Semantic Tree Kernels for Statistical Natural Language Learning

Danilo Croce, Roberto Basili and Alessandro Moschitti

Abstract A central topic in Natural Language Processing (NLP) is the design of effective linguistic processors suitable for the target applications. Within this scenario, Convolution Kernels provide a powerful method to directly apply Machine Learning algorithms to complex structures representing linguistic information. The main topic of this work is the definition of the semantically Smoothed Partial Tree Kernel (SPTK), a generalized formulation of one of the most performant Convolution Kernels, i.e. the Tree Kernel (TK), by extending the similarity between tree structures with node similarities. The main characteristic of SPTK is its ability to measure the similarity between syntactic tree structures, which are partially similar and whose nodes can differ but are nevertheless semantically related. One of the most important outcomes is that SPTK allows for embedding external lexical information in the kernel function only through a similarity function among lexical nodes. The SPTK has been evaluated in three complex automatic Semantic Processing tasks: Question Classification in Question Answering, Verb Classification and Semantic Role Labeling. Although these tasks address different problems, state-of-the-art results have been achieved in every evaluation.

Keywords Kernel methods · Tree kernels · Semantic role labeling · Verb classification

D. Croce (✉) · R. Basili
Department of Computer Science, Systems and Production,
University of Roma Tor Vergata, Rome, Italy
e-mail: croce@info.uniroma2.it

R. Basili
e-mail: basili@info.uniroma2.it

A. Moschitti
Department of Computer Science and Information Engineering,
University of Trento, Povo (TN), Italy
e-mail: moschitti@disi.unitn.it

R. Basili et al. (eds.), *Harmonization and Development of Resources and Tools for Italian Natural Language Processing within the PARLI Project*,
Studies in Computational Intelligence 589, DOI 10.1007/978-3-319-14206-7_5

1 Introduction

Most human knowledge is represented and expressed using language and modern systems in Information Technology need to access the huge amount of information that is stored and constantly produced in the Web. This source of information can be represented in structured form, e.g. stored inside Databases or Data Warehouses, but the vast majority is still produced in an unstructured form, e.g. documents written in natural language. In such a scenario which is also recurrent in real time marketing, semantic web-search, security or exploratory data analysis, the proper application of Natural Language Processing (NLP) techniques allows for more sophisticated access to information, hence providing more natural human-machine interfaces. Traditionally, Information Retrieval (IR) has dealt with representation, storage, organization of, and access to information items, e.g. documents, as described in [1]. However, given the rapid growth of the Web, although people can browse and generate linguistic contents, they still do not provide any effective enrichment of the produced information, e.g. a description of the linguistic content that can be exploited by search engines. The open research questions are: How to exploit this huge source of information? How do we *interpret* this large amount of textual data? Information Retrieval faces nowadays contemporary challenges such as Question Answering (QA) [2] or Sentiment Analysis (SA) [3]: in such tasks, complex and fine-grained linguistic information are involved and a principled model of both linguistic content and background knowledge is needed.

In this scenario, the main goal of Computational Natural Language Learning is to acquire knowledge and models needed to turn texts into meaningful structures (i.e. interpretations). The application of such models provides language learning systems, as largely described in [4, 5]. These allow for generalizing linguistic observations into rules and patterns as statistical models of higher level semantic inferences. Statistical learning methods make the assumption that lexical or grammatical observations are useful hints for modeling different semantic inferences, such as in document topical classification, predicate and role recognition in sentences as well as in question classification in Question Answering. Lexical features here include lemmas, multi-word expressions or Named Entities that can be directly observed in the texts. Features are then generalized into predictive components in the final model, induced from the training examples. A proper model of the linguistic observation is needed as a computational representation. A manual feature encoding, where an expert emphasizes the informative properties with respect to the target problem, represents one solution. This activity produces an artificial representation of the linguistic observations which can be employed by a learning system. One important drawback of such process is the cost of the definition of the proper features for a novel task. Even if the learning algorithm can select the most informative ones, they still need to be defined. Moreover, this activity is very tied to the target application and cannot be easily reused for different tasks. The support for the fast design of accurate automatic systems requires to implicitly derive this information from

the data distribution itself for an automatic engineering of syntactic and semantic properties.

Kernel methods, discussed in [6], have been employed in NLP, as in [7], in order to provide a statistical model able to decouple the problem representation and learning algorithm, still satisfying the above requirements. A kernel function [8], allows us to express the similarity between two objects, that are explanatory of the target problem, without defining their explicit representation and, most importantly, it can be used along with kernel-based learning algorithms, e.g. Support Vector Machines, that represent the state-of-the-art machine learning algorithms applied to NLP tasks. The main idea is that the algorithm can effectively learn the target phenomenon by focusing on the notion of similarity among observations, instead of their representations. A linguistic phenomenon can nevertheless be modeled at a more abstract level making the modeling process easier. For example, which representation would be employed to learn the difference between a correct and incorrect syntactic parse tree? By using the parse tree itself, the learner would focus only on the properties useful for the sake of making a decision. This idea is expanded in Tree Kernels, introduced by [7], that allow to model similarity betweentraining examples as a function of the shared syntactic information, in terms of shared syntactic tree fragments, in the corresponding parses.

In this work, we provide the definition of a semantically **Smoothed Partial Tree Kernel** (**SPTK**) that augments the existing Tree Kernel formulations with node similarity and allows to design effective language learning systems. The underlying idea is to provide a similarity score among lexical nodes depending on the semantic similarity between their labels. SPTK can therefore automatically provide the learning algorithm with a huge set of generalized structural patterns by simply applying the kernel function to the structural representation of the target task instances. Within this scenario, a meaningful similarity measure is thus crucial; in fact the lack of proper lexical generalization is often quoted to bear the main responsibility for significant performance drops in out-of-domain semantic processing tasks, e.g. Semantic Role Labeling, as discussed in [9]. Moreover, due to the expensiveness of developing large scale lexical Knowledge Bases, corpus driven methods will be used to acquire meaning generalizations in an unsupervised fashion, as suggested in [10–12]. A distributional paradigm will enable the extension of the SPTK through the adoption of vector based models of lexical meaning. A large-scale corpus is statistically analyzed and a geometrical space (the Word Space discussed in [11]) is defined: here words are modeled as vectors whose dimensions reflect the words co-occurrence statistics over texts, and the similarity (or distance) among vectors models a notion of semantic similarity between the corresponding words.

A large-scale empirical evaluation of SPTK will be discussed to assess its applicability and robustness. The same kernel will be thus applied to different complex semantic tasks: the *Question Classification* task in a Question Answering setting [13], which represents a sentence classification task; the *Verb Classification* task [14], which is a fundamental topic of computational linguistics research given its importance in understanding the role of verbs in conveying semantics of natural language; the *FrameNet based Semantic Role Labeling* task, which represents a

complex semantic annotation task [15]. In such tasks, the proposed model will not rely on manual feature engineering for linguistic phenomena: the employed discriminative learning algorithm, i.e. Support Vector Machines, will select the most informative features for the target problem without any explicit definition. Furthermore, the lexical information provided by the proposed distributional perspective will be investigated and compared with information obtained from hand-built dictionaries.

