Learning Contextual Embeddings for Structural Semantic Similarity using Categorical Information

Massimo Nicosia[◇] and **Alessandro Moschitti**
[◇]DISI, University of Trento 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 34110, Doha, Qatar
{m.nicosia,amoschitti}@gmail.com

Abstract

Tree kernels (TKs) and neural networks are two effective approaches for automatic feature engineering. In this paper, we combine them by modeling context word similarity in semantic TKs. This way, the latter can operate subtree matching by applying neural-based similarity on tree lexical nodes. We study how to learn representations for the words in context such that TKs can exploit more focused information. We found that neural embeddings produced by current methods do not provide a suitable contextual similarity. Thus, we define a new approach based on a Siamese Network, which produces word representations while learning a binary text similarity. We set the latter considering examples in the same category as similar. The experiments on question and sentiment classification show that our semantic TK highly improves previous results.

1 Introduction

Structural Kernels (Moschitti, 2006) can automatically represent syntactic and semantic structures in terms of substructures, showing high accuracy in several tasks, e.g., relation extraction (Nguyen et al., 2009; Nguyen and Moschitti, 2011; Plank and Moschitti, 2013; Nguyen et al., 2015) and sentiment analysis (Nguyen and Shirai, 2015).

At the same time, deep learning has demonstrated its effectiveness on a plethora of NLP tasks such as Question Answering (QA) (Severyn and Moschitti, 2015a; Rao et al., 2016), and parsing (Andor et al., 2016), to name a few. Deep learning models (DLMs) usually do not include traditional features; they extract relevant signals from distributed representations of words, by applying a sequence of linear and non linear functions to the input. Word representations are learned from large corpora, or directly from the training data of the task at hand.

Clearly, joining the two approaches above would have the advantage of easily integrating structures with kernels, and lexical representations with embeddings into learning algorithms. In this respect, the Smoothed Partial Tree Kernel (SPTK) is a noticeable approach for using lexical similarity in tree structures (Croce et al., 2011). SPTK can match different tree fragments, provided that they only differ in lexical nodes. Although the results were excellent, the used similarity did not consider the fact that words in context assume different meanings or weights for the final task, i.e., it does not consider the context. In contrast, SPTK would benefit to use specific word similarity when matching subtrees corresponding to different constituency. For example, the two questions:

– *What famous model was married to Billy Joel?*

– *What famous model of the Universe was proposed?*

are similar in terms of structures and words but clearly have different meaning and also different categories: the first asks for a human (the answer is Christie Brinkley) whereas the latter asks for an entity (an answer could be *the Expanding Universe*). To determine that such questions are not similar, SPTK would need different embeddings for the word *model* in the two contexts, i.e., those related to *person* and *science*, respectively.

In this paper, we use distributed representations generated by neural approaches for computing the lexical similarity in TKs. We carry out an extensive comparison between different methods, i.e., word2vec, using CBOW and SkipGram, and

Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017), pages 260–270,
Vancouver, Canada, August 3 - August 4, 2017.

Glove, in terms of their impact on convolution semantic TKs for question classification (QC). We experimented with composing word vectors and alternative embedding methods for bigger unit of text to obtain context specific vectors.

Unfortunately, the study above showed that standard ways to model context are not effective. Thus, we propose a novel application of Siamese Networks to learn word vectors in context, i.e., a representation of a word conditioned on the other words in the sentence. Since a comprehensive and large enough corpus of disambiguated senses is not available, we approximate them with categorical information: we derive a classification task that consists in deciding if two words extracted from two sentences belong to the same sentence category. We use the obtained contextual word representations in TKs. Our new approach tested on two tasks, question and sentiment classification, shows that modeling the context further improves the semantic kernel accuracy compared to only using standard word embeddings.

2 Related Work

Distributed word representations are an effective and compact way to represent text and are widely used in neural network models for NLP. The research community has also studied them in the context of many other machine learning models, where they are typically used as features.

SPTK is an interesting kernel algorithm that can compute word to word similarity with embeddings (Croce et al., 2011; Filice et al., 2015, 2016). In our work, we go beyond simple word similarity and improve the modeling power of SPTK using contextual information in word representations. Our approach mixes the syntactic and semantic features automatically extracted by the TK, with representations learned with deep learning models (DLMs).

Early attempts to incorporate syntactic information in DLMs use grammatical relations to guide the composition of word embeddings, and recursively compose the resulting substructural embeddings with parametrized functions. In Socher et al. (2012) and Socher et al. (2013), a parse tree is used to guide the composition of word embeddings, focusing on a single parametrized function for composing all words according to different grammatical relations. In Tai et al. (2015), several LSTM architectures that follow an order determined by syntax are presented. Considering embeddings only, Levy and Goldberg (2014) proposed to learn word representations that incorporate syntax from dependency-based contexts. In contrast, we inject syntactic information by means of TKs, which establish a hard match between tree fragments, while the soft match is enabled by the similarities of distributed representations.

