A Walk-based Semantically Enriched Tree Kernel Over Distributed Word Representations

Shashank Srivastava[1] **Dirk Hovy**[2] **Eduard Hovy**[1]
(1) Carnegie Mellon University, Pittsburgh
(2) Center for Language Technology, University of Copenhagen, Denmark
{ssrivastava,hovy}@cmu.edu, mail@dirkhovy.com

Abstract

In this paper, we propose a walk-based graph kernel that generalizes the notion of tree-kernels to continuous spaces. Our proposed approach subsumes a general framework for word-similarity, and in particular, provides a flexible way to incorporate distributed representations. Using vector representations, such an approach captures both distributional semantic similarities among words as well as the structural relations between them (encoded as the structure of the parse tree). We show an efficient formulation to compute this kernel using simple matrix operations. We present our results on three diverse NLP tasks, showing state-of-the-art results.

1 Introduction

Capturing semantic similarity between sentences is a fundamental issue in NLP, with applications in a wide range of tasks. Previously, tree kernels based on common substructures have been used to model similarity between parse trees (Collins and Duffy, 2002; Moschitti, 2004; Moschitti, 2006b). These kernels encode a high number of latent syntactic features within a concise representation, and compute the similarity between two parse trees based on the matching of node-labels (words, POS tags, etc.), as well as the overlap of tree structures. While this is sufficient to capture *syntactic similarity*, it does not capture *semantic similarity* very well, even when using discrete semantic types as node labels. This constrains the utility of many traditional tree kernels in two ways: i) two sentences that are syntactically identical, but have no semantic similarity can receive a high matching score (see Table 1, top) while ii) two sentences with only local syntactic overlap, but high semantic similarity can receive low scores (see Table 1, bottom).

tree pairs	semantic	syntactic	score
love (we, toys (green)) — crush (they, puppies (little))	✗	✓	high
kissed (she, cat (her)) — gave (she, friend (her, feline), kiss (a))	✓	✗	low

Table 1: Traditional tree kernels do not capture semantic similarity

In contrast, distributional vector representations of words have been successful in capturing fine-grained semantics, but lack syntactic knowledge. Resources such as Wordnet, dictionaries and ontologies that encode different semantic perspectives can also provide additional knowledge infusion.

In this paper, we describe a generic walk-based graph kernel for dependency parse trees that subsumes general notions of word-similarity, while focusing on vector representations of words to capture lexical semantics. Through a convolutional framework, our approach takes into account the distributional semantic similarities between words in a sentence as well as the structure of the parse tree. Our main contributions are:

1. We present a new graph kernel for NLP that extends to distributed word representations, and diverse word similarity measures.

2. Our proposed approach provides a flexible framework for incorporating both syntax and semantics of sentence level constructions.

3. Our generic kernel shows state-of-the-art performance on three eclectic NLP tasks.

2 Related Work

Tree kernels in NLP Tree kernels have been extensively used to capture syntactic information about parse trees in tasks such as parsing (Collins and Duffy, 2002), NER (Wang et al., 2010; Cumby and Roth, 2003), SRL (Moschitti et al., 2008) and relation extraction (Qian et al., 2008). These kernels are based on the paradigm that parse trees are similar if they contain many common substructures, consisting of nodes with identical labels (Vishwanathan and Smola, 2003; Collins and Duffy, 2002). Moschitti (2006a) proposed a partial tree kernel that adds flexibility in matching tree substructures. Croce et al. (2011) introduce a lexical semantic tree kernel that incorporates continuous similarity values between node labels, albeit with a different focus than ours and would not match words with different POS. This would miss the similarity of "feline friend" and "cat" in our examples, as it requires matching the adjective "feline" with "cat", and verb "kissed" with "kiss".