In the rest of the paper, Sect. 2 discusses limits of traditional Tree Kernel functions and introduces Distributional Models of Lexical Semantics. Section 3 defines the Smoothed Partial Tree Kernel. Section 4 provides the experimental evaluation. Finally, conclusions are derived in Sect. 5.

2 Tree Kernels and Distributional Models of Lexical Semantics

In order to better understand Tree Kernels and discuss their intrinsic limits, let us describe a task where these kernels have been successfully applied, i.e. Semantic Role Labeling (SRL), as proposed in [15, 16]. Since late 70s, *frame semantics* [17] has been proposed as a model of real world situations or events: a linguistic predicate, called *frame*, is evoked in a sentence through the occurrence of specific *lexical units*, i.e. words (e.g. nouns or verbs) that linguistically express the intended situation. A frame characterizes the set of prototypical semantic roles that describe the participants in the event for all lexical units. SRL is thus the task of automatic recognition of individual predicates together with their main roles, as they are semantically and grammatically realized in input sentences. For example, the following two sentences evoke the STATEMENT frame, i.e. the situation of communicating the act of a SPEAKER or a MEDIUM to address a MESSAGE to some ADDRESSEE using language:

$$[\textit{President Kennedy}]_{\text{SPEAKER}}\ \underline{\text{said}}\ [\textit{to an astronaut}]_{\text{ADDRESSEE}}[\text{“}\textit{Man is still the most extraordinary computer of all.”}]_{\text{MESSAGE}} \quad (1)$$

$$[\textit{The report}]_{\text{MEDIUM}}\ \underline{\text{stated}}\ [\textit{that some problems needed to be solved.}]_{\text{MESSAGE}} \quad (2)$$

The frame is evoked through the lexical units *say* and *state*, and the considered roles are SPEAKER, MEDIUM and MESSAGE. SRL is crucial to support reliable and accurate analysis of unstructured text, in order to enrich it with semantic meta-data and other kinds of information which is implicit in texts.

SRL has been a popular task since the availability of the PropBank [18] and FrameNet [19] annotated corpora and the successful CoNLL evaluation campaigns [20]. In SRL, the role of grammatical information has been outlined since the seminal work by [16], where syntactic parse trees are shown to relate a predicate word to its arguments. State-of-the-art approaches to SRL are based on Support Vector Machines, trained over manually built representations derived from syntactic parse trees (e.g. [9, 21]). As discussed in [22, 23], syntactic information of annotated

examples can be effectively generalized in SRL through the adoption of tree kernel-based learning, without relying on manual feature engineering: as tree kernels model similarity between two training examples as a function of their shared tree fragments, discriminative information is automatically selected by the learning algorithm, e.g., Support Vector Machines.

However, when the availability of training data is limited, the information derived from structural patterns may be not sufficient to discriminate examples. In fact, one important limitation of Tree Kernels is that only string matching between node labels is applied when estimating the number of common substructures. Consequently, this entails a poor lexical generalization. Let us consider the example in sentences 1 and 2. Two phrases like *"President Kennedy said…"* and *"The report stated…"* both evoke the JUDGMENT_COMMUNICATION frame, but the two logical subjects represent two different roles: *President Kennedy* represents a human being, then associated with the SPEAKER role, while *report* is a means of communication, therefore associated with the MEAN role. When a kernel function is applied between the above phrases and *"The mail says…"*, the word *mail* differs both from *president* and *report*, therefore it does not provide any contribution to the overall similarity estimation. Nevertheless, it should be considered that *mail* and *report* are semantically related in the inductive inference process, in order to associate the MEAN role with the above text. On the contrary, the resulting learning algorithm should be provided with all examples where the subject of a verb like *say* is a means of communication in order to learn differences between the SPEAKER and MEAN roles. Problems thus arise when the availability of training data is scant: lexical information should be properly generalized to obtain more informative structural patterns.

A significant research has been done on the study of Distributional Models of lexical semantics to automatically acquire meaningful word generalizations: these models follow the distributional hypothesis [24] and characterize lexical meanings in terms of *context of use* [25]. By inducing geometrical notions of vectors and norms through corpus analysis, they provide a topological definition of semantic similarity, i.e., distance in a space. They can capture the similarity between words such as *report* and *mail*. In supervised language learning, when few examples are available, DMs support cost-effective lexical generalizations, often outperforming knowledge based resources (such as WordNet, as in [26]). Obviously, the choice of the context type determines the type of targeted semantic properties. Wider contexts (e.g., entire documents) are shown to suggest topical relations. Smaller contexts tend to capture more specific semantic aspects, e.g. the syntactic behavior, and better capture paradigmatic relations, such as synonymy. In particular, word space models, as described in [11], define contexts as the words appearing in a n-sized window, centered around a target word. Co-occurrence counts are thus collected in a words-by-words matrix, where each element records the number of times two words co-occur within a single window of word tokens. Moreover, robust weighting schemas are used to smooth counts against too frequent co-occurrence pairs: Pointwise Mutual Information (PMI) scores [27] are commonly adopted. In such statistical paradigm, robust representations can be obtained through intelligent dimensionality reduction methods. According the Latent Semantic Analysis (LSA) technique [28], the original

word-by-word matrix M can be decomposed through Singular Value Decomposition (SVD) [29] into the product of three new matrices: U, S, and V so that S is diagonal and $M = USV^T$. M is then approximated by $M_k = U_k S_k V_k^T$, where only the first k columns of U and V are used, corresponding to the first k greatest singular values. This approximation supplies a way to project a generic term w_i into the k-dimensional space using $W = U_k S_k^{1/2}$, where each row corresponds to the representation vector $\boldsymbol{w_i}$. The original statistical information about M is captured by the new k-dimensional space, which preserves the global structure while removing low-variant dimensions, i.e., distribution noise. Given two words w_1 and w_2, the term similarity function σ is estimated as the cosine similarity between the corresponding projections $\boldsymbol{w_1}$, $\boldsymbol{w_2}$ in the LSA space, i.e. $\sigma(w_1, w_2) = \frac{\boldsymbol{w_1} \cdot \boldsymbol{w_2}}{\|\boldsymbol{w_1}\| \|\boldsymbol{w_2}\|}$. This is known as *Latent Semantic Kernel* (*LSK*), proposed in [30], as it defines a positive semi-definite Gram matrix $\boldsymbol{G} = \sigma(w_1, w_2)\, \forall w_1, w_2$ [8]. σ is thus a valid kernel and can be combined with other kernels, as discussed in the next session.

3 Semantically Smoothed Partial Tree Kernel

The main drawback of pure lexical information is due to its non-compositional nature as the grammatical structure of the sentences is ignored and it is not designed to compute the meanings of phrases. As already addressed in recent works, e.g. [31], the definition of methods able to express the meaning of phrases or sentences as composition operations over geometric representations is a complex problem, and a still largely open issue. Some studies, e.g. [32–36], propose classes of algebraic operators (e.g. tensor products) as effective combination of lexical information. Their focus is to explicitly combine vectors representing words in a phrase in order to obtain a new vector representing the semantics of the entire phrase. These works propose algebraic models of words composition with constraints imposed by the targeted phrase structure. However, these models still work on simple syntactic structures, e.g. they provide a composition between two or three words, although they lack the proper expressivity to be employed in complex tasks.