DLMs have been applied to the QC task. Convolutional neural neworks are explored in Kalchbrenner et al. (2014) and Kim (2014). In Ma et al. (2015), convolutions are guided by dependencies linking question words, but it is not clear how the word vectors are initialized. In our case, we only use pre-trained word vectors and the output of a parser, avoiding intensive manual feature engineering, as in Silva et al. (2010). The accuracy of these models are reported in Tab. 1 and can be compared to our QC results (Table 4) on the commonly used test set. In addition, we report our results in a cross-validation setting to better assess the generalization capabilities of the models.

To encode words in context, we employ a Siamese Network, a DLM that has been widely used to model sentence similarity. In a Siamese setting, the same network is used to encode two sentences, and during learning, the distance between the representations of similar sentences is minimized. In Mueller and Thyagarajan (2016), an LSTM is used to encode similar sentences, and their Manhattan distance is minimized. In Neculoiu et al. (2016), a character level bidirectional LSTM is used to determine the similarity between job titles. In Tan et al. (2016), the problem of question/answer matching is treated as a similarity task, and convolutions and pooling on top of LSTM states are used to extract the sentence representations. The paper reports also experiments that include neural attention. Those mechanisms are excluded in our work, since we do not want to break the symmetry of the encoding model.

In Siamese Networks, the similarity is typically computed between pair of sentences. In our work, we compute the similarity of word representations extracted from the states of a recurrent network. Such representations still depend on the entire sentence, and thus encode contextual information.

3 Tree Kernels-based Lexical Similarity

TKs are powerful methods for computing the similarity between tree structures. They can effec-

Model	Features	Accuracy
SVM	Unigram, syntactic information, parser output, WordNet features, hand-coded features	95.0
DCNN	Unsupervised vectors	93.0
CNN_{ns}	CBOW fine-tuned vectors	93.6
DepCNN	Depencency guided filters	95.6
SPTK	SPTK and LSA word vectors	94.8

Table 1: QC accuracy (%) and description of SVM (Silva et al., 2010), DCNN (Kalchbrenner et al., 2014), CNN_{ns} (Kim, 2014), DepCNN, (Ma et al., 2015) and SPTK (Croce et al., 2011) models.

tively encode lexical, syntactic and semantic information in learning algorithms. For this purpose, they count the number of substructures shared by two trees. In most TKs, two tree fragments match if they are identical. In contrast, Croce et al. (2011) proposed the Smoothed Partial Tree Kernel (SPTK), which can also match fragments differing in node labels. For example, consider two constituency tree fragments which differ only for one lexical node. SPTK can establish a soft match between the two fragments by associating the lexicals with vectors and by computing the cosine similarity between the latter. In previous work for QC, vectors were obtained by applying Latent Semantic Analysis (LSA) to a large corpus of textual documents. We use neural word embeddings as in Filice et al. (2015) to encode words. Differently from them, we explore specific embeddings by also deriving a vector representation for the context around each word. Finally, we define a new approach based on the category of the sentence of the target word.

3.1 Smoothed Partial Tree Kernel

SPTK can be defined as follows: let the set $\mathcal{F} = \{f_1, f_2, \ldots, f_{|\mathcal{F}|}\}$ be a tree fragment space and $\chi_i(n)$ be an indicator function, equal to 1 if the target f_i is rooted at node n, and equal to 0 otherwise. A TK function over T_1 and T_2 is:

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

where N_{T_1} and N_{T_2} are the sets of nodes of T_1 and T_2, and $\Delta(n_1, n_2) = \sum_{i=1}^{|\mathcal{F}|} \chi_i(n_1)\chi_i(n_2)$. The latter is equal to the number of common fragments rooted in the n_1 and n_2 nodes. The Δ function for

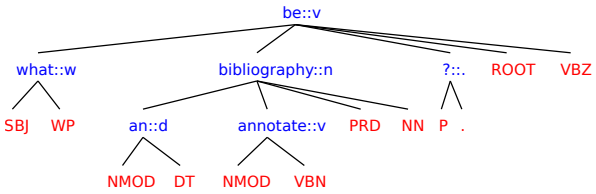

Figure 1: The Lexical Centered Tree (LCT) of the lemmatized sentence: "*What is an annotated bibliography?*".

SPTK[1] defines a rich kernel space as follows:

1. If n_1 and n_2 are leaves then $\Delta_\sigma(n_1, n_2) = \mu\lambda\sigma(n_1, n_2)$; else

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

where σ is any similarity between nodes, e.g., between their lexical labels, $\mu, \lambda \in [0, 1]$ 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. c is one of the children of the node n, also indexed by $\vec{I}$. SPTK has been shown to be rather efficient in practice (Croce et al., 2011, 2012).

3.2 Structural representation for text

Syntactic and semantic structures can play an important role in building effective representations for machine learning algorithms. The automatic extraction of features from tree structured representations of text is natural within the TK framework. Therefore, several studies have shown the power of associating rich structural encoding with TKs (Severyn et al., 2013; Tymoshenko and Moschitti, 2015).

