and rows of $\mathbf{G}$ and define $\mathbf{X} = [\mathbf{X}_1 \quad \mathbf{X}_2]$ such that:

$$\mathbf{G} = \mathbf{X}\mathbf{X}^\top = \begin{bmatrix} \mathbf{W} & \mathbf{X}_1^\top \mathbf{X}_2 \\ \mathbf{X}_2^\top \mathbf{X}_1 & \mathbf{X}_2^\top \mathbf{X}_2 \end{bmatrix} \quad \text{and} \quad \mathbf{C} = \begin{bmatrix} \mathbf{W} \\ \mathbf{X}_2^\top \mathbf{X}_1 \end{bmatrix} \tag{5}$$

where $\mathbf{W} = \mathbf{X}_1^\top \mathbf{X}_1$, i.e., the subset of $\mathbf{G}$ that contains only landmarks. The Nyström approximation can be defined as:

$$\mathbf{G} \approx \widetilde{\mathbf{G}} = \mathbf{C}\mathbf{W}^\dagger \mathbf{C}^\top \tag{6}$$

where $\mathbf{W}^\dagger$ denotes the Moore–Penrose inverse of $\mathbf{W}$. The Singular Value Decomposition (SVD) is used to obtain $\mathbf{W}^\dagger$ as it follows. First, $\mathbf{W}$ is decomposed so that $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are both orthogonal matrices, and $\mathbf{S}$ is a diagonal matrix containing the (non-zero) singular values of $\mathbf{W}$ on its diagonal. Since $\mathbf{W}$ is symmetric and positive definite $\mathbf{W} = \mathbf{U}\mathbf{S}\mathbf{U}^\top$. Then $\mathbf{W}^\dagger = \mathbf{U}\mathbf{S}^{-1}\mathbf{U}^\top = \mathbf{U}\mathbf{S}^{-\frac{1}{2}}\mathbf{S}^{-\frac{1}{2}}\mathbf{U}^\top$ and the Eq. (6) can be rewritten as

$$\mathbf{G} \approx \widetilde{\mathbf{G}} = \mathbf{C}\mathbf{U}\mathbf{S}^{-\frac{1}{2}}\mathbf{S}^{-\frac{1}{2}}\mathbf{U}^\top \mathbf{C}^\top = \left(\mathbf{C}\mathbf{U}\mathbf{S}^{-\frac{1}{2}}\right)\left(\mathbf{C}\mathbf{U}\mathbf{S}^{-\frac{1}{2}}\right)^\top = \widetilde{\mathbf{X}}\widetilde{\mathbf{X}}^\top \tag{7}$$

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

$$\widetilde{\mathbf{x}} = \mathbf{c}\mathbf{U}\mathbf{S}^{-\frac{1}{2}}, \tag{8}$$

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 Eq. (8). 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 $\mathcal{O}(kl)$. For generating of the projection matrix $\mathbf{U}\mathbf{S}^{-\frac{1}{2}}$ we need first to evaluate $\mathbf{W}$, and then to decompose it using the SVD algorithm. The evaluation of $\mathbf{W}$ requires the computation of all the pairwise kernel operations between the landmarks for matrix $\mathbf{W}$, therefore its computational cost is $\mathcal{O}(kl^2)$. Instead, the SVD cost using standard linear algebra libraries is $\mathcal{O}(l^3)$. Considering again that l is typically few hundreds, the overall cost for generating the Nyström projection matrix is quite low, almost negligible if compared to the training cost of a kernel based SVM on a large dataset.

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. On the expressiveness of different semantic kernel spaces

The tree kernel functions discussed in Section 2 produce very rich feature spaces where learning algorithms can acquire accurate models for semantic inferences. Moreover, in the previous Section, we introduced the Nyström method as an efficient technique to approximate such spaces in terms of low dimensional embeddings, that

Table 1
Analysis of the Semantic Textual Similarity task.

Model	Pearson
$\cos_{BoW}$	0.077
$\cos_{W2V}$	0.086
PTK	0.202
Ny_{300}^{PTK}	0.189
Ny_{400}^{PTK}	0.202
SPTK	0.262
Ny_{300}^{SPTK}	0.252
Ny_{400}^{SPTK}	0.263

reconstruct the implicit features generated by kernels. In this Section, we want to support the kernel formulations provided in the previous chapters via an empirical analysis that aims at confirming that (*i*) adopted semantic kernels are very effective in capturing semantic and syntactic aspects of sentences, (*ii*) the low dimensional embeddings produced by the Nyström method preserve the expressiveness of the original kernel spaces.

For this reason, we first investigate the application of kernels and Nyström embeddings over the task of Semantic Textual Similarity (Agirre et al., 2016) that is representative of the overall grammatical and semantic phenomena expressed by natural language sentences. We will verify that the tree kernel operation, as well as its Nyström approximation, can effectively capture the semantic similarity between two sentences.

Then, we will use these similarities as the basis of a clustering process over sentences: this will allow us to verify if the topology of the embedding spaces is still able to group texts in agreement with human intuition. In particular, we will focus on the clustering of questions, whose data are well understood and widely used for empirical research.

Expressive kernels for semantic textual similarity. Semantic Textual Similarity (STS) is the task of measuring the degree of equivalence in the underlying semantics of two snippets of text. This assessment is performed using an ordinal scale that ranges from complete semantic equivalence to complete semantic dissimilarity.

State-of-the-art systems in STS are based on supervised methods that exploit rich features sets, complex alignment models and deep learning techniques (e.g., Rychalska et al., 2016; Brychcín and Svoboda, 2016). In this analysis we do not aim at competing with such systems. We just want to demonstrate that the adopted kernel functions provide a good indicator of the semantic relatedness between two sentences: in a completely unsupervised fashion, we will evaluate the semantic similarity between two sentences by directly using the tree kernel functions. Then, we will verify whether such similarity correlates with the similarity scores provided by the annotators. To run this analysis we adopted the question-question portion of the STS dataset from SemEval-2016 (Agirre et al., 2016). It includes 209 question pairs extracted from the Stack Exchange Data Dump,[3] whose topics range from highly technical areas such as programming and mathematics, to more casual topics like cooking and fitness. Human assigned gold labels range from 1 (semantic dissimilarity) to 4 (semantic equivalence). Table 1 reports the Pearson correlation to the gold labels of different kernel similarities. We include two baselines model to better assess our results. The $\cos_{BoW}$ is the cosine similarity of bag-of-words vectors. These vectors consider only lexical information as their dimensions reflect the occurrences of words into a text, totally ignoring word ordering or syntactic information. This produces a high-dimensional sparse space (with as many dimensions as words in a dictionary) in which matching between different but semantically related words is completely neglected. Word Spaces can capture this linguistic information, as discussed in Section 2.2, where words are represented via low-dimensional embeddings where distance reflects semantic relations among represented lexical items (Sahlgren (2006)). Here, $\cos_{W2V}$ is the cosine similarity of the vectors obtained by averaging the word embeddings associated to the words of each sentence. We used 250-dimensional word embeddings generated by applying the Word2vec tool with a Skip-gram model (Mikolov et al., 2013) to the entire Wikipedia.