Walk based kernels Kernels for structured data derive from the seminal Convolution Kernel formalism by Haussler (1999) for designing kernels for structured objects through local decompositions. Our proposed kernel for parse trees is most closely associated with the random walk-based kernels defined by Gartner et al. (2003) and Kashima et al. (2003). The walk-based graph kernels proposed by Gartner et al. (2003) count the common walks between two input graphs, using the adjacency matrix of the product graph. This work extends to graphs with a finite set of edge and node labels by appropriately modifying the adjacency matrix. Our kernel differs from these kernels in two significant ways: (i) Our method extends beyond label matching to continuous similarity metrics (this conforms with the very general formalism for graph kernels in Vishwanathan et al. (2010)). (ii) Rather than using the adjacency matrix to model edge-strengths, we modify the product graph and the corresponding adjacency matrix to model node similarities.

3 Vector Tree Kernels

In this section, we describe our kernel and an algorithm to compute it as a simple matrix multiplication formulation.

3.1 Kernel description

The similarity kernel K between two dependency trees can be defined as:

$$K(T_1, T_2) = \sum_{\substack{h_1 \subseteq T_1, h_2 \subseteq T_2 \\ len(h_1)=len(h_2)}} k(h_1, h_2)$$

where the summation is over pairs of equal length walks h_1 and h_2 on the trees T_1 and T_2 respectively. The similarity between two n length walks, $k(h_1, h_2)$, is in turn given by the pairwise similarities of the corresponding nodes ${v_i}^h$ in the respective walks, measured via the node similarity kernel κ:

$$k(h_1, h_2) = \prod_{i:1}^{n} \kappa(v_i^{h_1}, v_i^{h_2})$$

In the context of parse trees, nodes $v_i^{h_1}$ and $v_i^{h_2}$ correspond to words in the two parse trees, and thus can often be conveniently represented as vectors over distributional/dependency contexts. The vector representation allows us several choices for the node kernel function κ. In particular, we consider:

1. Gaussian : $\kappa(v_1, v_2) = \exp\big(-\frac{\|v_1 - v_2\|^2}{2\sigma^2}\big)$
2. Positive-Linear: $\kappa(v_1, v_2) = \max(v_1^T v_2, 0)$
3. Sigmoid: $\kappa(v_1, v_2) = \big(1 + \tanh(\alpha v_1^T v_2)\big)/2$

We note that the kernels above take strictly non-negative values in $[0, 1]$ (assuming word vector representations are normalized). Non-negativity is necessary, since we define the walk kernel to be the product of the individual kernels. As walk kernels are products of individual node-kernels, boundedness by 1 ensures that the kernel contribution does not grow arbitrarily for longer length walks.

The kernel function K puts a high similarity weight between parse trees if they contain common walks with semantically similar words in corresponding positions. Apart from the Gaussian kernel, the other two kernels are based on the dot-product of the word vector representations. We observe that the positive-linear kernel defined above is *not* a Mercer kernel, since the max operation makes it non-positive semidefinite (PSD). However, this formulation has desirable properties, most significant being that all walks with one or more node-pair mismatches are strictly penalized and add no score to

the tree-kernel. This is a more selective condition than the other two kernels, where mediocre walk combinations could also add small contributions to the score. The sigmoid kernel is also non-PSD, but is known to work well empirically (Boughorbel et al., 2005). We also observe while the summation in the kernel is over equal length walks, the formalism can allow comparisons over different length paths by including self-loops at nodes in the tree.

With a notion of similarity between words that defines the local node kernels, we need computational machinery to enumerate all pairs of walks between two trees, and compute the summation over products in the kernel $K(T_1, T_2)$ efficiently. We now show a convenient way to compute this as a matrix geometric series.

3.2 Matrix Formulation for Kernel Computation

Walk-based kernels compute the number of common walks using the adjacency matrix of the product graph (Gartner et al., 2003). In our case, this computation is complicated by the fact that instead of *counting* common walks, we need to compute a product of node-similarities for each walk. Since we compute similarity scores over nodes, rather than edges, the product for a walk of length n involves $n+1$ factors.

