Linguistically-based Deep Unstructured Question Answering

Ahmad Aghaebrahimian
Charles University
Faculty of Mathematics and Physics
Institute of Formal and Applied Linguistics
Malostranske nam. 25, 11800 Praha 1, Czech Republic
ebrahimian@ufal.mff.cuni.cz

Abstract

In this paper, we propose a new linguistically-based approach to answering non-factoid open-domain questions from unstructured data. First, we elaborate on an architecture for textual encoding based on which we introduce a deep end-to-end neural model. This architecture benefits from a bilateral attention mechanism which helps the model to focus on a question and the answer sentence at the same time for phrasal answer extraction. Second, we feed the output of a constituency parser into the model directly and integrate linguistic constituents into the network to help it concentrate on chunks of an answer rather than on its single words for generating more natural output. By optimizing this architecture, we managed to obtain near-to-human-performance results and competitive to a state-of-the-art system on SQuAD and MS-MARCO datasets respectively.

1 Introduction

Reading, comprehending and reasoning over texts and answering a question about them (i.e. Question Answering) is a fundamental aspect of computational intelligence. Question Answering (QA), as a measure of intelligence, has been even suggested to replace Turing test (Clark and Etzioni, 2016).

The development of large datasets of QA in recent years (Hermann et al., 2015; Hill et al., 2015; Bordes et al., 2015; Rajpurkar et al., 2016) advanced the field especially for two significant branches of QA namely factoid (Aghaebrahimian and Jurčíček, 2016a,b) and non-factoid QA[1] (Rajpurkar et al., 2016). Non-factoid QA or QA over unstructured data is a somewhat new challenge in open-domain QA. A non-factoid QA system answers questions by reading and comprehending a context. The context in which we assume the answer is mentioned may have different granularities from a single sentence or paragraph to larger units of text. A QA system is supposed to extract a phrase answer from the provided paragraph or sentence depending on its granularity level.

The context for answering questions is usually extracted using an Information Retrieval (IR) technique. Then, a QA system should extract the best answer sentence. There are many studies about extracting answer sentences including but not limited to (He et al., 2015; He and Lin, 2016; Yih et al., 2013; Yu et al., 2016; Rao et al., 2016; Aghaebrahimian, 2017a).

Extracting the final or shortest possible answer from a set of candidate answer sentences is addressed in many studies as well (Zhang et al., 2017; Gong and Bowman, 2017; Shen et al., 2016; Weissenborn et al., 2017a). Instead of reasoning over and making inference on linguistic symbols (i.e., words or characters), almost all of these models use a neural architecture to encode contexts and questions into a vector representation and to reason over them.

A typical pattern in most of the current models is the use of a variant of uni- or bi-directional attention schemes (question to context and vice-versa) to encode the semantic content of questions' words with a focus on their context's words (Seo et al., 2016; Xiong et al., 2016; Weissenborn et al., 2017b; Chen et al., 2017; Wang et al., 2017). Compared to these models the novelty of our work is in explicitly conducting attention over both the context and the question for each candidate constituent[2] answer (every constituent

[1] While the answer to a non-factoid question is a chunk of one or more adjacent words, the answer to a factoid question is only an entity.

[2] Form now on and for the sake of brevity by constituents we mean linguistic constituents as they are referred to in Phrase Structure Grammar (Chomsky, 1957).

Proceedings of the 22nd Conference on Computational Natural Language Learning (CoNLL 2018), pages 433–443
Brussels, Belgium, October 31 - November 1, 2018.

Constituents Type	Training set	Development set
NP	59 %	62 %
ROOT	8 %	6 %
NNP	5 %	4 %
NN	4 %	2 %
JJ	3 %	1 %
VP	3 %	4 %
CD	3 %	2 %
PP	2 %	4 %
S	2 %	2 %
others	11 %(each $< 2\%$)	13 %(each $< 2\%$)

Table 1: The distribution of constituent types of answers in SQuAD training and development sets. For constituency parsing, we used the Standford CoreNLP tool (Manning et al., 2014). To see the full list of available constituency types in the dataset, please refer to Appendix A.

in the context). The fact that this is better than attending only to the question words is investigated and proved according to the results reported in Section 7.

Another observation is that a majority of recent studies are purely based on data science where one can barely see a linguistic intuition towards the problem. We show that a pure linguistic intuition could help neural reasoning and attention mechanisms to achieve quantitatively and qualitatively better results in QA.

By analyzing a human-generated QA dataset called SQuAD (Rajpurkar et al., 2016), we realized that people tend to answer questions in units called constituents (Table 1). We expect an answer to a question to be a valid constituent otherwise it would probably not be grammatical.

Constituents and Constituency relations are the bases of Phrase Structure Grammar first proposed by Noam Chomsky (Chomsky, 1957). Phrase Structure Grammar and many of its variants including Government and Binding theory (Chomsky, 1993) or Generalized and Head-driven Phrase Structure Grammar (Gazdar et al., 1994; Pollard and Sag, 1994) define hierarchical binary relations between the constituents of a text, and hence help to realize an exact and natural answer boundary for answer extraction.