The poor result achieved by the $\cos_{BoW}$ suggests that lexical overlap between texts is not particularly beneficial in this task. $\cos_{W2V}$ obtains a similar Pearson correlation: word embeddings need a better way to be combined, by using for instance the syntactic information (the SPTK is actually a way to achieve such target). We then investigated tree

[3] https://archive.org/details/stackexchange

Table 2
Some pairs from the STS dataset. They are sorted with respect to their gold label similarity. The last four columns indicate their ranking position with respect to different models. In case of ties multiple positions are reported.

		Rank			
Sentence 1	Sentence 2	Gold	$\cos_{BoW}$	$\cos_{W2V}$	Ny_{300}^{SPTK}
How do I remove paint from a wood floor?	*How do I remove paint from a porous table top?*	4	1	3	4
How do I remove paint from a wood floor?	*How do I remove a thick layer of paint from tiles?*	3	3–4	1	3
How do I remove paint from a wood floor?	*How can I remove paint from a deck?*	2	2	4	2
How do I remove paint from a wood floor?	*How can I remove small paint specks from a wooden floor?*	1	3–4	2	1

kernels[4] on LOCT tree representation, where all nodes are words, and edges reflect dependency relations. Such syntactic information is crucial: both PTK and SPTK largely improve the baselines. The similarity score between two questions is measured in terms of the kernel function between the corresponding parse trees, without any kind of supervision.

Most importantly, when the Nyström approximation of the kernel spaces is generated, overall results are not impacted. An approximated semantic kernel space generated by using only 300 landmarks, i.e., Ny_{300}^{PTK} and Ny_{300}^{SPTK}, achieve a Pearson Correlation which is only slightly lower than the one achieved by the corresponding tree kernels, while using 400 landmarks, i.e., Ny_{400}^{PTK} and Ny_{400}^{SPTK}, no difference is observed. This demonstrates that the sentence embeddings derived by applying the Nyström method to tree kernel spaces are a semantically rich representation for text, which is largely more expressive than common text representations, such as the Bag-of-words model.

To better appreciate the impact of different representations we reported a few example pairs in Table 2. Pairs are sorted w.r.t. their gold label similarity. While $\cos_{BoW}$ and $\cos_{W2V}$ models introduce many errors in their rankings, the Ny_{300}^{SPTK} produces the correct ranking. In particular, the $\cos_{BoW}$ cannot match semantically similar words such as *wood* and *wooden*, resulting in a poor similarity between the last pair, i.e., the one with the highest gold label similarity. Conversely, $\cos_{W2V}$ can capture this kind of matches, however its results are still low. Probably using the average vector for combining word embeddings is not a good choice: the syntactic information of the question is completely ignored and the word embeddings have the same contribution, regardless their syntactic/semantic role in the sentence. The Ny_{300}^{SPTK}, approximating a tree kernel operating on syntactic trees,[5] overcomes this limit, as demonstrated by its good results.

Clustering linguistic structure in semantic kernel spaces. In order to prove the expressiveness of the generated semantic space, we also investigated the application of clustering techniques within the approximated Nyström spaces. The positive impact of Kernel-Based clustering methods has been already demonstrated in several works, such as Schölkopf et al. (1998) and Kulis et al. (2005), where kernel functions enable the effective clustering of data even when complex and/or non-linear topologies are involved.

To run our experiments, we used the Question Classification (QC) task, which consists of mapping a question into a closed set of answer types in a Question Answering system. As an example a user the question "*Who is the President of Pergament?*" expect an answer referring to a HUMAN being. The adopted UIUC dataset (Li and Roth, 2006) includes a training and test set of 5452 and 500 questions, respectively, organized in 6 classes (like ENTITY or HUMAN). Tree Kernels were very effective, as shown in Croce et al. (2011) and Annesi et al. (2014).

We employed the clustering methods formulated in Kulis et al. (2005) and implemented in KeLP.[6] We first applied a traditional K-means algorithm in the explicit geometrical space generated by the BoW representation of questions. Then, we evaluated a Kernel-based K-means formulation adopting a CSPTK kernel applied to the CGRCT representation reported in Section 2.2.[7] Finally, we approximated the above kernel function by using 500 landmarks. We evaluated the clustering quality in terms of purity, i.e., the percentage of the most frequent class in each cluster.

[4] We used default values for the kernel parameters λ and μ, both set to 0.4. The terminal factor has been tuned via grid-search, and the estimated optimal value was 0.001.

[5] For the examples in Table 2 the Ny_{300}^{SPTK} ranking corresponds to the one produced by the SPTK.

[6] http://www.kelp-ml.org/?page_id=799

[7] This configuration obtained best results in the experimental evaluation reported in Section 4.1.

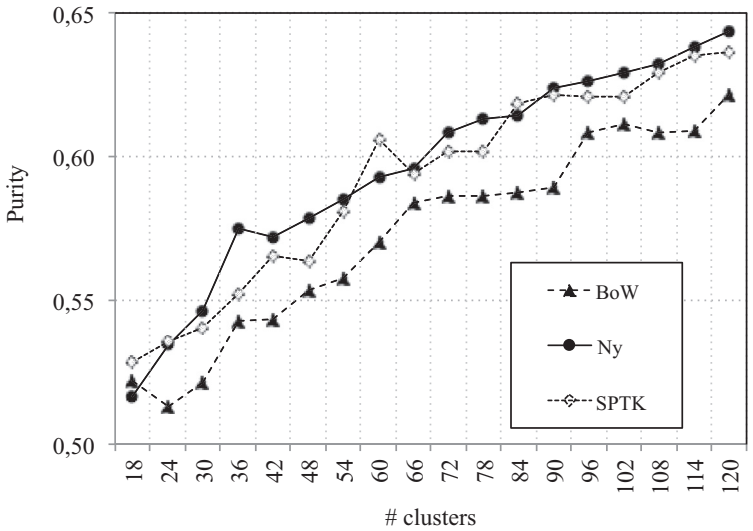

Fig. 7. Cluster purity w.r.t. the number of clusters on the task of question classification.

Table 3
Example of question clusters in the semantic kernel space.