However, we can still compute the tree kernel K as a simple sum of matrix products. Given two trees $T(V, E)$ and $T'(V', E')$, we define a modified product graph $G(V_p, E_p)$ with an additional ghost node u added to the vertex set. The vertex and edge sets for the modified product graph are given as:

$$V_p := \{(v_{i1}, {v_{j1}}') : v_{i1} \in V, {v_{j1}}' \in V'\} \cup u$$

$$E_p := \{((v_{i1}, {v_{j1}}'), (v_{i2}, {v_{j2}}')) : (v_{i1}, v_{i2}) \in E, ({v_{j1}}', {v_{j2}}') \in E'\} \bigcup \{(u, (v_{i1}, {v_{j1}}')) : v_{i1} \in V, {v_{j1}}' \in V'\}$$

The modified product graph thus has additional edges connecting u to all other nodes. In our formulation, u now serves as a starting location for all random walks on G, and a $k+1$ length walk of G corresponds to a pair of k length walks on T and T'. We now define the weighted adjacency matrix W for G, which incorporates the local node kernels.

$$W_{(v_{i1}, {v_{j1}}'), (v_{i2}, {v_{j2}}')} = \begin{cases} 0 : {}_{((v_{i1}, {v_{j1}}'), (v_{i2}, {v_{j2}}'))} \notin E_p \\ \kappa(v_{i2}, {v_{j2}}') : otherwise \end{cases}$$

$W_{u,(v_{i1}, {v_{j1}}')} = \kappa(v_{i1}, {v_{j1}}')$
$W_{(v,u)} = 0 \ \ \forall\, v \in V_p$

There is a straightforward bijective mapping from walks on G starting from u to pairs of walks on T and T'. Restricting ourselves to the case when the first node of a $k+1$ length walk is u, the next k steps allow us to efficiently compute the products of the node similarities along the k nodes in the corresponding k length walks in T and T'. Given this adjacency matrix for G, the sum of values of k length walk kernels is given by the u^{th} row of the $(k+1)^{th}$ exponent of the weighted adjacency matrix (denoted as W^{k+1}). This corresponds to $k+1$ length walks on G starting from u and ending at any node. Specifically, $W_{u,(v_i, v'_j)}$ corresponds to the sum of similarities of all common walks of length n in T and T' that end in v_i in T and v'_j in T'. The kernel K for walks upto length N can now be calculated as :

$$K(T, T') = \sum_{i}^{|V_p|} S_{u,i}$$

where

$$S = W + W^2 + ...W^{N+1}$$

We note that in out formulation, longer walks are naturally discounted, since they involve products of more factors (generally all less than unity).

The above kernel provides a similarity measure between any two pairs of dependency parse-trees. Depending on whether we consider directional relations in the parse tree, the edge set E_p changes, while the procedure for the kernel computation remains the same. Finally, to avoid larger trees yielding larger values for the kernel, we normalize the kernel by the number of edges in the product graph.

4 Experiments

We evaluate the Vector Tree Kernel (VTK) on three NLP tasks. We create dependency trees using the FANSE parser (Tratz and Hovy, 2011), and use distribution-based SENNA word embeddings by Collobert et al. (2011) as word representations. These embeddings provide low-dimensional vector

representations of words, while encoding distributional semantic characteristics. We use LibSVM for classification. For sake of brevity, we only report results for the best performing kernel.

We first consider the Cornell Sentence Polarity dataset by Pang and Lee (2005). The task is to identify the polarity of a given sentence. The data consists of 5331 sentences from positive and negative movie reviews. Many phrases denoting sentiments are lexically ambiguous (cf. "*terribly* entertaining" vs "*terribly* written"), so simple lexical approaches are not expected to work well here, while syntactic context could help disambiguation.

Next, we try our approach on the MSR paraphrase corpus. The data contains a training set of 4077 pairs of sentences, annotated as paraphrases and non-paraphrases, and a test-set of 1726 sentence pairs. Each instance consists of a pair of sentences, so the VTK cannot be directly used by a kernel machine for classification. Instead, we generate 16 kernel values based for each pair on different parameter settings of the kernel, and feed these as features to a linear SVM.