In this work a different approach is pursued based upon to the idea of convolution kernels: rather than providing an explicit representation of the sentence semantics in terms of word composition, a method is instead defined to estimate the similarity between sentences, embedding this lexical information directly in the similarity function. In this perspective, one interesting approach, proposed in [37], encoded lexical similarity in tree kernels. The model is essentially the Syntactic Tree Kernel (STK), defined in [7], in which syntactic fragments from constituency trees can be matched even if they differ in the leaf nodes (i.e., they are constituted by related words with different surface forms). This kernel has been named *Semantic Syntactic Tree Kernel* (SSTK) and its computation is recursively carried out by the following Δ_{SSTK} function:

- if n_1 and n_2 are not pre-terminals and the productions at n_1 and n_2 are different then

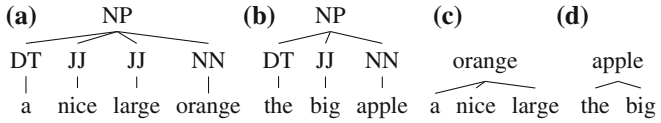

Fig. 1 Examples of syntactic parse trees

$$\Delta_{SSTK}(n_1, n_2) = 0$$

- if n_1 and n_2 are pre-terminals and $label(n_1) = label(n_2)$ then

$$\Delta_{SSTK}(n_1, n_2) = \lambda K_S(ch^1_{n_1}, ch^1_{n_2})$$

- if n_1 and n_2 are not pre-terminals and the productions at n_1 and n_2 are the same[1] then:

$$\Delta_{SSTK}(n_1, n_2) = \lambda \prod_{j=1}^{n_c(n_1)} (1 + \Delta_{SSTK}(ch^j_{n_1}, ch^j_{n_2}))$$

where $label(n_i)$ is the label of node n_i and K_S is a valid term similarity kernel. Note that in constituency parse trees n_1 and n_2 are pre-terminals and they can have only one child (i.e. $ch^1_{n_1}$ and $ch^1_{n_2}$) and such children are words. This kernel uses matching scores between fragments (i.e., features) that depend on the semantic similarity K_S between the corresponding leaves in the syntactic fragments. This allows to match fragments having the same structure but different leaves by assigning a score which is proportional to the product of the lexical similarities of each leaf pair.

Notwithstanding the aforementioned idea is promising and the SSTK provided good results in several NL tasks, such as Question Classification in [37] and Textual Entailment Recognition in [38]. However, the SSTK inherits the intrinsic limitations that reduce the effectiveness of semantic smoothing: in Fig. 1a, b, two simple fragments from a constituency parse tree are shown, representing the two nominal syntagmas "*a nice large orange*" and "*the big apple*", respectively. These short texts are semantically related and a proper lexical similarity could acquire this information by comparing words like *a/the*, *big/large* or *orange/apple*. However, the SSTK does not estimate this similarity among leaves because the production rules `[NP [DT JJ JJ NN]]` and `[NP [DT JJ NN]]` are not the same. Moreover, the SSTK cannot be applied to information represented through dependency parse trees. In Fig. 1c, d, two trees derived from the noun phrases as dependency graphs are shown; it is worth noting that the graph governor is the tree root, while the dependents are the leaves. As the SSTK estimates the K_S only between tree leaves, it is trivial that it cannot be applied to such trees, as their roots are different.

Hereafter, a more general tree kernel is defined and it can be applied to any tree and exploit any combination of lexical similarities thought respecting the syntax enforced

[1] It implies that $n_c(n_1) = n_c(n_2)$.

by the tree. To overcome such issues, the tree kernel proposed in [39], namely the Partial Tree Kernel (PTK), is augmented with node similarity. This allows to use any tree and any lexical similarity metrics between nodes for any position of the tree (not just on the leaves as in [37]). In other words, the new Smoothed PTK (SPTK) can automatically provide the learning algorithm, e.g., Support Vector Machines (SVMs), with a huge set of generalized structural patterns by simply applying it to the structural representation of instances of the target task. Combining lexical and structural kernels provides clear advantages on all-vs-all word similarity, which tends to semantically diverge. Indeed syntax provides the necessary restrictions to compute an effective semantic similarity.

3.1 Smoothed Partial Tree Kernel Definition

As for the evaluation of PTK, the evaluation of the common SPTK rooted in nodes n_1 and n_2 requires the selection of the shared child subsets of the two nodes. Due to the importance of the order of the children, we can use subsequence kernels for their generation. More in detail, let $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ be the set of all possible PT fragment and let the indicator function $I_i(n)$ be equal to 1 if the target f_i is rooted at node n and 0 otherwise, we define the SPTK as:

- If n_1 and n_2 are leaves then $\Delta_{SPTK}(n_1, n_2) = \mu\lambda\sigma_\tau(n_1, n_2)$
- else

$$\Delta_{SPTK}(n_1, n_2) = \mu\sigma_\tau(n_1, n_2) \times \left(\lambda^2 + \sum_{\boldsymbol{I}_1, \boldsymbol{I}_2, l(\boldsymbol{I}_1)=l(\boldsymbol{I}_2)} \lambda^{d(\boldsymbol{I}_1)+d(\boldsymbol{I}_2)} \prod_{j=1}^{l(\boldsymbol{I}_1)} \Delta_{SPTK}(c_{n_1}(\boldsymbol{I}_{1j}), c_{n_2}(\boldsymbol{I}_{2j}))\right) \quad (3)$$

Here the formulation is similar to the PTK, c_{n_1} and c_{n_2} are the ordered child sequences of n_1 and n_2 respectively, while $I_1 = \langle I_{11}, I_{12}, I_{13}, \ldots\rangle$ and $I_2 = \langle I_{21}, I_{22}, I_{23}, \ldots\rangle$ are index sequences associated with the ordered child sequences such that $c_{n_1}(\boldsymbol{I}_{1j})$ and $c_{n_2}(\boldsymbol{I}_{2j})$ are the jth children in the two sequences respectively. The function $l(\cdot)$ returns the sequence length. As for PTK, two decay factors are employed: $0 < \mu \leq 1$ for the height of the tree and $0 < \lambda \leq 1$ for the length of the child sequences. It follows that both larger trees and subtrees built on child subsequences that contain gaps are penalized depending on the exponent $d(\boldsymbol{I}_1) = \boldsymbol{I}_{1l(\boldsymbol{I}_1)} - \boldsymbol{I}_{11}$ and $d(\boldsymbol{I}_2) = \boldsymbol{I}_{2l(\boldsymbol{I}_2)} - \boldsymbol{I}_{21}$, i.e. the width of the production rule.