Having these two points in mind and inspired by attentive pooling networks by Santos et al. (2016), we designed an attentive bilateral model and trained it on the constituents of questions and answers. We attempted to use some information from the parser, so to go beyond a simple word-based or vector-based representations. The results obtained by the model are near to human performance on SQuAD dataset and competitive to a state-of-the-art system on MS-MARCO dataset. The contributions of our work are:

- A bilateral linguistically-based attention model for Question Answering
- Integrating linguistic constituents into a DNN architecture for the QA task.

In the next section, we review some of recent QA systems with a focus on unstructured QA. In Section 3, we briefly explain the constituency types. Then in Section 4, we discuss the details of our system architecture. In Section 5, we talk about the datasets and the way we prepared them for training. In Sections 6 and 7, we describe the training details and present the results of our experiments. Finally, we explain some ablation studies and error analysis in Section 8 before we conclude in Section 9.

2 Related work

In recent years, QA has been largely benefited from the development of Deep Neural Network (DNN) architectures largely in the form of Convolution Neural Networks (CNN) (LeCun et al., 1998) or Recurrent Neural Networks (RNN) (Elman, 1990). QA systems based on semantic parsing (Clarke et al., 2010; Kwiatkowski et al., 2010), IR-based systems (Yao and Durme, 2014), cloze-type (Kadlec et al., 2016; Hermann et al., 2015), factoid (Aghaebrahimian and Jurčíček, 2016b; Bordes et al., 2015) and non-factoid systems (Aghaebrahimian, 2017a; Rajpurkar et al., 2016) are some of the QA variants that have been improved by DNNs. Among all of these varieties, factoid and non-factoid are two most widely studied branches of QA systems.

In Factoid QA like air traffic information systems (ATIS) or dialogue systems, we answer the questions by extracting an entity from a structured database like relational databases or knowledge graphs. In contrast, in non-factoid systems, answers are extracted mostly from unstructured data like Wikipedia.

Unstructured QA in recent years has been studied with a few distinguishable different settings such as answer selection, answer trigging, and answers extraction. In answer selection (Aghaebrahimian, 2017a; Wang et al., 2016; Yu et al.,

2016) and answer trigging (Jurczyk et al., 2016) the goal is to find the best answer sentence given each question. These answer sentences may be non-existent in the provided context for answer triggering. In answer extraction (Shen and Klakow, 2006; Sultan et al., 2016), we extract a chunk of a sentence as the shortest possible answer.

Answer selection can be used as a measure of machine comprehension (Kadlec et al., 2016; Hermann et al., 2015). In this setting, a typical QA system reads a text and then answers either multiple-answer (cloze-type) or free-text questions. Cloze-type answers are limited to multiple distinct entities (usually 4 or 5) while a span of words answers free-text questions.

The performance of cloze-type systems is a good indication of machine comprehension. However, in QA systems for a real-life application like in dialogue systems or scientists' assistants, the answers, their boundaries and their types (e.g., proper noun, adjective or noun phrase) are not known in advance, and it makes this type of QA more challenging. In this setting, free-text QA or QA over unstructured data (Aghaebrahimian, 2017b; Rajpurkar et al., 2016; Cui et al., 2016) is advocated where answers are spans of multiple consecutive words in large repositories of textual data like Wikipedia.

Many successful studies have been performed for free-text (i.e., phrase) answer extraction from SQuAD since its release in 2016. Almost all of these models benefited from a form of DNN architecture and a majority of them integrated a kind of the attention mechanism. Some of these studies integrated attention to predicting the physical location of answers (Xiong et al., 2016; Cui et al., 2016; Hu et al., 2017; Seo et al., 2016). Others made an effort to find a match between queries and their contexts (Cui et al., 2016) or to compute a global distribution over the tokens in the context given a query (Wang and Jiang, 2016). Still, some other models integrated other mechanisms like memory networks (Pan et al., 2017) or reinforcement learning (Shen et al., 2016) to enhance their attention performance.

To add to these efforts, we intend to propose a new perspective on using attention and to enhance it by using linguistic constituents as a linguistically motivated feature.

3 Linguistic Constituents

There is hardly a universal agreement upon the definition of the term 'constituent'. In general, a constituent is an inseparable unit that can appear in different places of a sentence. Instead of defining what a constituent is, linguists define a set of experiments such as replacement or expansion to distinguish between constituents and non-constituents.

For instance, let's consider the sentence 'Plans for the relay were announced on April 26, 2007, in Beijing, China.' We can replace or expand some of its constituents and rephrase the sentence as 'on April 26, 2007, plans for the relay were announced , in Beijing, China.' or 'Plans for the great and important relay were announced on April 26, 2007, in Beijing, China.' while we are sure that these rephrases are not only both syntactically and semantically correct but also convey the same meaning as the original sentence. Some of the earliest works which tried to integrate more linguistic structures into QA are (Zhang et al., 2017; Xie and Eric, 2017). Using TreeLSTM, Zhang et al. (2017) tried to integrate linguistic structure into QA implicitly. At the prediction step, they used pointer network (Vinyals et al., 2017) to detect the beginning and the end of answer chunks. In contrast, (Xie and Eric, 2017) explicitly modeled candidate answers as sequences of constituents by encoding individual constituents using a chain of-trees LSTM (CT-LSTM) and tree-guided attention mechanism. However, their formulation of constituents is more complicated than ours and as we will see, a direct use of constituents as answer chunk is much less complicated and yields better results.

