aMV-LSTM: an attention-based model with multiple positional text matching

Thiziri Belkacem
Paul Sabatier University, IRIT laboratory
thiziri.belkacem@irit.fr

Taoufiq Dkaki
Jean Jaures University, IRIT laboratory
taoufiq.dkaki@irit.fr

Jose G. Moreno
Paul Sabatier University, IRIT laboratory
jose.moreno@irit.fr

Mohand Boughanem
Paul Sabatier University, IRIT laboratory
mohand.boughanem@irit.fr

ABSTRACT

Deep models are getting a wide interest in recent NLP and IR state-of-the-art. Among the proposed models, position-based models and attention-based models take into account the word position in the text, in the former, and the importance of a word among other words in the latter. The positional information are some of the important features that help text representation learning. However, the importance of a given word among others in a given text, which is an important aspect in text matching, is not considered in positional features. In this paper, we propose a model that combines position-based representation learning approach with the attention-based weighting process. The latter learns an importance coefficient for each word of the input text. We propose an extension of a position-based model MV-LSTM with an attention layer, allowing a parameterizable architecture. We believe that when the model is aware of both word position and importance, the learned representations will get more relevant features for the matching process. Our model, namely aMV-LSTM, learns the attention based coefficients to weight words of the different input sentences, before computing their position-based representations. Experimental results, in question/answer matching and question pairs identification tasks, show that the proposed model outperforms the MV-LSTM baseline and several state-of-the-art models.

KEYWORDS

Attention models, positional, text representation, text matching

1 INTRODUCTION

Deep neural networks have been used in a large scope of natural language processing (NLP) and information retrieval (IR) models, including several models for text representation and matching. In most state-of-the-art studies [7, 14, 17, 23], the input text is first mapped to a set of word vectors, then different mechanisms are used to capture the most important representation features. Words of the input texts are considered according to their semantic similarities. *Position-based* models [9, 22] rely on the word position to compute representations of the input text. The positional information of the word, such as proximity, word dependencies and the sequence structure, are important in learning text representations and can have an important impact in the matching process. *Attention-based* representation models [16, 24, 27], learn coefficient vectors for the weighting process. These coefficients enable to designate the words from the input text that deserve more attention, regardless of their position. But, it would be interesting for a representation learning model to have information about the most important words, as well as their positions in the input text.

In this paper, we propose a model architecture that combines attention-based and position-based representation learning. Our model, namely aMV-LSTM, is an extension of the positional sentence representa-tion model (MV-LSTM) [22]. We adopted different architectures of the aMV-LSTM model, to compute the importance weight of every term in the input text, using learnable attention coefficients, then construct a position-based sentence representations. In this work, our contributions are as follow:

(1) We combine the position-based and the attention-based representation learning algorithms. To do so, we propose a model architecture that extends an existing position-based model, with an attention-based layer. Our aim is to focus more on the most important and informative words of the input text making more effective the succeeding text matching layers.
(2) We conduct a comparative experimental setup, where we compare different models of the state-of-the-art and the proposed representation learning model. We use two question-answering (QA) tasks: question-answer matching in the

https://doi.org/10.1145/3297280.3297355

WikiQA[1] dataset and question-question matching (question-pairs identification) in QuoraQP[2] dataset. The results show the effectiveness of the proposed model compared to the different baselines.

2 RELATED WORK

Among the different neural models proposed for text matching, we are interested in position-based and attention-based deep models. In this section, we first, present some deep models for text matching. Then, we present the attention-based and the position-based concepts, in text representation learning and matching, with some corresponding models from the state-of-the-art.

2.1 Deep models for text matching

Representation and text matching are central for many real world applications, such as document retrieval [7], question answering [17] and paraphrase identification [14]. Several approaches based on neural networks have been proposed, to go beyond the classical similarity of bag-of-words [12, 20].

Huang et al [7], propose the Deep Structured Semantic Model (DSSM). This model uses a DNN architecture in order to map a high-dimensional sparse features vector, of a given input text, into a low-dimensional dense vector. The architecture of DSSM is made of a multiple layers percetron (MLP) with three processing layers: a *word hashing* layer that constructs a high dimensional sparse vector followed by two *projection layers* that produce a low-dimensional semantic representation vector. A convolution based version of DSSM, namely C-DSSM, is proposed by Shen et al [18]. C-DSSM extends the DSSM model with a convolutional layer, such that a 3-max-pooling is applied to the computed hash vector in order to extract stronger representation features. The experimental results show that the C-DSSM outperforms the DSSM model. In [5], the authors proposed two convolutional models for sentence matching: ARC-I and ARC-II. The first constructs a sentence representation using a sequence of convolution and pooling layers, where the output features vector is the corresponding sentence representation. An MLP layer is then used to compute the matching degree of the input sentences. ARC-II applies a series of convolution and pooling layers to the input matching matrix. In both ARC-I and ARC-II, the input sentences are mapped to their embedded word vectors. In [14], Pang et al proposed a text matching model that matches two input sequences as in image recognition, namely MatchPyramid. This model, first, considers the embedded vectors of the input text sequences, using embeddings trained in the target corpus, then computes the matching tensor using cosine and dot-product similarity functions. The computed matching tensor is then fed to a sequence of convolution and pooling layers in order to extract high level interaction signals between the two input texts. Reported results, in paraphrase identification and paper citation matching tasks, show that the model outperforms several classical and state-of-the-art models.

