Structural Representations for Learning Relations between Pairs of Texts

Simone Filice and **Giovanni Da San Martino** and **Alessandro Moschitti**
ALT, Qatar Computing Research Institute, Hamad Bin Khalifa University
{sfilice,gmartino,amoschitti}@qf.org.qa

Abstract

This paper studies the use of structural representations for learning relations between pairs of short texts (e.g., sentences or paragraphs) of the kind: the second text answers to, or conveys exactly the same information of, or is implied by, the first text. Engineering effective features that can capture syntactic and semantic relations between the constituents composing the target text pairs is rather complex. Thus, we define syntactic and semantic structures representing the text pairs and then apply graph and tree kernels to them for automatically engineering features in Support Vector Machines. We carry out an extensive comparative analysis of state-of-the-art models for this type of relational learning. Our findings allow for achieving the highest accuracy in two different and important related tasks, i.e., Paraphrasing Identification and Textual Entailment Recognition.

1 Introduction

Advanced NLP systems, e.g., IBM Watson system (Ferrucci et al., 2010), are the result of effective use of syntactic/semantic information along with relational learning (RL) methods. This research area is rather vast including, extraction of syntactic relations, e.g., (Nastase et al., 2013), predicate relations, e.g., Semantic Role Labeling (Carreras and Màrquez, 2005) or FrameNet parsing (Gildea and Jurafsky, 2002) and relation extraction between named entities, e.g., (Mintz et al., 2009).

Although extremely interesting, the above methods target relations only between text constituents whereas the final goal of an intelligent system would be to interpret the semantics of larger pieces of text, e.g., sentences or paragraphs. This line of research relates to three broad fields, namely, Question Answering (QA) (Voorhees and Tice, 1999), Paraphrasing Identification (PI) (Dolan et al., 2004) and Recognition of Textual Entailments (RTE) (Giampiccolo et al., 2007). More generally, RL from text can be denied as follows: given two text fragments, the main goal is to derive relations between them, e.g., either if the second fragment answers the question, or conveys exactly the same information or is implied by the first text fragment. For example, the following two sentences:

- *License revenue slid 21 percent, however, to $107.6 million.*
- *License sales, a key measure of demand, fell 21 percent to $107.6 million.*

express exactly the same meaning, whereas the next one:

- *She was transferred again to Navy when the American Civil War began, 1861.*

implies:

- *The American Civil War started in 1861.*

Automatic learning a model for deriving the relations above is rather complex as any of the text constituents, e.g., *License revenue*, *a key measure of demand*, in the two sentences plays an important role. Therefore, a suitable approach should exploit representations that can structure the two sentences and put their constituents in relation. Since the dependencies between constituents can be an exponential number and representing structures in learning algorithms is rather challenging, automatic feature engineering through kernel methods (Shawe-Taylor and Cristianini, 2004; Moschitti, 2006) can be a promising direction.

In particular, in (Zanzotto and Moschitti, 2006), we represented the two evaluating sentences for the RTE task with syntactic structures and then applied tree kernels to them. The resulting system was very accurate but, unfortunately, it could not scale to large datasets as it is based on a compu-

Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing, pages 1003–1013

tationally exponential algorithm. This prevents its application to PI tasks, which typically require a large dataset to train the related systems.

In this paper, we carry out an extensive experimentation using different kernels based on trees and graphs and their combinations with the aim of assessing the best model for relation learning between two entire sentences (or even paragraphs). More in detail, (i) we design many models for RL combining state-of-the-art tree kernels and graph kernels and apply them to innovative computational structures. These innovative combinations use for the fist time semantic/syntactic tree kernels and graph kernels for the tackled tasks. (ii) Our kernels provide effective and efficient solutions, which solve the previous scalability problem and, at the same time, exceed the state of the art on both RTE and PI. Finally, our study suggests research directions for designing effective graph kernels for RL.

2 Related Work

In this paper, we apply kernel methods, which enable an efficient comparison of structures in huge, possibly infinite, feature spaces. While for trees, a comparison using all possible subtrees is possible, designing kernel functions for graphs with such property is an NP-Hard problem (i.e., it shows the same complexity of the graph isomorphism problem) (Gartner et al., 2003). Thus most kernels for graphs only associate specific types of substructures with features, such as paths (Borgwardt and Kriegel, 2005; Heinonen et al., 2012), walks (Kashima et al., 2003; Vishwanathan et al., 2006) and tree structures (Cilia and Moschitti, 2007; Mahé and Vert, 2008; Shervashidze et al., 2011; Da San Martino et al., 2012).

We exploit structural kernels for PI, whose task is to evaluate whether a given pair of sentences is in the paraphrase class or not, (see for example (Dolan et al., 2004)). Paraphrases can be seen as a restatement of a text in another form that preserves the original meaning. This task has a primary importance in many other NLP and IR tasks such as Machine Translation, Plagiarism Detection and QA. Several approaches have been proposed, e.g., (Socher et al., 2011) apply a recursive auto encoder with dynamic pooling, and (Madnani et al., 2012) use eight machine translation metrics to achieve the state of the art. To our knowledge no previous model based on kernel methods has been applied before: with such methods, we outperform the state of the art in PI.

A description of RTE can be found in (Giampiccolo et al., 2007): it is defined as a directional relation extraction between two text fragments, called *text* and *hypothesis*. The implication is supposed to be detectable only based on the text content. Its applications are in QA, Information Extraction, Summarization and Machine translation. One of the most performing approaches of RTE 3 was (Iftene and Balahur-Dobrescu, 2007), which largely relies on external resources (i.e., WordNet, Wikipedia, acronyms dictionaries) and a base of knowledge developed ad hoc for the dataset. In (Zanzotto and Moschitti, 2006), we designed an interesting but computationally expensive model using simple syntactic tree kernels. In this paper, we develop models that do not use external resources but, at the same time, are efficient and approach the state of the art in RTE.

