Cross-Pair Text Representations for Answer Sentence Selection

Kateryna Tymoshenko
DISI, University of Trento (Adeptmind scholar)
38123 Povo (TN), Italy
kateryna.tymoshenko@unitn.it

Alessandro Moschitti*
Amazon
Manhattan Beach, CA 90266, USA
amosch@amazon.com

Abstract

High-level semantics tasks, e.g., paraphrasing, textual entailment or question answering, involve modeling of text pairs. Before the emergence of neural networks, this has been mostly performed using intra-pair features, which incorporate similarity scores or rewrite rules computed between the members within the same pair. In this paper, we compute scalar products between vectors representing similarity between members of different pairs, in place of simply using a single vector for each pair. This allows us to obtain a representation specific to any pair of pairs, which delivers the state of the art in answer sentence selection. Most importantly, our approach can outperform much more complex algorithms based on neural networks.

1 Introduction

Answer sentence selection (AS) is an important subtask of open-domain Question Answering (QA). Its input are a question Q and a set of candidate answer passages $A = \{A_1, A_2, ..., A_N\}$, which may, for example, be the output of a search engine. The objective consists in selecting A_i, $i \in \{1, ..., N\}$ that contain correct answers.

Pre-deep learning renaissance approaches to AS typically addressed the task by modeling Q-to-A (**intra-pair**) similarities (Yih et al., 2013; Wang et al., 2007; Heilman and Smith, 2010; Wang and Manning, 2010). Q-to-A similarity and alignment are indeed crucial, but, in practice, it is very difficult to automatically extract meaningful relations between Q and A. For example, consider two positive Q/A pairs in Table 1. If we want to learn a model based only on the intra-pair Q-to-A matches, simple lexical matching (marked with italics) will not be enough. One would need to conduct more complex processing and identify that *movie* and *film* are synonyms, and that the n-gram *play in the movie* or *be in the movie* can be paraphrased as *star*. While the former can be easily detected using an external lexical resource, e.g., WordNet (Fellbaum, 1998), the latter would require more complex inference.

On the other hand, Q_1 and Q_2 contain the same pattern **who ... in the movie ...**, and their respective answers contain **film ... starring ...**. If we know that $P_1 = (Q_1, A_1)$ is a positive AS example and want to classify $P_2 = (Q_2, A_2)$, then high Q_2-to-Q_1 and A_2-to-A_1 **cross-pair** similarities can suggest that P_1 and P_2 are likely to have the same label. This idea, for example, was exploited by Severyn and Moschitti (2012), whose system measures syntactic-semantic similarities directly between structural syntactic tree representations of Q_1/Q_2 and A_1/A_2. This model still exhibited state-of-the-art performance in 2016 (Tymoshenko et al., 2016a).

Deep neural networks (DNNs) also naturally use such cross-pair similarity when modeling two input texts, and then further combine it with intra-pair similarity, for example, by means of attention mechanisms (Shen et al., 2017), compare-aggregate architectures (Bian et al., 2017; Wang and Jiang, 2017), or fully-connected layers (Severyn and Moschitti, 2015, 2016; Rao et al., 2016).

In this work, we observe that: (i) the high accuracy of the kernel model by Severyn and Moschitti (2012) was due not only to the use of syntactic structures, but also to the use of cross-pair similarities; and (ii) the success of DNNs for QA can be partially attributed to an implicit combination of cross- and intra-pair similarity.

More specifically, we investigate, whether simple similarity metrics, e.g., cosine similarity between standard vector representations, can perform competitively to the state-of-the-art neural models when employed as cross-pair kernels.

*Professor at the University of Trento.

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 2162–2173
Brussels, Belgium, October 31 - November 4, 2018.

	Question		Answer	Label
Q_1	**who** plays ***mary poppins*** **in the movie**?	A_1	*Mary Poppins* is a 1964 musical **film** **starring** Julie Andrews, Dick Van Dyke, David Tomlinson, and Glynis Johns, produced by Walt Disney, and based on the *Mary Poppins* books series by P. L. Travers.	TRUE
Q_2	**WHO** WAS **IN THE MOVIE** *I CONFESS* WITH *MONTGOMERY CLIFT*	A_2	*I Confess* is a 1953 drama **film** directed by Alfred Hitchcock, and **starring** *Montgomery Clift* as Fr. Michael William Logan, a Catholic priest, Anne Baxter as Ruth Grandfort, and Karl Malden as Inspector Larrue .	TRUE

Table 1: Question/Answer Sentence pairs from WikiQA corpus. We use italic font to mark intra-pair lexical matches between Q_1 and A_1, Q_2 and A_2, and bold font to mark the cross-pair matches between Q_1 and Q_2, A_1 and A_2.

To this end, we apply linear and cosine kernels to Q_i/Q_j and A_i/A_j pairs ($i, j = 1, ..., N$) represented as a bag-of-words (BoW) or an averaged sum of their pretrained word embeddings. Then, we combine them with the cross-pair Tree Kernels (TKs) and kernels applied to the traditional Q/A intra-pair similarity feature vector representations in a composite kernel and use it in an SVM model.

We experiment with three reference datasets, WikiQA (Yang et al., 2015), TREC13 (Yao et al., 2013; Wang et al., 2007) and SemEval-2016, Task 3.A (Nakov et al., 2016), using a number of lexical-overlap/syntactic kernels. The latter challenge refers to a community question answering (cQA) task. It consists in reranking the responses to user questions from online forums. It is the same setting as AS, but the text of questions and answer sentences can be ungrammatical due to the nature of the online forum language.

We obtain competitive results on WikiQA and SemEval tasks, showing that: (i) simple BoW representations, when used in cross-pair kernels, perform comparably to and even outperform hand-crafted intra-pair features. (ii) In cQA, simple cross-pair embedding- and BoW-based similarity features outperform domain-specific similarity features, which are hand-crafted from intra-pair members. The simple features also perform comparably to syntactic TKs. (iii) We show that a combination of simple cosine- intra- and cross-pair kernels with TKs can outperform the most recent state-of-the-art DNN architectures.

Assuming the conjecture of our paper correct, *cross-pair modeling is the major neural network contribution*, the last point above is not surprising as on relatively small datasets kernels-based models can exploit syntactic information very effectively while neural models cannot.

The paper is structured as follows. We describe the kernels incorporating intra- and cross-pair matches in Sec. 3.2, list the simple cross- and intra-pair features in Sec. 3.3, describe strong hand-crafted baseline features in Sec. 4, and report the experimental results in Sec. 5.