[1] https://www.microsoft.com/en-us/download/details.aspx?id=52419
[2] http://qim.ec.quoracdn.net/quora_duplicate_questions.tsv

2.2 Attention-based text matching models

The *"attention"* concept, resulting from machine translation [1], has brought a significant gain in several NLP applications, including sentiment classification [16, 26] and paraphrase identification [27]. Attention-based models, identify the kernel information to be considered in a given sequence and allow focusing on some discriminated elements. The main idea is as follows:
Given an element sequence S, the attention-based model should learn a coefficient vector α that determines how much attention should be given for each element of S. The elements in S will be then weighted accordingly. Hence, the attention vector α is used in order to elaborate an optimal weighting process according to the task to be performed.

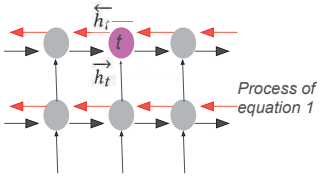
Severyn et al [17] proposed a simple attention model to explore contextual information of input sentences, in order to handle the QA task. The proposed model is based on convolution layers, applied to the word level, combined with additional tf-idf features, such as word overlap features and BM25 scores, to compute the final matching score. Yang et al [26] propose a hierarchical attention network for document classification. The model combines a word and a sentence attention levels in a recurrent model architecture. The input text is first represented using word embeddings, then a first bidirectional GRU[3] (bi-GRU) layer is applied to compute a representation vector for the input text. This vector is then passed through an MLP layer in order to construct an intermediate representation vector that will be used to learn attention coefficients of the different words. A second bi-GRU layer is used to compute the sentence level representation and learn the corresponding attention coefficients. The same process as in the word level is repeated in the sentence level. The final score is then computed based on these representations. Yang et al [24] propose an attention-based neural model (aNMM) for question-answer matching. aNMM uses a DNN architecture with a value shared weighting scheme and question words gating. In aNMM, some elements of the fully connected layers share the same connection weights, according to the interval of these elements values which represent the matching signal strength. The input question and answer are represented by their embedded word vectors. Then an interaction matrix is computed using the cosine similarity. Then a bin-sum technique is used, where elements of the same raw having interaction signal of the same interval will be added. Hence, these elements share the same connection weights. In addition to the value shared weights, the aNMM model uses question gates, where input question word vectors are used to compute attention scores for the shared weights. The question gates determine the attention vector that enable the model to focus on the strongest interaction signals from the value shared weights.

2.3 Position-based models

The word position is already considered in convolutional and recurrent models for text matching. In convolutional models [5, 18], the convolution layers process slighted windows or n-grams of the input text in order to extract contextual features. In recurrent models

[3] Is a specific LSTM architecture that, differently from the basic LSTM, GRU uses a gating mechanism to track the state of a given sequences without using separate memory cells [1]

Figure 1: The bidirectional process in a bi-LSTM network with 2 layers. Forward (continue thick arrows) and backward (dashed thick arrows) hidden states are computed by the equation 1.



[11, 13, 21], the words ordering of a given input sentence is preserved. In these models, the hidden activation vector, corresponding to the last input word, is considered as the embedding vector of the whole sentence. However, these models do not consider the position of individual words in a given sequence. Position-based models [9, 22] consider the position of an individual word, in a given text, as an important factor to determine its importance weight. Hui et al [9] proposed a representation learning model, based on word positions, for a relevance-based text matching. The model computes first the semantic similarity matrix using the embedded word vectors of the input document and query terms. The similarity matrix is then distilled, by selecting the k most significant matching signals along the document words dimension. The objective is to localize the relevance matching over all the matrix entries. This matrix is then fed to the DRMM[4][3] as input, to compute the final matching score.

MV-LSTM [22] is a semantic matching model based on different positions for sentence representation. The model uses bi-LSTM networks to construct a position-aware representations for the input sentences. Each input sentence acquires a different representation at each position provided by the bi-LSTM units, as follows: Let $S_t = \left[\overrightarrow{h_t}, \overleftarrow{h_t}\right]$ be a bidirectional representation vector, at position t of the input sentence $S = (w_0, w_1, ..., w_l)$. S contains $l + 1$ words, where w_t is the embedded vector of the word w at position t. In a bi-LSTM unit, a similar process is repeated on both forward and backward directions, to compute $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$ vectors respectively. The resulted vectors are concatenated to construct a unique representation h_t at a given position t. This process is highlighted in figure 1 in a given position t. The pipeline equations 1 describe how one hidden state h_t is computed by an LSTM unit at one direction:

$$\begin{aligned} i_t &= \sigma\left(W_{x_i} w_t + W_{h_i} h_{t-1} + b_i\right), \\ f_t &= \sigma\left(W_{x_f} w_t + W_{h_f} h_{t-1} + b_f\right), \\ m_t &= f_t m_{t-1} + i_t \; tanh\left(W_{x_m} w_t + W_{h_m} h_{t-1} + b_m\right), \\ o_t &= \sigma\left(W_{x_o} w_t + W_{h_o} h_{t-1} + b_o\right), \\ h_t &= o_t \; tanh(m_t) \end{aligned} \tag{1}$$

where i, f, m and o are respectively: the input and the forget information, the information to memorize by the cell through the current pass and the output information. W_x corresponds to the weighting matrices of the different gates f, m and o of the LSTM network. In MV-LSTM [22], the positional sentence representation y_t is computed, based on the embedded word vectors of that sentence, by the equation 2.