We finally look at the annotated Metaphor corpus by (Hovy et al., 2013). The dataset consists of sentences with specified target phrases. The task here is to classify the target use as literal or metaphorical. We focus on target phrases by upweighting walks that pass through target nodes. This is done by simply multiplying the corresponding entries in the adjacency matrix by a constant factor.

5 Results

5.1 Sentence Polarity Dataset

	Prec	Rec	F1	Acc
Albornoz et al	0.63	–	–	0.63
WNA+synsets	0.61	–	–	0.61
WNA	0.53	–	–	0.51
DSM	0.54	0.55	0.55	0.54
SSTK	0.49	0.48	0.48	0.49
VTK	**0.65**	**0.58**	**0.62**	**0.67**

Table 2: Results on Sentence Polarity dataset

On the polarity data set, Vector Tree Kernel (VTK) significantly outperforms the state-of-the-art method by Carrillo de Albornoz et al. (2010), who use a hybrid model incorporating databases of affective lexicons, and also explicitly model the effect of negation and quantifiers (see Table 2). Lexical approaches using pairwise semantic similarity of SENNA embeddings (DSM), as well as Wordnet Affective Database-based (WNA) labels perform poorly (Carrillo de Albornoz et al., 2010), showing the importance of syntax for this particular problem. On the other hand, a syntactic tree kernel (SSTK) that ignores distributional semantic similarity between words, fails as expected.

5.2 MSR Paraphrase Dataset

	Prec	Rec	F1	Acc
BASE	0.72	0.86	0.79	0.69
Zhang et al	0.74	0.88	0.81	0.72
Qiu et al	0.73	0.93	0.82	0.72
Malakasiotis	0.74	0.94	0.83	0.74
Finch	**0.77**	0.90	**0.83**	**0.75**
VTK	0.72	**0.95**	0.82	0.72

Table 3: Results on MSR Paraphrase corpus

On the MSR paraphrase corpus, VTK performs competitively against state-of-the-art-methods. We expected paraphrasing to be challenging to our method, since it can involve little syntactic overlap. However, data analysis reveals that the corpus generally contains sentence pairs with high syntactic similarity. Results for this task are encouraging since ours is a general approach, while other systems use multiple task-specific features like semantic role labels, active-passive voice conversion, and synonymy resolution. In the future, incorporating such features to VTK should further improve results for this task .

5.3 Metaphor Identification

	Acc	P	R	F1
CRF	0.69	**0.74**	0.50	0.59
SVM+DSM	0.70	0.63	0.80	0.71
SSTK	0.75	0.70	0.80	0.75
VTK	**0.76**	0.67	**0.87**	**0.76**

Table 4: Results on Metaphor dataset

On the Metaphor corpus, VTK improves the previous score by Hovy et al. (2013), whose approach uses an conjunction of lexical and syntactic tree kernels (Moschitti, 2006b), and distributional vectors. VTK identified several templates of metaphor usage such as "warm heart" and "cold shoulder". We look towards approaches for automatedly mining such metaphor patterns from a corpus.

6 Conclusion

We present a general formalism for walk-based kernels to evaluate similarity of dependency trees.

Our method generalizes tree kernels to take distributed representations of nodes as input, and capture both lexical semantics and syntactic structures of parse trees. Our approach has tunable parameters to look for larger or smaller syntactic constructs. Our experiments shows state-of-the-art performance on three diverse NLP tasks. The approach can generalize to any task involving structural and local similarity, and arbitrary node similarity measures.

References

Sabri Boughorbel, Jean-Philippe Tarel, and Nozha Boujemaa. 2005. Conditionally positive definite kernels for svm based image recognition. In *ICME*, pages 113–116.

Jorge Carrillo de Albornoz, Laura Plaza, and Pablo Gervás. 2010. A hybrid approach to emotional sentence polarity and intensity classification. In *Proceedings of the Fourteenth Conference on Computational Natural Language Learning*, pages 153–161. Association for Computational Linguistics.

