Modeling Relational Information in Question-Answer Pairs with Convolutional Neural Networks

Aliaksei Severyn[*]
Google Research
Zurich, Switzerland
severyn@google.com

Alessandro Moschitti
Qatar Computing Research Institute, HBKU
5825 Doha, Qatar
amoschitti@qf.org.qa

Abstract

In this paper, we propose convolutional neural networks for learning an optimal representation of question and answer sentences. Their main aspect is the use of relational information given by the matches between words from the two members of the pair. The matches are encoded as embeddings with additional parameters (dimensions), which are tuned by the network. These allows for better capturing interactions between questions and answers, resulting in a significant boost in accuracy. We test our models on two widely used answer sentence selection benchmarks. The results clearly show the effectiveness of our relational information, which allows our relatively simple network to approach the state of the art.

1 Introduction

Modeling text pairs to compute their semantic similarity is at the core of many NLP tasks. The most common approach is to encode them with many complex lexical, syntactic and semantic features and then compute various similarity measures between the obtained representations. Recently, it has been shown that the problem of semantic text matching can be tackled using distributional word matching, e.g., for matching questions with candidate answers (Yih et al., 2013).

Deep learning approaches generalize the distributional word matching problem to matching sentences and take it one step further by learning the optimal sentence representations for a given task. Deep neural networks are able to effectively capture the compositional process of mapping the meaning of individual words in a sentence to a continuous representation of the sentence. In particular, it has been recently shown that convolutional neural networks are able to efficiently learn to embed input sentences into low-dimensional vector space preserving important syntactic and semantic aspects of the input sentence, which leads to state-of-the-art results in many NLP tasks (Kalchbrenner et al., 2014; Kim, 2014; Yu et al., 2014).

In this paper, we capitalize on our previous work (Severyn and Moschitti, 2015) extending it with a novel deep learning architecture for modelling question-answer pairs for answer sentence reranking. The main building blocks of our architecture are two distributional sentence models based on convolutional neural networks (ConvNets). These underlying sentence models work in parallel, mapping questions and answer sentences to their distributional vectors, which are then used to learn the semantic similarity between them. To compute question-answer similarity score we adopt an approach used in the deep learning model of (Yu et al., 2014), which produces excellent results on the answer sentence selection task. However, their model only operates on unigram or bigrams, while our architecture learns to extract and compose n-grams of higher degrees, thus allowing for capturing longer range dependencies. Additionally, our architecture uses not only the intermediate representations of questions and answers to compute their similarity but also includes them in the final representation, which constitutes a much richer representation of the question-answer pairs.

The main novelty of our architecture is the way we choose to model relational information in a pair. Yu et al. (2014) combine the output of their deep learning model with additional features in the final logistic regression model. Such features count the number word overlaps between the two pair members. This provides a sort of relational information, which significantly improves the network accuracy. In contrast, our model uses a completely different approach, which injects relational information about

---

[*]This work was carried out during his PhD in the University of Trento.

matching words directly into the word embeddings as additional dimensions. The augmented word embeddings are thus passed through the layers of the convolutional feature extractors: this enables the automatic encoding of the relations between question-answer pairs in a more structured manner. Moreover, our embedding dimensions encoding matches are parameters of the network and are tuned during the training.

In summary, the distinctive properties of our model are: (i) we use a state-of-the-art distributional sentence model for learning to map input sentences to vectors, which are then used to measure the similarity between them; (ii) our model encodes question-answer pairs in a richer representation using not only their similarity score but also their intermediate representations; (iii) we augment the word embeddings with additional dimensions to encode the fact that certain words overlap in a given question-answer pair and let the network tune these parameters; (iv) the architecture of our network makes it straightforward to include any additional features encoding question-answer similarities; and finally (v) our model is trained end-to-end starting from the input sentences to producing a final score that is used to rerank answers. We only require to initialize word embeddings trained on some large unsupervised corpora. However, given a large training set the network can also optimize the embeddings directly for the task, thus omitting the need for pre-training of the word embeddings.

We test our model on a popular answer sentence selection benchmark TREC13 (Wang et al., 2007) and on the more recent dataset WikiQA (Yang et al., 2015). The results show the importance of using relational information and on WikiQA our network reaches the state of the art, i.e., an MRR of 71.07 and an MAP of 69.51.