The novelty of SPTK is represented by the embedding of a similarity function σ_τ between nodes which are typed according to τ. It is more general than the SSTK as it depends on the position of the node pairs within the trees, i.e. non terminals nodes and leaves. Furthermore, the overall SPTK is neutral with respect to the target linguistic problems discussed in this work. Obviously, the similarity function between nodes must be carefully designed in order to grant effectiveness in the target semantic processing task: in fact, the SPTK would enumerate and compare any possible node

Algorithm 1 $\sigma_\tau(n_1, n_2, lw)$

$\sigma_\tau \leftarrow 0$,
if $\tau(n_1) = \tau(n_2) = \text{SYNT} \wedge label(n_1) = label(n_2)$ **then**
  $\sigma_\tau \leftarrow 1$
end if
if $\tau(n_1) = \tau(n_2) = \text{POS} \wedge label(n_1) = label(n_2)$ **then**
  $\sigma_\tau \leftarrow 1$
end if
if $\tau(n_1) = \tau(n_2) = \text{LEX} \wedge pos(n_1) = pos(n_2)$ **then**
  $\sigma_\tau \leftarrow \sigma_{LEX}(n_1, n_2) \times lw$
end if
return σ_τ

pairs, including non terminal nodes. From a linguistic perspective this is problematic as each node reflects a specific aspect of data and the comparison between nodes of different nature, e.g. syntactic nodes like `NP` or `VP`, and lexical nodes like `apple` or `orange` should be avoided. The similarity function $\sigma_\tau(n_1, n_1)$ between two nodes n_1 and n_2 must depend on the nodes' type τ. An example of σ_τ is shown by Algorithm 1: given two nodes n_1 and n_2, it applies a different similarity for each node type. Types are described by τ and are divided into: syntactic categories (i.e., $\tau = \text{SYNT}$), POS-Tag labels (i.e., $\tau = \text{POS}$) or a lexical (i.e., $\tau = \text{LEX}$) type. In this example we require a hard match between non lexical nodes, i.e. assigning 0/1 similarity for SYNT and POS nodes. For LEX type, a lexical kernel K_{LEX}, introduced in Sect. 2, is applied between words sharing the same POS-Tag. It means that words that belong to different shallow grammatical classes are never considered compatible, e.g., nouns with a verbs or adjectives.

The lexical similarity function is therefore crucial in order to provide a meaningful kernel estimation. As discussed in the following sections when focusing on empirical evaluations, this lexical kernel can be acquired from an existing lexicon or directly through Distributional modeling. Indeed, such general formulation also allows for using weighting schemes with different similarity functions. For examples, in Algorithm 1 the contribution of the lexical information is amplified (or reduced) trough a *lexical weight* (lw), that multiplies the similarity function between lexemes.

The underlying principle that allows employing SPTK in a kernel based learning algorithms, e.g. Support Vector Machine, is that SPTK must be a valid kernel. In order to demonstrate its validity, let us consider the node similarity function σ as a string matching between node labels and $\lambda = \mu = 1$. Each recursive step of Eq. 3 can be seen as a summation of $(1 + \prod_{j=1}^{l(\boldsymbol{I}_1)} \Delta_{STK}(c_{n_1}(\boldsymbol{I}_{1j}), c_{n_2}(\boldsymbol{I}_{2j})))$, i.e. the Δ_{STK} recursive equation, for all subsequences of children $c_{n_1}(\boldsymbol{I}_{1j})$. In other words, PTK is a summation of an exponential number of STKs, which are valid kernels. It follows that PTK is a kernel. Note that the multiplication by λ and μ elevated to any power only depends on the target fragment. Thus, it just gives an additional weight to the fragment and does not violate the Mercer's condition, that is discussed in [6]. In contrast, the multiplication by $\sigma(n_1, n_2)$ does depend on both comparing

examples, i.e. on n_1 and n_2. However, if the matrix $\left[\sigma(n_1, n_2)\right]\forall n_1, n_2 \in f \in \mathcal{F}$ is positive semi-definite, a decomposition exists such that $\sigma(n_1, n_2) = \phi(n_1)\phi(n_2) \Rightarrow \Delta_\sigma(n_1, n_2)$ can be written as $\sum_{i=1}^{|\mathcal{F}|} \phi(n_1)\chi_i(n_1)\phi(n_2)\chi_i(n_2) = \sum_{i=1}^{|\mathcal{F}|} \phi_\sigma(n_1)\phi_\sigma(n_2)$, which proves SPTK to be a valid kernel.

3.2 Proposed Computational Structures

The feature space generated by the structural kernels, presented in the previous section, obviously depends on the input structures. In case of PTK and SPTK different tree representations may lead to engineer more or less effective syntactic/semantic feature spaces, as discussed in [7, 39]. Due to their nature, *constituency parse* trees can be easily employed in the TK estimation. Given the following sentence:

(s1) *What is the width of a football field?*

The representation tree for a phrase structure paradigm leaves little room for variations as shown by the constituency tree (CT) in Fig. 2. We apply lemmatization to the lexemes to improve generalization and, at the same time, we add to them a generalized PoS-tag, i.e. noun (n::), verb (v::), adjective (::a), determiner (::d) and so on. This is useful in forcing similarity to insist only between lexemes of the same grammatical category.

In contrast, the conversion of *dependency structures* in computationally effective trees (for the above kernels) is not straightforward. We need to define the role of lexemes, PoS-tags and grammatical functions (GR). In order to transform the dependency graph in a tree structure, the edge label can be associated with tree nodes to surrogate the syntactic information. The basic idea of our structures is to use (i) one of the three kinds of information above the central nodes, from which dependencies are drawn and (ii) all the other information as features (in terms of dominated nodes) attached to the formed ones.

We define three main versions to represent dependency trees, such as the one shown in Fig. 3:

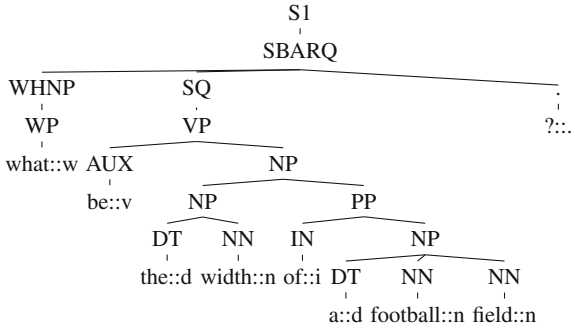

Fig. 2 Constituent Tree (CT)