$$y_t = \varepsilon(S_t) \tag{2}$$

where $S_t = w_0, w_1, ..., w_t$ is the sub-sequence of the first $t + 1$ words from S. ε stands for the computational pipeline, described in equation 1, applied on forward and backward directions.

Given two input sentences S_1 and S_2, MV-LSTM computes interaction matrices based on the bidirectional representations of the two sentences. These matrices are then passed through a pooling layer, where k strongest interactions are selected and fed to the final layer. Named *k-Max pooling*, this process is used in several neural text matching models [10, 15, 17, 19]. The objective is to remove the noisy features from the interaction tensor. The final score s is computed using the features vector z resulted from the pooling layer as described in equation 3.

$$s = \delta(Uz + b) \tag{3}$$

where U and b stands for the MLP training parameters and δ is the output activation function.

2.4 Discussion

In this paper, we consider short text matching models for QA tasks. Most of the the reviewed models in the previous section treat the input text sequences as a whole: words are represented by their embedded vectors, then an interaction matrix is computed, using functions such as the cosine similarity [5, 14]. Attention-based models [24, 26] have shown the effectiveness of the learnable attention coefficients in the weighting process, for the input text representation learning and during the matching process. The attention-based models associate an attention score to each word in the input text, these scores represent the importance of each word among the others, independently from their positions in the input text. While in the position based models [11, 22], the importance of a given word is mainly determined by its positional features. Position-based models motivate the importance of the positional information, such as the words ordering and individual positions of words in a given input sequence. But, words are not weighted according to the other words of an input sentence. These models basically rely on the word position in the input text, ignoring the mutual differences between its words that can determine the kernel words of that text. Note that, attention-based and position-based models complement each other and can be used together to exploit their advantages. However, none of previously overviewed models have provided the way to compute the representation of the input text, based on the attention scores and positions of its words.

3 ATTENTION-BASED POSITIONAL TEXT MATCHING

3.1 Motivation

Position based neural models [22] focuses at the *word position* as an asset to construct complex representations for input sequences. Different features are learned from different positions independently of the words themselves in the sentence. The position of a word considerably influences its importance in the sentence as well as its

[4]The architecture of the DRMM model [3] takes as input an interaction matrix of two given texts, then applies a histogram function to map all input pairs to the same length, based on intervals (beans) of elements of the interaction matrix. The histograms are then fed to an MLP to compute the matching score.

significant contribution to the meaning of that sentence. However, there are some other parameters that can measure the contribution of a given word in the sentence, such as the relatedness of this word with the other words of that sentence. We believe that, in a given sentence, there are always a few words that convey the main information. Therefore, these words are the key words of that sentence regardless of their position. So the matching model should focus more on these words and give them more attention. To highlight more this concept, let us look at the following example taken from the WikiQA dataset: *Q: "how are glacier caves formed?"*
A1: "Ice formations in the Titlis glacier cave"
A2: "A glacier cave is a cave formed within the ice of a glacier"
A3: "Glacier caves are often called ice caves, but this term is properly used to describe bedrock caves that contain formed year-round ice"
How can a position-based model identify the suitable answer from *A1*, *A2* and *A3* for the question *Q*?
By focusing on the position of the question words on the different answers, we can eliminate the *A1* since the position of the word stem *"form"* with respect to the position of the word *"glacier"* does not correspond to the appropriate position ordering in the question *Q*. But, the position alone could not help to find out the most appropriate answer from the other two answers, *A2* and *A3*, where positions of the question words corresponds to the positions in these answers. In this case, the model must focus on some words differently than the other words. In the question and in the given answers: the word *"form"* needs more focus then the words *"glacier"* and *"cave"* in the question *Q* and both the answers *A2* and *A3*, because we are asking about the *formatting* process. In order to determine how many attention the model should give to each word of the sentence, we propose to use an attention layer, before computing a position-based representation for a given sentence. The attention layer will learn an attention coefficient to each word, based on its semantic representation (embedded vector), in order to provide the positional representation layer with the relevant information about the importance of the word at the position to be processed.

