Convolutional Neural Networks vs. Convolution Kernels: Feature Engineering for Answer Sentence Reranking

Kateryna Tymoshenko[†] and **Daniele Bonadiman**[†] and **Alessandro Moschitti**
[†]DISI, University of Trento, 38123 Povo (TN), Italy
Qatar Computing Research Institute, HBKU, 5825 Doha, Qatar
{kateryna.tymoshenko,d.bonadiman}@unitn.it
amoschitti@qf.org.qa

Abstract

In this paper, we study, compare and combine two state-of-the-art approaches to automatic feature engineering: Convolution Tree Kernels (CTKs) and Convolutional Neural Networks (CNNs) for learning to rank answer sentences in a Question Answering (QA) setting. When dealing with QA, the key aspect is to encode relational information between the constituents of question and answer in learning algorithms. For this purpose, we propose novel CNNs using relational information and combined them with relational CTKs. The results show that (i) both approaches achieve the state of the art on a question answering task, where CTKs produce higher accuracy and (ii) combining such methods leads to unprecedented high results.

1 Introduction

The increasing use of machine learning for the design of NLP applications pushes for fast methods for feature engineering. In contrast, the latter typically requires considerable effort especially when dealing with highly semantic tasks such as QA. For example, for an effective design of automated QA systems, the question text needs to be put in relation with the text passages retrieved from a document collection to enable an accurate extraction of the correct answers from passages. From a machine learning perspective, encoding the information above consists in manually defining expressive rules and features based on syntactic and semantic patterns.

Therefore, methods for automatizing feature engineering are remarkably important also in the light of fast prototyping of commercial applications. To the best of our knowledge, two of the most effective methods for engineering features are: (i) kernel methods, which naturally map feature vectors or directly objects in richer feature spaces; and more recently (ii) approaches based on deep learning, which have been shown to be very effective.

Regarding the former, in (Moschitti et al., 2007), we firstly used CTKs in Support Vector Machines (SVMs) to generate features from a question (Q) and their candidate answer passages (AP). CTKs enable SVMs to learn in the space of convolutional subtrees of syntactic and semantic trees used for representing Q and AP. This automatically engineers syntactic/semantic features. One important characteristic we added in (Severyn and Moschitti, 2012) is the use of relational links between Q and AP, which basically merged the two syntactic trees in a relational graph (containing relational features).

Although based on different principles, also CNNs can generate powerful features, e.g., see (Kalchbrenner et al., 2014; Kim, 2014). CNNs can effectively capture the compositional process of mapping the meaning of individual words in a sentence to a continuous representation of the sentence. This way CNNs can efficiently learn to embed input sentences into low-dimensional vector space, preserving important syntactic and semantic aspects of the input sentence. However, engineering features spanning two pieces of text such as in QA is a more complex task than classifying single sentences. Indeed, only very recently, CNNs were proposed for QA by Yu et al. (2014). Although, such network achieved high accuracy, its design is still not enough to model relational features.

Proceedings of NAACL-HLT 2016, pages 1268–1278,
San Diego, California, June 12-17, 2016.

In this paper, we aim at comparing the ability of CTKs and CNNs of generating features for QA. For this purpose, we first explore CTKs applied to shallow linguistic structures for automatically learning classification and ranking functions with SVMs.

At the same time, we assess a novel deep learning architecture for effectively modeling Q and AP pairs generating relational features we initially modeled in (Severyn and Moschitti, 2015; Severyn and Moschitti, 2016). The main building blocks of our approach are two sentence models based on CNNs. These work in parallel, mapping questions and answer sentences to fixed size vectors, which are then used to learn the semantic similarity between them. To compute question-answer similarity score we adopt the approach used by Yu et al. (2014). Our main novelty is the way we model relational information: we inject overlapping words directly into the word embeddings as additional dimensions. The augmented word representation is then passed through the layers of the convolutional feature extractors, which encode the relatedness between Q and AP pairs in a more structured manner. Moreover, the embedding dimensions encoding overlapping words are parameters of the network and are tuned during training.

We experiment with two different QA benchmarks for sentence reranking TREC13 (Wang et al., 2007) and WikiQA (Yang et al., 2015). We compare CTKs and CNNs and then we also combine them. For this purpose, we design a new kernel that sum together CTKs and different embeddings extracted from different CNN layers. Our CTK-based models achieve the state of the art on TREC 13, obtaining an MRR of 85.53 and an MAP of 75.18 largely outperforming all the previous best results. On WikiQA, our CNNs perform almost on par with tree kernels, i.e., an MRR of 71.07 vs. 72.51 of CTK, which again is the current state of the art on such data. The combination between CTK and CNNs produces a further boost, achieving an MRR of 75.52 and an MAP of 73.99, confirming that the research line of combining these two interesting machine learning methods is very promising.