2 Our Deep Learning Model

This section explains the architecture of our deep learning model for modelling question-answer pairs to rerank answer sentences. We treat the answer sentence selection problem as a simple binary classification where answer candidates with higher prediction scores are ranked above the ones with lower scores. More formally, each question $\mathbf{q}_i \in Q$ is associated with a list of labelled candidate answer
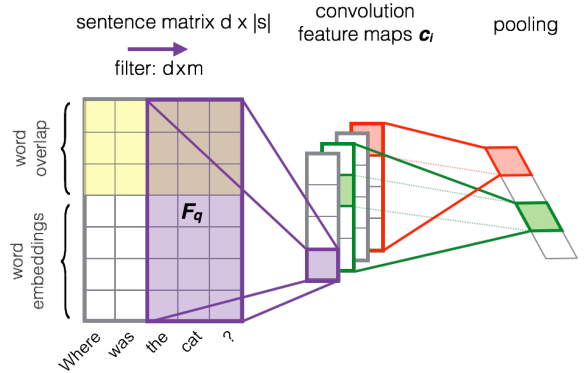
Figure 1: Our sentence model for mapping input sentences to their intermediate representations.

sentences $\{(y_{i1}, \mathbf{a}_{i1}), \ldots, (y_{in}, \mathbf{a}_{in})\}$, where labels $y_{ij} \in \{0, 1\}$ with 1 corresponding to answers that contain a correct answer and 0 otherwise. Our goal is to learn a decision function that maps each question-answer pair to a score reflecting their similarity: $f(\theta, \psi(\mathbf{q}_i, \mathbf{a}_{ij}))$, where $\psi(\cdot)$ is a function encoding question-answer pairs into a joint feature space, and θ are model parameters.

Given that we choose to model the answer sentence selection task as a binary classification, our main effort lies in designing a deep learning architecture for learning an optimal representation of question-answer pairs. Its main building blocks are two distributional *sentence models* based on convolutional neural networks. These underlying sentence models work in parallel mapping question and answer sentences to their distributional vectors, which are then used to learn the similarity between them.

In the following, we first describe our *sentence model* for mapping queries and documents to their intermediate representations and then describe how they can be used for learning semantic matching between input query-document pairs.

2.1 Distributional sentence model

The architecture of our network for mapping sentences to feature vectors is shown on Fig. 1. It is mainly inspired by the convolutional architectures used in (Kalchbrenner et al., 2014; Kim, 2014) for performing various sentence classification tasks. However, different from previous work the goal of our distributional sentence model is to learn intermediate representations of questions and answers used to compute their semantic matching.

Our sentence model is composed of a single *wide*
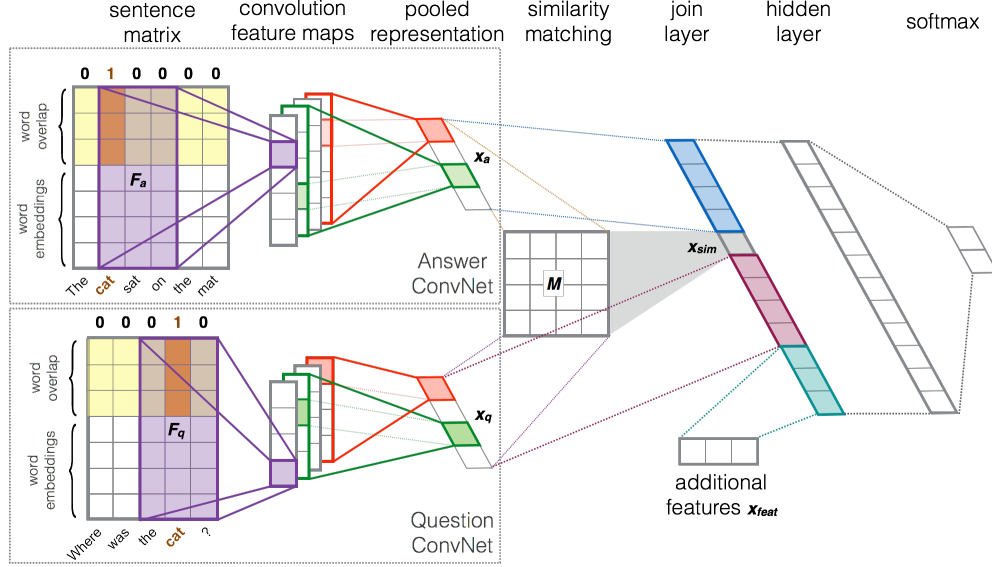
Figure 2: Our deep learning architecture for reranking question-answer pairs. The relational information in a pair is modelled by augmenting word embeddings with additional dimensions to encode overlapping words, e.g., we feed the network with additional *word overlap* indicator features whose values equal to 1 correspond to words that overlap in a pair, e.g., a non-stop word `cat`.

convolutional layer followed by a non-linearity and simple *max* pooling. In the following we give a brief explanation of its main components: sentence matrix, activations, convolutional and pooling layers.