3.2 Attention gating

We consider the MV-LSTM [22] model for short text matching and propose a new model architecture that extends this model, to take into account the attention-based coefficients in the weighting process. Namely aMV-LSTM, the architecture of the model that we propose is described in figure 2. This figure show the attention layer that is used to extend the architecture of the MV-LSTM [22] model.
Given two input sentences to be compared: $S_1 = ({w_0}^1, {w_1}^1, ..., {w_{l_1}}^1)$ of l_1 words and $S_2 = ({w_0}^2, {w_1}^2, ..., {w_{l_1}}^2)$ of l_2 words. In both S_1 and S_2, ${w_t}^i$ is the embedded vector of the word w at position t of the sentence S_i. We compute attention weight vectors α^i for all words of each sentence, using a gating function [24], as represented in equation 4.

$$ {\alpha_t}^i = \frac{exp({V_i}^T.{w_t}^i)}{\sum_{j=1}^{l_i} exp({V_i}^T.{w_j}^i)} \tag{4} $$

Where V_i is a model parameter that represents the attention coefficients vector for the input sentence S_i, $i \in \{1, 2\}$. u^T represents the transposed vector of u.

The α_i weights, are computed with dependence on the words in the sentence from each other. Hence, it will provide the positional model with an information about words requiring more attention while constructing the representation of the whole sentence. The input sentence representations are then computed using their embedded word vectors, scaled with the attention coefficients:

$$ S'_1 = {w_0}^1 \times {\alpha_0}^1, {w_1}^1 \times {\alpha_1}^1, ..., {w_{l_1}}^1 \times {\alpha_{l_1}}^1 $$
$$ S'_2 = {w_0}^2 \times {\alpha_0}^2, {w_1}^2 \times {\alpha_1}^2, ..., {w_{l_2}}^2 \times {\alpha_{l_2}}^2 $$

The attention-based representations S'_1 and S'_2 are then provided to a bi-LSTM layer, to compute a positional sentence representation, as described in equation 2. The final matching score, is then computed by equation 3 as in the original MV-LSTM model [22].

3.3 Model training

The aMV-LSTM model could be trained with, two different loss functions, according to the learning objective:

Rank hinge loss. is an objective function for a ranking tasks, this function is used in several text matching models [3, 4]. Given a sequence S and two other different sequences S^+ and S^-, such that S^+ is most similar to S and must be ranked better then S^-. The loss function L is defined in equation 5.

$$ L(S, S^+, S^-; \theta) = max(0, 1 - s(S, S^+) + s(S, S^-)) \tag{5} $$

where θ represents the model parameters and $s(S, S^*)$ is the matching score predicted by the model for the input sentences S and S^*.

Categorical cross entropy. is an objective function for a classification tasks [28]. Given two sequences S^1 and S^2, the objective is to compute the probability that event $x = S^1 \approx S^2$ (S^1 is similar to S^2) may occur. The loss function H is defined in equation 6.

$$ H(p, q; \theta) = -\sum_x p(x) log\,(q(x)) \tag{6} $$

where $q(x)$ is the observed probability computed by the model and $p(x)$ is the truth value.

4 EXPERIMENTS AND RESULTS

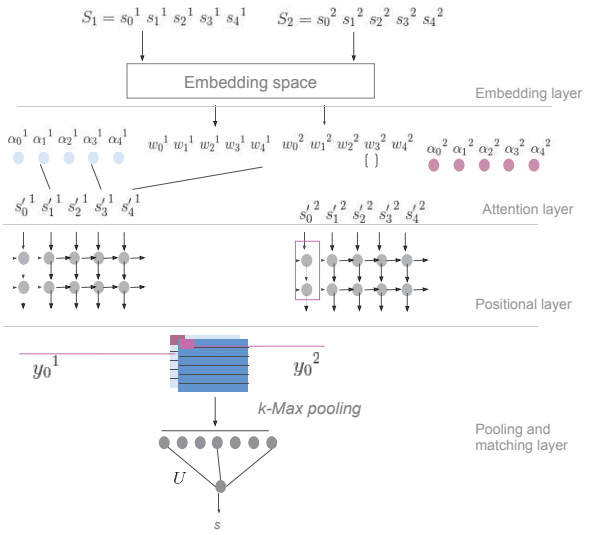
4.1 Experimental protocol

Tool. The model implementation and the experimental process are made using the MatchZoo framework [2]. MatchZoo is a framework for implementing, experimenting and comparing neural based text matching models. This framework is developed with Python using different deep learning libraries such as keras[5], in addition to other libraries.

Datasets. In our experiments, we used two datasets: Microsoft Research WikiQA Corpus [25], which is a set of question and sentence pairs. The questions have been collected from Bing query logs. Each question is linked to a Wikipedia page that potentially has the answer. The candidate answers are the sentences of the summary section of a corresponding Wikipedia page that provides the most important information about the topic. The second corpus consists of question pairs which are either duplicate[6] or not. A data sample of the QuoraQP dataset is presented in table 1. Statistics about

[5] https://keras.io/
[6] Duplicate questions are questions that mean the same thing.

Figure 2: The aMV-LSTM model architecture. The input sentences S_1 and S_2 are first mapped to their embedded word vectors via the *Embedding layer*. Then, trainable attention coefficients ${\alpha_i}^1$ and ${\alpha_j}^2$ are used to weight the embedded representations of the input sentences, in the *Attention layer*. The *Position layer* aims to learn specific representations at each position of the input sentences. These representations are then used by the *Pooling and matching* layer to compute positional interaction matrices, for the forward and backward representations. Then most important matching signals are combined to compute the final matching score s.



both WikiQA and QuoraQP datasets are presented in table 2. In both datasets, we used 80% for training, 10% for test and 10% for validation of the proposed model. All the evaluated models use the same experimental setup.