Cluster 1		Cluster 2	
DESC	*What are vermicilli, rigati, zitoni, and tubetti?*	HUM	*Who is the President of Pergament?*
DESC	*What are liver enzymes?*	HUM	*Who is the leader of Brunei?*
DESC	*What are amaretto biscuits?*	HUM	*Who is the president of Bolivia?*
DESC	*What are tonsils for?*	HUM	*Who is the President of Ghana?*
DESC	*What are hook worms?*	HUM	*Who is the leader of India?*
DESC	*What are some chemical properties of mendelevium?*	HUM	*Who was the president of Vichy France?*
ENTY	*What are birds descendents of?*	HUM	*Who was the 1st U.S. President?*
DESC	*What are some children 's rights?*	HUM	*Who is the prime minister of Japan?*
		HUM	*Who was the oldest U.S. president?*
Cluster 3		**Cluster 4**	
LOC	*What two countries ' coastlines border the Bay of Biscay?*	ENTY	*What basketball maneuver did Bert Loomis invent?*
LOC	*What country is bounded in part by the Indian Ocean and Coral and Tasman seas?*	HUM	*What college did Joe Namath play football for?*
LOC	*What country do the Galapagos Islands belong to?*	HUM	*What hockey team did Wayne Gretzky play for?*
LOC	*What part of Britain comprises the Highlands, Central Lowlands, and Southern Uplands?*	HUM	*What dumb-but-loveable character did Maurice Gosfield play on The Phil Silvers Show?*
LOC	*What two Caribbean countries share the island of Hispaniola?*	HUM	*What Cruise Line does Kathie Lee Gifford advertise for?*
LOC	*What country surrounds San Marino, the world 's smallest Republic?*	HUM	*What team did baseball 's St. Louis Browns become?*
LOC	*What mountain range marks the border of France and Spain?*	ENTY	*What war did Johnny Reb and Billy Yank fight?*
LOC	*What strait links the Mediterranean Sea and the Atlantic Ocean?*	HUM	*What feathered cartoon characters do Yugoslavians know as Vlaja, Gaja, and Raja?*
LOC	*What U.S. state includes the San Juan Islands?*	HUM	*What college did Dikembe Mutombo play basketball for?*

Fig. 7 shows the purity obtained with different values of the clustering parameter *K*. Since the seed of the K-means formulation and the selection of the landmarks are random, we iterated this evaluation 5 times and reported the average purity across the iterations. The plot clearly shows that the adoption of SPTK improves the purity w.r.t. the *BoW* representation. Most importantly, the results achieved in the approximated space (the *Ny* curve) overlap the result achieved by the kernel counterpart, demonstrating that the Nyström approximation does not introduce any gap by obscuring of significant information.

A deeper analysis of the clusters obtained in the reduced space is reported in Table 3, which reports 4 of the 100 clusters obtained by the standard K-mean algorithm over the approximated Nyström space.

The syntactic information captured by the tree kernel is clearly shown by the items in the first two clusters, that are in the form "*What are/is*" and "*What is*". Most noticeably, in the second cluster all questions refer to *leader*s,

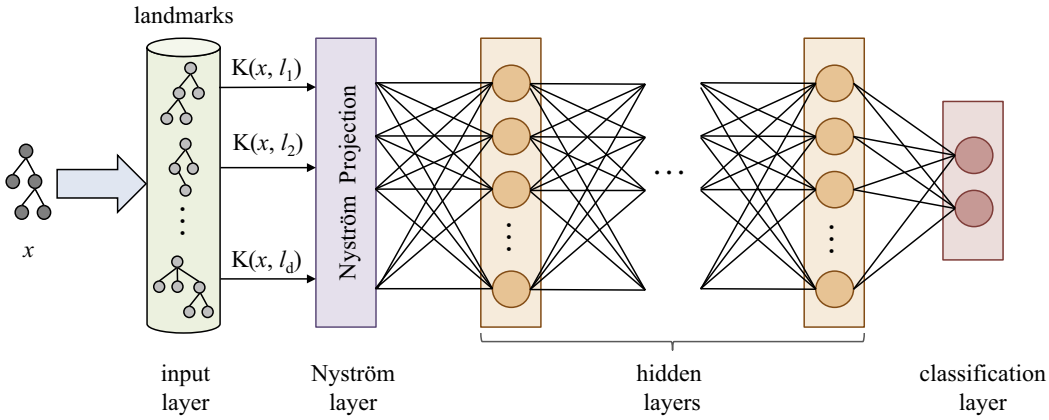

Fig. 8. Kernel-based deep architecture.

presidents or *prime minister*, as these are semantically related according the adopted lexical word embedding. The third and forth clusters are more interesting because they do not contain questions sharing the same structure, but the linguistic generalization is more evident, where locations (such as *countries* and *mountains*) and group of people are addressed by the questions. Most importantly, this combination of lexical, syntactic and semantic information is coded in the 500 dimensions of the approximated kernel space.

The derived clusters are very expressive in linguistic terms. In fact almost all clusters correspond more or less explicitly to one or more syntactic-semantic patterns, such as

Cluster 3.
What [LOC] {*border, surround, is bounded by, comprise*} [LOC]?
or **Cluster 4.**
What [HUM]*(did)* [HUM]*{become}*?
What [HUM]*{did, does}* [HUM] { { *play* } [sport]}, *advertise*} {*for*}?

3.3. A kernel-based deep architecture

As discussed in Section 3.1, the Nyström representation $\widetilde{\mathbf{x}}$ 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. (8). We will refer to this architecture, shown in Fig. 8, 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 $\mathbf{C}$ matrix associated to an example o. Notice that, for adopting the KDA, the values of the matrix $\mathbf{C}$ should be all available. In the training stage, these values are usually 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 of the l landmarks.

The input layer is mapped to the *Nyström* layer, through the projection in Eq. (8). Notice that the embedding provides also the proper weights, defined by $\mathbf{US}^{-\frac{1}{2}}$, so that the mapping can be expressed through the Nyström matrix $\mathbf{H}_{Ny} = \mathbf{US}^{-\frac{1}{2}}$: it corresponds to a pre-trained stage derived through SVD, as discussed in Section 3.1. Eq. (8) provides a static definition for $\mathbf{H}_{Ny}$ whose weights can be left invariant during the neural network training. However, the values of $\mathbf{H}_{Ny}$ can be made available for the standard back-propagation adjustments applied for training.[8] Formally, the low-dimensional embedding of an input example o, is $\widetilde{\mathbf{x}} = \mathbf{c}\ \mathbf{H}_{Ny} = \mathbf{c}\ \mathbf{US}^{-\frac{1}{2}}$.

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

[8] In our preliminary experiments, adjustments to the $\mathbf{H}_{Ny}$ 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.

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

$$\mathbf{x}_t = f(\mathbf{x}_{t-1}\mathbf{H}_t + \mathbf{b}_t) \tag{9}$$

The final layer of KDA is the *classification layer*, realized through the output matrix $\mathbf{H}_O$ 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., $\mathbf{H}_O \in \mathbb{R}^{h_{O_{-1}} \times |Y|}$ 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 $\widehat{y} = softmax(\mathbf{x}_{O_{-1}}\mathbf{H}_O + \mathbf{b}_O)$.

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[9] $\mathbf{H}_t$ and $\mathbf{H}_O$.

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 λ. The final loss function is thus

$$L(y, \widehat{y}) = \sum_{(o,y) \in \mathcal{L}} y \; log(\widehat{y}) + \lambda \sum_{\mathbf{H} \in \{\mathbf{H}_t\} \cup \{\mathbf{H}_O\}} \| \mathbf{H} \|^2$$

where $\widehat{y}$ 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

We conducted an extensive experimental investigation in order to demonstrate that the proposed KDA is an effective solution for combining the expressiveness of kernel methods with the powerful learning capabilities of Deep Learning. Furthermore, we will show that the KDA is very efficient and that it can easily scale to very large datasets. Finally, we investigated the impact of linguistic information on the performance reachable by a KDA by studying the benefits that different kernels (each characterized by a growing expressive power) can bring to the accuracy in semantic inference tasks.