In Croce et al. (2011), a wide array of representations derived from the parse tree of a sentence are evaluated. The Lexical Centered Tree (LCT) is shown to be the best performing tree layout for the QC task. An LCT, as shown in Figure 1, contains lexicals at the pre-terminal levels, and their grammatical functions and POS-tags are added as leftmost children. In addition, each lexical node is encoded as a word lemma, and has a suffix which is composed by a special `::` symbol and the first

[1]For a similarity score between 0 and 1, a normalization in the kernel space, i.e. $\frac{TK(T_1,T_2)}{\sqrt{TK(T_1,T_1)\times TK(T_2,T_2)}}$ is applied.

letter of the POS-tag of the word. These marked lexical nodes are then mapped to their corresponding numerical vectors, which are used in the kernel computation. Only lemmas sharing the same POS-tag are compared in the semantic kernel similarity.

4 Context Word Embeddings for SPTK

We propose to compute the similarity function σ in SPTK as the cosine similarity of word embeddings obtained with neural networks. We experimented with the popular Continuous Bag-Of-Words (CBOW), SkipGram models (Mikolov et al., 2013), and GloVe (Pennington et al., 2014).

4.1 Part-of-speech tags in word embeddings

As in (Croce et al., 2011), we observed that embeddings learned from raw words are not the most effective in the TK computation. Thus, similarly to Trask et al. (2015), we attach a special :: suffix plus the first letter of the part-of-speech (POS) to the word lemmas. This way, we differentiate words by their tags, and learn specific embedding vectors for each of them. This approach increases the performance of our models.

4.2 Modeling the word context

Although a word vector encodes some information about word co-occurrences, the context around a word, as also suggested in Iacobacci et al. (2016), can explicitly contribute to the word similarity, especially when the target words are infrequent. For this reason, we also represent each word as the concatenation of its embedding with a second vector, which is supposed to model the context around the word. We build this vector as (i) a simple average of the embeddings of the other words in the sentence, and (ii) with a method specifically designed to embed longer units of text, namely paragraph2vec (Le and Mikolov, 2014). This is similar to word2vec: a network is trained to predict a word given its context, but it can access to an additional vector specific for the paragraph, where the word and the context are sampled.

5 Recurrent Networks for Encoding Text

As described in Sec. 2, a Siamese Network encodes two inputs into a vectorial representation, reusing the network parameters. In this section, we briefly describe the standard units used in our Siamese Network to encode sentences.

5.1 Recurrent neural network units

Recurrent Neural Networks (RNNs) constitute one of the main architectures used to model sequences, and they have seen a wide adoption in the NLP literature. Vanilla RNNs consume a sequence of vectors one step at the time, and update their internal state as a function of the new input and their previous internal state. For this reason, at any given step, the internal state depends on the entire history of previous states. These networks suffer from the vanishing gradient problem (Bengio et al., 1994), which is mitigated by a popular RNN variant, the Long Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997). An LSTM can control the amount of information from the input that affects its internal state, the amount of information in the internal state that can be forgotten, and how the internal state affects the output of the network.

The Gated Recurrent Unit (GRU) (Chung et al., 2014) is an LSTM variant with similar performance and less parameters, thus faster to train. Since we use this recurrent unit in our model, we briefly review it. Let x_t and s_t be the input vector and state at timestep t, given a sequence of input vectors $(x_1, ..., x_T)$, the GRU computes a sequence of states $(s_1, ..., s_T)$ according to the following equations:

$$z = \sigma(x_t U^z + s_{t-1} W^z)$$
$$r = \sigma(x_t U^r + s_{t-1} W^r)$$
$$h = tanh(x_t U^h + (s_{t-1} \circ r) W^h)$$
$$s_t = (1 - z) \circ h + z \circ s_{t-1}$$

The GRU has an update, z, and reset gate, r, and does not have an internal memory beside the internal state. The U and W matrices are parameters of the model. σ is the logistic function, the $\circ$ operator denotes the elementwise (Hadamard) product, and $tanh$ is the hyperbolic tangent function. All the non-linearities are applied elementwise.

5.2 Bidirectional networks

The aforementioned recurrent units consume the input sequence in one direction, and thus earlier internal states do not have access to future steps. Bidirectional RNNs (Schuster and Paliwal, 1997) solve this issue by keeping a forward and backward internal states that are computed by going through the input sequence in both directions. The state at any given step will be the concatenation of

the forward and backward state at that step, and, in our case, will contain useful information from both the left and right context of a word.

6 Contextual Word Similarity Network

The methods to model the context described in Sec. 4.2 augment the target word vector with dimensions derived from the entire sentence. This provides some context that may increase the discriminative power of SPTK. The latter can thus use a similarity between two words dependent on the sentences which they belong to. For example, when SPTK carries out a QC task, the sentences above have higher probability to share similar context if they belong to the same category. Still, this approach is rather shallow as two words of the same sentence would be associated with almost the same context vector. That is, the approach does not really transform the embedding of a given word as a function of its context.