2.1.1 Sentence matrix

The input to the network are raw words that need to be translated into real-valued feature vectors to be processed by subsequent layers of the network.

The input is a sentence **s** treated as a sequence of words: $[w_1, \ldots, w_{|s|}]$, where each word is drawn from a finite-sized vocabulary V. The architecture of neural networks is not well suited for dealing with discrete words, hence, they are represented by low-dimensional, real-valued, dense vectors $\mathbf{w} \in \mathbb{R}^{d_w}$ looked up in a matrix $\mathbf{W} \in \mathbb{R}^{d_w \times |V|}$ (whose columns correspond to words in V). The mapping from words to their word embeddings is performed by a lookup table operation $LT_{\mathbf{W}}(w_i) = \mathbf{w}_i$. Hence, for each input sentence **s** we build a sentence matrix **S** where each i-th column corresponds to a word embedding $\mathbf{w}_i$.

To learn to capture and compose features of individual words in a given sentence from low-level word embeddings into higher level semantic concepts, the neural network applies a series of transformations to the input sentence matrix **S** using convolution, non-linearity and pooling operations, which

we describe next.

2.1.2 Convolutional feature maps

The aim of the convolutional layer is to extract patterns, i.e., discriminative word sequences that are common throughout the training instances.

More formally, the convolution operation $*$ between an input matrix $\mathbf{S} \in \mathbb{R}^{d \times |s|}$ and a filter (or a convolution kernel) $\mathbf{F} \in \mathbb{R}^{d \times m}$ of width m results in a vector $\mathbf{c} \in \mathbb{R}^{|s|+m-1}$ where each component is computed as follows:

$$\mathbf{c}_i = (\mathbf{S} * \mathbf{F})_i = \sum_{k,j} (\mathbf{S}_{[:,i-m+1:i]} \otimes \mathbf{F})_{kj} \qquad (1)$$

where $\otimes$ is the element-wise multiplication and $\mathbf{S}_{[:,i-m+1:i]}$ is a matrix slice of size m along the columns. Note that the convolution filter is of the same dimensionality d as the input sentence matrix. As shown in Fig. 1, it slides along the column dimension of **S** producing a vector $\mathbf{c} \in \mathbb{R}^{|s|-m+1}$ in output. Each component c_i is the result of computing an element-wise product between a column slice of **S** and a filter matrix **F**, which is then summed to a single value.

So far we have described a way to compute a convolution between the input sentence matrix and a single filter. To form a richer representation of the data, deep learning models apply a set of filters that work in parallel generating multiple feature maps

(also shown on Fig. 1). A set of filters form a filter bank $\mathbf{F} \in \mathbb{R}^{n \times d \times m}$ sequentially convolved with the sentence matrix $\mathbf{S}$ and producing a feature map matrix $\mathbf{C} \in \mathbb{R}^{n \times (|s| - m + 1)}$.

In practice, we also need to add a bias vector $\mathbf{b} \in \mathbb{R}^n$ to the result of a convolution – a single b_i value for each feature map $\mathbf{c}_i$. This allows the network to learn an appropriate threshold.

Activation units. To enable the learning of non-linear decision boundaries, each convolutional layer is typically followed by a non-linear activation function $\alpha()$ applied element-wise. We use a rectified linear (ReLU) function defined as simply $max(0, \mathbf{x})$ in our model since, as shown in (Nair and Hinton, 2010), it speeds up the training and sometimes produces more accurate results.

Pooling. The output from the convolutional layer (passed through the activation function) are then passed to the pooling layer, whose goal is to aggregate the information and reduce the representation. We use *max* pooling in our model which simply returns the maximum value. It operates on columns of the feature map matrix $\mathbf{C}$ returning the largest value: $pool(\mathbf{c}_i) : \mathbb{R}^{|s| + m - 1} \to \mathbb{R}$ (also shown schematically in Fig. 1).