2 Related Work

Relational learning from entire pieces of text concerns several natural language processing tasks, e.g., QA (Moschitti, 2008), Textual Entailment (Zanzotto and Moschitti, 2006) and Paraphrase Identification (Filice et al., 2015). Regarding QA, a referring work for our research is the IBM Watson system (Ferrucci et al., 2010). This is an advanced QA pipeline based on deep linguistic processing and semantic resources.

Wang et al. (2007) used quasi-synchronous grammar to model relations between a question and a candidate answer with syntactic transformations. (Heilman and Smith, 2010) applied Tree Edit Distance (TED) for learning tree transformations in a Q/AP pair. (Wang and Manning, 2010) designed a probabilistic model to learn tree-edit operations on dependency parse trees. (Yao et al., 2013) applied linear chain CRFs with features derived from TED to automatically learn associations between questions and candidate answers. Yih et al. (2013a) applied enhanced lexical semantics to build a word-alignment model, exploiting a number of large-scale external semantic resources.

Although the above approaches are very valuable, they required considerable effort to study, define and implement features that could capture relational representations. In contrast, we are interested in techniques that try to automatize the feature engineering step. In this respect, our work (Moschitti et al., 2007) is the first using CTKs applied to syntactic and semantic structural representations of the Q/AP pairs in a learning to rank algorithm based on SVMs. After this, we proposed several important improvement exploiting different type of relational links between Q and AP, i.e., (Severyn and Moschitti, 2012; Severyn et al., 2013; Severyn and Moschitti, 2013; Tymoshenko et al., 2014; Tymoshenko and Moschitti, 2015). The main difference with our previous approaches is usage of better-preprocessing algorithms and new structural representations, which highly outperform them.

Recently, deep learning approaches have been successfully applied to various sentence classification tasks, e.g., (Kalchbrenner et al., 2014; Kim, 2014), and for automatically modeling text pairs, e.g., (Lu and Li, 2013; Hu et al., 2014). Additionally, a number of deep learning models have been recently applied to question answering, e.g., Yih et al. (2014) applied CNNs to open-domain QA; Bordes et al. (2014b) propose a neural embedding model

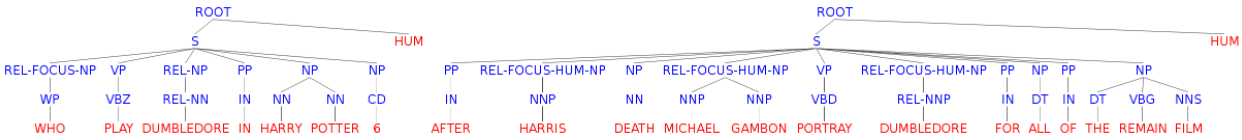

Figure 1: Shallow chunk-based tree for the Q/AP pair in the running example.

combined with the knowledge base for open-domain QA; Iyyer et al. (2014) applied recursive neural networks to factoid QA over paragraphs. (Miao et al., 2015) proposed a neural variational inference model and a Long-short Term Memory network for the same task. Recently (Yin et al., 2015) proposed a siamese convolutional network for matching sentences that employ an attentive average pooling mechanism, obtaining state-of-the-art results in various tasks and datasets. The work closest to this paper is (Yu et al., 2014) and (Severyn and Moschitti, 2015). The former presented a CNN architecture for answer sentence selection that uses a bigram convolution and average pooling, whereas in the latter we used convolution with k-max pooling. However, these models only partially captures relational information. In contrast, in this paper, we encode relational information about words that are matched betweem Q and AP.

3 Feature Engineering for QA with CTKs

Our approach to learning relations between two texts is to first convert them into a richer structural representation based on their syntactic and semantic structures, and then apply CTKs. To make our approach more effective, we further enriched structures with relational semantics by linking the related constituents with lexical and other semantic links.

3.1 Shallow Representation of Short Text Pairs

In our study, we employ a modified version of the shallow structural representation of question and answer pairs, **CH**, described in (Severyn et al., 2013; Tymoshenko and Moschitti, 2015). We represent a pair of short texts as two trees with lemmas at leaf level and their part-of-speech (POS) tags at the preterminal level. Preterminal POS-tags are grouped into chunk nodes and the chunks are further grouped into sentences. Figure 1 provides an example of this structure.

We enrich the above representation with the information about question class and question focus. Questions are classified in terms of their expected answer type. (Severyn et al., 2013) employed coarse-grained classes from (Li and Roth, 2002), namely HUM (person), ENTY (an entity), DESC (description), LOC (location), and NUM (number). In this work, we split the NUM class into three subcategories, DATE, QUANTITY, CURRENCY and train question classifiers as described in (Severyn et al., 2013). Differently from before, we add the question class node as the rightmost child of the root node both to the question and the answer structures.

We detect question focus using a focus classifier, FCLASS, trained as in (Severyn et al., 2013). However, in our previous model, we classified all words over the chunks in the question and picked the one with the highest FCLASS prediction score as a focus even if it is negative. In this work, if FCLASS assigns negative scores to all the question chunks, we consider the first question chunk, which is typically a question word, to be a focus. We mark the focus chunk by prepending the REL-FOCUS tag to its label.