An alternative approach is to train the context embedding using neural networks on a sense annotated corpus, which can remap the word embeddings in a supervised fashion. However, since there are not enough large disambiguated corpora, we need to approximate the word senses with coarse-grained information, e.g., the category of the context. In other words, we can train a network to decide if two target words are sampled from sentences belonging to the same category. This way, the states of the trained network corresponding to each word can be eventually used as word-in-context embeddings.

In the next sections, we present the classification task designed for this purpose, and then the architecture of our Siamese Network for learning contextual word embeddings.

6.1 Defining the derived classification task

The end task that we consider is the categorization of a sentence $s \in D = \{s_1, ..., s_n\}$ into one class $c_i \in C = \{c_1, ..., c_m\}$, where D is our collection of n sentences, and C is the set of m sentence categories. Intuitively, we define the derived task as determining if two words extracted from two different sentences share the same sentence category or not. Our classifier learns word representations while accessing to the entire sentence.

More formally, we sample a pair of labeled sentences $\langle s_i, c_i \rangle, \langle s_j, c_j \rangle$ from our training set, where $i \neq j$. Then, we sample a word from each sentence, $w_a \in s_i$ and $w_b \in s_j$, and we assign a label $y \in \{0, 1\}$ to the word pair. We set $y = 0$ if $c_i \neq c_j$, and $y = 1$ if $c_i = c_j$.

Our goal is to learn a mapping f such that:

$$sim(f(s_i, w_a), f(s_j, w_b)) \in [0, 1], \quad (2)$$

where sim is a similarity function between two vectors that should output values close to 1 when $y = 1$, and values close to 0 when $y = 0$.

6.2 Data construction for the derived task

To generate sentence pairs, we randomly sample sentences from different categories. Pairs labeled as positive are constructed by randomly sampling sentences from the same category, without replacement. Pairs labeled as negative are constructed by randomly sampling the first sentence from one category, and the second sentence from the remaining categories, again without replacement. Note that we oversample low frequency categories, and sample positive and negative examples several times to collect diverse pairs. We remove duplicates, and stop the generation process at approximately 500,000 sentence pairs.

6.3 Bidirectional GRUs for Word Similarity

We model the function f that maps a sentence and one of its words into a fixed size representation as a neural network. We aim at using the f encoder to map different word/sentence pairs into the same embedding space. Since the two input sentences play a symmetric role in our desired similarity and we need to use the same weights for both, we opt for a Siamese architecture (Chopra et al., 2005).

In this setting, the same network is applied to two input instances reusing the weights. Alternatively, the network can be seen as having two branches that share all the parameter weights.

The optimization strategy is what differentiates our Siamese Network from others that compute textual similarity. We do not compute the similarity (and thus the loss) between two sentences. Instead, we compute the similarity between the contextual representations of two random words from the two sentences.

This is clearly depicted in Fig. 2. The input words are mapped to integer ids, which are looked up in an embedding matrix to retrieve the corresponding embedding vectors. The sequence of vectors is then consumed by a 3-layer Bidirectional GRU (BiGRU). We selected a BiGRU for

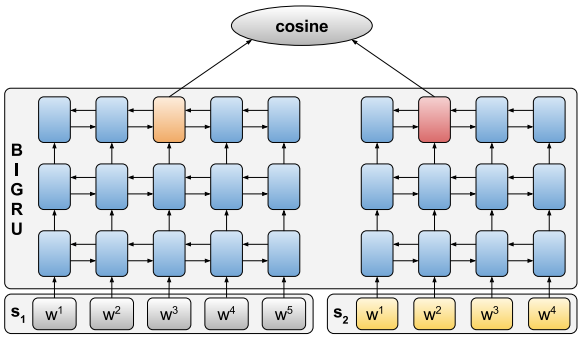

Figure 2: The architecture of the Siamese Network. The network computes $sim(f(s_1, 3), f(s_2, 2))$. The word embeddings of each sentence are consumed by a stack of 3 Bidirectional GRUs. The two branches of the network share the parameter weights.

	CBOW		SkipGram		GloVe
	hs	*ns*	*hs*	*ns*	-
dim					
50	89.8	89.8	91.0	91.6	89.8
100	93.0	93.6	94.2	92.8	91.6
150	94.2	94.0	94.2	93.8	92.4
200	94.6	93.6	93.2	94.2	93.2
250	94.4	94.4	94.2	94.2	93.6
300	94.2	94.0	94.4	94.0	93.8
500	95.2	95.0	94.8	93.8	94.4
750	94.8	94.6	95.0	94.4	94.2
1000	93.4	95.2	95.2	94.6	94.0

Table 2: QC test set accuracies (%) of NSPTK, given embeddings with window size equal to 5, and dimensionality ranging from 50 to 1,000.

our experiments as they are more efficient and accurate than LSTMs for our tasks. We tried other architectures, including convolutional networks, but RNNs gave us better results with less complexity and tuning effort. Note that the weights of the RNNs are shared between the two branches.