Convolutional layer passed through the activation function together with pooling layer acts as a non-linear feature extractor. Given that multiple feature maps are used in parallel to process the input, deep learning networks are able to build rich feature representations of the input.

This ends the description of our *sentence* model. In the following we present our deep learning architecture for learning to match short text pairs.

2.2 Our relational model

When learning to match text pairs, modelling the relational connections between sentences has been shown to greatly improve the accuracy of the semantic similarity models. For example, top performing systems on Semantic Textual Similarity benchmarks (Agirre et al., 2015) rely on similarity scores obtained by aligning the words and phrases between sentences in a pair. Yih et al. (2013) also uses latent word-alignment structure in their semantic similarity model to compute similarity between question and answer sentences. Yu et al. (2014) achieves large improvements by combining the output of their

deep learning model with word count features in a logistic regression model.

To allow our convolutional neural network capture the connections between related words in a pair we feed it with an additional binary-like input about overlapping words. In particular, for each word w in the input sentence we associate an additional *word overlap* indicator feature $o \in \{0, 1\}$, where 1 corresponds to words that overlap in a given pair and 0 otherwise (see Fig. 2). To decide if the words overlap, we perform string matching.

Hence, we require an additional lookup table layer for the word overlap features $LT_{\mathbf{W}_o}(\cdot)$ with parameters $\mathbf{W}_o \in \mathbb{R}^{d_o \times 2}$, where $d_o \in \mathbb{N}$ is the number of dimensions to encode word overlap features and is a hyper-parameter of the model. Effectively, we are augmenting 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 $\mathbf{w}_i \in \mathbb{R}^d$ (where $d = d_w + d_o$) is obtained by concatenating the output of two lookup table operations $LT_{\mathbf{W}}(w_i)$ and $LT_{\mathbf{W}_o}(w_i)$ (also see Fig. 2).

2.3 The matching model

The architecture of our model for matching question-answer pairs is presented in Fig. 2. Our sentence models (described in Sec. 2.1) learn to map input sentences to vectors, which can then be used to compute their similarity. These are then used to compute a similarity score, which together with the distributional vector of question and answer sentences are used in a single joint representation.

In the following we describe how the intermediate representations produced by the sentence model can be used to compute question-answer similarity scores and give a brief explanation of the remaining layers, e.g. hidden and softmax.

Similarity model. Given the output of our sentence models, their resulting vector representations $\mathbf{x}_q$ and $\mathbf{x}_a$, can be used to compute a question-answer similarity score. We follow the approach of (Bordes et al., 2014) that defines the similarity between $\mathbf{x}_q$ and $\mathbf{x}_a$ vectors as follows:

$$sim(\mathbf{x}_q, \mathbf{x}_a) = \mathbf{x}_q^T \mathbf{M} \mathbf{x}_a, \qquad (2)$$

where $\mathbf{M} \in \mathbb{R}^{d \times d}$ is a similarity matrix. Eq. 2 can be viewed as a model of the noisy channel approach

from machine translation, which has been widely used as a scoring model in information retrieval and QA (Echihabi and Marcu, 2003). In this model, we seek a transformation of the candidate document $\mathbf{x}'_a = \mathbf{M}\mathbf{x}_a$ that is the closest to the input query $\mathbf{x}_q$. The similarity matrix $\mathbf{M}$ is a parameter of the network and is optimized during the training.

Hidden and classification layers. Our model includes an additional hidden layer right before the softmax layer (described next) to allow for modeling interactions between the components of the intermediate representation. It computes the following transformation: $\alpha(\mathbf{w}_h \cdot \mathbf{x} + b)$, where $\mathbf{w}_h$ is the weight vector of the hidden layer and $\alpha()$ is non-linearity. Finally, to transform the output of the network to the probability distribution over the labels we apply a softmax function.

2.4 The information flow

The output of our sentence models (Sec. 2.1) are distributional representations of a question $\mathbf{x}_q$ and an answer $\mathbf{x}_a$, which are then matched using a similarity matrix $\mathbf{M}$ according to Eq. 2. This produces a single score x_{sim} capturing various aspects of similarity (syntactic and semantic) of the question-answer pair. Note that it is also straight-forward to add additional features $\mathbf{x}_{\text{feat}}$ to the model.