We explored three NLP tasks, namely Question Classification, Community Question Answering, and Automatic Boundary Detection in Semantic Role Labeling. We adopted the same architecture, without major differences, and the good performance obtained in these rather different tasks clearly confirm that the proposed framework is a general solution with an extremely large applicability.

General experimental settings: the Nyström projector has been implemented in the KeLP framework.[10] The neural network has been implemented in Tensorflow,[11] 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 λ 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 with fixed sizes. Results averaged against 5 such samplings are always hereafter reported. In the three tasks the only difference in the KDA configuration are the adopted Kernels, that will be described specifically for each task.

[9] The input layer and the Nyström layer are not modified during the learning process, and they are not regularized.
[10] http://www.kelp-ml.org, presented in Filice et al. (2018).
[11] https://www.tensorflow.org/

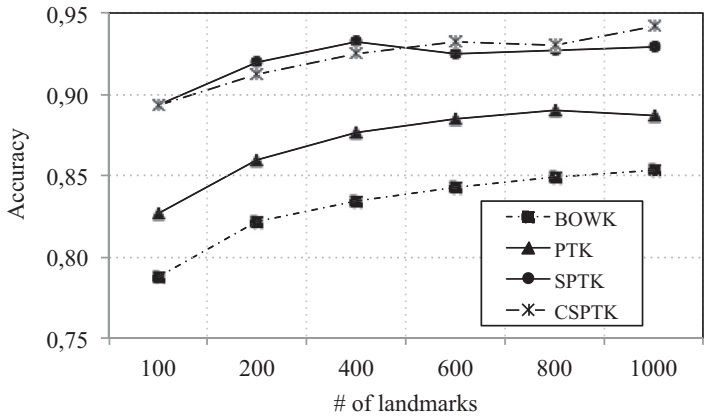

Fig. 9. QC task - accuracy measure curves w.r.t. the number of landmarks.

4.1. Question classification

In Section 3.2 we already introduced the Question Classification (QC) task and the UIUC dataset (Li and Roth, 2006). In that case, we used a completely unsupervised approach to show the quality of the low-dimensional semantic spaces generated by the Nyström methodology applied to tree kernel functions. In particular, we projected individual questions into the Nyström space and we showed that the resulting embeddings naturally group into clusters correlated with the 6 gold question classes. The UIUC dataset includes 5,452 and 500 questions as the training and test set, respectively.

In the following experiments, instead, we aim at understanding the impact of different kernels into the proposed KDA framework. The input vectors for the KDA are modeled using the Nyström method (with different kernels) based on a number of landmarks ranging from 100 to 1000. We tried different kernels with increasing expressiveness:

- BOWK: a linear kernel applied over bag-of-words vectors having lemmas as dimensions. It provides a pure lexical similarity.
- PTK: the partial tree kernel over the GRCT representations. It provides a lexical and syntactic similarity.
- SPTK: the smoothed partial tree kernel over the GRCT representations. It improves the reasoning of the PTK by including the semantic information derived by word embeddings.
- CSPTK: the compositionally smoothed partial tree kernel over the GRCT representations. It adds the semantic compositionality to the SPTK.

In the SPTK and CSPTK we used 250-dimensional word vectors generated by applying the Word2vec tool with a Skip-gram model (Mikolov et al., 2013) to the entire Wikipedia. The tree kernels have default parameters (i.e., $\mu = \lambda = 0.4$).

Fig. 9 shows the impact of different kernels in the proposed KDA model. Curves are computed by changing the number of landmarks in the Nyström formulation. The increasing complexity of the investigated kernels directly reflects on the accuracy achieved by the KDA. The BOWK is the simplest kernel and obtain poor results: it needs 800 landmarks to reach 85% of accuracy.

The contribution of the syntactic information provided by tree kernels is straightforward. The PTK achieves about 90% of accuracy starting from 600 landmarks. These results are improved by SPTK and CSPTK when the semantic information of the word embeddings is employed: even when only 100 landmarks are used, the KDA using these kernels can obtain 90% of accuracy and overcomes 94% with more landmarks. These achievements demonstrate that the KDA results directly depend on the involved kernel functions and that the improvement guaranteed by using a more expressive kernel cannot be obtained by the non-linear learning of the Neural Network.

We also performed a second set of experiments to show that (*i*) the proposed KDA is far more efficient than a pure kernel-based approach, and (*ii*) the powerful non-linear learning provided by the neural networks is necessary to take

Table 4
Results in terms of accuracy and saving in the question classification task.

Model	#Land.	Accuracy	Saving
CNN Kim (2014)	–	93.6%	–
LSTM Zhou et al. (2015)	–	93.2%	–
BiLSTM Zhou et al. (2015)	–	93.0%	–
C-LSTM Zhou et al. (2015)	–	94.6%	–
SVM_{ker}	–	95.0%	0.0%
	100	88.5% (84.1%)	97.4%
	200	92.2% (88.7%)	94.8%
	400	93.7% (91.6%)	89.7%
KDA (SVM_{lin})	600	**94.3%**(92.8%)	**84.5%**
	800	94.3% (93.0%)	79.3%
	1000	94.2% (93.6%)	74.2%

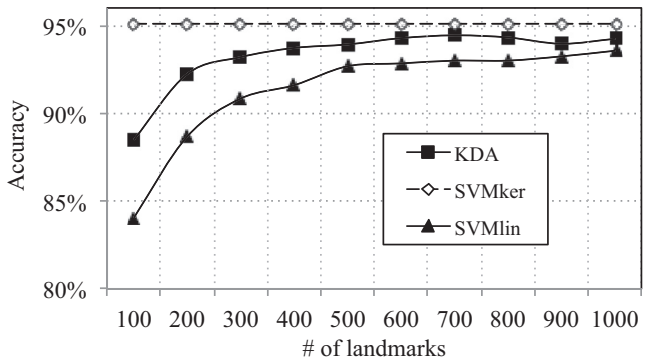

Fig. 10. QC task - accuracy curves w.r.t. the number of landmarks.

the best from the Nyström embeddings and achieve higher accuracy. In this case we focused on the most accurate kernel, i.e., the CSPTK. The kernel-based SVM formulation by Chang and Lin (2011), fed with the CSPTK (hereafter SVM_{ker}), 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.

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_{lin}.

Results are reported in Table 4: computational saving refers to the percentage of avoided kernel computations with respect to the application of the SVM_{ker} to classify each test instance. We also measured the state-of-the-art Convolutional Neural Network[12] (CNN) of Kim (2014), achieving the remarkable accuracy of 93.6%. Zhou et al. (2015) reports results from different version of Long-Short Term Memory (LSTM), including a Bidirectional LSTM (BiLSTM) and the combination of a Convolutional and a Recurrent Neural Network (namely C-LSTM) that leads to an Accuracy of 94.6%.

Notice that the linear classifier SVM_{lin} operating over the approximated kernel space achieves the same classification quality of the CNN when just 1,000 landmarks are considered. KDA improves these results, achieving 94.3% accuracy even with fewer landmarks (only 600), showing the effectiveness of non-linear learning over the Nyström input. Although SVM_{ker} 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*

[12] 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. A higher result is shown in Zhang et al. (2016) where a CNN is combined with a Recursive Neural Networks in the so-called DSCNN, leading to an accuracy of 95.4%: unfortunately this last work is not evaluated using the official train/test split and a direct comparison is not easily feasible.