2 Related work

Early approaches to AS typically focused on modeling intra-pair Q-to-A alignment similarities. For example, Yih et al. (2013) proposed a latent alignment model that employed lexical-semantical Q-to-A alignments, Wang et al. (2007) modeled syntactic alignments with probabilistic quasi-synchronous grammar, and Heilman and Smith (2010); Yao et al. (2013); Wang and Manning (2010) employed Tree Edit Distance-based Q-to-A alignments.

Originally, the idea of cross-pair similarity was proposed by Zanzotto and Moschitti (2006) and applied to the recognizing textual entailment task, which consists in detecting whether a text T entails a hypothesis H. They assumed that if two H/T pairs $\langle H_1, T_1 \rangle$ and $\langle H_2, T_2 \rangle$ share the same T-to-H "rewrite rules", they are likely to share the same label. Based on this idea, they proposed an algorithm applying TKs to (H_1, H_2) and (T_1, T_2) syntactic tree representations, enriched with H-to-T intra-pair rewrite rule information. More concretely, such algorithm aligns the constituents of H with T and then marks them with symbols directly in the trees. This way the alignment information can be matched by tree kernels applied to cross-pair members.

Then, a line of work on AS, started by Severyn and Moschitti (2012, 2013); Severyn et al. (2013), was inspired by a similar idea of incorporating "rewrite rules" directly into the tree representations of Q_1/A_1 and Q_2/A_2. They represent Q and A as syntactic trees enhanced with Q-to-A relational information, and apply TKs (Moschitti, 2006) to (Q_1, Q_2) and (A_1, A_2). Thus they model **cross-pair** similarity, and learn important patterns occurring in Q and A separately. As shown in (Tymoshenko et al., 2016a), this approach is competitive with convolutional neural networks (CNNs).

In our approach, instead of using only one TK, we employ a number of different word-based kernels, most of which can be computed more efficiently than TKs.

Most recent AS models are based on Deep Neural Networks (DNNs), which learn distributed representations of the input data. DNNs are trained to apply series of non-linear transformations to the input Q and A, represented as compositions of word or character embeddings. DNN architectures learn AS-relevant patterns using intra-pair similarities as well as cross-pair, Q-to-Q and A-to-A, similarities, when modeling the input texts. For example, the CNN network by (Severyn and Moschitti, 2015) has two separate embedding layers for Q and A, which are followed by the respective convolution layers, whose output is concatenated and then passed through the final fully-connected joint layer. The weights in the Q and A convolution layers are learned by means of the backpropagation algorithm on the training Q/A pairs. Thus, obviously, classifying a new Q/A pair is partially equivalent to performing the implicit cross-pair Q-to-Q and A-to-A comparison.

Additionally, the DNN approaches model the *Q-to-A relatedness* explicitly in a variety of ways, e.g., by: (i) using a Q-to-A transformation matrix and simple Q-to-A similarity features (Yu et al., 2014; Severyn and Moschitti, 2015), (ii) relying on RNN and LSTM architectures (Wang and Nyberg, 2015; Shen et al., 2017), (iii) employing attention components (Yin et al., 2016; Shen et al., 2017; Wang et al., 2016a), (iv) decomposing input into similarity and dissimilarity matches (Wang et al., 2016b) or (v) using the compare-aggregate method (Wang and Jiang, 2017; Bian et al., 2017).

We believe that the ability of DNNs to implicitly capture cross-pair relational matching, i.e., the capacity of learning from (Q_1, Q_2) and (A_1, A_2), is a very important factor to their high performance. This is of course paired with their ability to learn non-linear patterns and capture Q-to-A relatedness by means of attention mechanisms. It should be noted that the latter are typically hard-coded in kernel models as lexical matching/similarity (Severyn and Moschitti, 2012). This is effective as much as the attention approach, at least with standard-size dataset, also in neural models (Severyn and Moschitti, 2016).

In our work, we model Q-to-A, Q-to-Q and A-to-A similarities with intra- and cross-pair kernels and show that such combination also exhibits state-of-the-art performance on the reference corpora. In addition, our approach can be applied to smaller datasets as it utilizes less parameters, and can provide insights on future DNN design.

3 Cross-pair similarity kernels for text

3.1 Background on Kernel Machines

Kernel Machines (KMs) allow for replacing the dot product with kernel functions directly applied to examples, i.e., they avoid mapping examples into vectors. The main advantage of KMs is a much lower computational complexity than the dot product as the kernel computation does not depend on the size of the feature space.

KMs are linear classifiers: given a labeled training dataset $S = \{(\mathbf{x}_i, y_i) : i = 1, \ldots, n\}$, their classification function can be defined as:

$$f(\mathbf{x}) = \mathbf{w} \cdot \mathbf{x} + b = \sum_{i=1}^{n} \alpha_i y_i \mathbf{x}_i \cdot \mathbf{x} + b.$$

where $\mathbf{x}$ is a classification example, $\mathbf{w}$ is the gradient of the separating hyperplane, and b its bias. The equation shows that the gradient is a linear combination of the training points $\mathbf{x}_i \in \mathbb{R}^n$ multiplied by their labels $y_i \in \{-1, 1\}$ and their weights $\alpha_i \in \mathbb{R}^+$. Note that the latter are different from zero only for the support vectors: this reduces the classification complexity, which will be lower than O(n) for each example.

We can replace the scalar product with a kernel function directly defined over a pair of objects, $K(o_i, o) = \phi(o_i)\phi(o)$, where $\phi : O \rightarrow \mathbb{R}^n$ maps from objects to vectors of the final feature space. The new classification function becomes: $f(o) = \sum_{i=1}^{n} \alpha_i y_i K(o_i, o) + b$, which only needs the initial input objects.

3.2 Inter- and intra-pair match kernel

We cast AS as a text pair classification task: given a pair, $P = (Q, A)$, constituted by a question (Q) and a candidate answer sentence (A), we classify it as either correct or incorrect. We used KMs, where $K(\cdot, \cdot)$ operates on two pairs, $P_1 = (Q_1, A_1)$ and $P_2 = (Q_2, A_2)$.

