SwissAlps at SemEval-2017 Task 3: Attention-based Convolutional Neural Network for Community Question Answering

Jan Deriu
Zurich University of Applied Sciences
deri@zhaw.ch

Mark Cieliebak
Zurich University of Applied Sciences
ciel@zhaw.ch

Abstract

In this paper we propose a system for re-ranking answers for a given question. Our method builds on a siamese CNN architecture which is extended by two attention mechanisms. The approach was evaluated on the datasets of the SemEval-2017 competition for Community Question Answering (cQA), where it achieved 7^{th} place obtaining a MAP score of 86.24 points on the *Question-Comment Similarity* subtask.

1 Introduction

Community Question Answering (cQA) describes the task of finding a relevant answer to a never-before seen question (Nakov et al., 2017). The cQA task in SemEval-2017 is subdivided into three subtasks: (a) *Question-Comment Similarity*, (b) *Question-Question Similarity*, and (c) *Question-External Comment Similarity*. We participated at the *Question-Comment Similarity* subtask, which consists of re-ranking a set of 10 answers to a given question, such that all the relevant answers are ranked higher than the irrelevant answers. We evaluated this system on the dataset provided by SemEval-2017 for the *Question-Comment Similarity* subtask, wich consits of approximately 2000 questions with 10 answers each. Our system ranked 7^{th} place, achieving a MAP score of 86.2 which was outperformed by 2 points by the 1^{st} ranked system. In this paper we describe the implementation details of our system, which follows a siamese CNN architecture based on (Severyn and Moschitti, 2015) extended by the attention mechanisms introduced by (Yin et al., 2015).

Siamese Architecture Siamese architectures usually consist of two parallel CNNs, each processing one sentence and then using the representations for the classification. Siamese architectures have been proposed for various tasks, e.g. (Bromley et al., 1993) used the structure for signature verification, and they have been shown to be very useful for modelling sentence pairs: (He et al., 2015), (Severyn and Moschitti, 2015), and (Tan et al., 2015) used the siamese architecture to generate representations for both sentences which then are used for classification.

Attention Mechanisms Recently the notion of attention has been introduced in neural network architectures to mimic human behaviour, as we tend to focus on key parts of the sentences to extract relevant parts. Most of the work on attention mechanisms is focused on LSTMs: for instance in (Bahdanau et al., 2014) the authors use an attention mechanism for language translation, and in (Vinyals et al., 2015) the authors use it for generating parse trees. Regarding attention mechanisms for CNNs, we are only aware of (Yin et al., 2015), on which our system is based on.

The rest of the paper is structured as follows: in Section 2 we present our model showing the siamese architecture augmented with two different attention mechanisms. In Section 3 we describe our experimental setup and show the results obtained with our system. We conclude our discussion in Section 4.

2 Model

The input to the system are pairs of questions and answer candidates where the model should classify if the answer candidate is relevant to the question, thus, being a binary classification problem. Given such an input a question and an answer candidate, the parallel CNNs produce a representation

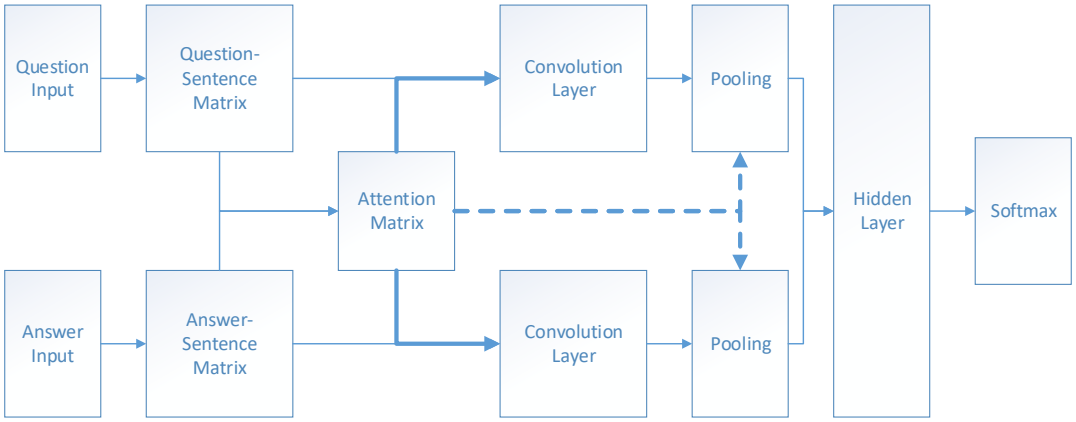

Figure 1: The architecture of the attention-based CNN used in our approach. The bold part highlights the first attention mechanism the dotted line highlight the second attention mechanism.

for both sentences, which are then concatenated and fed into a fully connected hidden layer before being fed into a softmax layer for classification (see Figure 1). The model is extended with two attention mechanisms: one modifies the input to the convolution and the second modifies the output of the convolution. Both methods aim at giving more weight to relevant parts of the sentences.