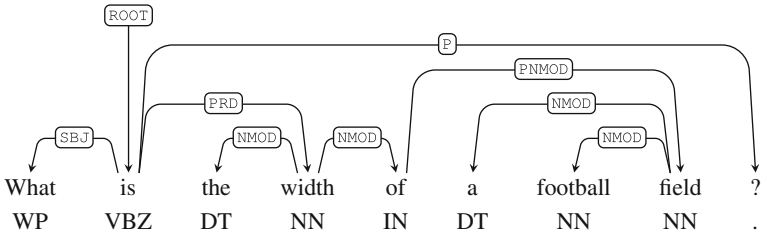

Fig. 3 Dependency Parse Tree

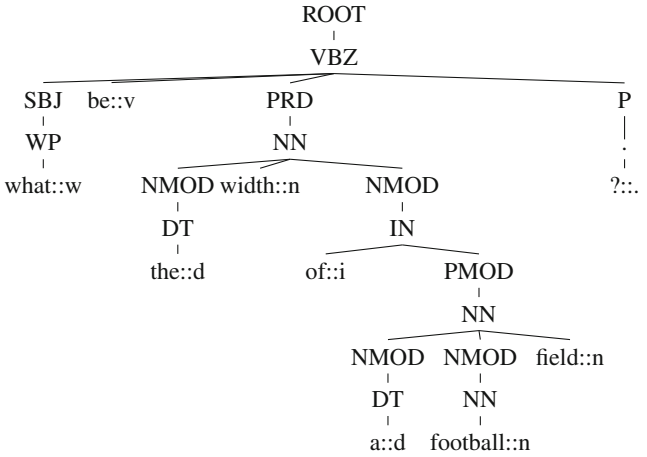

Fig. 4 PoS-Tag Centered Tree (PCT)

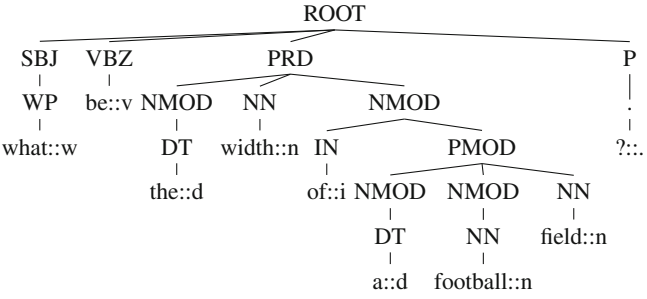

Fig. 5 Grammatical Relation Centered Tree (GRCT)

- the PoS-Tag Centered Tree (PCT), e.g. see Fig. 4, where the GR is added as father and the lexical as a child;
- the GR Centered Tree (GRCT), e.g. see Fig. 5, where the PoS-Tags are children of GR nodes and fathers of their associated lexemes;

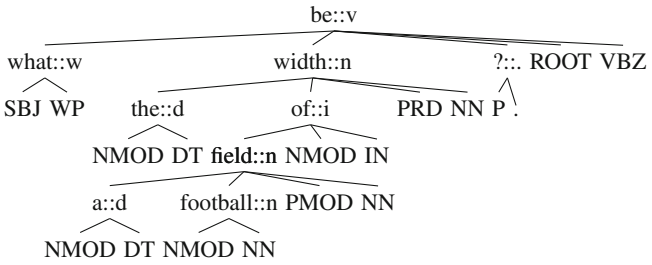

Fig. 6 Lexical Centered Tree (LCT)

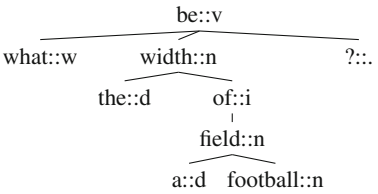

Fig. 7 Lexical Only Centered Tree (LOCT)

- the Lexical Centered Tree (LCT), e.g. see Fig. 6, in which both GR and PoS-Tag are added as the rightmost children.

To better study the role of the above dependency structures, especially from a performance perspective, we specify additional structures. Figure 7 shows the Lexical Only Centered Tree (LOCT) which is directly derived from the parse tree. It only accounts on the lexemes, where untyped binary relations are used for recursive structures. The grammatical generalization provided by the syntactic edge labels is thus neglected. In order to have a meaningful comparison, two trees whose structures does not reflect the sentence syntactic information are here defined. Figure 8 shows the Lexical and PoS-Tag Sequences Tree (LPST) in the form of a flattened tree with two levels, one for PoS-Tag information, where lexemes are simply added as leaves. Finally, in Fig. 9 only lexical items are leaves of a single root node. These

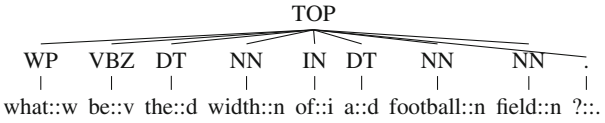

Fig. 8 Lexical and PoS-Tag Sequences Tree (LPST)

Fig. 9 Lexical Sequences Tree (LST)

two structures are interesting as they allow to employ a PTK or SPTK to surrogate the Sequence Kernel [40].

4 Experimental Evaluation

A large scale empirical evaluation is here discussed to describe the application of SPTK to a different semantic processing task: the Question Classification task in Sect. 4.2, the Verb Classification task in Sect. 4.3 and the Semantic Role Labeling task in Sect. 4.4. The aim of the following experiments is to analyze different levels of representation, i.e. structure, for syntactic dependency parses. Most importantly, the role of lexical similarity embedded in syntactic structures will be investigated.

4.1 General Experimental Setup

The following semantic processing task are modeled as a classification problem, where a SVM classifier is employed. For SVM learning, we extended the SVM-LightTK software[2] (which includes structural kernels in SVMLight [41]) with the smooth match between tree nodes. For generating constituency trees, we used the Charniak parser [42] whereas we applied LTH syntactic parser (described in [43]) to generate dependency trees. Lexical similarity is derived through the distributional analysis of UkWaC [44], which is a large scale document collection made by 2 billion tokens. More specifically, to build the matrix M, POS tagging is first applied so that its rows are pairs ⟨lemma, ::POS⟩, or lemma::POS in brief. The contexts of such items are the columns of M and are short windows of size $[-3, +3]$, centered on the items. This allows for better capturing syntactic properties of words. The most frequent 20,000 items are selected along with their 20 k contexts. The entries of M are the point-wise mutual information between them. The SVD reduction is then applied to M, with a dimensionality cut of $l = 250$. In Question Classification experiments the contribution of distributional models is compared with a resource based similarity derived from the word list (WL) provided in [13].

SVM-LightTK is applied to the different tree representations discussed in Sect. 3.2. We experiment with multi-classification, which we model through *one-vs-all* scheme

[2] http://disi.unitn.it/moschitti/Tree-Kernel.htm.

Table 1 Accuracy of several structural kernels on different structures for coarse and fine grained QC

	COARSE						FINE					
	NO		LSA		WL		NO		LSA		WL	
	lw	Acc. (%)	*lw*	Acc. (%)	*lw*	Acc. (%)	*lw*	Acc. (%)	*lw*	Acc. (%)	*lw*	Acc. (%)
CT	4	90.80	2	91.00	5	92.20	4	84.00	5	83.00	7	86.60
GRCT	3	**91.60**	4	92.60	2	**94.20**	3	83.80	4	83.20	2	85.00
LCT	1	90.80	1	**94.80**	1	**94.20**	0.33	**85.40**	1	86.20	0.33	**87.40**
LOCT	1	89.20	1	93.20	1	91.80	1	**85.40**	1	**86.80**	1	87.00
LST	1	88.20	1	85.80	1	89.60	1	84.00	1	80.00	1	85.00
LPST	3	89.40	1	89.60	1	92.40	3	84.20	4	82.20	1	84.60
PCT	4	91.20	4	92.20	5	93.40	4	84.80	5	84.00	5	85.20
CT-STK	–	91.20	–	–	–	–	–	82.20	–	–	–	–
BOW	–	88.80	–	–	–	–	–	83.20	–	–	–	–

by selecting the category associated with the maximum SVM margin. The quality of such classification is measured with accuracy. We determine the statistical signicance by using the model described in [45] and implemented in [46].