Parameters and embeddings. For all the models that use word embeddings, words are presented using the pre-trained Glove[7] word embedding. All the embedded vectors have been normalized. All the different models have been trained to optimize the ranking hinge loss function, defined in equation 5, for 400 epochs on the WikiQA dataset and for 500 epochs to optimize the categorical cross entropy, described in equation 6, on the QuoraQP dataset. For every model, we reported the results performance at the end of the training epochs. Concerning parameters of the baselines, we have adopted the configuration of the best result published in the corresponding paper. The same configuration of the neural network, with 50 bi-LSTM units and $k = 100$ in the *k-Max* pooling[8] layer, are used for MV-LSTM and aMV-LSTM.

Semantic interaction and features aggregation. We used the cosine similarity function, of the bidirectional representations computed for each input sentence, to compute the interaction matrices that are used in both MV-LSTM [22] and the proposed aMV-LSTM architectures. Let y_t^1 and $y_{t'}^2$ be representations computed with equation 2 for a given input sentences S_1 and S_2 (corresponding to S_1' and S_2' in aMV-LSTM), at positions t and t' respectively. We semantic similarity *sim* between these sentences is computed using the cosine function

$$sim(y_t^1, y_{t'}^2) = \frac{{y_t^1}^T y_{t'}^2}{\|y_t^1\| \cdot \|y_{t'}^2\|}$$

where $\|y\|$ stands for the L2 norm of then vector y.

4.2 Experimented models

aMV-LSTM configurations. We evaluated three architectures of the aMV-LSTM model, we refer to each as follows:

- *aMV-LSTM (Q)* refers to the architecture where the attention coefficients are learned for only the first element in the input layer (question on the question-answer matching task and first question only on the question-question matching task).
- *aMV-LSTM (A)* refers to the architecture where attention coefficients are learned for only the second element in the input layer (answer on the question-answer matching task and second question only on the question-question matching task).
- *aMV-LSTM (Q+A)* refers to the architecture described in figure 2, where the attention coefficients are learned for both the elements of input layer, at once.

Baselines. We considered the following state-of-the-art models:

- *DSSM* refers to the deep structured semantic IR model [8].
- *CDSSM* refers to extended version of DSSM model with a convolutional layer [19].
- *ARC-I and ARC-II* are two model architectures for text matching proposed by Hu et al [6].
- *MV-LSTM* is a position-based model [22] a model for position-aware representations of the input sentences.
- *ANMM* is an attention-based model with value shared weighting scheme [24].
- *MATCHPYRAMID* is a convolution based model for text matching [14].

4.3 Results and discussion

In this section, we compare results of the different aMV-LSTM model architectures, with those of the MV-LSTM baseline model. We considered two different tasks: question answering with WikiQA dataset and the classification task with QuoraQP dataset. Table 3 shows performance results of the different evaluated models, in terms of *ndcg*@3, *ndcg*@5 and *MAP*, in the WikiQA dataset. In this table, the aMV-LSTM (Q) performs better than the basic MV-LSTM model with more than 7%, in terms of MAP. The aMV-LSTM (A) and aMV-LSTM (Q+A) are less effective than the MV-LSTM baseline. Focusing on the words of the question, in aMV-LSTM (Q), is better than focusing on the words of the answer, in aMV-LSTM (A), or the words of both the question and the answer, in aMV-LSTM (Q+D). Besides, WikiQA is a collection of asymmetric data, the answer provides the information sought by the question. Therefore, it is more important to focus in the question in order to find the most

[7] http://nlp.stanford.edu/data/glove.840B.300d.zip
[8] The pooling layer is applied over both the forward and backward interaction matrices, corresponding to the bi-directional position-based representations of the input sentences.

Table 1: Sample of some questions from the QuoraQP dataset, where each question is given two different *ids* and the last column tells whether the question is duplicated or not (1 or 0 respectively).

id	qid1	qid2	question1	question2	is_duplicate
0	1	2	What is the step by step guide to invest in share market in india?	What is the step by step guide to invest in share market?	0
1	3	4	What is the story of Kohinoor (Koh-i-Noor) Diamond?	What would happen if the Indian government stole the Kohinoor (Koh-i-Noor) diamond back?	0
2	5	6	How can I increase the speed of my internet connection while using a VPN?	How can Internet speed be increased by hacking through DNS?	0
3	7	8	Why am I mentally very lonely? How can I solve it?	Find the remainder when [math]23^{24}[/math] is divided by 24,23?	0
4	9	10	Which one dissolve in water quikly sugar, salt, methane and carbon di oxide?	Which fish would survive in salt water?	0
5	11	12	Astrology: I am a Capricorn Sun Cap moon and cap rising...what does that say about me?	I'm a triple Capricorn (Sun, Moon and ascendant in Capricorn) What does this say about me?	1
6	13	14	Should I buy tiago?	What keeps childern active and far from phone and video games?	0
7	15	16	How can I be a good geologist?	What should I do to be a great geologist?	1

Table 2: Description of the experimental datasets: WikiQA and QuoraQP. In both datasets, we used 10% for test, 10% for validation and 80% for training.