3.2.1 Intra-pair similarity

A traditional baseline approach would (i) represent Q/A pairs as feature vectors, where the components are similarity metrics applied to Q and A, e.g., a world overlap-based similarity; and (ii)

train a classification model, e.g., an SVM using the following kernel:

$$K_{IP}(P_1, P_2) = K_v\left(V_{\langle Q_1,A_1\rangle}, V_{\langle Q_2,A_2\rangle}\right), \quad (1)$$

where K_v can be any kernel operating on the feature vectors, e.g., the polynomial or linear (as in our work) kernel. $V_{\langle T_1,T_2\rangle}$ is a vector built on N similarity features, $\langle f_1(\cdot,\cdot), f_2(\cdot,\cdot), ..., f_N(\cdot,\cdot)\rangle$, extracted by applying similarity metrics to two texts, T_1 and T_2 (see Sec. 3.3 for the list of the similarity metrics we used). K_{IP} merely uses intra-pair similarities.

3.2.2 Cross-pair similarity

We incorporate the intuition, *similar questions are likely to demand similar answer patterns*, by means of a cross-pair kernel, which measures similarity between questions and answers from P_1 and P_2 as follows:

$$\begin{aligned} K_{CP}(P_1, P_2) &= V_{\langle Q_1,Q_2\rangle} \cdot V_{\langle A_1,A_2\rangle} \\ &= \sum_{i=1}^{N} f_i(Q_1, Q_2) \cdot f_i(A_1, A_2) \end{aligned} \quad (2)$$

K_{CP} measures P_1-to-P_2 similarity in terms of a sum of the products of Q_1-to-Q_2 and A_1-to-A_2 similarities. Note, that within K_{IP}, $f_i(Q_i, A_i)$ is merely an i-th feature in the $V_{\langle Q_i,A_i\rangle}$ feature vector. At the same time, within K_{CP}, $f_i(\cdot,\cdot)$ becomes a kernel, which takes the (Q_1, Q_2) or (A_1, A_2) pairs as input. In other words, $V_{\langle Q_1,Q_2\rangle} \cdot V_{\langle A_1,A_2\rangle}$ is a sum of products of $f_i(\cdot,\cdot)$ kernels applied to the (Q_1, Q_2) and (A_1, A_2) pairs. K_{CP} is a valid kernel if the similarity metrics used to compute the $f_i(\cdot,\cdot)$ are valid kernel functions.

Finally, combining K_{IP} and K_{CP} enables learning of two different kinds of valuable cross- and intra-pair AS patterns. We combine various K_{IP} and K_{CP} by summing them or by training a meta-classifier on their outputs. See Section 5.4 for more details. Figure 1 summarizes the differences between the K_{IP} and K_{CP} computation processes described above.

3.3 Similarity features

We employ three similarity feature types as $f_i(\cdot,\cdot)$. Two of them are computed using the cosine similarity metrics and differ only in terms of the input texts, T_1 and T_2, representations. The other type is constituted by TKs applied to the structural representations of T_1 and T_2. Note that, since cosine similarity and TKs are valid kernels,

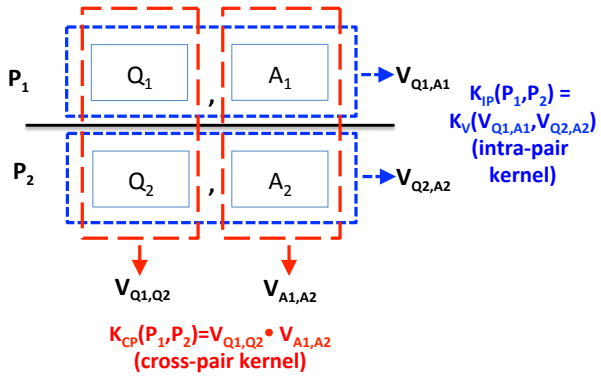

Figure 1: Feature extraction schema for two Q/A pairs, P_1 and P_2. V_{T_1,T_2} is a vector of similarity features extracted for a pair of texts T_1, T_2, the respective dashed boxes show from which pair of input texts they are extracted.

K_{CP} is also guaranteed to be a valid kernel when computed using these similarity features.

3.3.1 Bag-of-n-grams overlap (B)

$f_B^{t,l,s}(T_1, T_2)$ is a cosine similarity metric applied to the bag-of-n-grams vector representations of T_i, $BOW_{\{t,l,s\}}(T_i)$, $i = 1, 2$. The $\{t, l, s\}$ index describes an n-gram representation configuration: t denotes whether the n-grams are assembled of word lemmas (L), or their part-of-speech tags (POS), or lemmas concatenated with their respective POS-tags (L_{POS}); l is a (n_1, n_2) tuple, with n_1 and n_2 being the minimal and maximal length of n-grams considered, respectively; and s is YES if the representation discards the stopwords and NO, otherwise.

We used $\{t, l, s\}$ configurations from the following set: $C = (\{L, L_{POS}\} \times \{(1,2),(1,3),(1,4),(2,4),(2,3)\} \times \{YES, NO\}) \cup (\{POS\} \times \{(1,4),(2,4)\} \times \{YES\})$. It follows that $|C| = 23$, which means we have 23 similarity features, $f_B^{t,l,s}(T_1, T_2)$, in total, in the intra-pair setting. The respective cross-pair kernels are a composite kernel summing 23 products of cosine kernels applied to 23 different (Q_1, Q_2) and (A_1, A_2) bag-of-ngram representations.

3.3.2 Embedding-based similarities (E).

We represent an input text as an average of embeddings of its lemmas from pre-trained word embedding models. Then, the embedding feature $f_{model}^{E}(T_1, T_2)$ is the cosine kernel applied to the embedding-based representations of T_1 and T_2. We use two pretrained embeddings: Word2Vec (Mikolov et al., 2013) and

GloVe (Pennington et al., 2014), resulting in three[1] embedding-based features (see Sec. 5 for more technical details).

3.3.3 Tree-kernel based similarities

Following the framework defined in (Severyn et al., 2013; Tymoshenko et al., 2016a), we represent T_1 and T_2 as syntactico-semantic structures and use TKs as semantic similarity metrics. When computing K_{CP} with TK as similarities, in Eq.2, we employ summation instead of multiplication[2].

More specifically, we represent T_1 and T_2 as (i) constituency trees and apply subset TK (SST); or (ii) shallow chunk-based trees, similar to the one presented in Figure 2, and apply partial tree (PTK) kernel. In the shallow trees, lemmas are leaves and POS tags are pre-terminals. POS nodes are grouped under chunk nodes, and then under the sentences nodes. These representations encode also some intra-pair similarity information, e.g., prefix `REL` denotes the lexical Q-to-A match. In a structural representation, we prepend it to the parent and grand-parent nodes of lemmas which occur both in Q and A, e.g., "Mary" in the first example of Table 1.