Each RNN layer produces a state for each word, which is consumed by the next RNN in the stack. From the top layer, the state corresponding to the word in the similarity pair is selected. This state encodes the word given its sentential context. Thus, the first layer, $BiGRU'$, maps the sequence of input vectors $(x_1, ..., x_T)$, into a sequence of states $(s'_1, ..., s'_T)$, the second, $BiGRU''$, transforms those states into $(s''_1, ..., s''_T)$, and the third, $BiGRU'''$, produces the final representations of the words in context $(s'''_1, ..., s'''_T)$.

Eventually, the network computes the similarity of a pair of encoded words, selected from the two sentences. We optimize the cosine similarity to match the similarity function used in SPTK. We rescale the output similarity in the $[0, 1]$ range and train the network to minimize the log loss between predictions and true labels.

7 Experiments

We compare SPTK models with our tree kernel model using neural word embeddings (NSPTK) on question classification (QC), a central task for question answering, and on sentiment classfication (SC).

7.1 Experimental setup

Data. The QC dataset (Li and Roth, 2006) contains a set of questions labelled according to a two-layered taxonomy, which describes their expected answer type. The coarse layer maps each question into one of 6 classes: Abbreviation, Description, Entity, Human, Location and Number. Our experimental setting mirrors the setting of the original study: we train on 5,452 questions and test on 500.

The SC dataset is the one of SemEval Twitter'13 for message-level polarity classification (Nakov et al., 2013). The dataset is organized in a training, development and test sets containing respectively 9,728, 1,654 and 3,813 tweets. Each tweet is labeled as positive, neutral or negative. The only preprocessing step we perform on tweets is to replace user mentions and url with a <USER> and <URL> token, respectively.

In the cross-validation experiments, we use the training data to produce the training and test folds, whereas we use the original test set as our validation set for tuning the parameters of the network.

Word embeddings. Learning high quality word embeddings requires large textual corpora. We train all the vectors for QC on the ukWaC corpus (Ferraresi et al., 2008), also used in Croce et al. (2011) to obtain LSA vectors. The corpus includes an annotation layer produced with TreeTagger [2]. We process the documents by attaching the POS-tag marker to each lemma. We trained paragraph2vec vectors using the Gensim[3] toolkit. Word embeddings for the SC task are learned on a corpus of 50M English tweets collected from the Twitter API over two months, using word2vec and setting the dimension to 100.

Neural model. We use GloVe word embeddings (300 dimensions), and we fix them during training. Embeddings for words that are not present in

[2]http://www.cis.uni-muenchen.de/ schmid/tools/TreeTagger/
[3]https://radimrehurek.com/gensim/

	CBOW hs	SkipGram hs	GloVe -	LSA -
dim				
100	84.47	84.63	82.92	-
250	85.75	85.85	85.04	85.39
500	86.48	86.32	85.73	-

Table 3: QC cross-validation accuracies (%) of NSPTK given embeddings with the selected dimensionalities.

word	context	QC test accuracy	QC CV accuracy
w2v	-	95.2	86.48
w2v	w2v	95.4	86.08†
w2v	p2v	95.0	86.46
p2v	-	92.8	82.65†
p2v	p2v	93.6	83.47†

Table 4: QC accuracies for the word embeddings (CBOW vectors with 500 dimensions, trained using hierarchical softmax) and paragraph2vec.

the embedding model are randomly initalized by sampling a vector of the same dimension from the uniform distribution $U[-0.25, 0.25]$.

The size of the forward and backward states of the BiGRUs is set to 100, so the resulting concatenated state has 200 dimensions. The number of stacked bidirectional networks is three and it was tuned on a development set. This allows the network to have high capacity, fit the data, and have the best generalization ability. The final layer learns higher order representations of the words in context. We did not use dropout as a regularization mechanism since it did not show a significant difference on the performance of the network. The network parameters are trained using the Adam optimizer (Kingma and Ba, 2014), with a learning rate of 0.001.

The training examples are fed to the network in mini-batches. The latter are balanced between positive and negative examples by picking 32 pairs of sentences sharing the same category, and 32 pairs of sentences from different categories. Batches of 64 sentences are fed to the network. The number of words sampled from each sentence is fixed to 4, and for this reason the final loss is computed over 256 pairs of words in context, for each mini-batch. The network is then trained for 5 epochs, storing the parameters corresponding to the best registered accuracy on the validation set. Those weights are later loaded and used to encode the words in a sentence by taking their corresponding output states from the last BiGRU unit.

Structural models. We trained the tree kernel models using SVM-Light-TK (Moschitti, 2004), an SVM-Light extension (Joachims, 1999) with tree kernel support. We modified the software to lookup specific vectors for each word in a sentence. We preprocessed each sentence with the LTH parser[4] and used its output to construct the LCT. We used the parameters for the QC classifiers from Croce et al. (2011), while we selected them on the Twitter'13 dev. set for the SC task.

word	context	QC CV accuracy	Std. dev
w2v	-	86.48	.005
BiGRUs	-	84.61†	.027
w2v	BiGRUs	**88.32**†	.009

Table 5: QC accuracies for NSPTK, using the word-in-context vector produced by the stacked BiGRU encoder trained with the Siamese Network. Word vectors are trained with CBOW (hs) and have 500 dimensions.

word	context	SC F_1^{PN}
w2v	-	48.65
w2v	w2v	51.59
w2v	BiGRUs	**60.96**

SemEval system	SC F_1^{PN}
Castellucci et al. (2013)	58.27
Dong et al. (2015)	72.8

Table 6: SC results for NSPTK with word embeddings and the word-in-context embeddings. Runs of selected systems are also reported.