WikiQA		
Questions	Sentences	Answers
3047	29258	1473

QuoraQP		
Question pairs	Positive (duplicates)	Negative (non-duplicates)
404351	149306	255045

appropriate answer. Table 4 present the 3 first answers, corresponding to one sampled question from WikiQA dataset. The considered question is *"Where do crocodiles live?"* and there are 21 possible answers in the WikiQA dataset, where only one question is relevant ($label = 1$). Note that the answers retrieved by the MV-LSTM model as well as the different aMV-LSTM model architectures deal with the same subject as the question. However, the correct answer is ranked better by the aMV-LSTM (Q) model. Compared to the rank with aMV-LSTM (A) and aMV-LSTM (Q+A) models. This example shows the effectiveness of the focus on the input question words.

Figure 4 show the comparison of the accuracy evolution during the training and validation phase of the different architectures of the proposed model aMV-LSTM and the MV-LSTM baseline, in the QuoraQP dataset. The red line on the curves corresponding to aMV-LSTM (A), aMV-LSTM (Q) and aMV-LSTM (Q+A), show the accuracy value of the MV-LSTM model after training. We can notice that all aMV-LSTM architectures evolve in the same way and converge to higher accuracy values compared to the MV-LSTM performance. aMV-LSTM (Q) outperforms the MV-LSTM baseline with 3.8% in terms of accuracy. The attention-based weights has the same effect in the different aMV-LSTM architectures, in the QuoraQP dataset, because it is a symmetric dataset of inputs with the same nature, where the task consists of identifying whether an input question is a duplicate of another question or not. So, the comparison process considers inputs with approximately a same length and the same structure.

Comparison with state-of-the-art models

In table 3 and figure 3 we represent the performance results of our model architectures, compared to the MV-LSTM baseline and some other models from the state-of-the-art. Note that the proposed model outperforms the MV-LSTM baseline and state-of-the-art models. In table 3, the aMV-LSTM (Q) model outperforms all the evaluated models. Figure 3 shows results of the question-pairs identification (in QuoraQP dataset). In this figure, the dashed line corresponds to the maximum accuracy value given by the MatchPyramid model [14]. The continue red line corresponds to the accuracy of the MV-LSTM model. All the proposed aMV-LSTM model architectures perform better then the position-based MV-LSTM baseline. Results are also better then the attention-based aNMM model [24], where attention gates are applied to the final model layer, independently of the words position. However, better performances corresponds to the MATCHPYRAMID and ARC-II [6] models. These models are based on the interaction matrices of embedded word vectors of the input texts. According to the different results, in both WikiQA and QuoraQP datasets, we can conclude that the combination of attention coefficients learning with the position-based representation learning, helps to improve state-of-the-art results. When the word position only is used (MV-LSTM) or the word attention only is used (aNMM), The results are less performent then the proposed aMV-LSTM model, in both WikiQA and QuoraQP datasets. However, the convolution based models, ARC-II and MATCHPYRAMID, outperform all the different models in QuoraQP dataset.

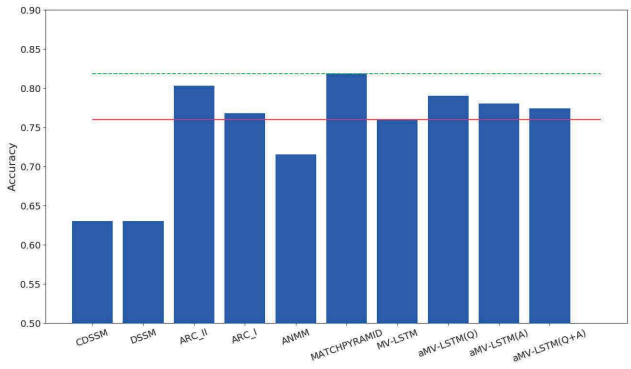
5 CONCLUSION

In this paper, we presented aMV-LSTM, a positional attention-based model for text matching. Allowing parameterizable architectures, aMV-LSTM uses an attention layer to weight input sequence words then learns a position-based representation. Three architectures were explored in our experiments: aMV-LSTM (A), aMV-LSTM (Q) and aMV-LSTM (Q+A) that corresponds, respectively, to the application of the attention layer to the answer input only, to the question input only or to both the question and answer inputs. Experimental results show that aMV-LSTM (Q) outperforms existing text-matching models, including a strong baseline MV-LSTM, with more than 3.8% and 7% in terms of accuracy and MAP, respectively. We conclude that attention-based coefficients of the input words,

Table 3: Experimental results showing the performance of the different models, in terms of $ndcg@3$, $ndcg@5$ and MAP on the WikiQA dataset. Results presented in Bold characters stands for the best performances over all the models.