Then, for factoid QA[3], we mark focus words in Q and entities in A, if the answer contains any named entities of types matching the question expected answer type (EAT)[4]. More specifically, we mark the semantic Q-to-A match by prepending the `REL-FOCUS-<EAT>` label to the answer chunk nodes that contain such named entities and also to the question focus word. Here, `<EAT>` stands for the EAT label. For example, in the Q_1/A_1 pair in Table 1, the Q_1 EAT is HUMan, and the matching named entities include "Julie Andrews", "David Tomlinson" and others. Figure 2 depicts Q_1 annotated both with `REL-` and `REL-FOCUS` links. We detect both question focus and EAT automatically. Due to the space limitations, we do not describe the structural representations and matching algorithms in more detail, but refer the reader to the works above.

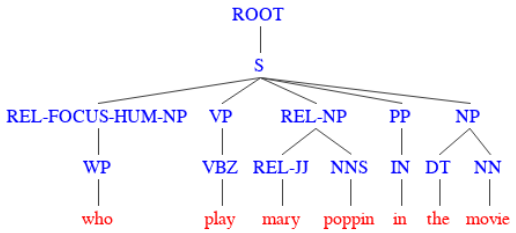

Figure 2: Shallow syntactic representation of A_1 from the running example in Table 1

(i) **cosine similarity** applied to the BoW representations of T_1 and T_2 in terms of word lemmas, bi-, three-, four-grams (computed twice with and without stopwords); POS-tags; dependency triplets;
(ii) **longest common string subsequence** measure w. and w/out stopwords;
(iii) **Jaccard similarity metric** applied to one-, two-, four, three-grams w. and w/out stopwords;
(iv) **word n-gram containment measure** on uni- and bi-grams w. and w/out stopwords (Broder, 1997);
(v) **greedy string tiling** (Wise, 1996) with minimum matching length of 3 ;
(vi) **string kernel similarity** (Lodhi et al., 2002);
(vii) **expected answer type match:** percentage of named entities (NE) in the answer passage compatible with the question class[5];
(viii) **WordNet-based similarity.** WordNet T_1/T_2 common lemma/synonym/hypernym overlap ratio;
(ix) **PTK (Moschitti, 2006) similarity** between constituency or dependency tree representations of input texts;

Table 2: Strong baseline fatures

4 Strong baseline feature vector

As a strong baseline, we use similarity feature vectors and intra-pair K_{IP} kernel.

For the **factoid answer sentence selection** task, we use 47 strong features listed in Table 2. This is a compilation of features used in the top-performing system at SemEval-2012 Semantic Text Similarity workshop (Bär et al., 2012) and earlier factoid QA work (Severyn and Moschitti, 2012), extended with few additional features.

For the **community question answering (cQA)** task, we employ instead a combination of similarity-based and thread-level features shown to be very effective for the cQA task (Nicosia et al., 2015; Barrón-Cedeño et al., 2016). We use the exact feature combination from (Barrón-Cedeño et al., 2016), which includes both lexical and syntactic similarity measures (cosine similarity of bag-of-words, PTK similarity over syntactic tree representations of the input texts) and thread-

[1] We use Word2Vec embeddings trained on two different corpora, which result in two features, and GloVe trained on one corpus.

[2] We have opted to use summation in this case to follow the earlier work.

[3] WikiQA and TREC13 are the factoid AS datasets, as their questions ask for a specific fact, e.g. date or a name.

[4] For example, the PERson named entity type matches the HUMan EAT. More specifically, we employ the following NER-to-EAT matching rules: PERson, ORGanization → HUMan; LOCation → LOCation; DATE, TIME, MONEY, PERCENTAGE, DURATION, NUMBER, SET → NUM; ORGanization, PERson, MISCellanious → ENTiTY. We employ the (Li and Roth, 2002) coarse-grained EAT taxonomy and Stanford CoreNLP (Manning et al., 2014) entity types.

MODE	TRAIN		DEV		TEST	
	Q	A	Q	A	Q	A
raw	2118	20360	296	2733	633	6165
no all$^-$	873	8672	126	1130	243	2351
clean	857	8651	121	1126	237	2341

Table 3: WikiQA corpus statistics

level domain specific features (*are the question and comment authored by the same person?*, *does the comment contain any questions?*, and so on.).

We cannot directly use these feature vectors in the K_{CP} kernels, as not all functions used to compute features are valid kernels, e.g., the longest common string subsequence is not a kernel function. Moreover, some of them can be computed only on the (Q, A) pairs, e.g., the expected type match feature (vii) in Tab. 2, or many of the cQA domain-specific features.

5 Experiments

We conduct experiments on three corpora, namely TREC13, WikiQA and SemEval, and evaluate the results in terms of Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). Our code is available at https://github.com/iKernels/RelTextRank.

5.1 Datasets

WikiQA dataset. WikiQA (Yang et al., 2015) is a factoid answer sentence selection dataset with Bing query logs as questions. Candidate answer sentences are extracted from Wikipedia and labeled manually. Some of the questions have no correct answer sentence (all$^-$) or have only correct answer sentences (all$^+$). Table 3 reports the statistics of the WikiQA corpus as distributed (raw), without all$^-$ questions, and without both all$^-$ and all$^+$ questions (clean). We train in the "no all$^-$" mode using 10 answer sentences per question[6] and test in the "clean" mode.

TREC13 dataset. A factoid answer sentence selection dataset originally presented in (Wang et al., 2007)[7], also frequently called **QASent** (Yang et al., 2015). We train on 1,229 automatically labeled TREC8-12 questions. We use only 10 candidate answer sentences per question. We test in the "clean" setting defined in (Rao et al., 2016), i.e., we discard the all$^+$ and all$^-$ questions, resulting in 65 DEV and 68 TEST questions, respectively. DEV and TEST contain 1117 and 1442 candidate associated answer sentences, respectively.

SemEval-2016, Task 3.A dataset. SemEval cQA dataset is a benchmark dataset in the SemEval 2016 Task 3. A question-to-comment similarity competition. It is a collection of user questions and the respective answer comment threads from Qatar Living forum, where the user comments to questions were manually labeled as correct or incorrect. Each question has 10 respective candidate answers. The training, dev and test sets have 1790, 244 and 327 questions, respectively. The AS task consists in reranking comments with respect to the question: most questions are non-factoid and the text is often noisy.

5.2 Models

We used the following notation:

B, **E**. Intra-pair K_{IP} kernels (see Sec. 3.2) using the eponymous similarity features from Sec. 3.3.