The join layer concatenates all intermediate vectors, the similarity score and any additional features into a single vector: $\mathbf{x}_{\text{join}} = [\mathbf{x}_q^T; x_{\text{sim}}; \mathbf{x}_a^T; \mathbf{x}_{\text{feat}}^T]$ This vector is then passed through a fully connected hidden layer, which allows for modelling interactions between the components of the joined representation vector. Finally, the output of the hidden layer is further fed to the softmax classification layer, which generates a distribution over the class labels.

2.5 Training

The model is trained to minimise the negative conditional log-likelihood of the training set:

$$\mathcal{C} = -\log \prod_{i=1}^{N} p(y_i | \mathbf{q}_i, \mathbf{a}_i; \theta)$$

where θ contains all the network parameters:

$$\theta = \{\mathbf{W}; \mathbf{W}_o; \mathbf{F}_q; \mathbf{b}_q; \mathbf{F}_a; \mathbf{b}_a; \mathbf{M}; \mathbf{w}_h; b_h; \mathbf{w}_s; b_s\},$$

namely the word embeddings matrix $\mathbf{W}$ and word overlap feature matrix $\mathbf{W}_o$, filter weights and biases of the convolutional layers, similarity matrix $\mathbf{M}$, parameters of the hidden and softmax layers.

The parameters of the network are optimized by stochastic gradient descent (SGD) using backpropogation algorithm to compute the gradients.

3 Experiments and Evaluation

This section describes the dataset and our experimental setup, also giving details about how we obtain the word embeddings matrix $\mathbf{W}$ and train our network.

3.1 Data and setup

We test our model on the manually curated TREC QA dataset[1] from Wang et al. (2007), which appears to be one of the most widely used benchmarks for answer sentence reranking. The set of questions are collected from TREC QA tracks 8-13. The manual judgement of candidate answer sentences is provided for the entire TREC 13 set and for the first 100 questions from TREC 8-12.

To enable a direct comparison with the previous work, we use the same `train`, `dev` and `test` sets. Table 1 summarizes the datasets used in our experiments. An additional training set TRAIN-ALL provided by Wang et al. (2007) contains 1,229 questions from the entire TREC 8-12 collection and comes with automatic judgements. This set represents a more noisy setting, nevertheless, it provides many more QA pairs for learning.

WikiQA. This is an the open domain QA dataset (Yang et al., 2015). Its questions were derived from the Bing query logs and the candidate answers were extracted from paragraphs of the associated Wikipedia pages. The training, test, and development set contain 2,118, 633 and 296 questions, respectively. Consistently with (Yin et al., 2015), we remove the questions without answers for our evaluations.

Evaluation. The two metrics used to evaluate the quality of our model are Mean Average Precision (MAP) and Mean Reciprocal Rank (MRR). We use the official `trec_eval` scorer to compute the above metrics.

Word vectors. While our model allows for learning the word embeddings directly, we keep the word matrix parameter $\mathbf{W}$ static. This is due to a common experience that a minimal size of the dataset

---

Table 1: TREC QA datasets for answer reranking.

Data	# Questions	# QA pairs	% Correct
TRAIN-ALL	1,229	53,417	12.0%
TRAIN	94	4,718	7.4%
DEV	82	1,148	19.3%
TEST	100	1,517	18.7%

required for tuning the word embeddings for a given task should be at least in the order of hundred thousands, while in our case the number of question-answer pairs is one order of magnitude smaller. Hence, similar to (Denil et al., 2014; Kim, 2014; Yu et al., 2014) we keep the word embeddings fixed and initialize the word matrix $\mathbf{W}$ from an unsupervised neural language model.

We run `word2vec` tool (Mikolov et al., 2013) on the English Wikipedia dump and the AQUAINT corpus[2] containing roughly 375 million words. We opt for a skipgram model with window size 5 and filtering words with frequency less than 5. We set the dimensionality of our word embeddings to 50 to be on the line with (Yu et al., 2014). The resulting model contains 50-dimensional vectors for about 3.5 million words. Embeddings for words not present in the `word2vec` model are randomly initialized with each component uniformly sampled.

We minimally preprocess the data only performing tokenization and lowercasing all words. To reduce the size of the resulting vocabulary V, we also replace all digits with 0. The size of the word vocabulary V for experiments using TRAIN set is 17,023 with approximately 95% of words initialized using `wor2vec` embeddings and the remaining 5% words are initialized at random. For the TRAIN-ALL setting the $|V| = 56,953$ with about 90% words found in the `word2vec` model.