Models	ndcg@3	ndcg@5	MAP
CDSSM	0.409214	0.477269	0.435007
DSSM	0.531882	0.609625	0.560120
ARC-II	0.540994	0.609524	0.560602
ARC-I	0.564213	0.638075	0.587890
ANMM	0.609504	0.612841	0.645765
MATCHPYRAMID	0.644247	0.690151	0.643604
MV-LSTM	0.610065	0.654855	0.604582
aMV-LSTM (Q)	**0.651922**	**0.694755**	**0.650664**
aMV-LSTM (A)	0.598445	0.641318	0.602153
aMV-LSTM (Q+A)	0.556163	0.614518	0.556163

Figure 3: Performances of the different models in terms of *accuracy*. The values corresponds to the ones at the end of the training process on the QuoraQP dataset. The dashed green line corresponds to the maximum accuracy value and the continue red line corresponds to the accuracy of the MV-LSTM model.



enable the model to focus on the most important content and improve the results. Our work opens an interesting direction for future research. Different neural matching models could be used within the proposed architecture. The future work will focus on the study of the proposed architecture with different state-of-the-art neural models, in order to study the impact of the asymmetric architecture in different datasets.

REFERENCES

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

[2] Yixing Fan, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. 2017. Matchzoo: A toolkit for deep text matching. *arXiv preprint arXiv:1707.07270* (2017).

[3] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 55–64.

[4] Till Haug, Octavian-Eugen Ganea, and Paulina Grnarova. 2018. Neural multi-step reasoning for question answering on semi-structured tables. In *European Conference on Information Retrieval*. Springer, 611–617.

[5] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2042–2050. http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences.pdf

[6] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional Neural Network Architectures for Matching Natural Language Sentences. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 2042–2050. http://papers.nips.cc/paper/5550-convolutional-neural-network-architectures-for-matching-natural-language-sentences.pdf

[7] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 2333–2338. https://doi.org/10.1145/2505515.2505665

[8] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning Deep Structured Semantic Models for Web Search Using Clickthrough Data. In *Proceedings of the 22Nd ACM International Conference on Information & Knowledge Management (CIKM '13)*. ACM, New York, NY, USA, 2333–2338. https://doi.org/10.1145/2505515.2505665

[9] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. 2017. Position-Aware Representations for Relevance Matching in Neural Information Retrieval. In *Proceedings of the 26th International Conference on World Wide Web Companion (WWW '17 Companion)*. International World Wide Web Conferences Steering Committee, Republic and Canton of Geneva, Switzerland, 799–800. https://doi.org/10.1145/3041021.3054258

[10] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. 2014. A Convolutional Neural Network for Modelling Sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Vol. 1. 655–665.

[11] Ryan Kiros, Yukun Zhu, Ruslan R Salakhutdinov, Richard Zemel, Raquel Urtasun, Antonio Torralba, and Sanja Fidler. 2015. Skip-Thought Vectors. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett (Eds.). Curran Associates, Inc., 3294–3302. http://papers.nips.cc/paper/5950-skip-thought-vectors.pdf

[12] Donald Metzler and W Bruce . 2004. Combining the language model and inference network approaches to retrieval. *Information processing & management* 40, 5 (2004), 735–750.

[13] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep Sentence Embedding Using Long Short-term Memory Networks: Analysis and Application to Information Retrieval. *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 24, 4 (April 2016), 694–707. https://doi.org/10.1109/TASLP.2016.2520371

[14] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. 2016. Text Matching as Image Recognition.. In *AAAI*. 2793–2799.

[15] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. DeepRank: A New Deep Architecture for Relevance Ranking in Information Retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management (CIKM '17)*. ACM, New York, NY, USA, 257–266. https://doi.org/10.1145/3132847.3132914

[16] Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933* (2016).

[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, 373–382.

[18] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*. ACM, New York, NY, USA, 373–374. https://doi.org/10.1145/2567948.2577348

[19] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning Semantic Representations Using Convolutional Neural Networks for Web Search. In *Proceedings of the 23rd International Conference on World Wide Web (WWW '14 Companion)*. ACM, New York, NY, USA, 373–374. https://doi.org/10.1145/2567948.2577348

[20] Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation* 28, 1 (1972), 11–21.