In previous work, we have shown the importance of encoding information about the relatedness between Q and AP into their structural representations. Thus, we employ lexical and question class match, described hereafter.

Lexical match. Lemmas that occur both in Q and AP are marked by prepending the **REL** tag to the labels of the corresponding preterminal nodes and their parents.

Question class match. We detect named entities (NEs) in AP and mark the NEs of type compatible[1] with the question class by prepending the REL-FOCUS-QC label to the corresponding pre-preterminals in the trees. The QC suffix in the labels

[1]Compatibility is checked using a predefined table, namely Person, Organization→ HUM, ENTY; Misc→ ENTY; Location→ LOC; Date, Time, Number→ DATE; Money, Number→ CURRENCY; Percentage, Number→ QUANTITY

is replaced by the question class in the given pair.

For example, in Figure 1, the Dumbledore lemma occurs in both Q and AP, therefore the respective POS and chunk nodes are marked with **REL**. The named entities, Harris, Michael Gambon and Dumbledore have the type Person compatible with the question class HUM, thus their respective chunk nodes are marked as REL-FOCUS-HUM (overriding the previously inserted REL tag for the Dumbledore chunk).

3.2 Reranking with Tree Kernels

We aim at learning reranker that can decide which Q/AP pair is more probably correct than others, where correct Q/AP pairs are formed by an AP containing a correct answer to Q along with a supporting justification. We adopt the following kernel for reranking: $P_K(\langle o_1, o_2\rangle, \langle o_1, o_2\rangle) = K(o_1, o_1) + K(o_2, o_2) - K(o_1, o_2) - K(o_2, o_1)$. In our case, $o_i = \langle Q_i, AP_i\rangle$ and $o_j = \langle Q_j, AP_j\rangle$, where Q and AP are the trees defined in the previous section, $K(o_i, o_j) = TK(Q_i, Q_j) + TK(AP_i, AP_j)$ and TK is a tree kernel function. Finally, we also add $(V(o_1) - V(o_2)) \cdot (V(o_1) - V(o_2))$ to P_K, where $V(o_i)$ is a feature vector representing Q/AP pairs.

4 Feature Engineering for QA with CNNs

The architecture of our convolutional neural network for matching Q and AP pairs is presented in Fig. 2. Its main components are: (i) sentence matrices $\mathbf{s}_i \in \mathbb{R}^{d \times |i|}$ obtained by the concatenation of the word vectors $\mathbf{w}_j \in \mathbb{R}^d$ (with d being the size of the embeddings) of the corresponding words w_j from the input sentences (Q and AP) $\mathbf{s}_i$; (ii) a convolutional sentence model $f : \mathbb{R}^{d \times |i|} \rightarrow \mathbb{R}^m$ that maps the sentence matrix of an input sentence $\mathbf{s}_i$ to a fixed-size vector representations $\mathbf{x}_{s_i}$ of size m; (iii) a layer for computing the similarity between the obtained intermediate vector representations of the input sentences, using a similarity matrix $\mathbf{M} \in \mathbb{R}^{m \times m}$ – an intermediate vector representation $\mathbf{x}_{s_1}$ of a sentence $\mathbf{s}_1$ is projected to a $\tilde{\mathbf{x}}_{s_1} = \mathbf{x}_{s_1}\mathbf{M}$, which is then matched with $\mathbf{x}_{s_2}$ (Bordes et al., 2014a), i.e., by computing a dot-product $\tilde{\mathbf{x}}_{s_1}\mathbf{x}_{s_2}$, thus resulting in a single similarity score x_{sim}; (iv) a set of fully-connected hidden layers that model the similarity between sentences using their vector representations produced by the sentence model (also integrating the single similarity score from the previous layer); and (v) a sigmoid layer that outputs probability scores reflecting how well the Q-AP pairs match with each other.

The choice of the sentence model plays a crucial role as the resulting intermediate representations of the input sentences will affect the successive step of computing their similarity. Recently, convolutional sentence models, where $f(\mathbf{s})$ is represented by a sequence of convolutional-pooling feature maps, have shown state-of-the-art results on many NLP tasks, e.g., (Kalchbrenner et al., 2014; Kim, 2014). In this paper, we opt for a convolutional operation followed by a k-max pooling layer with $k = 1$ as proposed in (Severyn and Moschitti, 2015).

Considering recent applications of deep learning models to the problem of matching sentences, our network is most similar to the models in (Hu et al., 2014) applied for computing sentence similarity and in (Yu et al., 2014) (answer sentence selection in QA) with the following difference. To compute the similarity between the vector representation of the input sentences, our network uses two methods: (i) computing the similarity score obtained using a similarity matrix $\mathbf{M}$ (explored in (Yu et al., 2014)), and (ii) directly modelling interactions between intermediate vector representations of the input sentences via fully-connected hidden layers (used by (Hu et al., 2014)). This approach, as proposed in (Severyn and Moschitti, 2015), results in a significant improvement in the task of question answer selection over the two methods used separately. Differently from the above models we do not add additional features in the join layer.