3 Structural kernels

Kernel Machines carry out learning and classification by only relying on the inner product between instances. This can be efficiently and implicitly computed by kernel functions by exploiting the following dual formulation of the model (hyperplane): $\sum_{i=1..l} y_i \alpha_i \phi(o_i) \cdot \phi(o) + b = 0$, where y_i are the example labels, α_i the support vector coefficients, o_i and o are two objects, ϕ is a mapping from the objects to feature vectors $\vec{x_i}$ and $\phi(o_i) \cdot \phi(o) = K(o_i, o)$ is the kernel function implicitly defining such mapping. In case of structural kernels, K maps objects in substructures, thus determining their size and shape. Given two structures S_1 and S_2, our general definition of structural kernels is the following:

$$K(S_1, S_2) = \sum_{s_1 \subseteq S_1, s_2 \subseteq S_2, s_i \in \mathcal{S}} k_{iso}(s_1, s_2), \quad (1)$$

where s_i are substructures of S_i, $\mathcal{S}$ is the set of admissible substructures, and k_{iso} determines if the two substructures are isomorphic, i.e., it outputs 1 if s_1 and s_2 are isomorphic and 0 otherwise.

In the following, we also provide a more computational-oriented definition of structural kernels to more easily describe those we use in our work:
Let the set $\mathcal{S} = \{s_1, s_2, \ldots, s_{|\mathcal{S}|}\}$ be the substructure space and $\chi_i(n)$ be an indicator function, equal to 1 if the target s_i is rooted at node n and

equal to 0 otherwise. A structural-kernel function over S_1 and S_2 is

$$K(S_1, S_2) = \sum_{n_1 \in N_{S_1}} \sum_{n_2 \in N_{S_2}} \Delta(n_1, n_2), \quad (2)$$

where N_{S_1} and N_{S_2} are the sets of the S_1's and S_2's nodes, respectively and

$$\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{S}|} \chi_i(n_1)\chi_i(n_2). \quad (3)$$

The latter is equal to the number of common substructures rooted in the n_1 and n_2 nodes. In order to have a similarity score between 0 and 1, a normalization in the kernel space, i.e., $\frac{K(S_1,S_2)}{\sqrt{K(S_1,S_1)\times K(S_2,S_2)}}$ is usually applied. From a practical computation viewpoint, it is convenient to divide structural kernels in two classes of algorithms working either on trees or graphs.

3.1 The Partial Tree Kernel (PTK)

PTK (Moschitti, 2006) generalizes a large class of tree kernels as it computes one of the most general tree substructure spaces. Given two trees S_1 and S_2, PTK considers any connected subset of nodes as possible feature of the substructure space, and counts how many of them are shared by S_1 and S_2. Its computation is carried out by Eq. 2 using the following Δ_{PTK} function:

if the labels of n_1 and n_2 are different $\Delta_{PTK}(n_1, n_2) = 0$; **else** $\Delta_{PTK}(n_1, n_2) =$

$$\mu\Big(\lambda^2 + \sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_{PTK}(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big)$$

where $\mu, \lambda \in [0, 1]$ are two decay factors, $\vec{I}_1$ and $\vec{I}_2$ are two sequences of indices, which index subsequences of children u, $\vec{I} = (i_1, ..., i_{|u|})$, in sequences of children s, $1 \leq i_1 < ... < i_{|u|} \leq |s|$, i.e., such that $u = s_{i_1}..s_{i_{|u|}}$, and $d(\vec{I}) = i_{|u|} - i_1 + 1$ is the distance between the first and last child. The PTK computational complexity is $O(p\rho^2|N_{S_1}||N_{S_2}|)$ (Moschitti, 2006), where p is the largest subsequence of children that we want to consider and ρ is the maximal outdegree observed in the two trees. However the average running time tends to be linear for natural language syntactic trees (Moschitti, 2006).

3.2 Smoothed Partial Tree Kernel ($SPTK$)

Constraining the application of lexical similarity to words embedded in similar structures provides clear advantages over all-vs-all words similarity, which tends to semantically diverge. Indeed, syntax provides the necessary restrictions to compute an effective semantic similarity. $SPTK$ (Croce et al., 2011) generalizes PTK by enabling node similarity during substructure matching. More formally, $SPTK$ is computed by Eq. 2 using the following $\Delta_{SPTK}(n_1, n_2) = \sum_{i,j=1}^{|\mathcal{S}|} \chi_i(n_1)\chi_j(n_2)\Sigma(s_i, s_j)$, where Σ is a similarity between structures[1]. The recursive definition of Δ_{SPTK} is the following:

1. **if** n_1 and n_2 are leaves $\Delta_{SPTK}(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)$;
2. **else** $\Delta_{SPTK}(n_1, n_2) = \mu\sigma(n_1, n_2) \times \Big(\lambda^2 +$

$$\sum_{\vec{I}_1, \vec{I}_2, l(\vec{I}_1)=l(\vec{I}_2)} \lambda^{d(\vec{I}_1)+d(\vec{I}_2)} \prod_{j=1}^{l(\vec{I}_1)} \Delta_\sigma(c_{n_1}(\vec{I}_{1j}), c_{n_2}(\vec{I}_{2j}))\Big),$$

where σ is any similarity between nodes, e.g., between their lexical labels, and the other variables are the same of PTK. The worst case complexity of $SPTK$ is identical to PTK and in practice is not higher than $O(|N_{S_1}||N_{S_2}|)$.

3.3 Neighborhood Subgraph Pairwise Distance Kernel ($NSPDK$)

When general subgraphs are used as features in a kernel computation, eq. 1 and 2 become computationally intractable (Gartner et al., 2003). To solve this problem, we need to restrict the set of considered substructures $\mathcal{S}$. (Costa and De Grave, 2010) defined $NSPDK$ such that the feature space is only constituted by pairs of subgraphs (substructures) that are (i) centered in two nodes n_1 and n_2 such that their distance is not more than D; and (ii) constituted by all nodes (and their edges) at an exact distance h from n_1 or n_2, where the distance between two nodes is defined as the number of edges in the shortest path connecting them. More formally, let G, N_G and E_G be a graph and its set of nodes and edges, respectively, the substructure space $\mathcal{S} = S_G(H, D)$ used by $NSPDK$ in eqs 2 and 3 is:

$$\{(\gamma_h(n), \gamma_h(n')) : 1 \leq h \leq H, n, n' \in N_G, d(n, n') \leq D\},$$

where $\gamma_h(n)$ returns the subgraph obtained by executing h steps of a breadth-first visit of G starting from node n and $d(n, n')$ is the distance between two nodes in the graph. Note that (i) any feature

[1] Note that this generalizes Eq. 3.

of the space is basically a pair of substructures; and (ii) there is currently no efficient (implicit) formulation for computing such kernel. In contrast, when H and D are limited, it is simple to compute the space $S_G(H, D)$ explicitly. In such case, the complexity of the kernel is given by the substructure extraction step, which is $O(|N_G| \times h\rho \log \rho)$.