The parameterization of each classifier is carried on a held-out set and concerns with the setting of the trade-off parameter (option -c) and the Leaf Weight (lw) (see Algorithm 1), which is used to linearly scale the contribution of the leaf nodes. In contrast, the cost-factor parameter of the SVM-LightTK is set as the ratio between the number of negative and positive examples for attempting to have a balanced Precision/Recall.

4.2 Question Classification

The typical architecture of a QA system includes three main phases: question processing, document retrieval and answer extraction [2]. Question processing is usually centered around the so called Question Classification task. It maps a question into one of k predefined answer classes, thus posing constraints on the search space of possible answers. For these experiments, we used the UIUC dataset [13]. It is composed by a training set of 5,452 questions and a test set of 500 questions.[3] Question classes are organized in two levels: 6 coarse-grained classes (like `ENTITY` or `HUMAN`) and 50 fine-grained sub-classes (e.g. `Plant`, `Food` as subclasses of `ENTITY`).

The outcome of the several kernels applied to several structures for the coarse and fine grained QC is reported in Table 1. Since PTK and SPTK are typically used in our experiments, to have a more compact acronym for each model, we associate the

[3] http://cogcomp.cs.illinois.edu/Data/QA/QC/.

latter with the name of the structure, i.e. this indicates that PTK is applied to it. Then the presence of the subscript $_{WL}$ and $_{LSA}$ indicates that SPTK is applied along with the corresponding similarity, e.g. LCT$_{WL}$ is the SPTK kernel applied to LCT structure, using WL similarity. The first column shows the experimented models, obtained by applying PTK/SPTK to the structures described in Sect. 3.2. The last two rows are: CT-STK, i.e. Syntactic Tree Kernel, proposed in [7] applied to a constituency tree and BOW, which is a linear kernel applied to lexical vectors. Column 2, 3 and 4 report the accuracy using no, LSA and WL similarity, where lw is the amplifying parameter, i.e. weight, associated with the leaves in the tree. The last three columns refer to the fine- grained task.

It is worth nothing that when no similarity is applied: (i) BOW produces high accuracy, i.e. 88.8 % but it is improved by STK, current state-of-the-art[4] in QC; (ii) PTK applied to the same tree of STK produces a slightly lower value (non-statistically significant difference); (iii) interestingly, when PTK is instead applied to dependency structures, it improves STK, i.e. 91.60 versus 91.40 % (although not significantly); and (iv) LCT, strongly based on lexical nodes, is the less accurate, i.e 90.80 % since it is obviously subject to data sparseness (fragments only composed by lexicals are very sparse). The very important results can be noted when lexical similarity is used, i.e. SPTK is applied: (a) all the syntactic-base structures using both LSA or WL improve the classification accuracy (b) CT gets the lowest improvement whereas LCT achieves an impressive result of 94.80 %, i.e. more than 41 % of relative error reduction. It seems that the lexical similar paths when driven by syntax produces accurate features. Indeed, when syntax is missing such as for the unstructured lexical path of LST$_{LSA}$, the accuracy does not highly improve or may also decrease. Additionally, the result of our best model is so high that its errors only refer to questions like *What did Jesse Jackson organize?*, where the classifier selected `Entity` instead of `Human` category. These are clear examples where a huge amount of background knowledge is needed. Finally, on the fine grained experiments LCT still produces the most accurate outcome again exceeding the state-of-the-art [47], where WL significantly improves on all models (CT included).

4.3 Verb Classification

Verb classification is a fundamental topic of computational linguistics research given its importance for understanding the role of verbs in conveying semantics of natural language (NL). Currently, a lot of interest has been devoted to the VerbNet verb categorization scheme [48]. However, the definition of models for optimally combining lexical and syntactic constraints is still far for being accomplished. In particular, the exhaustive design and experimentation of lexical and syntactic features for learning

[4] Note that in [37], higher accuracy values for smoothed STK are shown for different parameters but the best according to a validation set is not highlighted.

verb classification appears to be computationally problematic. For example, the verb **order** can belongs to the two VerbNet classes:

- The class 60.1, i.e., *order someone to do something* as shown in: *The Illinois Supreme Court **ordered** the commission to audit Commonwealth Edison's construction expenses and refund any unreasonable expenses.*
- The class 13.5.1: *order or request something* like in: *... Michelle blabs about it to a sandwich man while **ordering** lunch over the phone.*

Clearly, the syntactic realization can be used to discern the cases above but it would not be enough to correctly classify the following verb occurrence: "*... ordered the lunch to be delivered ...*" in Verb class 13.5.1. For such a case, selectional restrictions are needed.

The implicit feature space generated by structural kernels and the corresponding notion of similarity between verbs obviously depend on the input structures. First we employed the constituency tree (CT) representation, enriching the target verb node with the `target` label. Here, we apply tree pruning to reduce the computational complexity of tree kernels as it is proportional to the number of nodes in the input trees. Accordingly, we only keep the subtree dominated by the target `VP` by pruning from it all the `S`-nodes along with their subtrees (i.e., all nested sentences are removed). To encode dependency structure information in a tree we employed the GR Centered Tree (GRCT) and the Lexical Centered Tree (LCT); for both trees, the pruning strategy only preserves the verb node, its direct ancestors (father and siblings) and its descendants up to two levels (i.e., direct children and grandchildren of the verb node). Note that our dependency tree can capture the semantic head of the verbal argument along with the main syntactic construct, e.g., *to audit*.

In these experiments, we tested the impact of our different verb representations using different kernels, similarities and parameters. We also compared with simple bag-of-words (BOW) models and the state-of-the-art. In particular, we used the same verb classification setting of [14]: sentences are drawn from the Semlink corpus [49], which consists of the PropBanked Penn Treebank portions of the Wall Street Journal. It contains 113 K verb instances, 97 K of which are verbs represented in at least one VerbNet class. Semlink includes 495 verbs, whose instances are labeled with more than one class (including one single VerbNet class or none). We used all instances of the corpus for a total of 45,584 instances for 180 verb classes. When instances labeled with the *none* class are not included, the number of examples becomes 23,719. We used 70 % of instances for training and 30 % for testing.