V. Linear kernel applied to the strong intra-pair feature vector representation defined in Section 4. Note that, as already mentioned in Sec. 4, due to the slightly different nature of the factoid and community question answering tasks, we used different strong feature groups for WikiQA, TREC13 (Table 2) and SemEval-2016.

B$_{cr}$, **E**$_{cr}$. Cross-pair K_{CP} kernel applied to B and E similarity feature vectors respectively. More specifically, B$_{cr}$ and E$_{cr}$ are a sum of 23 and 3 cross-pair kernel products, respectively (see Eq. 2 and Sec. 3.2).

PTK, **SST** are the cross-pair PTK and SST tree kernels applied to the shallow chunk- and constituency-based representations (see Sec. 3.3).

"+" denotes kernel summation. We use this symbol to denote that we sum the gram-matrices for the distinct standalone kernels, and use the resulting kernel matrix as input to SVMs.

META$_{BASE;PTK}$, **META**$_{BASE;SST}$. Logistic regression metaclassifiers trained on the outputs of two standalone systems, namely (i) V+B$_{cr}$+E$_{cr}$+E (we denote it as $BASE$ to simplify the notation), and (ii) PTK or SST, respectively. We ran 10-fold cross-validation on the training set and used the resulting predictions as training data for the ensemble classifier. We did not use the development or training sets for any parameter tuning, thus we report the results both on the DEV and TEST sets.

SUM$_{BASE;PTK}$, **SUM**$_{BASE;SST}$. Simple metaclassifiers, summing the output of the BASE and

[6] The 10 answer sentences per question limit speeds up training time without loss in performance

[7] We use the version distributed by (Yao et al., 2013) in https://code.google.com/p/jacana/

PTK or SST systems, respectively.

5.3 Toolkits

We trained the models using *scikit-learn*[8] by Pedregosa et al. (2012) using the SVC version of SVM with precomputed K_{IP} and K_{CP} kernel matrices and default parameters. We trained the ensemble model using the scikit *LogisticRegression* classifier implementation with the default parameters. We used spaCy library[9] and scikit to obtain bag-of-n-gram representations for the B similarity features, and to compute B- and E- base gram matrices.

We used the RelTextRank framework[10] (Tymoshenko et al., 2017b) to generate the structural representations for the TK similarity features and to extract the strong baseline feature vectors from Sec. 4. We used KeLP (Filice et al.) to compute the TK gram matrices.

Regarding the Embedding-based similarities (E), we obtain three similarity features by using three word embedding models to generate the representations of the input texts, T_1 and T_2, namely GloVe vectors trained on common crawl data[11], Word2Vec vectors pre-trained on Google News[12], and another Word2Vec vectors model[13] pre-trained on Aquaint[14] plus Wikipedia.

5.4 Results and discussion

Table 4 reports the results obtained with the intra- and cross-pair kernels K_{IP}, K_{CP} and their combinations. In the following, we describe the results according to the model categories above.

Intra-pair kernels. Taking into account intra-pair similarity is the standard approach in the majority of the previous non-DNN work. In our experiments, we implement this approach as K_{IP} using B, E, V groups of similarity features. K_{IP} performs worse than the state-of-art (SoA) DNN systems on all the datasets (see tables 5, 6 and 7, for the SoA systems).

The results on WikiQA are particularly low even when the best K_{IP} system, B+V+E, is used, which scores up to 15 points less than the state of the art. This confirms the Yang et al. (2015) observation on WikiQA, according to which, simple word matching methods are likely to underperform on its data, considering how it was built. Nevertheless, despite its simplicity, B+V+E performs comparably to the Yang et al. (2015) reimplementation of LCLR, the complex latent structured approach employing rich lexical and semantic intra-pair similarity features (Yih et al., 2013). Yang et al. (2015) report that on WikiQA LCLR obtains MRR of 60.86 and MAP of 59.93.

Then, on TREC13 and SemEval-2016, the intra-pair *V*, *V+E* and *B+V+E* kernels exhibit rather high performance, however, they are still significantly below the state of the art, thus confirming our hypothesis that intra-pair similarity alone does not guarantee top results.

Cross-pair kernels. B_{cr} and E_{cr} obtain rather high results on WikiQA and SemEval. On WikiQA, both B_{cr} and $B_{cr} + E_{cr}$ outperform all the intra-pair kernels by a large margin, while, on SemEval, they perform comparably to the manually engineered domain-specific V features of Nicosia et al. (2015). On the contrary, on TREC13, V outperforms both B_{cr} and E_{cr}, thus showing that TREC13 is indeed biased towards intra-pair relatedness features by construction.

More complex PTK and SST cross-pair kernels, both alone and combined with B_{cr}, E_{cr}, typically outperform the standalone B_{cr} and E_{cr} on all the corpora (PTK on TREC13 and WikiQA, and SST and B_{cr}+E_{cr}+PTK on SemEval). This can be explained by the fact that PTK and SST are able to learn complex syntactic patterns and also contain some information about intra-pair relations, namely `REL-` labels described in Sec. 3.2. Thus, it is natural that they outperform simpler cross-pair kernels. Nevertheless, on WikiQA-DEV, B_{cr}+E_{cr} performs very close to PTK. Moreover, on SemEval, B_{cr}+E_{cr} outperforms PTK and is behind SST for less than 1 point in terms of MAP. This can be explained by the fact that Q and A, in SemEval, are frequently ungrammatical as the cQA corpus is collected from online forums.

Finally, note that the B_{cr}+E_{cr}+PTK system, which does not use any cQA domain-specific features, is only 0.56 MAP points behind KeLP, the best-performing system in the SemEval competition (see Line 1 of Table 7).

Kernels combining the intra- and cross-pair similarities. The V+B_{cr}+E_{cr}+E combination (we

[8] http://scikit-learn.org/

[9] https://spacy.io/

[10] This tool employs Stanford CoreNLP 3.6.0 (Manning et al., 2014) for text processing; and DKProSimilarity (Bär et al., 2013) to extract features (ii)-(v).