3.4 Kernel Combinations

Previous sections have shown three different kernels. Among them, $NSPDK$ is actually an explicit kernel, where the features are automatically extracted with a procedure. In NLP, features are often manually defined by domain experts, who know the linguistic phenomena involved in the task. When available, such features are important as they encode some of the background knowledge on the task. Therefore, combining different feature spaces is typically very useful. Fortunately, kernel methods enable an easy integration of different kernels or feature spaces, i.e., the kernel sum produces the joint feature space and it is still a valid kernel. In the next section, we show representations of text, i.e., structures and features, specific to PI and RTE.

4 Representations for RL from text

The kernels described in the previous section can be applied to generic trees and graphs. Automatic feature engineering using structural kernels requires the design of structures for representing data examples that are specific to the learning task we want to tackle. In our case, we focus on RL, which consists in deriving the semantic relation between two entire pieces of text. We focus on two well-understood relations, namely, paraphrasing and textual implications. The tasks are simply defined as: given two texts a_1 and a_2, automatically classify if (i) a_1 is a paraphrase of a_2 and/or (ii) a_1 implies a_2. Although the two tasks are linguistically and conceptually rather different, they can be modeled in a similar way from a shallow representation viewpoint. This is exactly the perspective we would like to keep for showing the advantage of using kernel methods. Therefore, in the following, we define sentence representations that can be suitably used for both tasks and then we rely on structural kernels and the adopted learning algorithm for exploring the substructures relevant to the different tasks.

4.1 Tree Representations

An intuitive understanding of our target tasks suggests that syntactic information is essential to achieve high accuracy. Therefore, we consider the syntactic parse trees of the pair of sentences involved in the evaluation. For example, Fig. 1 shows the syntactic constituency trees of the sentences reported in the introduction (these do not include the green label REL and the dashed edges). Given two pairs of sentences, $p_a = \langle a_1, a_2 \rangle$ and $p_b = \langle b_1, b_2 \rangle$, an initial kernel for learning the tasks, can be the simple tree kernel sum, e.g., $PTK(a_1, b_1) + PTK(a_2, b_2)$ as was defined in (Moschitti, 2008). This kernel works in the space of the union of the sets of all subtrees from the upper and lower trees, e.g.:

a_1: {[PP [TO [to::t]][NP [QP [$ [$::$]][QP [CD [107.6::c]]]]]], [PP [TO][NP [QP [$][QP [CD [107]]]]]], [PP [TO][NP [QP [QP [CD]]]]], [PP [NP [QP [QP]]]], ...}

$\bigcup$

a_2: {[NP [NP [DT [a::d]] [JJ [key::j] NN]][PP]], [NP [NP [DT] [JJ NN]][PP]], [NP [NP [JJ NN]][PP]], [NP [NP [NN]][PP]], [NP [NP [JJ]][PP]], ...}

However, such features cannot capture the relations between the constituents (or semantic lexical units) from the two trees. In contrast, these are essential to learn the relation between the two entire sentences[2].

To overcome this problem, in (Zanzotto and Moschitti, 2006), we proposed the use of *placeholders* for RTE: the main idea was to annotate the matches between the constituents of the two sentences, e.g., *107.6 millions*, on both trees. This way the tree fragments in the generated kernel space contained an index capturing the correspondences between a_1 and a_2. The critical drawback of this approach is that other pairs, e.g., p_b, will have in general different indices, making the representation very sparse. Alternatively, we experimented with models that select the best match between all possible placeholder assignments across the two pairs. Although we obtained a good improvement, such solution required an exponential computational time and the selection of the max

[2] Of course assuming that text meaning is compositional.

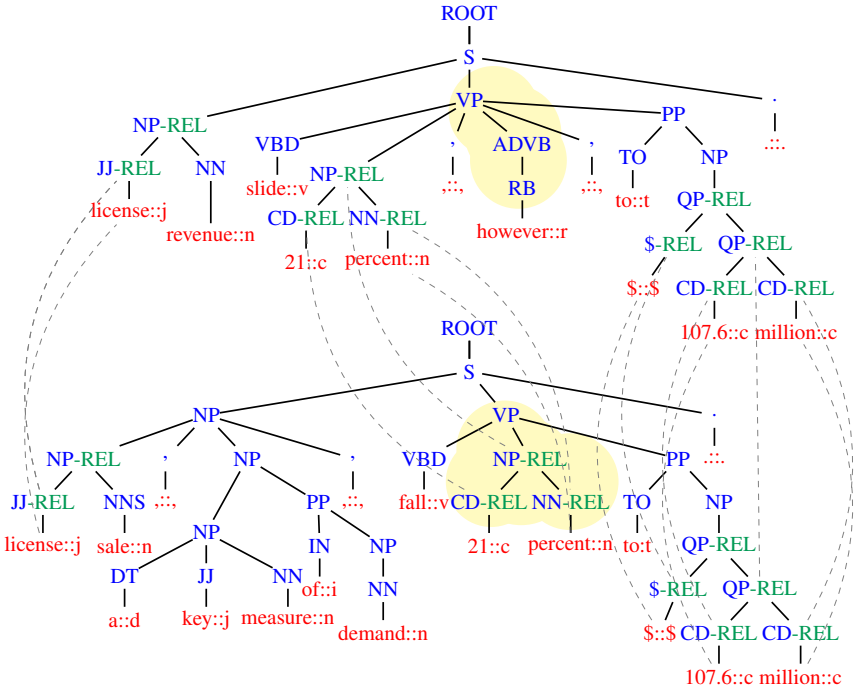

Figure 1: Text representations for PI and RTE: (i) pair of trees, a_1 (upper) and a_2 (lower), (ii) combined in a graph with dashed edges, and (iii) labelled with the tag REL (in green). The nodes highlighted in yellow constitute a feature for the $NSPDK$ kernel ($h = 1$, $D = 3$) centered at the nodes `ADVB` and `NP-REL`.

assignment made our similarity function a non-valid kernel.