2.1 Language Model

We use word embeddings based on word2vec (Mikolov et al., 2013) as input to the convolutions. As described in (Mikolov et al., 2013) we first learn representations for phrases by learning which bigrams and trigrams appear frequently together. These n-grams are replaced by a unique token, e.g. 'New York' is replaced by the token 'New_York'. The word embeddings are generated using the skip-gram model setting the context window to 5 and the dimensionality to $d = 200$. The data used to create the word embeddings is a large corpus of 200M English Twitter messages. The word embeddings are stored as a matrix $E \in \mathbb{R}^{n \times d}$ where n is the number of tokens in the vocabulary. We generate a mapping V from each token t to the index of the corresponding word vector in the matrix E where $V(t)$ denotes the index of token t.

2.2 Siamese CNN Architecture

Input Layer A minimal preprocessing is applied to both sentences. First, each sentence is lower-cased and tokenized. Each token t is replaced with the corresponding vocabulary index $V(t)$. Thus, each sentence is represented as a vector s of indices. We denote the length of the vector as s_q for the length of the question and s_a for the length of the answer.

Embedding Layer The embedding layer uses the indices provided by the input layer to select and concatenate the vectors from the embedding matrix E , thus, creating a matrix representation S for the sentence. For the question we have $S^q \in \mathbb{R}^{s_q \times d}$ and for the answer candidate $S^a \in \mathbb{R}^{s_a \times d}$.

Convolution Layer This layer applies a set of m convolutional filters of length h over the sentence matrix $S \in \{S^q, S^a\}$. Let $S_{[i:i+h]}$ denote the concatenation of word vectors S_i to S_{i+h}. A feature c_i is generated for a given filter $\mathbb{F}$ by:

$$c_i := \sum_{k,j} (S_{[i:i+h]})_{k,j} \cdot \mathbb{F}_{k,j} \qquad (1)$$

The concatenation of all vectors in a sentence defines a feature vector $c \in \mathbb{R}^{s-h+1}$, where s denotes the sentence length. The vectors are then aggregated from all m filters into a feature map matrix $C \in \mathbb{R}^{m \times (s-h+1)}$. The output of the convolutional layer is passed through the *relu*-activation function (Nair and Hinton, 2010), before entering a pooling layer.

Zero Padding When computing the convolution at the boundary of the sentence, the convolutional filter is off the edge. Zero Padding is applied by adding $h-1$ zero vectors at the beginning and the end of the sentence matrix. The padded sentence matrix is of the form: $S^{zq} \in \mathbb{R}^{s_q+2*(h-1) \times d}$ and $S^{za} \in \mathbb{R}^{s_a+2*(h-1) \times d}$ for the question and answer candidate respectively. Note that the feature map

matrix has the form: $C \in \mathbb{R}^{m \times (s+h-1)}$ if the input is padded.

Pooling Layer The pooling layer aggregates the vectors in the feature map matrix C by taking the maximum value for each feature vector. This reduces the representation of both the question and the answer candidate to $c_{q,pooled}, c_{a,pooled} \in \mathbb{R}^m$.

Hidden Layer The two vectors $c_{q,pooled}$ and $c_{a,pooled}$ are concatenated to a vector $x \in \mathbb{R}^{2m}$ and passed into a fully connected hidden layer which computes the following transformation: $x_h = relu(W * x + b)$, where $W \in \mathbb{R}^{2m \times 2m}$ is the weight matrix and $b \in \mathbb{R}^{2m}$ the bias vector.

Softmax Finally, the outputs of the previous layer $x \in \mathbb{R}^{2m}$ are fully connected to a softmax regression layer, which returns the class $\hat{y} \in [1, K]$ with largest probability, i.e.,

$$\begin{aligned} \hat{y} &= \arg\max_j P(y = j \mid \mathbf{x}, \mathbf{w}, \mathbf{a}) \\ &= \arg\max_j \frac{e^{\mathbf{x}^\intercal \mathbf{w}_j + a_j}}{\sum_{k=1}^{K} e^{\mathbf{x}^\intercal \mathbf{w}_k + a_k}}, \end{aligned} \tag{2}$$

where $\mathbf{w}_j$ and a_j denotes the weights and bias of class j.