[11] http://nlp.stanford.edu/data/glove.42B.300d.zip

[12] https://code.google.com/archive/p/word2vec/

[13] https://github.com/aseveryn/deep-qa

[14] https://catalog.ldc.upenn.edu/LDC2002T31

		TREC13				WikiQA				Semeval-2016			
		DEV		TEST		DEV		TEST		DEV		TEST	
		MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP	MRR	MAP
Intra-pair kernel (K_{IP})	B	61.91	57.69	65.68	55.63	51.36	51.41	54.69	53.8	57.90	51.45	65.70	57.28
	V	85.38	78.01	78.82	69.18	47.07	46.53	50.98	50.26	69.75	63.68	81.94	73.89
	E	71.15	67.88	73.27	66.48	48.25	48.49	53.14	52.42	61.50	54.25	67.05	60.25
	V+E	**88.48**	**79.64**	**79.63**	**69.57**	49.24	49.08	53.32	52.87	**70.98**	**64.43**	**82.06**	**74.13**
	B+V+E	84.84	77.31	79.22	68.83	**55.47**	**55.75**	**60.19**	**59.52**	**70.98**	64.37	80.84	74.09
Cross-pair kernel (K_{CP})	B_{cr}	72.64	65.81	63.70	56.16	72.71	72.00	68.84	67.67	72.28	65.20	81.63	74.68
	E_{cr}	72.22	66.36	76.94	66.14	64.61	64.38	60.77	59.42	70.07	63.92	80.60	73.62
	PTK	**89.49**	**81.02**	84.09	76.06	75.56	74.27	**75.60**	**74.67**	70.45	63.04	81.96	74.30
	SST	77.52	68.8	78.73	70.91	72.38	71.78	69.49	68.1	73.77	66.00	82.37	75.65
	B_{cr}+E_{cr}	71.52	66.35	69.24	62.16	74.62	74.18	70.03	68.76	71.56	65.26	81.81	75.06
	B_{cr}+E_{cr}+PTK	84.74	79.24	**85.07**	**77.18**	75.34	74.91	71.01	69.61	**75.99**	**68.82**	**85.26**	**78.63**
	B_{cr}+E_{cr}+SST	73.80	67.69	72.43	65.75	**75.90**	**75.59**	71.23	69.72	73.38	66.46	82.82	75.91
Combination	V+E_{cr}	86.92	78.34	**85.65**	76.72	71.97	71.77	67.63	66.40	75.83	68.75	84.37	77.60
	V+B_{cr}	88.85	80.51	81.99	74.46	75.84	75.22	74.59	73.22	75.42	68.06	85.78	78.47
	V+PTK	87.44	80.38	84.25	75.74	75.19	74.81	72.78	71.70	74.15	67.83	**86.68**	79.22
	V+SST	86.41	78.58	84.12	76.86	72.96	72.74	71.24	69.74	75.29	68.71	85.69	79.07
	V+B_{cr}+E_{cr}+E	**90.51**	**82.94**	82.40	75.98	76.65	76.29	**74.68**	**73.34**	76.07	69.75	85.54	78.83
	V+B_{cr}+E_{cr}+E+PTK	89.10	81.23	83.14	**76.77**	**76.88**	**76.59**	74.05	72.71	**76.60**	**69.97**	86.52	**79.79**
	V+B_{cr}+E_{cr}+E+SST	90.56	82.95	84.24	76.63	76.73	76.11	74.08	72.73	76.00	69.89	86.23	79.43
Ensemble	$SUM_{BASE;SST}$	89.41	80.02	84.07	77.35	77.12	76.74	74.07	72.52	76.49	69.50	85.45	78.92
	$SUM_{BASE;PTK}$	88.33	81.37	**86.89**	**77.65**	**77.94**	**77.43**	**77.00**	**75.59**	74.60	67.69	84.94	78.65
	$META_{BASE;PTK}$	88.26	81.26	**86.20**	**77.53**	**77.61**	**77.29**	**76.01**	**74.63**	75.90	69.94	85.34	79.08
	$META_{BASE;SST}$	**90.77**	**82.51**	83.33	77.05	76.61	76.20	75.01	73.54	**76.80**	**70.49**	**85.78**	**79.46**

Table 4: Results on TREC13, WikiQA and SemEval-2016 datasets. Best results in each feature category are highlighted with bold, overall best results are underlined. TK is SST for Semeval and PTK for WikiQA and TREC13. BASE refers to the V+B_{cr}+E_{cr}+E configuration.

will refer to it as *BASE*), outperforms the standalone domain-specific handcrafted cQA features, V, and both PTK and SST on SemEval 2016 TEST and DEV by at least 2.3 points in all metrics.

Moreover, V+B_{cr}+E_{cr}+E is only less than 0.5 points behind the #1 system of the SemEval-2016 competition (see Tab. 7). We recall that V+B_{cr}+E_{cr}+E only uses basic n-gram overlap-based cross- and intra- similarity features and embedding-based cosine similarities.

Finally, when we add tree kernel models to the combination, i.e., V+B_{cr}+E_{cr}+E+PTK or V+B_{cr}+E_{cr}+E+SST, we note improvement for SemEval and TREC13 tasks.

Ensemble models. We ensemble cross- and intra-pair kernels-based models by summing the predictions of the standalone SVM classifiers (SUM models) or by training a logistic regression meta-classifier on them (META models). We build the meta-classifiers on the outputs of the standalone system *BASE* and TKs, namely PTK and SST. The "Ensemble" section of Table 4 shows that meta-system combinations mostly outperform the standalone kernels.

In general, combining cross-pair and intra-pair similarities (with kernel sum or meta-classifiers) provides state-of-the-art results without using deep learning. Additionally, the outcome is deterministic, while the DNN accuracy may vary depending on the type of the hardware used or the random initialization parameters (Crane, 2018).

5.5 Comparison with the state of the art

Tables 5, 6 and 7 report the performance of the most recent state-of-the-art systems on WikiQA, TREC13 and SemEval in comparison with our best results. We discuss them with respect to the different datasets.

WikiQA. As already mentioned earlier, WikiQA contains many questions without correct answer (see Tab. 3). When evaluated on the full data, even the oracle system will achieve at most 38.38 points of MAP. Moreover, as originally observed in (Wang et al., 2007), the questions that do not have either correct answers or incorrect answers are not useful for comparing the performance of different answer sentence selection systems. Therefore, they are typically removed from WikiQA and TREC13 before the evaluation.