Word matching features. In contrast to the word embeddings matrix, the size of the vocabulary V_o to encode word overlap features is tiny. Given such a small parameter space it is possible to tune the vectors even on small sized datasets. Hence, we keep this as a parameter optimized by our network. We set the size of the space, d_o, to 5 and randomly initialize the entries of the matrix $\mathbf{W}^o$ by sampling from the uniform distribution.

[2] `https://catalog.ldc.upenn.edu/ LDC2002T31`

3.2 Training and hyperparameters
The parameters of our deep learning model were chosen on a `dev` set: the width m of the convolution filters is 5 and the number of convolutional feature maps is 100. We use ReLU activation function and a simple max-pooling. The size of the hidden layer is equal to the size of the $\mathbf{x}_{\text{join}}$ vector obtained after concatenating question and answer sentence vectors from the distributional models, similarity score and additional features.

To train the network we use stochastic gradient descent with shuffled mini-batches. We eliminate the need to tune the learning rate by using the Adadelta update rule (Zeiler, 2012). The batch size is set to 50 examples. The network is trained for 25 epochs with early stopping, i.e., we stop the training if no update to the best accuracy on the `dev` set has been made for the last 5 epochs. The accuracy computed on the `dev` set is the MAP score. At test time we use the parameters of the network obtained with the best MAP score on the `dev` set: we compute the MAP score after each 10 mini-batch updates and save the network parameters if a new best `dev` score is obtained. In practice, the training converges after a few epochs.

3.3 Results and discussion
Our goal is to evaluate the impact of using our: (i) more powerful convolutional network for sentence modeling; (ii) distributional representations of questions and answers in addition to the similarity score; and (iii) approach to model matching words by augmenting word embeddings with additional dimensions vs. providing the network with a pre-computed feature vector of overlapping word counts as in (Yu et al., 2014).

3.3.1 Distributional sentence models
Table 2 summarises the results for the setting when the network is trained using only input question-answer pairs without using any additional features, i.e., we omit the word overlap features.

First, we report the results of our model when using only a similarity score x_{sim}. It should be noted that the network by Yu et al. (2014), similarly to ours, relies on a convolutional neural network to learn intermediate representations. However, their convolutional neural network operates only on unigram or bigrams, while in our architecture we use a

Table 2: Results on TREC QA using only similarity score (Sim) and also including the distributional representation (Dist) of question and answer sentences.

	Train		Train-ALL	
	Sim	Dist	Sim	Dist
MAP	.5884	.6258	.6521	.6709
MRR	.6036	.6591	.7010	.7280

Table 3: Results on TREC QA when augmenting the deep learning model with relational information about overlapping words.

	Train		Train-ALL	
	fvec	Emb.	fvec	Emb.
MAP	.7275	.7325	.7459	.7654
MRR	.7796	.8018	.8078	.8186

Table 4: Survey of the results on the TREC QA answer selection task (after score rescaling).

Model	MAP	MRR
Wang et al. (2007)	.6029	.6852
Heilman and Smith (2010)	.6091	.6917
Wang and Manning (2010)	.5951	.6951
Yao et al. (2013)	.6307	.7477
Severyn & Moschitti (2013)	.6781	.7358
Yih et al. (2013)	.7092	.7700
Yu et al. (2014)	.7113	.7846
Wang and Ittycheriah (2015)	.7063	.7740
Yin et al. (2015)	.6951	.7633
Miao et al. (2015)	**.7339**	**.8117**
CNN_R on (TRAIN)	.6857	.7660
CNN_R on (TRAIN-ALL)	.7186	.7828

larger width of the convolution filter, thus allowing for capturing longer range dependencies.

Additionally, along with the question-answer similarity score from Eq. 2, our architecture includes intermediate representations of the question and the answer x_q and x_a into the final vector representation x_{join}, which together constitute a much richer representation for computing the final score. We call this network simply CNN.

3.3.2 Relational models

Yu et al. (2014) shows that combining the output of their deep learning system with a simple feature vector that includes word overlap counts in a logistic regression model, provides a significant boost in accuracy and yields new state-of-the-art results.