2.3 Attention Mechanism

We implemented the two different ways of introducing an attention mechanism into the siamese structure. The first manipulates the input to the convolution directly, the second modifies output to the convolution. Both approaches are based on an attention matrix.

Attention Matrix The attention matrix $A \in \mathbb{R}^{s_q \times s_a}$ is derived from the sentence matrices S_q and S_a by computing the pairwise Euclidean similarity between the word embeddings of S_q and the word embeddings of S_a. Thus, $A_{i,j} = (1 + |S_{q_i} - S_{a_j}|)^{-1}$ denotes the similarity of the *i*-th word in the question with the *j*-th word in the answer candidate.

Convolution Modification The first mechanism modifies the input to the convolution by applying a linear transformation to the attention matrix A to create the attention features. For this, two weight matrices are used: one for the question $W_q \in \mathbb{R}^{s_c \times d}$ and one for the answer candidate $W_a \in \mathbb{R}^{s_q \times d}$. To attention matrix is multiplied with the weight matrices to generate the attention features: $A_q = A * W_q$ and $A_a = A^T * W_a$ with $A_q \in \mathbb{R}^{s_q \times d}$ and $A_a \in \mathbb{R}^{s_a \times d}$, where the weight matrices are learned during the training phase. The attention features are stacked on top of the sentence matrix, creating an order-3-tensor: $S_q^2 \in \mathbb{R}^{s_q \times d \times 2}$ for the question and $S_a^2 \in \mathbb{R}^{s_a \times d \times 2}$ for the answer candidate. These tensors are used as the input into the convolution layer, giving more weight to the relevant regions in the sentence. As in (Yin et al., 2015) we refer to this architecture as *ABCNN 1*.

Attention Based Pooling The second mechanism modifies the output of the convolution. First a sliding window is applied on h consecutive columns of the feature map matrix $C^w_{:,i} = \sum_{k=i:i+h} C_{:,k}$ where $i \in [1..s_q]$ and the window size h is the same as the filter length h used for the convolution. The values of the resulting feature map matrix $C^w \in \mathbb{R}^{m \times s_q}$ are weighted to include the attention values. The attention values are generated by summing the attention matrix column-wise for the question and row-wise for the answer candidate. Thus, $a^q = \sum A_{j,:} \in \mathbb{R}^{s_q}$ and $a^a = \sum A_{:,j} \in \mathbb{R}^{s_a}$ represent the attention values for each token in the question and the answer candidate, respectively. These vectors are used to weight the feature map matrix, thus, we get $C^q_{:,i} = a^q_i * C^{wq}_{:,i}$ for the question and $C^a_{:,i} = a^a_i * C^{wa}_{:,i}$ where C^{wq} and C^{wa} denote the window averaged feature map matrices for the question and answer candidate, respectively. Finally, standard max pooling is applied to the attention weighted feature map matrices. As in (Yin et al., 2015) we refer to this architecture as *ABCNN 2*.

3 Experiments

For the experiments we compared the three different architectures: (i) the siamese architecture without the attention mechanism; we refer to this as *siamese CNN* (sCNN) (ii) the *ABCNN 1* architecture, and (iii) the *ABCNN 2* architecture.

3.1 Setup

For all experiments we used the same pre-trained 200-dimensional word embeddings introduced in Section 2.1. We employ *AdaDelta* (Zeiler, 2012) as optimizer and L_2 regularization to avoid overfitting. Table 1 gives an overview of the hyperparameters chosen via grid search. Furthermore we employ early stopping on the *SemEval-2016 test-set*, using a patience of 50 epochs. The final

	lr	ϵ	ρ	m	h	L_2
sCNN	0.1	$1e^{-6}$	0.95	500	3	0.001
ABCNN1	0.05	$1e^{-8}$	0.95	200	3	0.0005
ABCNN2	0.01	$1e^{-8}$	0.95	200	3	0.0001

Table 1: Hyperparameters of the system. lr: learning rate,ρ and ϵ: AdaDelta hyperparameters , h: filter width, m: number of filters

ranking of the answer candidates for a question is derived from the softmax probability, i.e. the answer candidates are sorted by their probability of being relevant.