Michael Collins and Nigel Duffy. 2002. New ranking algorithms for parsing and tagging: Kernels over discrete structures, and the voted perceptron. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 263–270. Association for Computational Linguistics.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1034–1046. Association for Computational Linguistics.

Chad Cumby and Dan Roth. 2003. On kernel methods for relational learning. In *In Proc. of the International Conference on Machine Learning*, pages 107–114.

Andrew Finch. 2005. Using machine translation evaluation techniques to determine sentence-level semantic equivalence. In *In IWP2005*.

Thomas Gartner, Peter Flach, and Stefan Wrobel. 2003. On graph kernels: Hardness results and efficient alternatives. In *Proceedings of the Annual Conference on Computational Learning Theory*, pages 129–143. Springer.

David Haussler. 1999. Convolution kernels on discrete structures. Technical Report Technical Report UCS-CRL-99-10, UC Santa Cruz.

Dirk Hovy, Shashank Srivastava, Sujay Kumar Jauhar, Mrinmaya Sachan, Kartik Goyal, Huiying Li, Whitney Sanders, and Eduard Hovy. 2013. Identifying metaphorical word use with tree kernels. In *Proceedings of NAACL HLT, Meta4NLP Workshop*.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *Proceedings of the Twentieth International Conference on Machine Learning*, pages 321–328. AAAI Press.

Prodromos Malakasiotis. 2009. Paraphrase recognition using machine learning to combine similarity measures. In *Proceedings of the ACL-IJCNLP 2009 Student Research Workshop*, ACLstudent '09, pages 27–35, Stroudsburg, PA, USA. Association for Computational Linguistics.

Alessandro Moschitti, Daniele Pighin, and Roberto Basili. 2008. Tree kernels for semantic role labeling. *Computational Linguistics*, 34(2):193–224.

Alessandro Moschitti. 2004. A study on convolution kernels for shallow semantic parsing. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*, pages 335–es. Association for Computational Linguistics.

Alessandro Moschitti. 2006a. Efficient convolution kernels for dependency and constituent syntactic trees. In *Machine Learning: ECML 2006*, pages 318–329. Springer.

Alessandro Moschitti. 2006b. Making Tree Kernels Practical for Natural Language Learning. In *In Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics*.

Bo Pang and Lillian Lee. 2005. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. In *Proceedings of the ACL*.

Longhua Qian, Guodong Zhou, Fang Kong, Qiaoming Zhu, and Peide Qian. 2008. Exploiting constituent dependencies for tree kernel-based semantic relation extraction. In *Proceedings of the 22nd International Conference on Computational Linguistics-Volume 1*, pages 697–704. Association for Computational Linguistics.

Long Qiu, Min-Yen Kan, and Tat-Seng Chua. 2006. Paraphrase recognition via dissimilarity significance classification. In *Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing*, EMNLP '06, pages 18–26, Stroudsburg, PA, USA. Association for Computational Linguistics.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *In Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149.

Stephen Tratz and Eduard Hovy. 2011. A fast, accurate, non-projective, semantically-enriched parser. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, EMNLP '11, pages 1257–1268, Stroudsburg, PA, USA. Association for Computational Linguistics.

S. V. N. Vishwanathan and Alexander J. Smola. 2003. Fast kernels for string and tree matching. In *Advances In Neural Information Processing Systems 15*, pages 569–576. MIT Press.

S. V. N. Vishwanathan, Nicol N. Schraudolph, Risi Kondor, and Karsten M. Borgwardt. 2010. Graph kernels. *J. Mach. Learn. Res.*, 99:1201–1242, August.

Xinglong Wang, Jun'ichi Tsujii, and Sophia Ananiadou. 2010. Disambiguating the species of biomedical named entities using natural language parsers. *Bioinformatics*, 26(5):661–667.

Yitao Zhang and Jon Patrick. 2005. Paraphrase identification by text canonicalization. In *In Proceedings of the Australasian Language Technology Workshop 2005*.