Q: ***Can I obtain Driving License my QID is written Employee***
Can I obtain Driving license my QID is written Employee, i saw list in gulf times bit there isnt mentioned EMPLOYEE QID Profession.

$\mathbf{A}_1$ *the word employee is a general term that refers to all the staff in your company either the manager, secretary up to the lowest position or whatever positions they have. you are all considered employees of your company.*

$\mathbf{A}_2$ *your qid should specify what is the actual profession you have. i think for me, your chances to have a drivers license is low.*

$\mathbf{A}_3$ *dear richard, his asking if he can obtain. means he have the driver license*

$\mathbf{A}_4$ *Slim chance . . .*

Fig. 11. Example from SemEval-2015 Task 3.

the analysis of questions and can be used as a sort of pre-training strategy. Moreover, even if the application of the KDA does not outperform the results obtained by the C-LSTM (even if the results are almost the same), it is worth noting that the proposed classifier is a very simple multi-layered feed-forward network applied in a very informative space. Further extensions which use more complex architectures in such spaces represent an important research direction). Fig. 10 shows the accuracy curves according to various approximations of the kernel space, i.e., number of landmarks.

4.2. Community question-answering

The Community Question Answering (cQA) task is an evolution of a typical QA setting framed in a Web forum context, where users are allowed to freely ask questions and to expect some good, honest answers. Unfortunately, many answers are only poorly related to the actual question, and some even may miss the topic. This is a real problem, as a question can have a long thread of comments containing a very small set of good answers. Therefore, finding the desired information through a long list of answers might be very time-consuming.

In the SemEval-2016 task 3 challenge (Nakov et al., 2016), participants are asked to automatically provide good answers from the Qatar Living forum.[13] In particular, the subtask A is defined as follows: given a question and a large collection of question-comment threads created by a user community, the task consists in (re-)ranking the comments according to their utility in answering the question. A simplified example is shown in Fig. 11, where answers 2 and 4 are *good* , answer 1 is *potentially useful*, and answer 3 is *bad* .[14]

This task is interesting as kernel methods achieved the highest performance in the cQA task, as demonstrated by the KeLP team (Filice et al., 2016).[15] In particular, Subtask A is 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 with a *good* comment and the negative label includes the *potential* and *bad* comments. The classification score is used to sort the instances and produce the final ranking, so that *good* comments will receive positive (i.e., higher) scores w.r.t. the other comments. According to the above setting, a train and test dataset made of 20,340 and 3270 examples are generated.

In Filice et al. (2016), a Kernel-based SVM classifier achieved state-of-the-art results by adopting a kernel combination of tree kernels and a linear kernel. In particular the linear kernel is applied on feature vectors containing (*i*) 21 linguistic similarity scores between the texts in a pair (including lexical similarities of lemmas, syntactic similarities on PoS tags, and semantic similarities based on 250 dimensional Word2vec Embeddings word embeddings generated by applying word2vec (Mikolov et al., 2013) to the entire Qatar Living corpus from SemEval 2015[16]); (*ii*) 61 features capturing task-specific information, such as whether the comment contains URLs, emails, acknowledges; etc.

Tree kernels are also applied to evaluate inter-pair similarities between question-comment pairs. As shown in Fig. 12, a question-comment pair is represented as pair of their corresponding shallow parse trees, where common or semantically similar lexical nodes are linked using a tagging strategy (which is propagated to their upper

[13] http://www.qatarliving.com/forum
[14] According to the official evaluation, there is no distinction between the classes *bad* and *potentially useful*.
[15] http://alt.qcri.org/semeval2016/task3/data/uploads/semeval2016_task3_results.pdf
[16] http://alt.qcri.org/semeval2015/task3

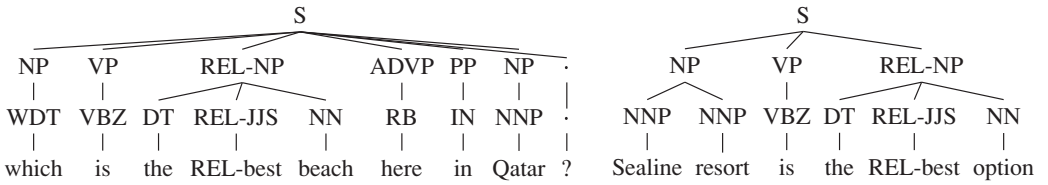

Fig. 12. Structural representation of a question-answer pair.

Table 5
Results in terms of F_1 and savings in the community question answering task.

Model	#Land.	F_1	Saving
SVM_{ker}	–	0.644	0.0%
ConvKN Barrón-Cedeño et al. (2016)	–	0.662	–
	100	0.638 (0.596)	99.1%
	200	0.635 (0.627)	98.2%
	400	0.657 (0.637)	96.5%
KDA (SVM_{lin})	600	0.669 (0.645)	94.7%
	800	**0.680** (0.653)	**92.9%**
	1000	0.674 (0.644)	91.2%

constituents), following the approach proposed in Filice et al. (2015b). This approach discriminates aligned sub-fragments from non-aligned ones, allowing the learning algorithm to capture relational patterns, e.g., *the* REL-*best beach* and *the* REL-*best option*. Given two question-comment pairs $p_a = \langle q_1, c_1 \rangle$ and $p_b = \langle q_2, c_2 \rangle$, the following tree kernel combination is defined:

$$\mathrm{PTK}^{+}(p_a, p_b) = \mathrm{PTK}(q_1, q_2) + \mathrm{PTK}(c_1, c_2) \tag{10}$$

This method can capture emerging pairwise patterns and should be effective in recognizing valid question/answer pairs, even in those cases in which the two texts have few words in common that would cause the failure of any intra-pair approach.

The above 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 SVM_{ker}. We varied the Nyström dimensions from 100 to 1000 landmarks, i.e., a much lower number than the support vectors of SVM_{ker}.

Table 5 reports the results: very high F_1 scores 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_{lin} is 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

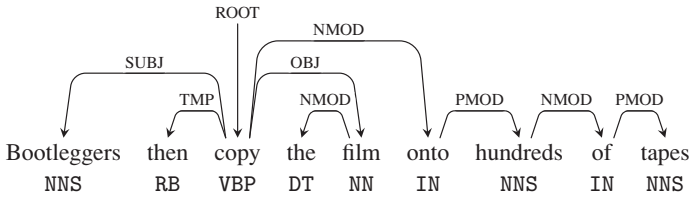

Fig. 13. Example of dependency parse tree.