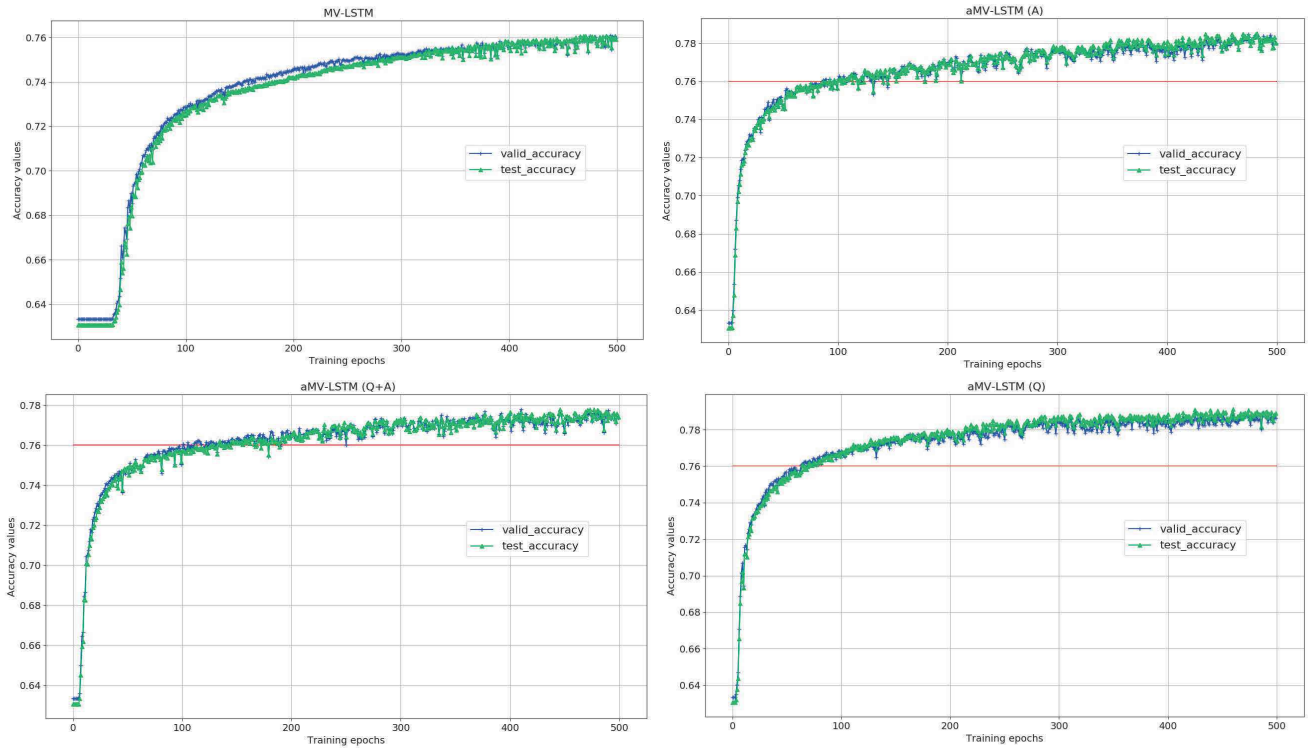
[21] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 27*, Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger (Eds.). Curran Associates, Inc., 3104–3112. http://papers.nips.cc/paper/5346-sequence-to-sequence-learning-with-neural-networks.pdf

[22] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. 2016. A Deep Architecture for Semantic Matching with Multiple Positional

Table 4: Comparison of the 3 first answers retrieved by the MV-LSTM model, among 21 possible answers that correspond to the question *"Where do crocodiles live?"*, sampled from the WikiQA dataset. We show how these answers are ranked by the proposed architectures of the aMV-LSTM model. (.) corresponds to the retrieval rank according to each model.

First answers	Predicted scores and rank values (.)				True *label*
	MV-LSTM	aMV-LSTM			
		Q	A	Q+A	
Crocodiles (subfamily Crocodylinae) or true crocodiles are large aquatic tetrapods that live throughout the tropics in Africa, Asia, the Americas and Australia.	(3) 0.3646	(1) 0.9003	(2) 0.2871	(3) 0.3640	1
Crocodiles have more webbing on the toes of the hind feet and can better tolerate saltwater due to specialized salt glands for filtering out salt, which are present but non-functioning in alligators.	(1) 0.8989	(2) 0.8826	(1) 3.5154	(1) 2.0885	0
Also when the crocodile 's mouth is closed, the large fourth tooth in the lower jaw fits into a constriction in the upper jaw.	(2) 0.6290	(4) -1.1083	(4) -0.2245	(6) -0.2575	0
They are carnivorous animals, feeding mostly on vertebrates such as fish, reptiles, birds and mammals, and sometimes on invertebrates such as molluscs and crustaceans, depending on species and age.	(6) -2.0848	(3) -0.6799	(3) 0.2831	(2) 0.5409	0

Figure 4: Comparison of the *validation* and *test accuracy* values, after each training epoch on the QuoraQP dataset, of the proposed aMV-LSTM model architectures compared with the MV-LSTM baseline. The red line on the curves corresponding to aMV-LSTM (A), aMV-LSTM (Q) and aMV-LSTM (Q+A) shows the accuracy value of the MV-LSTM model after training.



Sentence Representations.. In *AAAI*, Vol. 16. 2835–2841.

[23] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. 2017. End-to-End Neural Ad-hoc Ranking with Kernel Pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '17)*. ACM, New York, NY, USA, 55–64. https://doi.org/10.1145/3077136.3080809

[24] Liu Yang, Qingyao Ai, Jiafeng Guo, and W Bruce Croft. 2016. aNMM: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*. ACM, 287–296.

[25] 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 (EMNLP)*. Association for Computational Linguistics, Lisbon, Portugal.

[26] Zichao Yang, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 1480–1489.

[27] Wenpeng Yin, Hinrich Schütze, Bing Xiang, and Bowen Zhou. 2016. ABCNN: Attention-Based Convolutional Neural Network for Modeling Sentence Pairs. *Transactions of the Association of Computational Linguistics* 4, 1 (2016), 259–272.

[28] Peng Zhou, Zhenyu Qi, Suncong Zheng, Jiaming Xu, Hongyun Bao, and Bo Xu. 2016. Text classification improved by integrating bidirectional LSTM with two-dimensional max pooling. *arXiv preprint arXiv:1611.06639* (2016).