4.1 Representation Layers

It should be noted that NNs non-linearly transform the input at each layer. For instance, the output of the convolutional and pooling operation $f(s_i)$ is a fixed-size representation of the input sentence s_i. In the reminder of the paper, we will refer to these vector representations for the question and the answer passage as the question embedding (QE) and the answer embedding (AE), respectively. Similarly, the output of the penultimate layer of the network (the hidden layer whose output is fed to the final classification layer) is a compact representation of the input Question and Answer pair, which we call Joint

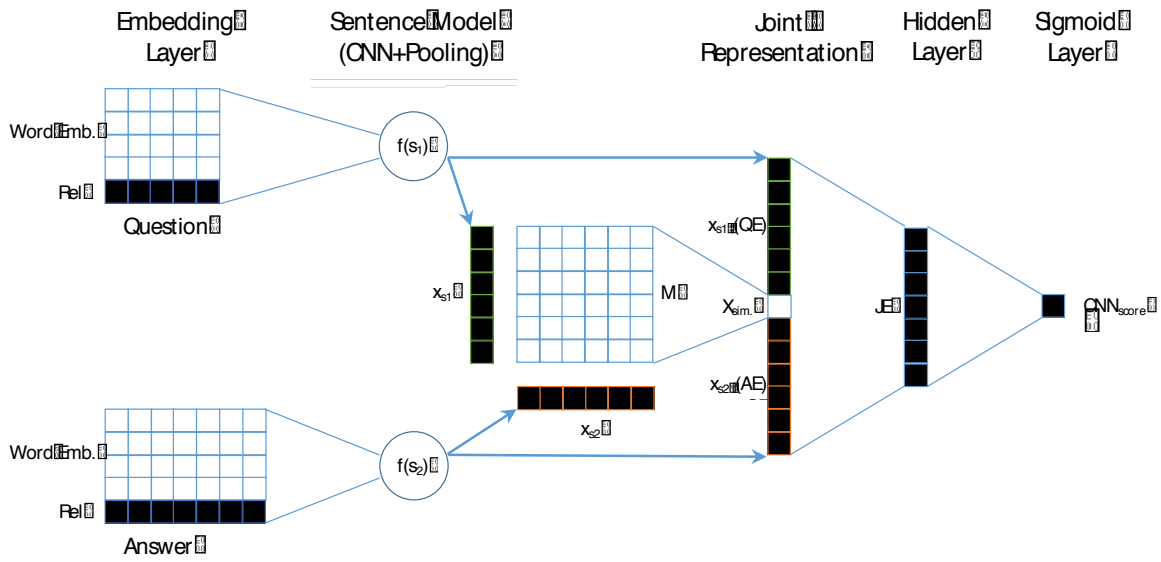

Figure 2: CNN for computing the similarity between question and answer.

Embedding (JE).

4.2 Injecting Relational Information in CNNs

Sec. 3 has shown that establishing relational links (REL nodes) between Q and A pairs is very important for solving the QA task. Yih et al. (2013b) also use latent word-alignment structure in their semantic similarity model to compute similarity between question and answer sentences. Yu et al. (2014) achieve large improvement by combining the output of their deep learning model with word count features in a logistic regression model. Differently from (Yu et al., 2014; Severyn and Moschitti, 2015) we do not add additional features such as the word count in the join layer. We allow our convolutional neural network to capture the connections between related words in a pair and we feed it with an additional binary-like input about overlapping words (Severyn and Moschitti, 2016).

In particular, in the input sentence, we associate an additional word overlap indicator feature $o \in \{0, 1\}$ with each word w, where 1 corresponds to words that overlap in a given pair and 0 otherwise. To decide if the words overlap, we perform string matching. Basically this small feature vector plays the role of REL tag added to the CTK structures.

Hence, we require an additional lookup table layer for the word overlap features $LT_{W_o}(\cdot)$ with parameters $W_o \in R^{d_o \times 2}$, where $d_o \in N$ is a hyper-parameter of the model, which indicates the number of dimensions used for encoding the word overlap features. Thus, we augment word embeddings with additional dimensions that encode the fact that a given word in a pair is overlapping or semantically similar and let the network learn its optimal representation. Given a word w_i, its final word embedding $w_i \in R^d$ (where $d = d_w + d_o$) is obtained by concatenating the output of two lookup table operations $LT_W(w_i)$ and $LT_{W_o}(w_i)$.

5 Experiments

In these experiments, we compare the impact in accuracy of two main methods for automatic feature engineering, i.e., CTKs and CNNs, for relational learning, using two different answer sentence selection datasets, WikiQA and TREC13. We propose several strategies to combine CNNs with CTKs and we show that the two approaches are complementary as their joint use significantly boosts both models.