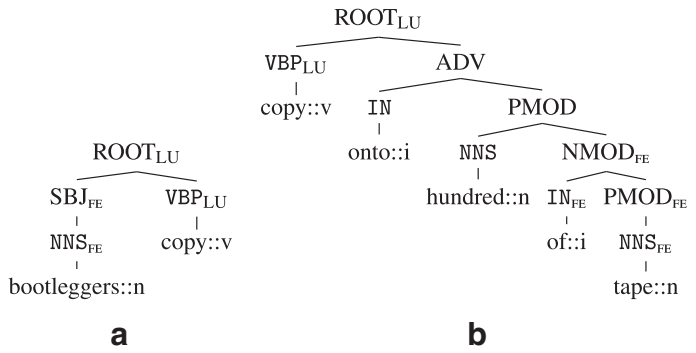

Fig. 14. Dependency graphs, expressed as marked GRCTs, that are examples of argument boundaries. Tree in Fig. 14(a) is a positive example (as *bootleggers* represents the CREATOR argument) while in Fig. 14(c) a negative one (as the span *of tape* does not represent any argument).

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 Trees (GRCT) derived from dependency grammar (Fig. 13) are employed and marked, as shown in Fig. 14. Each node is considered as a candidate in covering a potential *argument*. The nodes of the subtree covering the argument words 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. Each marked tree can be a positive or negative example. In particular, the structure in Fig. 14 (a) shows a positive example. On the contrary, in Fig. 14(c) the NMOD node only covers the phrase "*of tapes*", i.e., a subset of the correct GOAL role, and thus represents a negative example.

We selected all the sentences whose predicate word (lexical unit) is a verb (they are about 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 into 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 (SVM_{ker}) achieves an F_1 of 70.2%, over the same test set used in Croce and Basili (2016) that includes 146,399 examples. The SVM_{ker} 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_{lin}, and the KDA proposed in this work. Table 6 and Fig. 15 presents the results in terms of F_1 and saved kernel operations. Although SVM_{lin} with 500 landmarks already achieves 0.713 F_1, a score higher than SVM_{ker}, it is 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 SVM_{ker}. 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 SVM_{ker}).

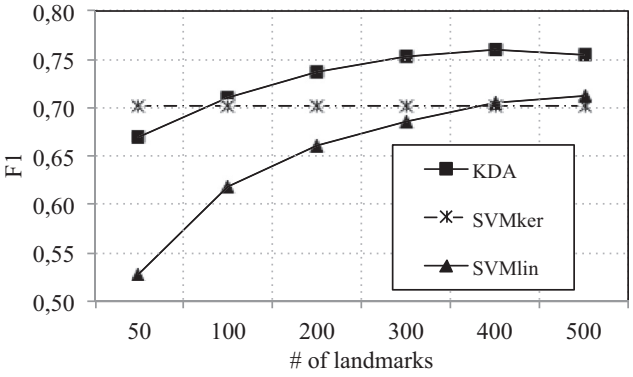

Fig. 15. ABD task: F_1 measure curves w.r.t. the number of landmarks.

Table 6
Results in terms of F_1 and saving in the argument boundary detection task.

Model	Land.	Tr.Size	F_1	Saving(%)
SVM_{ker}	–	10k	0.702	0.0
	100	100k	0.711 (0.618)	96.7
	200	100k	0.737 (0.661)	93.3
KDA (SVM_{lin})	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 then applied to learn classification models. This operation corresponds to learning non-linear decision functions on structured data in complex kernel space. As a result, accurate models can be obtained while reducing the underlying computational complexity w.r.t. to pure kernel methods.

At the best of our knowledge, this work is one of the few attempts to systematically integrate 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 input. 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 the so-called Convolutional Kernel Networks. Other task-driven 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 experimental outcomes discussed in the paper confirm that no critical loss in performance is observed for different

semantic kernels (such as PTK, SPTK and CSPTKs) across all the three different tasks tested. In some cases, the applicability of the Nyström methodology allows to scale up to larger training datasets so that a higher accuracy can be achieved. 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 here proposed 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. The novel learning strategy capitalizes the ability of kernels to represent complex search spaces in combination with the ability of neural networks to find non-linear solutions 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.

This work paves the way to several different future research directions. First, experimentation with larger scale datasets will be useful to assess the scalability properties and better understand the contribution of landmarks. It is useful to better understand suitable selection policies that may improve the random selection applied in this paper. Moreover, the trade-off between the accuracy of the kernel approximation of the Nyström method and the over-fitting possibly introduced in a neural network architecture is an interesting topic as it sheds light on the compromise between representation and generalization capabilities of these methods.

A second direction refers to the optimization of the KDA methodology. Suitable parallelization methods in fact can be designed over multi-core architectures in order to improve the efficiency benefits brought by the method and amplify its scalability.

Finally, the combination of non-linear neural learning algorithms with kernelized representations here proposed allows the adoption of *linguistically motivated explanatory methods that provide justifications about the network decisions*. These methods aim at making the neural learning more *readable* by tracing back the portions of the network input that mostly contribute to the output decision. Several works have been proposed where network propagation techniques are used to identify the patterns of a given input item (e.g., an image) that are linked to the particular deep neural network prediction (Erhan et al., 2010; Zeiler and Fergus, 2013). Usually, these are based on backward algorithms that layer-wise reuse arc weights to propagate the prediction from the output down to the input, thus leading to the re-creation of *meaningful* patterns in the input space. Typical examples are deconvoution heatmaps, used to approximate through Taylor series, the partial derivatives at each layer (Simonyan et al., 2013), or the so-called Layer-wise Relevance Propagation (LRP), that redistributes back positive and negative evidence across the layers (Bach et al., 2015).

This work also triggers a relevant research direction, that is the study on how the input used to stimulate the KDA network is correlated with its output and how this property can be used to semantically describe possible causal relations. As these dependencies would be the outcome of the network, i.e. the trained neural model of the decision, they could work as explanations. As pointed out in Croce et al. (2018), the first layer of the KDA (the so-called Nyström layer) reflects real examples, i.e., the landmarks. The application of algorithms able to trace the activations backward across layers allows to correlate a KDA decision with one or more involved landmarks, e.g. those characterized by the highest similarity with the input instance. Given the semantic nature of the semantic kernels (that are arbitrarily complex in a KDA), this correlation may well act as an explanation. For example, a KDA that assigns the question "*What French ruler was defeated at the battle of Waterloo?*" to the class Human in a Question Classification task is able to emphasize the contribution of the correlated landmark, such as "*Who was the first woman killed in the Vietnam War?*". This landmark embodies high level of correlation as for its lexical, syntactic and semantic properties that are naturally readable. KDA thus opens the opportunity of a language driven method for producing the explanation of a classification. As an example, the method proposed in Croce et al. (2018) proposes to use expressions such as *I think "What French ruler ⋯" refers to a* Human *since it reminds me of "Who was the first woman ⋯ "*. Each decision can be thus motivated through analogies with real examples that are linguistically related to the input. Such kind of arguments are supposed to inspire, in the reference user, a higher level of trust towards the machine

decision. Evidence from the linguistic properties shared between the input sentence and the explanation one as well their role in the output decision has been shown to effectively support the rejection of wrong decisions. This fosters effective auditing stages of a reference classification model as the underlying KDA framework makes it explainable *by design*.