Thus, for this paper, we prefer to rely on a more recent solution we proposed for passage reranking in the QA domain (Severyn and Moschitti, 2012; Severyn et al., 2013a; Severyn et al., 2013b), and for Answer Selection (Severyn and Moschitti, 2013). It consists in simply labeling matching nodes with a special tag, e.g., REL, which indicates the correspondences between words. REL is attached to the father and grandfather nodes of the matching words. Fig. 1 shows several green REL tags attached to the usual POS-tag and constituent node labels of the parse trees. For example, the lemma *license* is matched by the two sentences, thus both its father, JJ, and its grandfather, NP, nodes are marked with REL. Thanks to such relational labeling the simple kernel, $PTK(a_1, b_1) + PTK(a_2, b_2)$, can generate relational features from a_1, e.g., `[NP [NP-REL [JJ-REL] NN]][PP]`, `[NP [NP-REL [NN]][PP]]`, `[NP [NP-REL [JJ-REL]][PP]]`,... If such features are matched in b_1, they provide the fuzzy information: *there should be a match similar to* `[NP [NP-REL [JJ-REL]]` *also between* a_2 *and* b_2. This kind of matches establishes a sort of relational pair features.

It should be noted that we proposed more complex REL tagging policies for Passage Reranking, exploiting additional resources such as Linked Open Data or WordNet (Tymoshenko et al., 2014). Another interesting application of this RL framework is the Machine Translation Evaluation (Guzmán et al., 2014). Finally, we used a similar model for translating questions to SQL queries in (Giordani and Moschitti, 2012).

4.2 Graph Representations

The relational tree representation can capture relational features but the use of the same REL tag for any match between the two trees prevents to deterministically establish the correspondences between nodes. For exactly representing such matches (without incurring in non-valid kernels or sparsity problems), a graph representation is needed. If we connect matching nodes (or also nodes labelled as REL) in Fig. 1 (see dashed lines), we obtain a relational graph. Substructures of such graph clearly indicate how constituents, e.g., NPs, VPs, PPs, from one sentence map into the other sentence. If such mappings observed in a pair of paraphrase sentences are matched in another sentence pair, there may be evidence that also the second pair contains paraphrase sen-

tences.

Unfortunately, the kernel computing the space of all substructures of a graph (even if only considering connected nodes) is an intractable problem as mentioned in Sec. 3.3. Thus, we opt for the use of $NSPDK$, which generates specific pairs of structures. Intuitively, the latter can capture relational features between constituents of the two trees. Figure 1 shows an example of features generated by the $NSPDK$ with parameters $H = 1$, $D = 3$ (the substructures are highlighted in yellow), i.e., `[ADVB [VP] [RB]]`, `[NP-REL [VP] [CD-REL] [NN-REL]]`.

4.3 Basic Features

In addition to structural representations, we also use typical features for capturing the degrees of similarity between two sentences. In contrast, with the previous kernels these similarities are computed intra-pair, e.g., between a_1 and a_2. Note that any similarity measure generates only one feature. Their description follows:

– *Syntactic similarities*, which apply the cosine function to vectors of n-grams (with $n = 1, 2, 3, 4$) of word lemmas and part-of-speech tags.
– *Kernel similarities*, which use PTK or $SPTK$ applied to the sentences within the pair.

We also used similarity features from the DKPro of the UKP Lab (Bär et al., 2012), tested in the Semantic Textual Similarity (STS) task:
– *Longest common substring measure* and *Longest common subsequence measure*, which determine the length of the longest substring shared by two text segments.
– *Running-Karp-Rabin Greedy String Tiling* provides a similarity between two sentences by counting the number of shuffles in their subparts.
– *Resnik similarity* based on the WordNet hierarchy.
– *Explicit Semantic Analysis* (ESA) similarity (Gabrilovich and Markovitch, 2007) represents documents as weighted vectors of concepts learned from Wikipedia, WordNet and Wiktionary.
– *Lexical Substitution* (Szarvas et al., 2013): a supervised word sense disambiguation system is used to substitute a wide selection of high-frequency English nouns with generalizations, then Resnik and ESA features are computed on the transformed text.

4.4 Combined representations

As mentioned in Sec. 3.4, we can combine kernels for engineering new features. Let K be PTK or $SPTK$, given two pairs of sentences $p_a = \langle a_1, a_2 \rangle$ and $p_b = \langle b_1, b_2 \rangle$, we build the following kernel combinations for the RTE task:

(i) $K^+(p_a, p_b) = K(a_1, b_1) + K(a_2, b_2)$, which simply sums the similarities between the first two sentences and the second two sentences whose implication has to be derived.

(ii) An alternative kernel combines the two similarity scores above with the product: $K^\times(p_a, p_b) = K(a_1, b_1) \cdot K(a_2, b_2)$.

(iii) The symmetry of the PI task requires different kernels. The most intuitive applies K between all member combinations and sum all contributions: $all^+_K(p_a, p_b)$=$K(a_1, b_1) + K(a_2, b_2) + K(a_1, b_2) + K(a_2, b_1)$.

(iv) It is also possible to combine pairs of corresponding kernels with the product: $all^\times_K(p_a, p_b) = K(a_1, b_1)K(a_2, b_2) + K(a_1, b_2)K(a_2, b_1)$.