7.2 Context Embedding Results

Table 2 shows the QC accuracy of NSPTK with CBOW, SkipGram and GloVe. The results are reported for vector dimensions (*dim*) ranging from 50 to 1000, with a fixed window size of 5.

The performance for the CBOW hierarchical softmax (hs) and negative sampling (ns), and for the SkipGram hs settings are similar. For the SkipGram ns settings, the accuracy is slightly lower for smaller dimension sizes. GloVe embeddings yield a lower accuracy, which steadily increases with the size of the embeddings. In general, a higher dimension size produces higher accuracy, but also makes the training more expensive. 500 dimensions seem a good trade-off between performance and computational cost.

To better validate the performance of NSPTK, and since the usual test set may have reached a saturation point, we cross-validate some models.

[4] http://nlp.cs.lth.se

Question	Wrong w2v	Correct BiGRU
1) What is the occupation of Nicholas Cage ?	enty	hum
2) What level of government (...) is responsible for dealing with racism?	num	hum
3) What is the Motto for the State of Maryland?	loc	desc
4) What is a virtual IP address?	loc	desc
5) What function does a community's water tower serve?	loc	desc

Table 7: Sample of sentences where NSPTK with word vectors fails, and the BiGRU model produces correct classifications.

We use the training set to perform a 5-fold stratified cross-validation (CV), such that the distribution of labels in each fold is similar. Table 3 shows the cross-validated results for a subset of word embedding models. Neural embeddings seem to give a slightly higher accuracy than LSA. A more substantial performance edge may come from modeling the context, thus we experimented with word embeddings concatenated to context embeddings.

Table 4 shows the results of NSPTK using different word encodings. The *word* and *context* columns refer to the model used for encoding the word and the context, respectively. These models are word2vec (*w2v*) and paragraph2vec (*p2v*). The word2vec vector for the context is produced by averaging the embedding vectors of the other words in the sentence, i.e., excluding the target word. The paragraph2vec model has its own procedure to embed the words in the context. CV results marked with † are significant with a p-value < 0.005. The cross-validation results reveal that word2vec embeddings without context are a tough baseline to beat, suggesting that standard ways to model the context are not effective.

7.3 Results of our Bidirectional GRU for Word Similarity

Table 5 shows the results of encoding the words in context using a more sophisticated approach: mapping the word to a representation learned with the Siamese Network that we optimize on the derived classification task presented in Section 6.1. The NSPTK operating on word vectors (best vectors from Table 3) concatenated with the word-in-context vectors produced by the stacked BiGRU encoder, registers a significant improvement over word vectors alone. In this case, the results marked with † are significant with a p-value < 0.002. This indicates that the strong similarity contribution coming from word vectors is successfully affected by the word-in-context vectors from the network. The original similarities are thus modulated to be more effective for the final classification task. Another possible advantage of the model is that unknown words, which do not participate in the context average of simpler model, have a potentially more useful representation in the internal states of the network.

7.4 Sentiment Classification

Table 6 reports the results on the SC task. This experiment shows that incorporating the context in the similarity computation slightly improves the performance of the NSPTK. The real improvement, 12.31 absolute percent points over using word vectors alone, comes from modeling the words in context with the BiGRU encoder, confirming it as an effective strategy to improve the modeling capabilities of NSPTK.

Interestingly, our model with a single kernel function and without complex text normalization techniques outperforms a multikernel system (Castellucci et al., 2013), when the word-in-context embeddings are incorporated. The multikernel system is applied on preprocessed text and includes a Bag-Of-Words Kernel, a Lexical Semantic Kernel, and a Smoothed Partial Tree Kernel. State-of-the-art systems (Dong et al., 2015; Severyn and Moschitti, 2015b) include many lexical and clustering features, sentiment lexicons, and distant supervision techniques. Our approach does not include any of the former.

7.5 Wins of the BiGRU model

An error analysis on the QC task reveals the *What* questions as the most ambiguous. Table 7 contains some of the successes of the BiGRU model with respect to the model using only word vectors. Those wins can be explained by the effect of the contextual word vectors on the kernel similarity. In Question 1, the meaning of *occupation* is affected by the presence of a person name. In Question 2, the word *level* loses its prevalent association with quantities. In questions 3 to 5, the underlined words are a strong indicator of locations/places, and the kernel similarity may be

dominated by their corresponding word vectors. BiGRU vectors are instead able to effectively re-modulate the kernel similarity and induce a correct classification.