Table 3 provides the results when we include the information about overlapping words in two modes: (i) feature vector (*fvec*) mode – when we include overlapping word counts replicating (Yu et al., 2014), which is represented by a feature vector x_{feat} that is plugged into the final representation x_{join} (see Fig. 2); and (ii) *embeddings* mode – when we augment the representation of input words with additional word *overlap* indicator features (as described in Sec. 2.1.1)[3]. First, we note that the results are significantly better than in Table 2 when no overlap information is used. Adding word overlap information in the form of a feature vector x_{feat} results in a considerable generalization improvement of the network. As argued by Yu et al. (2014), distributional word embeddings have certain shortcomings especially when dealing with proper nouns and cardinal numbers, which are frequent in factoid questions.

In contrast, our approach to encode the relational information about overlapping words in a pair (*embeddings*) directly into word embeddings shows even larger improvement on TRAIN, achieving the

---

[3]a combined model using both *fvec* and *embeddings* modes yielded the same performance as using the *embeddings* model.

best results on TRAIN-ALL with a MAP score of 76.54% and an MRR of 81.86%. We call this network using relational information, CNN_R.

3.4 Comparing with the state of the art

It should be noted that, to be consistent with the results of previous work on TREC13, it is required to evaluate our models in the same setting as (Wang et al., 2007; Yih et al., 2014), i.e., we need to (i) remove the questions having only correct or only incorrect answer sentence candidates and (ii) use 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 score of the systems. This basically lowered the performance of CNN_R from an MAP and an MRR of .7654 and .8186 to .7186 and .7828, respectively. We used these numbers in Table 4 for exactly comparing with the results of previously published systems. We note that our model is almost on par with previous models. However, TREC13 is too small to assess the rank of our approach. Therefore,

	MAP	MRR	P@1
State of the art			
CNN_c (Yang et al., 2015)	.6520	.6652	n/a
ABCNN (Yin et al., 2015)	.6914	**.7127**	n/a
$LSTM_{a,c}$ (Miao et al., 2015)	.6855	.7041	n/a
$NASM_c$ (Miao et al., 2015)	.6886	.7069	n/a
Our Models			
CNN	.6661	.6851	.5401
CNN_R	**.6951**	.7107	.5720

Table 5: Performance on the WikiQA dataset

we evaluated our best systems on WikiQA, which being larger, enables a more reliable system comparison. Tab. 5 reports the system performance on WikiQA. It shows that our model reaches the accuracy of the best system, ABCNN, and outperforms $NASM_c$ by Miao et al. (2015), which was superior to our models on TREC13. Finally, the difference between CNN and CNN_R is again remarkable confirming the benefit of relational information.

4 Related Work

Most of the previous work to tackle the answer sentence selection task use various approaches to model transformations of syntactic trees between a question and its candidate answer sentence, e.g., Wang et al. (2007) use quasi-synchronous grammar, Heilman & Smith (2010) develop an improved Tree Edit Distance (TED) model, Wang & Manning (2010) develop a probabilistic model to learn tree-edit operations on dependency parse trees, while Yao et al. (2013) applies linear chain CRFs with features derived from TED. Severyn and Moschitti (2013) applied SVM with tree kernels to shallow syntactic representations. Yih et al. (2013) use distributional models based on lexical semantics to match semantic relations of aligned words in QA pairs.

Recently, deep learning approaches have been successfully applied to various sentence classification tasks, e.g., (Kalchbrenner et al., 2014; Kim, 2014), and for modelling text pairs, e.g. (Lu and Li, 2013; Hu et al., 2014), where in the latter model they use up to 3 convolution-pooling layers, while in our experiments deeper architectures were severely overfitting and we compensate our more shallow sentence ConvNets by using a more powerful relational model.

Additionally, a number of deep learning models have been recently applied to question answering, e.g., Yih et al. (2014) applied convolutional neural networks to open-domain question answering; Bordes et al. (2014) propose a neural embedding model combined with the knowledge base for open-domain QA; Iyyer et al. (2014) applied recursive neural networks to the factoid QA over paragraphs.

The work closest to ours is (Yu et al., 2014), where they present a deep learning architecture for answer sentence selection. However, their sentence model to map questions and answers to vectors operates only on unigrams or bigrams. Our sentence model is based on a convolutional neural network that uses a larger width of the convolution filter, thus allowing the network to capture longer range dependencies. Moreover, our architecture along with the similarity score also encodes vector representations of questions and answers used to compute the final score. Hence, our model constructs and learns a richer representation of the question-answer pairs, which results in superior results on the answer sentence selection dataset. Moreover, we use a completely different way to encode relational information about words that overlap in a pair. Finally, our deep learning model is trained end-to-end, while in (Yu et al., 2014) they use the output of their neural network in a separate logistic scoring model.