(v) An alternative kernel selects only the best between the two products above: $M_K(p_a, p_b) = \max(K(a_1, b_1)K(a_2, b_2), K(a_1, b_2)K(a_2, b_1))$. This is motivated by the observation that before measuring the similarity between two pairs, we need to establish which a_i is more similar to b_j. However, the max operator causes M_K not to be a valid kernel function, thus we substitute it with a *softmax* function, which is a valid kernel, i.e., $SM_K(p_a, p_b)$ = *softmax*$(K(a_1, b_1)K(a_2, b_2), K(a_1, b_2)K(a_2, b_1))$, where *softmax*$(x_1, x_2) = \frac{1}{c} log(e^{cx_1} + e^{cx_2})$ (c=100 was accurate enough).

The linear kernel (LK) over the basic features (described previously) and/or $NSPDK$ can be of course added to all the above kernels.

5 Experiments

5.1 Setup

MSR Paraphrasing: we used the Microsoft Research Paraphrase Corpus (Dolan et al., 2004) consisting of 4,076 sentence pairs in the training set and 1,725 sentence pairs in test set, with a distribution of about 66% between positive and negative

		Vs Test				5 Fold Cross Validation			
	Kernel	Acc (%)	P	R	F1	Acc (%)	P	R	F1
without REL tagging	LK	75.88	0.784	0.881	0.829	75.54 ± 0.45	0.786 ± 0.009	0.876 ± 0.019	0.828 ± 0.004
	GK	72.81	0.720	0.967	0.825	72.49 ± 1.22	0.723 ± 0.014	0.957 ± 0.011	0.824 ± 0.008
	SM_{PTK}	72.06	0.722	0.943	0.818	72.04 ± 1.08	0.725 ± 0.009	0.940 ± 0.017	0.819 ± 0.009
	$SM_{SPTK_{LSA}}$	72.12	0.722	0.943	0.818	72.56 ± 1.10	0.731 ± 0.010	0.937 ± 0.017	0.821 ± 0.009
	$SM_{SPTK_{W2V}}$	71.88	0.719	0.946	0.817	72.23 ± 1.07	0.727 ± 0.009	0.938 ± 0.017	0.820 ± 0.009
	$all^{\times}_{PTK}$	71.42	0.718	0.939	0.814	71.57 ± 0.86	0.724 ± 0.007	0.933 ± 0.015	0.815 ± 0.008
	$all^{\times}_{SPTK_{LSA}}$	72.29	0.725	0.941	0.819	72.06 ± 0.62	0.730 ± 0.007	0.928 ± 0.014	0.817 ± 0.006
	$all^{\times}_{SPTK_{W2V}}$	71.59	0.717	0.947	0.816	71.61 ± 0.76	0.725 ± 0.008	0.931 ± 0.013	0.815 ± 0.007
	all^{+}_{PTK}	70.78	0.716	0.930	0.809	70.76 ± 0.91	0.720 ± 0.008	0.924 ± 0.017	0.809 ± 0.009
	$all^{+}_{SPTK_{LSA}}$	71.48	0.720	0.934	0.813	71.42 ± 0.91	0.727 ± 0.008	0.920 ± 0.020	0.812 ± 0.009
	$all^{+}_{SPTK_{W2V}}$	70.72	0.714	0.935	0.809	71.19 ± 1.22	0.723 ± 0.010	0.927 ± 0.018	0.812 ± 0.011
	M_{PTK}	72.17	0.725	0.935	0.817	72.31 ± 0.67	0.731 ± 0.007	0.930 ± 0.015	0.819 ± 0.007
	$M_{SPTK_{LSA}}$	72.00	0.725	0.934	0.816	72.32 ± 0.44	0.732 ± 0.006	0.927 ± 0.014	0.818 ± 0.005
	$M_{SPTK_{W2V}}$	71.71	0.722	0.933	0.814	71.99 ± 0.96	0.730 ± 0.008	0.926 ± 0.016	0.816 ± 0.008
with REL tagging	GK	75.07	0.752	0.933	0.833	74.69 ± 2.52	0.749 ± 0.029	0.940 ± 0.008	0.834 ± 0.018
	SM_{PTK}	76.17	0.765	0.927	0.838	75.42 ± 0.86	0.771 ± 0.007	0.903 ± 0.012	0.832 ± 0.008
	$SM_{SPTK_{LSA}}$	76.52	0.767	0.929	0.840	75.62 ± 0.90	0.772 ± 0.007	0.905 ± 0.013	0.833 ± 0.007
	$SM_{SPTK_{W2V}}$	76.35	0.766	0.929	0.839	75.64 ± 0.77	0.771 ± 0.004	0.907 ± 0.012	0.833 ± 0.007
	$all^{\times}_{PTK}$	75.36	0.767	0.905	0.830	74.76 ± 0.71	0.769 ± 0.006	0.892 ± 0.016	0.826 ± 0.008
	$all^{\times}_{SPTK_{LSA}}$	75.65	0.770	0.903	0.831	74.83 ± 0.92	0.771 ± 0.009	0.891 ± 0.011	0.826 ± 0.008
	$all^{\times}_{SPTK_{W2V}}$	75.88	0.772	0.905	0.833	75.26 ± 0.81	0.771 ± 0.008	0.898 ± 0.011	0.830 ± 0.008
	all^{+}_{PTK}	74.49	0.762	0.895	0.824	73.99 ± 1.04	0.767 ± 0.010	0.880 ± 0.013	0.820 ± 0.009
	$all^{+}_{SPTK_{LSA}}$	75.07	0.767	0.899	0.827	73.87 ± 0.85	0.766 ± 0.009	0.880 ± 0.010	0.819 ± 0.007
	$all^{+}_{SPTK_{W2V}}$	75.42	0.772	0.894	0.829	74.16 ± 0.75	0.768 ± 0.008	0.882 ± 0.012	0.821 ± 0.007
	GK+$SM_{SPTK_{W2V}}$	76.70	0.782	0.901	0.837	76.12 ± 0.96	0.787 ± 0.008	0.885 ± 0.015	0.833 ± 0.009
	LK+GK	78.67	0.802	0.902	0.849	77.85 ± 1.00	0.804 ± 0.008	0.886 ± 0.015	0.843 ± 0.009
	LK+$SM_{SPTK_{W2V}}$	77.74	0.794	0.899	0.843	77.52 ± 1.41	0.802 ± 0.011	0.885 ± 0.016	0.841 ± 0.011
	LK+GK+$SM_{SPTK_{W2V}}$	**79.13**	0.807	0.901	**0.852**	78.11 ± 0.94	0.811 ± 0.005	0.879 ± 0.016	0.844 ± 0.009
	(Socher et al., 2011)	76.8	–	–	0.836	–	–	–	–
	(Madnani et al., 2012)	77.4	–	–	0.841	–	–	–	–

Table 1: Results on Paraphrasing Identification

examples. These pairs were extracted from topically similar Web news articles, applying some heuristics that select potential paraphrases to be annotated by human experts.