8 Conclusions

In this paper, we applied neural network models for learning representations with semantic convolution tree kernels. We evaluated the main distributional representation methods for computing semantic similarity inside the kernel. In addition, we augmented the vectorial representations of words with information coming from the sentential content. Word vectors alone revealed to be difficult to improve upon. To better model the context, we proposed word-in-context representations extracted from the states of a recurrent neural network. Such network learns to decide if two words are sampled from sentences which share the same category label. The resulting embeddings are able to improve on the selected tasks when used in conjunction with the original word embeddings, by injecting more contextual information for the modulation of the kernel similarity. We show that our approach can improve the accuracy of the convolution semantic tree kernel.

Acknowledgments. This work has been supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action). The first author was supported by the Google Europe Doctoral Fellowship Award 2015. Many thanks to the anonymous reviewers for their valuable suggestions.

References

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 2442–2452. http://www.aclweb.org/anthology/P16-1231.

Y. Bengio, P. Simard, and P. Frasconi. 1994. Learning Long-term Dependencies with Gradient Descent is Difficult. *Trans. Neur. Netw.* 5(2):157–166. https://doi.org/10.1109/72.279181.

Giuseppe Castellucci, Simone Filice, Danilo Croce, and Roberto Basili. 2013. UNITOR: Combining Syntactic and Semantic Kernels for Twitter Sentiment Analysis. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 369–374. http://www.aclweb.org/anthology/S13-2060.

Sumit Chopra, Raia Hadsell, and Yann LeCun. 2005. Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*. IEEE, volume 1, pages 539–546.

Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555* .

Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2011. Structured Lexical Similarity via Convolution Kernels on Dependency Trees. In *In EMNLP*. Edinburgh, Scotland, UK. http://www.aclweb.org/anthology/D11-1096.

Danilo Croce, Alessandro Moschitti, Roberto Basili, and Martha Palmer. 2012. Verb Classification using Distributional Similarity in Syntactic and Semantic Structures. In *ACL (1)*. The Association for Computer Linguistics, pages 263–272.

Li Dong, Furu Wei, Yichun Yin, Ming Zhou, and Ke Xu. 2015. Splusplus: A Feature-Rich Two-stage Classifier for Sentiment Analysis of Tweets. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 515–519. http://www.aclweb.org/anthology/S15-2086.

Adriano Ferraresi, Eros Zanchetta, Marco Baroni, and Silvia Bernardini. 2008. Introducing and evaluating ukWaC, a very large web-derived corpus of English. In *Proceedings of the 4th Web as Corpus Workshop (WAC-4) Can we beat Google?*. page 47.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. In *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*. Association for Computational Linguistics, San Diego, California, pages 1116–1123. http://www.aclweb.org/anthology/S16-1172.

Simone Filice, Giovanni Da San Martino, and Alessandro Moschitti. 2015. Structural Representations for Learning Relations between Pairs of Texts. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume*

1: Long Papers). Association for Computational Linguistics, Beijing, China, pages 1003–1013. http://www.aclweb.org/anthology/P15-1097.

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

Ignacio Iacobacci, Mohammad Taher Pilehvar, and Roberto Navigli. 2016. Embeddings for Word Sense Disambiguation: An Evaluation Study. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 897–907. http://www.aclweb.org/anthology/P16-1085.

Thorsten Joachims. 1999. Making Large-scale Support Vector Machine Learning Practical. In Bernhard Schölkopf, Christopher J. C. Burges, and Alexander J. Smola, editors, *Advances in Kernel Methods*, MIT Press, Cambridge, MA, USA, pages 169–184. http://dl.acm.org/citation.cfm?id=299094.299104.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Baltimore, Maryland, pages 655–665. http://www.aclweb.org/anthology/P14-1062.

Yoon Kim. 2014. Convolutional Neural Networks for Sentence Classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, Doha, Qatar, pages 1746–1751. http://www.aclweb.org/anthology/D14-1181.

Diederik P. Kingma and Jimmy Ba. 2014. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*.

Quoc V. Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents. *CoRR* abs/1405.4053. http://arxiv.org/abs/1405.4053.

Omer Levy and Yoav Goldberg. 2014. Dependency-Based Word Embeddings. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Association for Computational Linguistics, Baltimore, Maryland, pages 302–308. http://www.aclweb.org/anthology/P14-2050.

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

Mingbo Ma, Liang Huang, Bowen Zhou, and Bing Xiang. 2015. Dependency-based Convolutional Neural Networks for Sentence Embedding. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. Association for Computational Linguistics, Beijing, China, pages 174–179. http://www.aclweb.org/anthology/P15-2029.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *ICLR Workshop* .

Alessandro Moschitti. 2004. A Study on Convolution Kernels for Shallow Semantic Parsing. In *Proceedings of the 42nd Annual Meeting on Association for Computational Linguistics*. Association for Computational Linguistics, Stroudsburg, PA, USA, ACL '04.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. In *Proceedings of the 17th European Conference on Machine Learning*. Springer-Verlag, Berlin, Heidelberg, ECML'06, pages 318–329. https://doi.org/10.1007/11871842_32.