5.1 Experimental Setup

We utilized two datasets for testing our models:

TREC13. This is the factoid open-domain TREC QA corpus prepared by (Wang et al., 2007). The training data was assembled from the 1,229 TREC8-12 questions. The answers for the training questions were automatically marked in sentences by applying regular expressions, therefore the dataset can be

noisy. The test data contains 68 questions, whose answers were manually annotated. We used 10 answer passages for each question for training our classifiers and all the answer passages available for each question for testing.

WikiQA. TREC13 is a small dataset with an even smaller test set, which makes the system evaluation rather unstable, i.e., a small difference in parameters and models can produce very different results. Moreover, as pointed by (Yih et al., 2013b), it has significant lexical overlap between questions and answer candidates, therefore simple lexical match models may likely outperform more elaborate methods if trained and tested on it. WikiQA dataset (Yang et al., 2015) is a larger dataset, created for open domain QA, which overcomes these problems. Its questions were sampled from the Bing query logs and candidate answers were extracted from the summary paragraphs of the associated Wikipedia pages. The train, test, and development sets contain 2,118, 633 and 296 questions, respectively. There is no correct answer sentence for 1,245 training, 170 development and 390 test questions. Consistently with (Yin et al., 2015), we remove the questions without answers for our evaluations.

Preprocessing. We used the Illinois chunker (Punyakanok and Roth, 2001), question class and focus classifiers trained as in (Severyn and Moschitti, 2013) and the Stanford CoreNLP (Manning et al., 2014) toolkit for the needed preprocessing.

CTKs. We used SVM-light-TK[2] to train our models. The toolkit enables the use of structural kernels (Moschitti, 2006) in SVM-light (Joachims, 2002). We applied (i) the partial tree kernel (PTK) with its default parameters to all our structures and (ii) the polynomial kernel of degree 3 on all feature vectors we generate.

Metaclassifier. We used the scikit[3] logistic regression classifier implementation to train the metaclassifier on the outputs of CTKs and CNNs.

CNNs. We pre-initialize the word embeddings by running the `word2vec` tool (Mikolov et al., 2013) on the English Wikipedia dump and the jacana corpus as in (Severyn and Moschitti, 2015). We opt for a skipgram model with window size 5 and filtering words with frequency less than 5. The dimensionality of the embeddings is set to 50. The input sentences are mapped to fixed-sized vectors by computing the average of their word embeddings. We use a single non-linear hidden layer (with hyperbolic tangent activation, Tanh), whose size is equal to the size of the previous layer. The network is trained using SGD with shuffled mini-batches using the Adam update rule (Kingma and Ba, 2014). The batch size is set to 100 examples. The network is trained for a fixed number of epochs (i.e., 3) for all the experiments. We decided to avoid using early stopping, in order to do not overfit the development set and have a fair comparison with the CTKs models.

QA metrics. We used common QA metrics: Precision at rank 1 (P@1), i.e., the percentage of questions with a correct answer ranked at the first position, the Mean Reciprocal Rank (MRR) and the Mean Average Precision (MAP).

	MRR	MAP	P@1
State of the art			
CNN_c (Yang et al., 2015)	66.52	65.20	n/a
ABCNN (Yin et al., 2015)	71.27	69.14	n/a
$LSTM_{a,c}$ (Miao et al., 2015)	70.41	68.55	n/a
$NASM_c$ (Miao et al., 2015)	70.69	68.86	n/a
Our Individual Models			
CNN_R	71.07	69.51	57.20
CH_{coarse}	71.63	70.45	56.79
CH	72.30	71.25	58.44
V_{AE+QE}	68.29	67.24	55.56
V_{JE}	67.07	65.76	52.26
Our Model Combinations			
CH+V_{AE+QE}	72.51	71.29	59.26
CH+V_{JE}	73.18	71.56	60.49
*CH+V_{AE+QE}	**75.88**	**74.17**	**64.61**
*CH+V_{JE}	75.52	73.99	63.79
Meta: CH, V_{JE}, CNN_R	**75.28**	**73.69**	**62.96**
Meta: CH, V_{JE}	75.08	73.64	62.55
Meta: CH+V_{JE}, CNN_R	73.94	72.25	61.73

Table 1: Performance on the WikiQA dataset

5.2 Experiments on WikiQA

State of the art. Table 1 reports the results obtained on the WikiQA test set by state-of-the-art systems (lines 1-4) and our models, when removing the questions with no correct answers (this to be aligned with previous work). More in detail:

CNN_c is the Convolutional Neural Network with word count,

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

[3] http://scikit-learn.org/stable/index.html

	TRAIN50			DEV		
	MRR	MAP	P@1	MRR	MAP	P@1
CH	69.97	68.77	55.14	67.23	65.93	51.44
V_{AE+QE}	68.70	67.18	54.32	68.14	66.46	54.73
V_{JE}	70.43	68.67	57.61	68.90	67.14	55.56
Model Combinations						
CH+V_{AE+QE}	**74.40**	**72.63**	**62.55**	**70.01**	**68.60**	57.61
CH+V_{JE}	73.53	71.69	60.49	**70.10**	**68.55**	**58.44**
Metaclassifiers:						
CH, V_{JE}, CNN_R	74.01	72.31	62.14	n/a	n/a	n/a
CH, V_{JE}	73.95	72.15	62.14	n/a	n/a	n/a
CH+V_{JE}, CNN_R	73.43	71.58	60.49	n/a	n/a	n/a

Table 2: Performance on the WikiQA using the development set or half of the training set for training

	TRAIN			TRAIN50		
	MRR	MAP	P@1	MRR	MAP	P@1
CH	74.87	74.17	63.49	71.31	70.45	57.94
V_{AE+QE}	70.32	69.75	56.35	71.06	70.33	57.14
V_{JE}	69.86	69.24	55.56	71.11	70.43	57.14
CH+V_{AE+QE}	71.29	70.79	57.94	72.62	72.18	59.52
CH+V_{JE}	71.36	70.81	57.94	71.96	71.55	59.52
*CH+V_{AE+QE}	**76.66**	**75.50**	**66.67**	**75.23**	**74.54**	**64.29**

Table 3: Performance on the WikiQA on the development set

ABCNN is the Attention-Based CNN,
LSTM$_{a,c}$ is the long short-term memory network with attention and word count, and
NASM$_c$ is the neural answer selection model with word count.
CNN$_R$ is the relational CNN described in Section 4.
CH[4] is a tree kernel-based SVM reranker trained on the shallow pos-chunk tree representations of question and answer sentences (Sec. 3.1), where the subscript $_{coarse}$ refers to the model with the coarse-grained question classes as in (Tymoshenko and Moschitti, 2015).
V is a polynomial SVM reranker, where the subscripts AE, QE, JE indicate the use of the answer, question or joint embeddings (see Sec. 4.1) as the feature vector of SVM and + means that two embeddings were concatenated into a single vector.

The results show that our CNN_R model performs comparably to ABCNN (Yin et al., 2015), which is the most recent and accurate NN model and to CH_{coarse}. The performance drops when the embeddings AE, QE and JE are used in a polynomial SVM reranker. In contrast, CH (using our tree structure enriched with fine-grained categories) outperforms all the models, showing the importance of syntactic relational information for the answer sentence selection task.

5.2.1 Combining CNN with CTK on WikiQA

We experiment with two ways of combining CTK with CNN_R: (i) at the kernel level, i.e., summing tree kernels with the polynomial kernel over different embeddings, i.e., CH+V, and (ii) using the predictions of SVM and CNN_R models (computed on the development set) as features to train logistic regression meta-classifiers (again only on the development set). These are reported in the last three lines of Table 1, where the name of the classifiers participating with their outputs are illustrated as a comma-separated list. The results are very interesting as all kinds of combinations largely outperform the state of the art, e.g., by around 3 points in terms of MRR, 2 points in terms of MAP and 5 points in terms of P@1 with respect to the strongest standalone system, CH. Directly using the predictions of the CNN_R as features in the meta-classifier does not impact the overall performance. It should be noted that the meta-classifier could only be trained on the

[4]Models marked by * use an improved version of the preference ranking framework we described in Section 3.2. It is important to show such results as they provide a referring baseline for future research in this field.

development data to avoid predictions biased by the training data.

5.2.2 Using less training data

Since we train the weights of CNN_R on the training set of WikiQA, to obtain the embeddings minimizing the loss function, we risk to have overfitted, i.e., "biased", JE, AE and QE on the questions and answers of the training set. Therefore, we conducted another set of experiments to study this case. We randomly split the training set into two equal subsets. We train CNN_R on one of them and in the other subset, (referred to as **TRAIN50**) we produce the embeddings of questions and answers.

Table 2 reports the results on the WikiQA test set which we obtained when training SVM on TRAIN50 and on the development set, **DEV**. We trained the meta-classifier on the predictions of the standalone models on DEV. Consistently with the previous results, we obtain the best performance combining the CNN_R embeddings with CTK. Even when we train on the 50% of the training data only, we still outperform the state of the art, and our best model CH+V_{JE} performs only around 2 points lower in terms of MRR, MAP and P@1 than when training on the full training set.

Finally, Table 3 reports the performance of our models when tested on the development set and demonstrates that the improvement obtained when combining CTK and CNN_R embeddings also holds on it. Note, that we did not use the development set for any parameter tuning and we train all the models with the default parameters.