3.2 Data

The training data provided by SemEval consist of approx. 2000 questions with 10 answer candidates each. Each answer candidate is manually labelled as either *Relevant*, *Irrelevant*, or *Potentially Useful*. Table 2 gives an overview of the data. For the training phase we combined the *Training Part 1*, *Training Part 2*, and *Dev 2016*, and we used *Test 2016* as validation set for early stopping. Furthermore, we aggregated the *Irrelevant* and the *Potentially Useful* pairs to reduce the problem to a binary classification task.

	Relevant	Irrelevant	Pot. Useful	Total
Training Part 1	5287	6362	2461	14110
Training Part 2	1364	1777	649	3790
Dev 2016	2440	1209	413	818
Test 2016	1329	1485	456	3270

Table 2: Overview of the datasets used for training and validation.

3.3 Results

Table 3 shows the results obtained by the systems on the *Test 2016*. The results show that the attention based mechanism boosts the MAP score by 3-4 points, the AvgRec by 3 points, and the MRR score by 3-4 points compared to the *sCNN* architecture. We also observe that *ABCNN2* outperforms *ABCNN1* by 1 point in every metric.

	MAP	AvgRec	MRR	Prec	Rec	F1	Acc
sCNN	0.741	0.854	0.806	0.589	0.815	0.684	0.693
ABCNN1	0.776	0.878	0.833	0.807	0.313	0.451	0.690
ABCNN2	0.788	0.883	0.845	0.798	0.352	0.489	0.700

Table 3: Resutls on the *Test 2016* set.

Based on these results we decided to use *ABCNN2* as our primary submission and *ABCNN1* as the contrastive submission. Table 4 shows the results obtained on the *SemEval-2017 test-set*. We observe the same pattern as with *Test 2016*, i.e. *ABCNN2* outperforms *ABCNN1* by 1 point. We included the scores of the 1^{st}, 2^{nd}, and 3^{rd} placed submissions for comparison. Our system is outperformed by 2 points by the *KeLP* and the *Beihang-MSRA* submission and by only 0.6 points by the *IIT-UHH* submission.

	MAP	AvgRec	MRR	Prec	Rec	F1	Acc
ABCNN1	0.855	0.919	0.905	0.903	0.240	0.379	0.591
ABCNN2	0.862	0.922	0.908	0.907	0.284	0.433	0.613
KeLP(1^{st})	0.884	0.937	0.928	0.873	0.582	0.698	0.738
Beihang-MSRA(2^{nd})	0.882	0.938	0.923	0.519	1.0	0.684	0.519
IIT-UHH (3^{rd})	0.868	0.920	0.912	0.733	0.745	0.739	0.727

Table 4: Resutls on the *Test 2017* set.

4 Conclusion

We described a deep learning approach to question-answering. The proposed architecture is based on parallel CNNs that compute a sentence representation for the question and the answer. These representations are then concatenated and used to predict whether the answer is relevant to the question. The architecture is augmented by two different attention mechanisms which improve the performance. Our system was evaluated on the SemEval-2017 competition for Community Question Answering, where it ranked 7^{th} on the *Question-Comment* subtask. Our system performed poorly on the other two subtasks, thus, for future work we will improve our system to tackle these tasks with high performance.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473* .

Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Sackinger, and Roopak Shah. 1993. Signature verification using a "siamese" time delay neural network. In *International Journal of Pattern Recognition and Artificial Intelligence*.

Hua He, Kevin Gimpel, and Jimmy J Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *EMNLP*. pages 1576–1586.

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*. pages 3111–3119.

Vinod Nair and Geoffrey E Hinton. 2010. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*. pages 807–814.

Preslav Nakov, Doris Hoogeveen, Lluís Màrquez, Alessandro Moschitti, Hamdy Mubarak, Timothy Baldwin, and Karin Verspoor. 2017. SemEval-2017 task 3: Community question answering. In *Proceedings of the 11th International Workshop on Semantic Evaluation*. Association for Computational Linguistics, Vancouver, Canada, SemEval '17.

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*. ACM, pages 373–382.

Ming Tan, Cicero dos Santos, Bing Xiang, and Bowen Zhou. 2015. Lstm-based deep learning models for non-factoid answer selection. *arXiv preprint arXiv:1511.04108* .

Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. 2015. Grammar as a foreign language. In *Advances in Neural Information Processing Systems*. pages 2773–2781.

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* .

Matthew D Zeiler. 2012. Adadelta: an adaptive learning rate method. *arXiv preprint arXiv:1212.5701* .