Given that kernel methods allows to decouple the representation from the learning algorithm, they make it possible to exploit the above advantages in the KDA (as it operates over H_{Ny}). Landmarks embed lexical, syntactic and semantic information according to complex tree kernels and this leaves the deep neural network to non linearly explore the kernel space. If this is effective for a given task then it can also be used to make the overall statistical inference more transparent. In the resulting framework in fact, the non-linearity of the neural learning is preserved in the space of the semantic patterns provided by the landmarks. The reconstruction algorithms (e.g., deconvolution heatmaps) should allow to easily reconstruct the semantic patterns characterizing an output decision. A major advantage is thus semantic transparency: explanatory knowledge, made of the syntactic and semantic patterns positively or negatively correlated with the output decisions, is made available as a meaningful, though inexpensive, outcome of a KDA decision. KDA in fact can adopt any semantic kernel, i.e., the pre-training supported by the Nyström H_{Ny} layer. Given the critical role of DNNs in current AI, this epistemological transparency aspect certainly constitutes a crucial topic for future research on the KDA method proposed in this paper.

References

Agirre, E., Banea, C., Cer, D.M., Diab, M.T., Gonzalez-Agirre, A., Mihalcea, R., Rigau, G., Wiebe, J., 2016. Semeval-2016 task 1: semantic textual similarity, monolingual and cross-lingual evaluation. In: Bethard, S., Cer, D.M., Carpuat, M., Jurgens, D., Nakov, P., Zesch, T. (Eds.), Proceedings of the SemEval@NAACL-HLT. The Association for Computer Linguistics, pp. 497–511.

Annesi, P., Croce, D., Basili, R., 2014. Semantic compositionality in tree kernels. In: Proceedings of the CIKM. ACM.

Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., Samek, W., Suárez, Ó.D., 2015. On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. PLoS ONE 10 (7), 1–46.

Baker, C.F., Fillmore, C.J., Lowe, J.B., 1998. The Berkeley FrameNet project. In: Proceedings of the COLING-ACL. Montreal, Canada.

Baldi, P., Azencott, C., Swamidass, S.J., 2011. Bridging the gap between neural network and kernel methods: applications to drug discovery. In: Proceedings of the Twentieth Italian Workshop on Neural Nets.

Barrón-Cedeño, A., Da San Martino, G., Joty, S., Moschitti, A., Al-Obaidli, F., Romeo, S., Tymoshenko, K., Uva, A., 2016. ConvKN at SemEval-2016 task 3: answer and question selection for question answering on arabic and english fora. In: Proceedings of the SemEval-2016.

Bengio, Y., Courville, A., Vincent, P., 2013. Representation learning: a review and new perspectives. IEEE Trans. Pattern Anal. Mach. Intell. 35 (8), 1798–1828. doi: 10.1109/TPAMI.2013.50.

Brychcín, T., Svoboda, L., 2016. Uwb at semeval-2016 task 1: semantic textual similarity using lexical, syntactic, and semantic information. In: Proceedings of the Tenth International Workshop on Semantic Evaluation (SemEval-2016). Association for Computational Linguistics, San Diego, California, pp. 588–594.

Cancedda, N., Gaussier, É., Goutte, C., Renders, J.-M., 2003. Word-sequence kernels. J Mach Learn Res 3, 1059–1082.

Chali, Y., Joty, S.R., Hasan, S.A., 2009. Complex question answering: Unsupervised learning approaches and experiments. J Artif Intell Res (JAIR) 35 (1), 1–47.

Chang, C.-C., Lin, C.-J., 2011. Libsvm: A library for support vector machines. ACM Trans. Intell. Syst. Technol. 2 (3), 27:1–27:27. doi: 10.1145/1961189.1961199.

Cho, Y., Saul, L.K., 2009. Kernel methods for deep learning. In: Bengio, Y., Schuurmans, D., Lafferty, J.D., Williams, C.K.I., Culotta, A. (Eds.), Proceedings of the Advances in Neural Information Processing Systems 22. Curran Associates, Inc., pp. 342–350.

Collins, M., Duffy, N., 2001. Convolution kernels for natural language. In: Proceedings of the Neural Information Processing Systems (NIPS'2001), pp. 625–632.

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P., 2011. Natural language processing (almost) from scratch. J Mach Learn Res (JAIR) 12, 2493–2537.

Cortes, C., Vapnik, V., 1995. Support-vector networks. Mach. Learn 20 (3), 273–297. doi: 10.1023/A:1022627411411.

Croce, D., Basili, R., 2016. Large-scale kernel-based language learning through the ensemble Nystrom methods. In: Proceedings of the ECIR 2016.

Croce, D., Filice, S., Castellucci, G., Basili, R., 2017. Deep learning in semantic kernel spaces. In: Proceedings of the Fifty-Fifth Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, pp. 345–354. doi: 10.18653/v1/P17-1032.

Croce, D., Moschitti, A., Basili, R., 2011. Structured lexical similarity via convolution kernels on dependency trees. In: Proceedings of the EMNLP '11, pp. 1034–1046.

Croce, D., Rossini, D., Basili, R., 2018. Explaining non-linear classifier decisions within kernel-based deep architectures. In: Proceedings of the Workshop: Analyzing and Interpreting Neural Networks for NLP, BlackboxNLP@EMNLP 2018, pp. 16–24. Brussels, Belgium, November 1, 2018