5.3 Experiments on TREC13 dataset

TREC13 corpus has been used for evaluation in a number of works starting from 2007. Table 4 reports our as well as some state-of-the-art system results on TREC13. It should be noted that, to be consistent with the previous work, we evaluated our models in the same setting as (Wang et al., 2007; Yih et al., 2013a), i.e., we (i) remove the questions having only correct or only incorrect answer sentence candidates and (ii) used the same evaluation script and the gold judgment file as they used. As pointed out by Footnote 7 in (Yih et al., 2014), the evaluation script always considers 4 questions to be answered incorrectly thus penalizing the overall system score.

We note that our models, i.e., CNN_R, V_{JE}, $V_{\text{AE+QE}}$, again align with the state of the art. In contrast, our CTK using CH largely outperforms all previous work, e.g., 7.6 points more than CNN_R in terms of MRR. Considering that the evaluation of CH with a script that does not penalize systems would show real MRR and MAP of 90.56 and 80.08, respectively, there is little room for improvement with combinations. Indeed, the table shows no improvement of model combinations over CH.

Models	MRR	MAP
State of the art		
Wang et al. (2007)	68.52	60.29
Heilman and Smith (2010)	69.17	60.91
Wang and Manning (2010)	69.51	59.51
Yao et al. (2013)	74.77	63.07
Severyn and Moschitti (2013)	73.58	67.81
Yih et al. (2013a)	77.00	70.92
Yu et al. (2014)	78.64	71.13
Wang and Ittycheriah (2015)	77.40	70.63
Tymoshenko and Moschitti (2015)	82.29	73.34
Yang et al. (2015)	76.33	69.51
Miao et al. (2015)	81.17	73.39
Individual Models		
CNN_R	77.93	71.09
$\text{V}_{\text{AE+QE}}$	79.32	73.37
V_{JE}	77.24	71.34
CH	**85.53**	**75.18**
Model Combinations		
CH+V_{JE}	79.75	74.29
CH+$\text{V}_{\text{AE+QE}}$	79.74	75.06
Meta: CH, $\text{V}_{\text{AE+QE}}$, CNN_R	81.67	75.77
Model Combinations using simpler CH		
CH_{smpl}	78.66	71.18
CH_{smpl}+$\text{V}_{\text{AE+QE}}$	80.19	75.01
CH_{smpl}+V_{JE}	80.42	74.16

Table 4: Results on the TREC13, answer selection task.

Therefore, we trained a simplified version of CH, CH_{smpl}, which employs shallow chunk-based representations without the question focus or question class information, i.e., only using the basic relational information represented by the lexical match **REL** tags. CH_{smpl} performs comparably to CNN_R, and the combination with embeddings produced by CNN_R, i.e., CH_{smpl}+$V_{\text{AE+QE}}$, outperforms both CH_{smpl} and CNN_R.

6 Discussion

The main focus and novelty of this paper is comparing and combining CTKs and CNETs. We showed that the features they generate are complementary

as their combination improve both models. For the combinations, we used voting and our new method of combining network layers embedded in a polynomial kernels added to tree kernels.

We would like to stress that to the best of our knowledge we are the first to merge CNNs and CTK together. We showed that kernels based on different embedding layers learned with our CNNs, when used in SVMs, deliver the same accuracy of CNNs. This enables an effective combination between TK and CNNs at kernel level. Indeed, we experimented with different kernel combinations built on top of different CNN layers, improving the state of the art, largely outperforming all previous systems exactly using the same testing conditions. These results are important for developing future research as they provide indications on features/methods and referring baselines to compare with.

Finally, we generated modified structures and used better parsers outperforming our initial result in (Severyn and Moschitti, 2013) by more than 10 points.

6.1 Efficiency

An interesting question is the practical use of our models, which require the discussion of their efficiency. In this respect, our framework combines CTKs and CNNs by generating a global kernel. Thus, the time complexity during training is basically given by (i) training CNNs, (ii) extracting their embeddings and (iii) use these embeddings during the CTK training. The time for computing steps (i) and (ii) is linear with respect to the number of examples as the architecture and the number of optimization steps are fixed. In practice, the bottleneck of training our CNN architecture is in the number of weights.

Regarding Step (iii), since the embeddings just feed a polynomial kernel, which is slightly more efficient than CTKs, the overall complexity is dominated by the one of the CTK framework, i.e., $O(n^2)$. In practice, this is rather efficient, e.g., see the discussion in (Tymoshenko and Moschitti, 2015). The testing complexity is reduced to the number of kernel operations between the support vectors and the test examples (the worst case is $O(n^2)$), which are also parallelizable.

7 Conclusions

This paper compares two state-of-the-art feature engineering approaches, namely CTKs and CNNs, on the very complex task of answer reranking in a QA setting. In order to have a meaningful comparison, we have set the best configuration for CTK by defining and implementing innovative linguistic structures enriched with semantic information from statistical classifiers (i.e., question and focus classifiers). At the same time, we have developed powerful CNNs, which can embed relational information in their representations.