Our verb (multi) classifier is designed with the *one-vs-all* [50] multi-classification schema. This uses a set of binary SVM classifiers, one for each verb class (frame) i. The sentences whose verb is labeled with the class i are positive examples for the classifier i. The sentences whose verbs are compatible with the class i but evoking a different class or labeled with *none* (no current verb class applies) are added as negative examples. In the classification phase the binary classifiers are applied by (i) only considering classes that are compatible with the target verbs; and (ii) selecting the class associated with the maximum positive SVM margin. If all classifiers provide a negative score the example is labeled with *none*. To assess the performance of

our settings, we also derive a simple baseline based on the bag-of-words (BOW) model. For it, we represent an instance of a verb in a sentence using all words of the sentence (by creating a special feature for the predicate word). We also used a Sequence Kernel (SK) applied to the LST structure, described in Sect. 3.2; for efficiency reasons,[5] we only consider the 10 words before and after the predicate with subsequence features of length up to 5. Table 2 reports the accuracy of different models for VerbNet classification. It should be noted that: first, LST produces a much higher accuracy than BOW, i.e., 82.08 versus 79.08 %. On one hand, this is generally in contrast with standard text categorization tasks, for which n-gram models show accuracy comparable to the simpler BOW. On the other hand, it simply confirms that verb classification requires the dependency information between words (i.e., at least the sequential structure information provided by LST). Second, LST is 2.56 % points below the state-of-the-art achieved in [14] (BR), i.e., 82.08 versus 84.64. In contrast, STK applied to our representation (CT, GRCT and LCT) produces comparable accuracy, e.g., 84.83, confirming that syntactic representation is needed to reach the state-of-the-art. Third, PTK, which produces more general structures, improves over BR by almost 1.5 (statistically significant result) when using our dependency structures GRCT and LCT. CT does not produce the same improvement since it does not allow PTK to directly compare the lexical structure (lexemes are all leaf nodes in CT and to connect some pairs of them very large trees are needed). Finally, the best model of SPTK (i.e., using LCT) improves over the best PTK (i.e., using LCT) by almost 1 point (statistically significant result): this difference is only given by lexical similarity. SPTK improves on the state-of-the-art by about 2.08 absolute percent points, which, given the high accuracy of the baseline, corresponds to 13.5 % of relative error reduction.

Table 2 VerbNet accuracy with the *none* class

	STK		PTK		SPTK	
	lw	Acc. (%)	*lw*	Acc. (%)	*lw*	Acc. (%)
CT	–	83.83	8	**84.57**	8	84.46
GRCT	–	84.83	8	85.15	8	**85.28**
LCT	–	77.73	0.1	86.03	0.2	**86.72**
Br. et Al	84.64 %					
BOW	79.08 %					
LST	82.08 %					

[5] The average running time of the SK is much higher than the one of PTK. When a tree is composed by only one level PTK collapses to SK.

4.4 FrameNet Role Classification

To verify that our findings are general and that our syntactic/semantic dependency kernels can be effectively exploited for diverse NLP tasks, we experimented with a completely different application, i.e. FrameNet SRL classification. Given a predicate (the lexical unit, as described in Sect. 2) and a set of arguments, the Role Classification consists in the assignment of the proper role label to each argument. We used the FrameNet version 1.3 with the 90/10 % split between training and test set (i.e. 271,560 and 30,173 examples respectively), as defined in [9], one of the best system for FrameNet parsing. We used the LTH dependency parser. LSA was applied to the BNC corpus, the source of the FrameNet annotations.

For each of 648 frames, we applied SVM along with the best models for QC, i.e. GRCT and LCT, to learn its associated binary role classifiers (RC) for a total of 4,254 classifiers. For example, Fig. 10 shows the LCT representation of the first two roles of the following sentence:

$$[\textit{Bootleggers}]_{\text{CREATOR}}, \textit{then } \textbf{copy } [\textit{the film}]_{\text{ORIGINAL}}$$
$$[\textit{onto hundreds of VHS tapes}]_{\text{GOAL}}$$

Table 3 shows the results of the different multi-classifiers. GRCT and LCT show a large accuracy, i.e. 87.60 %. This improves up to 88.74 % by activating the LSA similarity. The combination GRCT_{LSA}+LCT_{LSA} significantly improves the above model, achieving 88.91 %. This is very close to the state-of-the-art of SRL for classification (using a single classifier, i.e. no joint model), i.e. 89.6 %, achieved in [9]. These results thus confirm the idea that a lexical generalization allows to improve the quality of the Argument Classification, especially for examples where the syntactic information alone is not discriminative, like the examples of Sentences 1 and 2. Finally, it should be noted that, to learn and test the SELF_MOTION multi-classifier, containing 14,584 examples, distributed on 22 roles, SVM-SPTK employed 1.5 h and 10 min, respectively.[6]

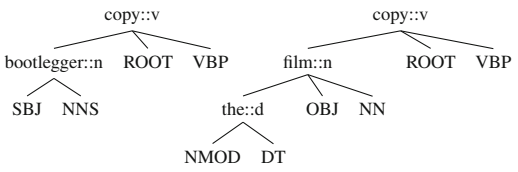

Fig. 10 LCT Examples for argument roles

[6] Using one of the 8 processors of an Intel(R) Xeon(R) CPU E5430 @ 2.66 GHz machine, 32 Gb Ram.

Table 3 Argument Classification Accuracy

Kernel	Accuracy (%)
GRCT	87.60
$GRCT_{LSA}$	88.61
LCT	87.61
LCT_{LSA}	88.74
GRCT + LCT	87.99
$GRCT_{LSA}$ + LCT_{LSA}	**88.91**

5 Conclusions

This paper has proposed a study on representation of dependency structures for the design of effective structural kernels. Most importantly, we have defined a new class of kernel functions, i.e. SPTK, that carry out syntactic and lexical similarity on the above structures. This allows for automatically generating feature spaces of generalized syntactic/semantic dependency substructures. To test our models, we carried out experiments on Question Classification, Verb Classification and Semantic Role Labeling. These show that by exploiting the similarity between two sets of words carried out according to their dependency structure leads to an unprecedented result, whereas no structure is used the accuracy does not significantly improves. We have also provided a fast algorithm for the computation of SPTK and empirically shown that it can easily scale. Such result enables many promising future research directions: the most important being the use of SPTK for many NLP tasks with many different similarities.

References

1. Baeza-Yates, R.A., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Boston (1999)
2. Kwok, C.C., Etzioni, O., Weld, D.S.: Scaling question answering to the web. In: World Wide Web, pp. 150–161 (2001)
3. Pang, B., Lee, L.: Opinion mining and sentiment analysis. Found. Trends Inf. Retr. **2**(1–2), 1–135 (2008)
4. Jelinek, F.: Statistical Methods for Speech Recognition. The MIT Press, Cambridge (1998)
5. Manning, C.D., Schütze, H.: Foundations of Statistical Natural Language Processing. MIT Press, Cambridge (1999)
6. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience, New York (1998)
7. Collins, M., Duffy, N.: Convolution kernels for natural language. In: Proceedings of Neural Information Processing Systems (NIPS'2001), pp. 625–632 (2001)
8. Shawe-Taylor, J., Cristianini, N.: Kernel Methods for Pattern Analysis. Cambridge University Press (2004).
9. Johansson, R., Nugues, P.: The effect of syntactic representation on semantic role labeling. In: Proceedings of COLING, Manchester, 18–22 Aug 2008
10. Pado, S., Lapata, M.: Dependency-based construction of semantic space models. Comput. Linguist. **33**(2), (2007)