5 Conclusions and future work

In this paper, we propose a novel deep learning architecture for answer sentence selection. Our experimental findings show that our model can achieve the accuracy of state-of-the-art networks, which are much more complex. This is largely due to our use of more expressive models for the input question and answer sentences, and our approach to inject relational information directly in the word embeddings. However, our word overlap indicator features are based on simple string matching, which is clearly a very coarse way to model relatedness between words in a question-answer pair.

Recently, deep learning architectures have been successfully applied to learn word alignments in machine translation, e.g., (Yang et al., 2013). It sounds promising to allow the network to learn to dynamically align the related words in a question and its answer. This in turn requires to maximize over the latent alignment configurations, thus making the optimization problem highly non-convex. Our preliminary experiments show that a far larger number of

text pairs are required to train such architectures. We leave it for the future work.

References

[Agirre et al.2015] Eneko Agirre, Carmen Banea, Claire Cardie, Daniel Cer, Mona Diab, Aitor Gonzalez-Agirre, Weiwei Guo, Iigo Lopez-Gazpio, Montse Maritxalar, Rada Mihalcea, German Rigau, Larraitz Uria, and Janyce Wiebe. 2015. SemEval-2015 Task 2: Semantic Textual Similarity, English, Spanish and Pilot on Interpretability. In *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*.

[Bordes et al.2014] Antoine Bordes, Jason Weston, and Nicolas Usunier. 2014. Open question answering with weakly supervised embedding models. In *ECML*.

[Denil et al.2014] Misha Denil, Alban Demiraj, Nal Kalchbrenner, Phil Blunsom, and Nando de Freitas. 2014. Modelling, visualising and summarising documents with a single convolutional neural network. Technical report, University of Oxford.

[Echihabi and Marcu2003] Abdessamad Echihabi and Daniel Marcu. 2003. A noisy-channel approach to question answering. In *ACL*.

[Heilman and Smith2010] Michael Heilman and Noah A. Smith. 2010. Tree edit models for recognizing textual entailments, paraphrases, and answers to questions. In *NAACL*.

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

[Iyyer et al.2014] 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*.

[Kalchbrenner et al.2014] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A convolutional neural network for modelling sentences. *ACL*.

[Kim2014] Yoon Kim. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751, Doha, Qatar, October.

[Lu and Li2013] Zhengdong Lu and Hang Li. 2013. A deep architecture for matching short texts. In *NIPS*.

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

[Mikolov et al.2013] 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.

[Nair and Hinton2010] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *ICML*.

[Severyn and Moschitti2013] Aliaksei Severyn and Alessandro Moschitti. 2013. Automatic feature engineering for answer selection and extraction. In *EMNLP*.

[Severyn and Moschitti2015] 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.

[Wang and Ittycheriah2015] Zhiguo Wang and Abraham Ittycheriah. 2015. Faq-based question answering via word alignment. *arXiv preprint arXiv:1507.02628*.

[Wang and Manning2010] Mengqiu Wang and Christopher D. Manning. 2010. Probabilistic tree-edit models with structured latent variables for textual entailment and question answer- ing. In *ACL*.

[Wang et al.2007] Mengqiu Wang, Noah A. Smith, and Teruko Mitaura. 2007. What is the jeopardy model? a quasi-synchronous grammar for qa. In *EMNLP*.

[Xuchen Yao and Callison-Burch2013] Peter Clark Xuchen Yao, Benjamin Van Durme and Chris Callison-Burch. 2013. Answer extraction as sequence tagging with tree edit distance. In *NAACL*.

[Yang et al.2013] Nan Yang, Shujie Liu, Mu Li, Ming Zhou, and Nenghai Yu. 2013. Word alignment modeling with context dependent deep neural network. In *ACL*.

[Yang et al.2015] 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.

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

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

[Yin et al.2015] 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*.

[Yu et al.2014] Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2014. Deep learning for answer sentence selection. *CoRR*.

[Zeiler2012] Matthew D. Zeiler. 2012. Adadelta: An adaptive learning rate method. *CoRR*.