RTE-3. We adopted the RTE-3 dataset (Giampiccolo et al., 2007), which is composed by 800 text-hypothesis pairs in both the training and test sets, collected by human annotators. The distribution of the examples among the positive and negative classes is balanced.

5.1.1 Models and Parameterization

We train our classifiers with the C-SVM learning algorithm (Chang and Lin, 2011) within KeLP[3], a Kernel-based Machine Learning platform that implements tree kernels. In both tasks, we applied the kernels described in Sec. 4, where the trees are generated with the Stanford parser[4].

$SPTK$ uses a node similarity function $\sigma(n_1, n_2)$ implemented as follows: if n_1 and n_2 are two identical syntactic nodes $\sigma = 1$. If n_1 and n_2 are two lexical nodes with the same POS tag, their similarity is evaluated computing the cosine similarity of their corresponding vectors in a wordspace. In all the other cases $\sigma = 0$. We generated two different wordspaces. The first is a co-occurrence LSA embedding as described in (Croce and Previtali, 2010). The second space is derived by applying a skip-gram model (Mikolov et al., 2013) with the word2vec tool[5]. $SPTK$ using the LSA will be referred to as $SPTK_{LSA}$, while when adopting word2vec it will be indicated with $SPTK_{W2V}$. We used default parameters both for PTK and $SPTK$ whereas we selected h and D parameters of $NSPDK$ that obtained the best average accuracy using a 5-fold cross validation on the training set.

5.1.2 Performance measures

The two considered tasks are binary classification problems thus we used Accuracy, Precision, Recall and F1. The adopted corpora have a predefined split between training and test sets thus we tested our models according to such settings for exactly comparing with previous work. Additionally, to better assess our results, we performed a 5-fold cross validation on the complete datasets. In case of PI, the same sentence can appear in multiple pairs thus we distributed the pairs such that the same sentence can only appear in one fold at a time.

[3] https://github.com/SAG-KeLP

[4] http://nlp.stanford.edu/software/corenlp.shtml

[5] https://code.google.com/p/word2vec/

		Vs Test				5 Fold Cross Validation			
	Kernel	Acc (%)	P	R	F1	Acc (%)	P	R	F1
without REL tagging	LK	62	0.608	0.729	0.663	62.94 ± 5.68	0.635 ± 0.057	0.679 ± 0.083	0.652 ± 0.049
	GK	55.375	0.555	0.651	0.599	55.63 ± 1.81	0.564 ± 0.022	0.612 ± 0.087	0.584 ± 0.032
	PTK^{+}	56.13	0.560	0.676	0.612	54.13 ± 3.26	0.547 ± 0.024	0.637 ± 0.051	0.587 ± 0.027
	$SPTK^{+}_{LSA}$	56.88	0.566	0.683	0.619	53.63 ± 2.50	0.543 ± 0.024	0.622 ± 0.060	0.578 ± 0.027
	$SPTK^{+}_{W2V}$	56.63	0.563	0.683	0.617	54.06 ± 2.34	0.546 ± 0.022	0.634 ± 0.060	0.585 ± 0.026
	$PTK^{\times}$	55.88	0.558	0.671	0.609	52.81 ± 1.99	0.535 ± 0.025	0.623 ± 0.055	0.574 ± 0.028
	$SPTK^{\times}_{LSA}$	56.25	0.561	0.671	0.611	53.56 ± 2.09	0.543 ± 0.022	0.616 ± 0.065	0.576 ± 0.026
	$SPTK^{\times}_{W2V}$	55.25	0.554	0.646	0.597	52.50 ± 1.77	0.533 ± 0.027	0.619 ± 0.071	0.571 ± 0.034
with REL tagging	GK	61.63	0.603	0.734	0.662	59.81 ± 3.84	0.599 ± 0.037	0.678 ± 0.071	0.634 ± 0.026
	PTK^{+}	66.00	0.627	0.829	0.714	67.75 ± 7.17	0.655 ± 0.067	0.817 ± 0.038	0.725 ± 0.046
	$SPTK^{+}_{LSA}$	65.38	0.622	0.824	0.709	67.81 ± 7.30	0.656 ± 0.069	0.816 ± 0.036	0.725 ± 0.047
	$SPTK^{+}_{W2V}$	66.38	0.629	0.837	0.718	68.00 ± 7.15	0.658 ± 0.068	0.816 ± 0.039	0.726 ± 0.046
	$PTK^{\times}$	66.13	0.629	0.827	0.714	67.75 ± 7.37	0.658 ± 0.071	0.804 ± 0.038	0.722 ± 0.049
	$SPTK^{\times}_{LSA}$	66.00	0.629	0.822	0.712	68.00 ± 7.62	0.661 ± 0.074	0.808 ± 0.039	0.725 ± 0.049
	$SPTK^{\times}_{W2V}$	**67.00**	0.636	0.834	0.722	67.69 ± 6.95	0.658 ± 0.069	0.804 ± 0.040	0.722 ± 0.043
	GK+$SPTK^{\times}_{W2V}$	66.38	0.634	0.815	0.713	66.00 ± 6.79	0.648 ± 0.069	0.769 ± 0.034	0.701 ± 0.044
	LK+GK	62.25	0.609	0.737	0.667	62.06 ± 5.49	0.620 ± 0.051	0.702 ± 0.053	0.656 ± 0.036
	LK+$SPTK^{\times}_{W2V}$	66.13	0.628	0.829	0.715	68.25 ± 7.54	0.663 ± 0.076	0.816 ± 0.032	0.728 ± 0.047
	LK+GK+$SPTK^{\times}_{W2V}$	66.00	0.633	0.800	0.707	66.31 ± 7.35	0.652 ± 0.075	0.770 ± 0.053	0.703 ± 0.052
	(Zanzotto et al., 2009)	66.75	0.667	–	–	–	–	–	–
	(Iftene and Balahur-Dobrescu, 2007)	69.13	–	–	–	–	–	–	–