We tested our models for answer passage reranking in QA on two benchmarks, WikiQA and TREC13. Thus, they are directly comparable with many systems from previous work. The results show that our models outperform the state of the art achieved by more complex networks.

In particular, CTKs outperform our CNNs but use more information, e.g., on TREC 13, CTKs obtain an MRR and MAP of 85.53 and 75.18 vs. 77.93 and 71.09 of CNNs. On WikiQA, CNNs combined with tree kernels achieves an MRR of 75.88 and an MAP of 74.17 largely outperforming the current state of the art, i.e., MRR of 71.27 and MAP 69.14 of ABCNN by Yin et al. (2015).

It should be noted that CTK models use syntactic parsing, two statistical classifiers for focus and question classification and a named entity recognizer whereas CNNs only use words and two additional unsupervised corpora.

In the future, we would like to embed CNN similarity in CTKs. A straightforward methods for achieving this is to use the Smoothed Partial Tree Kernel by Croce et al. (2011). Our preliminary experiments using word2vec were not successful. However, CNNs may provide a more effective similarity. Finally, it would be also very interesting to exploit structural kernels in the network layers.

Acknowledgements

This work has been partially supported by the EC project CogNet, 671625 (H2020-ICT-2014-2, Research and Innovation action) and by an IBM Faculty Award. Many thanks to the anonymous reviewers for their valuable suggestions.

References

Antoine Bordes, Sumit Chopra, and Jason Weston. 2014a. Question answering with subgraph embeddings. In EMNLP.

Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014b. Open question answering with weakly supervised embedding models. In ECML.

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

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Chris Welty. 2010. Building watson: An overview of the deepqa project. AI Magazine, 31(3).

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), pages 1003–1013, Beijing, China, July. Association for Computational Linguistics.

Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In NAACL.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In NIPS.

Mohit Iyyer, Jordan Boyd-Graber, Leonardo Claudino, Richard Socher, and Hal Daumé III. 2014. A neural network for factoid question answering over paragraphs. In EMNLP.

Thorsten Joachims. 2002. Optimizing search engines using clickthrough data. In Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '02, pages 133–142, New York, NY, USA. ACM.

Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, June.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. Doha, Qatar.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Xin Li and Dan Roth. 2002. Learning question classifiers. In COLING.

Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In NIPS.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pages 55–60.

Yishu Miao, Lei Yu, and Phil Blunsom. 2015. Neural variational inference for text processing. arXiv preprint arXiv:1511.06038.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In Advances in Neural Information Processing Systems 26, pages 3111–3119.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. 2007. Exploiting syntactic and shallow semantic kernels for question/answer classification. In ACL.

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

Alessandro Moschitti. 2008. Kernel Methods, Syntax and Semantics for Relational Text Categorization. In Proceeding of ACM 17th Conf. on Information and Knowledge Management (CIKM'08), Napa Valley, CA, USA.

V. Punyakanok and D. Roth. 2001. The use of classifiers in sequential inference. In NIPS, pages 995–1001.

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.

Aliaksei Severyn and Alessandro Moschitti. 2015. 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, pages 373–382. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. In Preprint arXiv:1604.01178.

Aliaksei Severyn, Massimo Nicosia, and Alessandro Moschitti. 2013. Learning adaptable patterns for passage reranking. CoNLL-2013, page 75.

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, pages 1451–1460. ACM.

Kateryna Tymoshenko, Alessandro Moschitti, and Aliaksei Severyn. 2014. Encoding semantic resources in syntactic structures for passage reranking. In Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, pages 664–672, Gothenburg, Sweden, April. Association for Computational Linguistics.

Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. arXiv preprint arXiv:1507.02628.

Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answering. In ACL.

Mengqiu Wang, Noah A Smith, and Teruko Mitamura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In EMNLP-CoNLL.

Yi Yang, Wen-tau Yih, and Christopher Meek. 2015. Wikiqa: A challenge dataset for open-domain question answering. In Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 2013–2018, Lisbon, Portugal, September. Association for Computational Linguistics.

Xuchen Yao, Benjamin Van Durme, Peter Clark, and Chris Callison-Burch. 2013. Answer extraction as sequence tagging with tree edit distance. In NAACL.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013a. Question answering using enhanced lexical semantic models. In Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1744–1753, Sofia, Bulgaria, August. Association for Computational Linguistics.

Wen-Tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013b. Question answering using enhanced lexical semantic models. In ACL, August.

Wen-Tau Yih, Xiaodong He, and Christopher Meek. 2014. Semantic parsing for single-relation question answering. In ACL.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2015. Abcnn: Attention-based convolutional neural network for modeling sentence pairs. arXiv preprint arXiv:1512.05193.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep Learning for Answer Sentence Selection. In NIPS Deep Learning Workshop, December.

F. M. Zanzotto and A. Moschitti. 2006. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In The Joint 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics (COLING-ACL), Sydney, Australia. Association for Computational Linguistics.