There has been some discrepancy in the community when evaluating on WikiQA. The original baselines proposed for the corpus in (Yang et al., 2015) were evaluated in the "clean" setting[15]. We

[15] According to the WikiQA gold reference files at https://www.microsoft.com/en-us/download/details.aspx?id=52419

	no all$^-$		clean	
	MRR	**MAP**	**MRR**	**MAP**
LCLR (Yang et al., 2015)	*61.83*	*60.92*	60.86	59.93
impl. of (Yih et al., 2013)				
HybridTK-NN (Tymoshenko et al., 2017a)	*74.72*	*72.88*	74.08	72.19
IWAN-att (Shen et al., 2017)	75.00	73.30	*74.37*	*72.62*
C/A, MULT (Wang and Jiang, 2017)	75.45	74.33	*74.83*	*73.68*
C/A, k-max (Bian et al., 2017)	76.40	75.40	*75.80*	*74.78*
C/A, listwise (Bian et al., 2017)	75.90	74.60	*75.29*	*73.96*
HyperQA (Tay et al., 2018)	72.70	71.20	*72.01*	*70.47*
Our model (PTK)	76.21	75.29	75.60	74.67
Our model (SUM$_{BASE;PTK}$)	77.57	76.19	**77.00**	**75.59**

Table 5: Comparison to the SoA on WikiQA

	MRR	**MAP**
Noise-contrastive estim. (Rao et al., 2016)	87.7	80.1
IWAN-att (Shen et al., 2017)	88.9	**82.2**
BIMPM (Wang et al., 2017)	87.5	80.2
C/A, k-threshold (Bian et al., 2017)	**89.9**	82.1
C/A-listwise (Bian et al., 2017)	**88.9**	81.0
HyperQA (Tay et al., 2018)	86.5	78.4
Our model (B$_{cr}$+E$_{cr}$+PTK)	85.07	77.18
Our model (V+E$_{cr}$)	85.65	76.72
Our model (SUM$_{BASE;PTK}$)	**86.89**	**77.65**

Table 6: Comparison to the SoA on TREC13

	MRR	**MAP**
Kelp [#1] (Filice et al., 2016)	86.42	79.19
Conv-KN [#2] (Barrón-Cedeño et al., 2016)	84.93	77.6
CTK$_C$+V$_{QF}$ (Tymoshenko et al., 2016b)	86.26	78.78
HyperQA (Tay et al., 2018)	n/a	79.5
AI-CNN (Zhang et al., 2017)	n/a	80.14
Our model (V+B$_{cr}$+E$_{cr}$+E+SST)	**86.52**	**79.79**

Table 7: Comparison to the SoA on SemEval-2016

also evaluate in the "clean" setting. However, the performance of the most recent state-of-the-art systems listed in the Tab. 5 is reported in the "no all$^-$" setting, in the respective papers, i.e., they keep the all$^+$ questions[16]. Thus, they have 6 extra questions always answered correctly by default. To account for this discrepancy, in Tab. 5, we report the results in both settings. It is trivial to convert the performance figures from one setting to another. In the table, we mark the conversion results with italic.

Our SUM$_{BASE;PTK}$ system (i) outperforms all the state-of-the-art systems, including the sophisticated architectures with attention, such as IWAN-att (Shen et al., 2017), and compare-aggregate (C/A) frameworks (Wang and Jiang, 2017; Bian et al., 2017) in terms of MRR; and (ii) has the same MAP as (Bian et al., 2017). Obviously, this improvement is not statistically significant with respect to C/A systems by Bian et al. (2017). Nevertheless, ours is a very promising result, considering that we only use linear models with simple kernels and do not tune any learning parameter of such models.

TREC13. As shown in Tab. 6, our models do not outperform the state of the art on TREC13, but they still perform comparably to the recent DNN HyperQA model (Tay et al., 2018). In general, our model is behind the state-of-the-art IWAN-att system by 4.55 points in terms of MAP. Note, however, that TREC13 test set contains only 68 questions, therefore this difference in performance is not likely to be statistically significant[17].

Semeval. Table 7 compares performance of B_{cr} + E_{cr} + V + E + SST system on Semeval to that of KeLP and ConvKN, the two top systems in the SemEval 2016 competition, and also to the performance of the recent DNN-based HyperQA and AI-CNN systems. In the Semeval 2016 competition, our model would have been the first[18], with #1 KeLP system being 0.6 MAP points behind. Then, it would have outperformed the state-of-the-art AI-CNN system by 0.35 MAP points.

6 Conclusions

This work proposes a simple, yet effective approach to the task of answer sentence selection based on the intuition that similar patterns in questions are likely to demand similar patterns in answers. We showed that this hypothesis provides an improvement on three benchmark datasets, WikiQA, TREC13, Semeval-2016, and, moreover, it enables simple features to achieve the state of the art on WikiQA and Semeval-2016, outperforming many of state-of-the-art DNN-based systems. There is significant room for further elaboration of this approach, for example, by expanding feature spaces with more syntactic and semantic features, employing new types of kernels for measuring the inter-question/answer pair similarity or trying to implement the same idea in DNN architectures.

Acknowledgments

The first author is supported by Adeptmind. Many thanks to the Area chairs, PC chairs and anonymous reviewers for their professional work and valuable suggestions.

[16] We deduced that from the corpus statistics reported by the authors of the papers. They all report having 243 test questions, which corresponds to the "no all$^-$" setting

[17] We cannot conduct the test since we do not have the outputs of such systems. However, 4 points more correspond to just a couple of correct questions more.

[18] Ranking available in (Nakov et al., 2016)

References

D Bär, C Biemann, I Gurevych, and T Zesch. 2012. Ukp: Computing semantic textual similarity by combining multiple content similarity measures. *First Joint Conference on Lexical and Computational Semantics (*SEM 2012)*, pages 435–440.

Daniel Bär, Torsten Zesch, and Iryna Gurevych. 2013. DKPro Similarity: An Open Source Framework for Text Similarity. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 121–126, Sofia, Bulgaria. Association for Computational Linguistics.

A. Barrón-Cedeño, D. Bonadiman, G. Da San Martino, S. Joty, A. Moschitti, F.A. Al Obaidli, S. Romeo, K. Tymoshenko, and A. Uva. 2016. ConvKN at SemEval-2016 task 3: Answer and question selection for question answering on Arabic and English fora. In *SemEval 2016 - 10th International Workshop on Semantic Evaluation, Proceedings*.

Weijie Bian, Si Li, Zhao Yang, Guang Chen, and Zhiqing Lin. 2017. A Compare-Aggregate Model with Dynamic-Clip Attention for Answer Selection. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 1987–1990, New York, NY, USA. ACM.

Andrei Z Broder. 1997. On the resemblance and containment of documents. In *Compression and Complexity of Sequences 1997. Proceedings*, pages 21–29. IEEE.

Matt Crane. 2018. Questionable Answers in Question Answering Research: Reproducibility and Variability of Published Results. *Transactions of the Association for Computational Linguistics*, 6:241–252.

Christiane Fellbaum. 1998. *WordNet: An Electronic Lexical Database*, volume 71.