11. Sahlgren, M.: The Word-space model. PhD thesis, Stockholm University (2006)
12. Schütze, H.: Word space. In: Advances in Neural Information Processing Systems 5, pp. 895–902. Morgan Kaufmann (1993)
13. Li, X., Roth, D.: Learning question classifiers. In: Proceedings of ACL'02 (2002)
14. Brown, S.W., Dligach, D., Palmer, M.: Verbnet class assignment as a WSD task. In: Proceedings of the Ninth International Conference on Computational Semantics, IWCS'11, pp. 85–94. Association for Computational Linguistics, Stroudsburg (2011)
15. Gildea, D., Palmer, M.: The necessity of parsing for predicate argument recognition. In: Proceedings of the 40th Annual Conference of the Association for Computational Linguistics (ACL-02), Philadelphia (2002)
16. Gildea, D., Jurafsky, D.: Automatic Labeling of Semantic Roles. Comput. Linguist. **28**(3), 245–288 (2002)
17. Fillmore, C.J.: Frames and the semantics of understanding. Quaderni di Semantica **4**(2), 222–254 (1985)
18. Palmer, M., Kingsbury, P., Gildea, D.: The proposition bank: an annotated corpus of semantic roles. Comput. Linguist. **31**(1), 71–106 (2005)
19. Baker, C.F., Fillmore, C.J., Lowe, J.B.: The Berkeley FrameNet project. In: Proceedings of COLING-ACL, Montreal, Canada (1998)
20. Carreras, X., Màrquez, L.: Introduction to the CoNLL-2005 Shared Task: Semantic Role Labeling. In: Proceedings of CoNLL-2005, Ann Arbor, Michigan, June 2005, pp. 152–164
21. Pradhan, S., Hacioglu, K., Krugler, V., Ward, W., Martin, J.H.: Support vector learning for semantic argument classification. Mach. Learn. J. **60**(1–3), 11–39 (2005)
22. Coppola, B., Moschitti, A., Riccardi, G.: Shallow semantic parsing for spoken language understanding. In: Proceedings of NAACL'09, pp. 85–88. Morristown, NJ (2009)
23. Moschitti, A., Pighin, D., Basili, R.: Tree kernels for semantic role labeling. Comput. Linguist. **34**(2), 193–224 (2008)
24. Firth, J.: A synopsis of linguistic theory 1930–1955. In: Studies in Linguistic Analysis. Philological Society, Oxford (1957) reprinted in Palmer, F. (ed.) Selected Papers of J. R. Firth, Longman, Harlow (1968)
25. Wittgenstein, L.: Philosophical Investigations. Blackwells, Oxford (1953)
26. Pantel, P., Bhagat, R., Coppola, B., Chklovski, T., Hovy, E.: ISP: Learning inferential selectional preferences. In: Proceedings of HLT/NAACL (2007)
27. Turney, P.D., Pantel, P.: From frequency to meaning: Vector space models of semantics. J. Artif. Intell. Res. **37**, 141–188 (2010)
28. Landauer, T., Dumais, S.: A solution to plato's problem: the latent semantic analysis theory of acquisition, induction and representation of knowledge. Psychol. Rev. **104** (1997)
29. Golub, G., Kahan, W.: Calculating the singular values and pseudo-inverse of a matrix. J. Soc. Ind. Appl. Math.: Ser. B, Numer. Anal. **2**(2), 205–224 (1965)
30. Cristianini, N., Shawe-Taylor, J., Lodhi, H.: Latent semantic kernels. In: Brodley, C., Danyluk, A. (eds.) Proceedings of the 18th International Conference on Machine Learning (ICML-01), pp. 66–73. Morgan Kaufmann Publishers, San Francisco, Williams College (2001)
31. Mitchell, J., Lapata, M.: Composition in distributional models of semantics. Cogn. Sci. **34**, 1388–1429 (2010)
32. Annesi, P., Storch, V., Basili, R.: Space projections as distributional models for semantic composition. In: Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing, CICLing'12. Springer (2012)
33. Baroni, M., Lenci, A.: One distributional memory, many semantic spaces. In: Proceedings of the GEMS 2009 Workshop. GEMS'09, pp. 1–8. Stroudsburg (2009)
34. Clark, S., Pulman, S.: Combining symbolic and distributional models of meaning. In: Proceedings of the AAAI Spring Symposium on Quantum Interaction, pp. 52–55 (2007)
35. Grefenstette, E., Sadrzadeh, M.: Experimental support for a categorical compositional distributional model of meaning. In: Proceedings of EMNLP 2011, Edinburgh, Scotland, UK (2011)

36. Zanzotto, F.M., Korkontzelos, I., Fallucchi, F., Manandhar, S.: Estimating linear models for compositional distributional semantics. In: Proceedings of the 23rd International Conference on Computational Linguistics (COLING'10), pp. 1263–1271. Association for Computational Linguistics, Stroudsburg (2010)
37. Bloehdorn, S., Moschitti, A.: Structure and semantics for expressive text kernels. In: Proceedings of CIKM (2007)
38. Mehdad, Y., Moschitti, A., Zanzotto, F.M.: Syntactic/semantic structures for textual entailment recognition. In: HLT-NAACL, pp. 1020–1028 (2010)
39. Moschitti, A.: Efficient convolution kernels for dependency and constituent syntactic trees. In: 17th European Conference on Machine Learning, Proceedings, Machine Learning: ECML 2006, pp. 318–329. ECML, Berlin, Germany, Sept 2006
40. Cancedda, N., Gaussier, E., Goutte, C., Renders, J.M.: Word sequence kernels. J. Mach. Learn. Res. **3**, 1059–1082 (2003)
41. Joachims, T.: Estimating the generalization performance of a SVM efficiently. In: Proceedings of ICML'00 (2000)
42. Charniak, E.: A maximum-entropy-inspired parser. In: Proceedings of NAACL'00 (2000)
43. Johansson, R., Nugues, P.: Dependency-based syntactic-semantic analysis with PropBank and NomBank. In: Proceedings of the Twelfth Conference on Natural Language Learning (CoNLL 2008), pp. 183–187. Manchester (2008)
44. Baroni, M., Bernardini, S., Ferraresi, A., Zanchetta, E.: The wacky wide web: a collection of very large linguistically processed web-crawled corpora. Lang. Res. Eval. **43**(3), 209–226 (2009)
45. Yeh, A.S.: More accurate tests for the statistical significance of result differences. In: COLING, pp. 947–953 (2000)
46. Padó, S.: User's guide to `sigf`: significance testing by approximate randomisation (2006)
47. Zhang, D., Lee, W.S.: Question classification using support vector machines. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Informaion Retrieval, pp. 26–32. ACM Press (2003)
48. Schuler, K.K.: VerbNet: A broad-coverage, comprehensive verb lexicon. PhD thesis, University of Pennsylyania (2005)
49. Loper, E., ting Yi, S., Palmer, M.: Combining lexical resources: mapping between propbank and verbnet. In: Proceedings of the 7th International Workshop on Computational Linguistics (2007)
50. Rifkin, R., Klautau, A.: In defense of one-vs-all classification. J. Mach. Learn. Res. **5**, 101–141 (2004)