Table 2: Results on Textual Entailment Recognition

5.2 Results on PI

The results are reported in Table 1. The first column shows the use of the relational tag REL in the structures (discussed in Sec. 4.1). The second column indicates the kernel models described in sections 3 and 4 as well as the combination of the best models. Columns 3-6 report Accuracy, Precision, Recall and F1 derived on the fixed test set, whereas the remaining columns regard the results obtained with cross validation. We note that:

First, when REL information is not used in the structures, the linear kernel (LK) on basic features outperforms all the structural kernels, which all perform similarly. The best structural kernel is the graph kernel, $NSPDK$ (GK in short). This is not surprising as without REL, GK is the only kernel that can express relational features.

Second, $SPTK$ is only slightly better than PTK. The reason is mainly due to the approach used for building the dataset: potential paraphrases are retrieved applying some heuristics mostly based on the lexical overlap between sentences. Thus, in most cases, the lexical similarity used in $SPTK$ is not needed as hard matches occur between the words of the sentences.

Third, when REL is used on the structures, all kernels reach or outperform the F1 (official measure of the challenge) of LK. The relational structures seem to drastically reduce the inconsistent matching between positive and negative examples, reflecting in remarkable increasing in Precision. In particular, $SM_{SPTK_{LSA}}$ achieves the state of the art[6], i.e., 84.1 (Madnani et al., 2012).

Next, combining our best models produces a significant improvement of the state of the art, e.g., $LK + GK + SM_{SPTK_{W2V}}$ outperforms the result in (Madnani et al., 2012) by 1.7% in accuracy and 1.1 points in F1.

Finally, the cross-validation experiments confirm the system behavior observed on the fixed test set. The Std. Dev. (specified after the $\pm$ sign) shows that in most cases the system differences are significant.

5.3 Results on RTE

We used the same experimental settings performed for PI to carry out the experiments on RTE. The results are shown in Table 2 structured in the same way as the previous table. We note that:

(i) Findings similar to PI are obtained.

(ii) Again the relational structures (using REL) provide a remarkable improvement in Accuracy (RTE challenge measure), allowing tree kernels to compete with the state of the art. This is an impressive result considering that our models do not use any external resource, e.g., as in (Iftene and Balahur-Dobrescu, 2007).

(iii) This time, $SPTK^{\times}_{W2V}$ improves on PTK by 1 absolute percent point.

[6]The performance of the several best systems improved by our models are nicely summarized at `http://aclweb.org/aclwiki/index.php?title=Paraphrase_Identification_(State_of_the_art)`

(iv) The kernel combinations are not more effective than $SPTK$ alone.

Finally, the cross-fold validation experiments confirm the fixed-test set results.

6 Discussion and Conclusions

In this paper, we have engineered and studied several models for relation learning. We utilized state-of-the-art kernels for structures and created new ones by combining kernels together. Additionally, we provide a novel definition of effective and computationally feasible structural kernels. Most importantly, we have designed novel computational structures for trees and graphs, which are for the first time tested in NLP tasks. Our kernels are computationally efficient thus solving one of the most important problems of previous work.

We empirically tested our kernels on two of the most representative tasks of RL from text, namely, PI and RTE. The extensive experimentation using many kernel models also combined with traditional feature vector approaches sheds some light on how engineering effective graph and tree kernels for learning from pairs of entire text fragments. In particular, our best models significantly outperform the state of the art in PI and the best kernel model for RTE 3, with Accuracy close to the one of the best system of RTE 3.

It should be stressed that the design of previous state-of-the-art models involved the use of several resources, annotation and heavy manually engineering of specific rules and features: this makes the portability of such systems on other domains and tasks extremely difficult. Moreover the unavailability of the used resources and the opacity of the used rules have also made such systems very difficult to replicate.

On the contrary, the models we propose enable researchers to:

(i) build their system without the use of specific resources. We use a standard syntactic parser, and for some models we use well-known and available corpora for automatically learning similarities with word embedding algorithms; and

(ii) reuse our work for different (similar) tasks (see paraphrasing) and data.

The simplicity and portability of our system is a significant contribution to a very complex research area such as RL from two entire pieces of text.

Our study has indeed shown that our kernel models, which are very simple to be implemented, reach the state of the art and can be used with large datasets.

Furthermore, it should be noted that our models outperform the best tree kernel approach of the RTE challenges (Zanzotto and Moschitti, 2006) and also its extension that we proposed in (Zanzotto et al., 2009). These systems are also adaptable and easy to replicate, but they are subject to an exponential computational complexity and can thus only be used on very small datasets (e.g., they cannot be applied to the MSR Paraphrase corpus). In contrast, the model we proposed in this paper can be used on large datasets, because its kernel complexity is about linear (on average).

We believe that disseminating these findings to the research community is very important, as it will foster research on RL, e.g., on RTE, using structural kernel methods. Such research has had a sudden stop as the RTE data in the latest challenges increased from 800 instances to several thousands and no tree kernel model has been enough accurate to replace our computational expensive models (Zanzotto et al., 2009).

In the future, it would be interesting defining graph kernels that can combine more than two substructures. Another possible extension regards the use of node similarity in graph kernels. Additionally, we would like to test our models on other RTE challenges and on several QA datasets, which for space constraints we could not do in this work.

Acknowledgments

This research is part of the Interactive sYstems for Answer Search (IYAS) project, conducted by the Arabic Language Technologies (ALT) group at Qatar Computing Research Institute (QCRI) within the Hamad Bin Khalifa University and Qatar Foundation.