Jonas Mueller and Aditya Thyagarajan. 2016. Siamese Recurrent Architectures for Learning Sentence Similarity. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. AAAI Press, AAAI'16, pages 2786–2792. http://dl.acm.org/citation.cfm?id=3016100.3016291.

Preslav Nakov, Sara Rosenthal, Zornitsa Kozareva, Veselin Stoyanov, Alan Ritter, and Theresa Wilson. 2013. SemEval-2013 Task 2: Sentiment Analysis in Twitter. In *Second Joint Conference on Lexical and Computational Semantics (*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*. Association for Computational Linguistics, Atlanta, Georgia, USA, pages 312–320. http://www.aclweb.org/anthology/S13-2052.

Paul Neculoiu, Maarten Versteegh, and Mihai Rotaru. 2016. Learning Text Similarity with Siamese Recurrent Networks. In *Proceedings of the 1st Workshop on Representation Learning for NLP*. Association for Computational Linguistics, Berlin, Germany, pages 148–157. http://anthology.aclweb.org/W16-1617.

Thien Hai Nguyen and Kiyoaki Shirai. 2015. Aspect-Based Sentiment Analysis Using Tree Kernel Based Relation Extraction. In Alexander Gelbukh, editor, *Computational Linguistics and Intelligent Text Processing: 16th International Conference, CICLing 2015*. Springer International Publishing, Cairo, Egypt, pages 114–125.

Thien Huu Nguyen, Barbara Plank, and Ralph Grishman. 2015. Semantic Representations for Domain Adaptation: A Case Study on the Tree Kernel-based Method for Relation Extraction. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International*

Joint Conference on Natural Language Processing (Volume 1: Long Papers). Association for Computational Linguistics, Beijing, China, pages 635–644. http://www.aclweb.org/anthology/P15-1062.

Truc Vien T. Nguyen and Alessandro Moschitti. 2011. End-to-End Relation Extraction Using Distant Supervision from External Semantic Repositories. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Portland, Oregon, USA, pages 277–282. http://www.aclweb.org/anthology/P11-2048.

Truc-Vien T. Nguyen, Alessandro Moschitti, and Giuseppe Riccardi. 2009. Convolution Kernels on Constituent, Dependency and Sequential Structures for Relation Extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, pages 1378–1387. http://www.aclweb.org/anthology/D/D09/D09-1143.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. GloVe: Global Vectors for Word Representation. In *Empirical Methods in Natural Language Processing*. pages 1532–1543. http://www.aclweb.org/anthology/D14-1162.

Barbara Plank and Alessandro Moschitti. 2013. Embedding Semantic Similarity in Tree Kernels for Domain Adaptation of Relation Extraction. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Sofia, Bulgaria, pages 1498–1507. http://www.aclweb.org/anthology/P13-1147.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-Contrastive Estimation for Answer Selection with Deep Neural Networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, New York, NY, USA, CIKM '16, pages 1913–1916.

Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing* 45(11):2673–2681.

Aliaksei Severyn and Alessandro Moschitti. 2015a. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, pages 373–382.

Aliaksei Severyn and Alessandro Moschitti. 2015b. UNITN: Training Deep Convolutional Neural Network for Twitter Sentiment Classification. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*. Association for Computational Linguistics, Denver, Colorado, pages 464–469. http://www.aclweb.org/anthology/S15-2079.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning Semantic Textual Similarity with Structural Representations. In *Proceedings of the 51st Annual Meeting of the ACL (Volume 2: Short Papers)*. ACL, pages 714–718. http://aclweb.org/anthology/P13-2125.

João Silva, Luísa Coheur, Ana Cristina Mendes, and Andreas Wichert. 2010. From symbolic to subsymbolic information in question classification. *Artificial Intelligence Review* 35(2):137–154.

Richard Socher, Brody Huval, Christopher D. Manning, and Andrew Y. Ng. 2012. Semantic Compositionality through Recursive Matrix-Vector Spaces. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*. Association for Computational Linguistics, Jeju Island, Korea, pages 1201–1211. http://www.aclweb.org/anthology/D12-1110.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Seattle, Washington, USA, pages 1631–1642. http://www.aclweb.org/anthology/D13-1170.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics, Beijing, China, pages 1556–1566. http://www.aclweb.org/anthology/P15-1150.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2016. Improved Representation Learning for Question Answer Matching. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Association for Computational Linguistics, Berlin, Germany, pages 464–473. http://www.aclweb.org/anthology/P16-1044.

Andrew Trask, Phil Michalak, and John Liu. 2015. sense2vec - A Fast and Accurate Method for Word Sense Disambiguation In Neural Word Embeddings. *CoRR* abs/1511.06388. http://arxiv.org/abs/1511.06388.

Kateryna Tymoshenko and Alessandro Moschitti. 2015. Assessing the Impact of Syntactic and Semantic Structures for Answer Passages Reranking. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management, CIKM 2015, Melbourne, VIC, Australia, October 19 - 23, 2015*. pages 1451–1460.