Simone Filice, Giuseppe Castellucci, Danilo Croce, Giovanni Da San Martino, Alessandro Moschitti, and Roberto Basili. KeLP: a Kernel-based Learning Platform in Java, year = 2015. In *The workshop on Machine Learning Open Source Software (MLOSS): Open Ecosystems*, Lille, France. International Conference of Machine Learning.

Simone Filice, Danilo Croce, Alessandro Moschitti, and Roberto Basili. 2016. KeLP at SemEval-2016 Task 3: Learning Semantic Relations between Questions and Answers. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 1116–1123.

Michael Heilman and Noah a. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. *The 2010 Annual Conference of the North American Chapter of the ACL*, pages 1011–1019.

Xin Li and Dan Roth. 2002. Learning question classifiers. In *COLING 2002: The 19th International Conference on Computational Linguistics*.

Huma Lodhi, Craig Saunders, John Shawe-Taylor, Nello Cristianini, and Chris Watkins. 2002. Text Classification using String Kernels. *Journal of Machine Learning Research*, 2:419–444.

Christopher D Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J Bethard, and David McClosky. 2014. The Stanford CoreNLP Natural Language Processing Toolkit. In *ACL*.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 1–9.

Alessandro Moschitti. 2006. Efficient Convolution Kernels for Dependency and Constituent Syntactic Trees. *Machine Learning ECML 2006*, 4212:318–329.

Preslav Nakov, Lluís Màrquez, Alessandro Moschitti, Walid Magdy, Hamdy Mubarak, abed Alhakim Freihat, Jim Glass, and Bilal Randeree. 2016. SemEval-2016 Task 3: Community Question Answering. *Proceedings of the 10th International Workshop on Semantic Evaluation (SemEval-2016)*, pages 525–545.

Massimo Nicosia, Simone Filice, Alberto Barrón-Cedeno, Iman Saleh, Hamdy Mubarak, Wei Gao, Preslav Nakov, Giovanni Da San Martino, Alessandro Moschitti, Kareem Darwish, and Others. 2015. QCRI: Answer selection for community question answering-experiments for Arabic and English. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, pages 203–209.

Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. 2012. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global Vectors for Word Representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

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 - CIKM '16*, pages 1913–1916.

Aliaksei Severyn and Alessandro Moschitti. 2012. Structural relationships for large-scale learning of answer re-ranking. *Proceedings of the 35th international ACM SIGIR conference on Research and development in information retrieval - SIGIR '12*, pages 741–750.

Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic Feature Engineering for Answer Selection and Extraction. In *Proc. Empirical Methods in Natural Language Processing*, October, pages 458–467.

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*, SIGIR '15, pages 373–382, New York, NY, USA. ACM.

Aliaksei Severyn and Alessandro Moschitti. 2016. Modeling relational information in question-answer pairs with convolutional neural networks. *CoRR*, abs/1604.01178.

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

Gehui Shen, Yunlun Yang, and Zhi-Hong Deng. 2017. Inter-Weighted Alignment Network for Sentence Pair Modeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1179–1189. Association for Computational Linguistics.

Yi Tay, Luu Anh Tuan, and Siu Cheung Hui. 2018. Hyperbolic Representation Learning for Fast and Efficient Neural Question Answering. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 583–591, New York, NY, USA. ACM.

K. Tymoshenko, D. Bonadiman, and A. Moschitti. 2016a. Convolutional neural networks vs. convolution kernels: Feature engineering for answer sentence reranking. In *2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL HLT 2016 - Proceedings of the Conference*.

K. Tymoshenko, D. Bonadiman, and A. Moschitti. 2016b. Learning to rank non-factoid answers: Comment selection in Web forums. In *CIKM*.

Kateryna Tymoshenko, Daniele Bonadiman, and Alessandro Moschitti. 2017a. Ranking Kernels for Structures and Embeddings: A Hybrid Preference and Classification Model. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, Copenhagen, Denmark. Association for Computational Linguistics.

Kateryna Tymoshenko, Alessandro Moschitti, Massimo Nicosia, and Aliaksei Severyn. 2017b. RelTextRank: An Open Source Framework for Building Relational Syntactic-Semantic Text Pair Representations. In *Proceedings of ACL 2017, System Demonstrations*, pages 79–84. Association for Computational Linguistics.

Bingning Wang, Kang Liu, and Jun Zhao. 2016a. Inner Attention based Recurrent Neural Networks for Answer Selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1288–1297.

Di Wang and Eric Nyberg. 2015. A Long Short-Term Memory Model for Answer Sentence Selection in Question Answering. pages 707–712.

Mengqiu Wang and Christopher D Manning. 2010. Probabilistic Tree-Edit Models with Structured Latent Variables for Textual Entailment and Question Answering. *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, (August):1164–1172.

Mengqiu Wang, Noah A. Smith, and Teruko Mitamura. 2007. What is the Jeopardy Model? A Quasi-Synchronous Grammar for QA. *Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 22–32.

Shuohang Wang and Jing Jiang. 2017. A compare-aggregate model for matching text sequences. *ICLR*.

Zhiguo Wang, Wael Hamza, and Radu Florian. 2017. Bilateral multi-perspective matching for natural language sentences. In *IJCAI International Joint Conference on Artificial Intelligence*, pages 4144–4150.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016b. Sentence Similarity Learning by Lexical Decomposition and Composition. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1340–1349, Osaka, Japan. The COLING 2016 Organizing Committee.

Michael J Wise. 1996. YAP3: Improved detection of similarities in computer program and other texts. *ACM SIGCSE Bulletin*, 28(1).

Yi Yang, Wen-Tau Yih, and Christopher Meek. 2015. WIKIQA: A Challenge Dataset for Open-Domain Question Answering. *Proceedings of EMNLP 2015*, pages 2013–2018.

Xuchen Yao, Benjamin Van Durme, Chris Callison-Burch, and Peter Clark. 2013. Answer Extraction as Sequence Tagging with Tree Edit Distance.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question Answering Using Enhanced Lexical Semantic Models. pages 1744–1753.

Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association for Computational Linguistics*, 4:259–272.

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

Fabio Massimo Zanzotto and Alessandro Moschitti. 2006. Automatic Learning of Textual Entailments with Cross-Pair Similarities. In *Proceedings of the 21st International Conference on Computational Linguistics and 44th Annual Meeting of the Association for Computational Linguistics*, pages 401–408, Sydney, Australia. Association for Computational Linguistics.

Xiaodong Zhang, Sujian Li, Lei Sha, and Houfeng Wang. 2017. Attentive Interactive Neural Networks for Answer Selection in Community Question Answering. *AAAI*, 24(4):505–509.