References

Daniel Bär, Chris Biemann, Iryna Gurevych, and Torsten Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. In *Proc. of SemEval '12*. ACL.

Karsten M Borgwardt and Hans-Peter Kriegel. 2005. Shortest-Path Kernels on Graphs. *ICDM*, 0:74–81.

Xavier Carreras and Lluís Màrquez. 2005. Introduction to the conll-2005 shared task: Semantic role labeling. In *Proc. of CONLL '05*, USA.

Chih-Chung Chang and Chih-Jen Lin. 2011. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27.

Elisa Cilia and Alessandro Moschitti. 2007. Advanced tree-based kernels for protein classification. In *AI*IA 2007: Artificial Intelligence and Human-Oriented Computing, 10th Congress of the Italian Association for Artificial Intelligence, Rome, Italy, September 10-13, 2007, Proceedings*, pages 218–229.

Fabrizio Costa and Kurt De Grave. 2010. Fast neighborhood subgraph pairwise distance kernel. In *ICML*, number v.

Danilo Croce and Daniele Previtali. 2010. Manifold learning for the semi-supervised induction of framenet predicates: An empirical investigation.

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured lexical similarity via convolution kernels on dependency trees. In *Proceedings EMNLP*.

Giovanni Da San Martino, Nicolò Navarin, and Alessandro Sperduti. 2012. A tree-based kernel for graphs. In *Proceedings of the Twelfth SIAM International Conference on Data Mining, Anaheim, California, USA, April 26-28, 2012.*, pages 975–986. SIAM / Omnipress.

Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *Proc. of COLING '04*, Stroudsburg, PA, USA.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya A Kalyanpur, Adam Lally, J William Murdock, Eric Nyberg, John Prager, et al. 2010. Building watson: An overview of the deepqa project. *AI magazine*, 31(3):59–79.

Evgeniy Gabrilovich and Shaul Markovitch. 2007. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In *Proc. of IJCAI-07*, pages 1606–1611.

Thomas Gartner, P Flach, and S Wrobel. 2003. On Graph Kernels : Hardness Results and Efficient Alternatives. *LNCS*, pages 129–143.

Danilo Giampiccolo, Bernardo Magnini, Ido Dagan, and Bill Dolan. 2007. The third pascal recognizing textual entailment challenge. In *Proc. of the ACL-PASCAL RTE '07 Workshop*, pages 1–9. ACL.

Daniel Gildea and Daniel Jurafsky. 2002. Automatic labeling of semantic roles. *Computational Linguistics*, 28:245–288.

Alessandra Giordani and Alessandro Moschitti. 2012. Translating questions to sql queries with generative parsers discriminatively reranked. In *COLING (Posters)*, pages 401–410.

Francisco Guzmán, Shafiq Joty, Lluís Màrquez, Alessandro Moschitti, Preslav Nakov, and Massimo Nicosia. 2014. Learning to differentiate better from worse translations. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 214–220, Doha, Qatar, October. Association for Computational Linguistics.

M Heinonen, N Välimäki, V Mäkinen, and J Rousu. 2012. Efficient Path Kernels for Reaction Function Prediction. *Bioinformatics Models, Methods and Algorithms*.

Adrian Iftene and Alexandra Balahur-Dobrescu. 2007. Hypothesis transformation and semantic variability rules used in recognizing textual entailment. In *RTE Workshop*, Prague.

Hisashi Kashima, Koji Tsuda, and Akihiro Inokuchi. 2003. Marginalized kernels between labeled graphs. In *ICML*, pages 321–328. AAAI Press.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Re-examining machine translation metrics for paraphrase identification. In *Proceedings of NAACL HLT '12*. ACL.

Pierre Mahé and Jean-Philippe Vert. 2008. Graph kernels based on tree patterns for molecules. *Machine Learning*, 75(1):3–35, October.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

Mike Mintz, Steven Bills, Rion Snow, and Daniel Jurafsky. 2009. Distant supervision for relation extraction without labeled data. In *ACL-AFNLP*.

Alessandro Moschitti. 2006. Efficient convolution kernels for dependency and constituent syntactic trees. In *Proc. of ECML'06*, pages 318–329.

Alessandro Moschitti. 2008. Kernel methods, syntax and semantics for relational text categorization. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 253–262, New York, NY, USA. ACM.

Vivi Nastase, Preslav Nakov, Diarmuid Saghdha, and Stan Szpakowicz. 2013. *Semantic Relations Between Nominals*. Morgan & Claypool Publishers.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. In *SIGIR*.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*, pages 458–467.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013a. Building structures from classifiers for passage reranking. In *Proceedings of*

the 22Nd ACM International Conference on Conference on Information & Knowledge Management, CIKM '13, pages 969–978, New York, NY, USA. ACM.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013b. Learning adaptable patterns for passage reranking. *Proceedings of the Seventeenth Conference on Computational Natural Language Learning*, pages 75–83.

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

Nino Shervashidze, Pascal Schweitzer, Erik Jan van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. 2011. Weisfeiler-Lehman Graph Kernels. *JMLR*, 12:2539–2561.

Richard Socher, Eric H. Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y. Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS 24*.

György Szarvas, Chris Biemann, and Iryna Gurevych. 2013. Supervised all-words lexical substitution using delexicalized features. In *NAACL*.

Kateryna Tymoshenko, Alessandro Moschitti, and Aliaksei Severyn. 2014. Encoding semantic resources in syntactic structures for passage reranking. *EACL 2014*, page 664.

S. V. N. Vishwanathan, Karsten M Borgwardt, and Nicol N Schraudolph. 2006. Fast Computation of Graph Kernels. In *NIPS*.

E. Voorhees and D. Tice, 1999. *The TREC-8 Question Answering Track Evaluation*.

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic learning of textual entailments with cross-pair similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 401–408, Sydney, Australia, July. Association for Computational Linguistics.

Fabio massimo Zanzotto, Marco Pennacchiotti, and Alessandro Moschitti. 2009. A machine learning approach to textual entailment recognition. *Nat. Lang. Eng.*, 15(4):551–582, October.