Dekel, O., Singer, Y., 2006. Support vector machines on a budget. In: Proceedings of the NIPS. MIT Press, pp. 345–352.
Drineas, P., Mahoney, M.W., 2005. On the nyström method for approximating a gram matrix for improved kernel-based learning. J Mach Learn Res 6, 2153–2175.
Erhan, D., Courville, A., Bengio, Y., 2010. Understanding Representations Learned in Deep Architectures. Technical Report 1355. Université de Montréal/DIRO.
Filice, S., Castellucci, G., Croce, D., Basili, R., 2015. KeLP: a kernel-based learning platform for natural language processing. In: Proceedings of the ACL: System Demonstrations. Beijing, China.
Filice, S., Castellucci, G., Martino, G.D.S., Aless, ro Moschitti, Croce, D., Basili, R., 2018. Kelp: a kernel-based learning platform. J Mach Learn Res 18 (191), 1–5.
Filice, S., Croce, D., Moschitti, A., Basili, R., 2016. KeLP at SemEval-2016 task 3: learning semantic relations between questions and comments. In: Proceedings of the SemEval '16.
Filice, S., Da San Martino, G., Moschitti, A., 2015. Structural representations for learning relations between pairs of texts. In: Proceedings of the ACL 2015. Beijing, China, pp. 1003–1013.
Fillmore, C.J., 1985. Frames and the semantics of understanding. Quaderni di Semantica 6 (2), 222–254.
Goldberg, Y., 2016. A primer on neural network models for natural language processing. J Artif Intell Res (JAIR) 57, 345–420.
Harris, Z., 1964. Distributional structure. In: Katz, J.J., Fodor, J.A. (Eds.), The Philosophy of Linguistics. Oxford University Press.
Hochreiter, S., Schmidhuber, J., 1997. Long short-term memory. Neural Comput 9 (8), 1735–1780.
Hsieh, C.-J., Chang, K.-W., Lin, C.-J., Keerthi, S.S., Sundararajan, S., 2008. A dual coordinate descent method for large-scale linear SVM. In: Proceedings of the ICML 2008. ACM, pp. 408–415.
Johansson, R., Nugues, P., 2008. The effect of syntactic representation on semantic role labeling. In: Proceedings of the COLING.
Katrenko, S., Adriaans, P., van Someren, M., 2010. Using local alignments for relation recognition. J Artif Intell Res (JAIR) 38 (1), 1–48.
Kim, Y., 2014. Convolutional neural networks for sentence classification. In: Proceedings of the EMNLP 2014. Doha, Qatar, pp. 1746–1751.
Kulis, B., Basu, S., Dhillon, I., Mooney, R., 2005. Semi-supervised graph clustering: a kernel approach. In: Proceedings of the ICML. ACM, pp. 457–464.
Kumar, S., Mohri, M., Talwalkar, A., 2012. Sampling methods for the Nyström method. J Mach Learn Res 13, 981–1006.
LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., 1998. Gradient-based learning applied to document recognition. Proc IEEE 86 (11), 1–46.
Levy, O., Goldberg, Y., Dagan, I., 2015. Improving distributional similarity with lessons learned from word embeddings. Trans. Assoc. Comput Linguist 3, 211–225.
Li, X., Roth, D., 2006. Learning question classifiers: the role of semantic information. Natural Lang Eng 12 (3), 229–249.
Mairal, J., Koniusz, P., Harchaoui, Z., Schmid, C., 2014. Convolutional kernel networks. In: Proceedings of the Advances in Neural Information Processing Systems.
Mikolov, T., Chen, K., Corrado, G., Dean, J., 2013. Efficient estimation of word representations in vector space. CoRR abs/1301.3781.
Mitchell, J., Lapata, M., 2010. Composition in distributional models of semantics. Cognit Sci 34 (8), 1388–1429.
Moschitti, A., 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In: Proceedings of the ECML. Berlin, Germany.
Moschitti, A., 2012. State-of-the-art kernels for natural language processing. In: Proceedings of the ACL (Tutorial Abstracts). The Association for Computer Linguistics, p. 2.
Moschitti, A., Pighin, D., Basili, R., 2008. Tree kernels for semantic role labeling. Comput Linguist 34, 193–224.
Robert Müller, K., Mika, S., Rätsch, G., Tsuda, K., Schölkopf, B., 2001. An introduction to kernel-based learning algorithms. IEEE Trans Neural Netw 12 (2), 181–201.
Nakov, P., Màrquez, L., Moschitti, A., Magdy, W., Mubarak, H., Freihat, A.A., Glass, J., Randeree, B., 2016. SemEval-2016 task 3: community question answering. In: Proceedings of the SemEval-2016.
Pado, S., Lapata, M., 2007. Dependency-based construction of semantic space models. Comput Linguist 33 (2), 161–199.
Rahimi, A., Recht, B., 2008. Random features for large-scale kernel machines. In: Platt, J.C., Koller, D., Singer, Y., Roweis, S.T. (Eds.), Proceedings of the Advances in Neural Information Processing Systems 20. Curran Associates, Inc., pp. 1177–1184.
Rychalska, B., Pakulska, K., Chodorowska, K., Walczak, W., Andruszkiewicz, P., 2016. Samsung Poland NLP team at SemEval-2016 task 1: necessity for diversity; combining recursive Autoencoders, Wordnet and ensemble methods to measure semantic similarity In: Proceedings of the Tenth International Workshop on Semantic Evaluation (SemEval-2016). San Diego, California, pp. 614–620.
Sahlgren, M., 2006. The Word-Space Model (Ph.D. thesis.). Stockholm University.
Schölkopf, B., Smola, A., Müller, K.-R., 1998. Nonlinear component analysis as a kernel eigenvalue problem. Neural Comput 10 (5), 1299–1319. doi: 10.1162/089976698300017467.
Severyn, A., Nicosia, M., Moschitti, A., 2013. Building structures from classifiers for passage reranking. In: Proceedings of the Twenty-Second ACM International Conference on Information and Knowledge Management, CIKM'13, pp. 969–978. doi: 10.1145/2505515.2505688. San Francisco, CA, USA, October 27 - November 1, 2013
Shawe-Taylor, J., Cristianini, N., 2004. Kernel Methods for Pattern Analysis. Cambridge University Press, New York, NY, USA.
Simonyan, K., Vedaldi, A., Zisserman, A., 2013. Deep inside convolutional networks: visualising image classification models and saliency maps. CoRRarXiv:1312.6034.
Socher, R., Perelygin, A., Wu, J., Chuang, J., Manning, C.D., Ng, A., Potts, C., 2013. Recursive deep models for semantic Compositionality over a sentiment treebank. In: Proceedings of the EMNLP '13.
Tai, K.S., Socher, R., Manning, C.D., 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 (Volume 1: Long Papers). Association for Computational Linguistics, Beijing, China, pp. 1556–1566 url: https://www.aclweb.org/anthology/P15-1150, doi:10.3115/v1/P15-1150.

Turian, J., Ratinov, L.-A., Bengio, Y., 2010. Word representations: a simple and general method for semi-supervised learning. In: Proceedings of the Forty-Eighth Annual Meeting of the Association for Computational Linguistics. Uppsala, Sweden, pp. 384–394.

Turney, P.D., Pantel, P., 2010. From frequency to meaning: vector space models of semantics. J Artif Intell Res 37, 141–188.

Tymoshenko, K., Bonadiman, D., Moschitti, A., 2016. Convolutional neural networks vs. convolution kernels: feature engineering for answer sentence reranking. In: Proceedings of the NAACL 2016.

Vapnik, V.N., 1998. Statistical Learning Theory. Wiley-Interscience.

Vishwanathan, S., Smola, A.J., 2002. Fast kernels on strings and trees. In: Proceedings of the Neural Information Processing Systems, pp. 569–576.

Wang, Z., Vucetic, S., 2010. Online passive-aggressive algorithms on a budget. J Mach Learn Res Proc Track 9, 908–915.

Williams, C.K.I., Seeger, M., 2001. Using the Nyström method to speed up kernel machines. In: Proceedings of the NIPS 2000.

Yu, K., Xu, W., Gong, Y., 2009. Deep learning with kernel regularization for visual recognition. In: Proceedings of the Advances in Neural Information Processing Systems 21. Curran Associates, Inc., pp. 1889–1896.

Zanzotto, F.M., Pennacchiotti, M., Moschitti, A., 2009. A machine learning approach to textual entailment recognition. Nat Lang Eng 15-04, 551–582. doi: 10.1017/S1351324909990143.

Zeiler, M. D., Fergus, R., 2013. Visualizing and understanding convolutional networks. CoR, abs:1311.2901.

Zhang, R., Lee, H., Radev, D.R., 2016. Dependency sensitive convolutional neural networks for modeling sentences and documents. In: Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies. Association for Computational Linguistics, San Diego, California, pp. 1512–1521.

Zhou, C., Sun, C., Liu, Z., Lau, F. C. M., 2015. A C-LSTM neural network for text classification. CoRR, abs:1511.08630.

Zhuang, J., Tsang, I.W., Hoi, S.C.H., 2011. Two-layer multiple kernel learning In: Proceedings of the AISTATS. JMLR.org, pp. 909–917.