4 System Architecture

In this section, we describe how to represent questions, sentences and answers in vector space in Subsection 4.1 and then we train the vectors in Subsection 4.2 using a specific loss function and distance measure.

4.1 Representation Learning

Our goal is to extract constituent answers by loading their vector representations with the semantic content of their question and their containing answer sentence. To achieve this end, we integrated a bilateral attention mechanism into our model which lets us estimate a joint vector representation

between answers when they are attending to questions' constituents and when they are attending to sentences' constituents.

To encode the semantic information in questions and sentences, we used a simple encoding unit (see Equations 1 to 6 and Figure 1). In this unit, $W_k \in \mathbb{R}^{|V|}$ are words in one-hot vector representations where k^{th} element of each vector is one and others are 0. V are all vocabularies in training questions and answers. $E \in \mathbb{R}^{|V| \times d_e}$ is the embedding matrix and d_e is the embedding dimension. The product of the multiplication in Equation 1 is the word embeddings in which each cell $W_{i,t}$ is the word in time step t in sample i. $W_{i,t}$ is the input of forward and backward RNN cells in Equations 2 and 4.

As RNN cell, we used Long Short-Term Memory architecture (LSTM) (Hochreiter and Schmidhuber, 1997). Pan et al. (2017) and Hu et al. (2017) show that bi-directional LSTM architectures provide more accurate representations of textual data. The common practice to form a bidirectional LSTM is to concatenate the last vectors in forward and backward LSTMs. Instead, we used a stepwise max pooling (SWMP) mechanism which takes the most important vectors from forward and backward LSTMs in Equations 3 and 5 and concatenate them in Equations 6.

$$W_{i,t} = E^\top W_k \tag{1}$$
$$\overrightarrow{enc}_{i,t} = LSTM(\overrightarrow{enc}_{i,t-1}, W_{i,t}) \tag{2}$$
$$\overrightarrow{enc}_i = SWMP(\overrightarrow{enc}_{i,t}) \tag{3}$$
$$\overleftarrow{enc}_{i,t} = LSTM(\overleftarrow{enc}_{i,t+1}, W_{i,t}) \tag{4}$$
$$\overleftarrow{enc}_i = SWMP(\overleftarrow{enc}_{i,t}) \tag{5}$$
$$enc_i = [\overrightarrow{enc}_i; \overleftarrow{enc}_i] \tag{6}$$

Using our encoding unit we encode questions and sentences and then concatenate the resulted vectors to generate a joint representation of questions and their answer sentences in Equation 7.

$$enc_i^{QS} = [enc_i^Q; enc_i^S] \tag{7}$$

In the next step, we need to encode the constituent answers. Our answer encoding unit has two modules, one with attention on questions enc_i^Q (equations 8-15) and the other with attention on sentences enc_i^S (equations 16-23). In both modules, we used an architecture similar to the one in the encoding unit with an additional attention unit.

In the answer encoding unit, again the input to LSTM cells are word embeddings generated by lookup table $W_{i,t}^A$. Two attention layers in this unit receive the output sequences of the forward and backward LSTM cells and focus once on questions and once on sentences. This is done using an attention mechanism similar to the one proposed by Santos et al. (2016).

In the end, the vectors generated by these two modules are concatenated in $h_i^{A_{QS}} = [h_i^{A_Q}; h_i^{A_S}]$[3] to form a general attentive representation of constituents with respect to their corresponding questions and sentences.

At training time, we try to learn the vector representations of questions, sentences, and their constituents jointly. However, we like to learn the vectors in a way that leads to a small distance between questions and their true constituents and a long distance between them and their false constituents. For this purpose, for each pair of question and sentence, we compute one true answer A_{QS}^+ and some false answer A_{QS}^- vectors.

We generated these vectors by passing a correct constituent A^+ and a random wrong constituent A^- through question-attentive (see equations 8-15) and sentence-attentive (see equations 16-23) modules and by concatenating the outputs.

4.2 Training

In this section, we train our model. Given a question and the constituents associated with its answer sentence, the model generates a score for each constituent. The score is an estimate of how similar the constituent to the gold answer is. Taking the argmax over the scores, the model returns the id of its true predicted constituent.

To train the model, we need to compute the distance between questions and their true constituents and to contrast it with the distance between questions and their false constituents.

There are various measures of distance or similarity between two vectors each with its own merits. Feng et al. (2015) did an exhaustive study on different distance measures for text classification and proposed some new measures including the Geometric mean of Euclidean and Sigmoid Dot product (GESD) (Formula 1) which outperformed other measures in their study.

We integrated GESD in our work to estimate

[3] All concatenations are performed on the last layer (i.e. data dimensions).

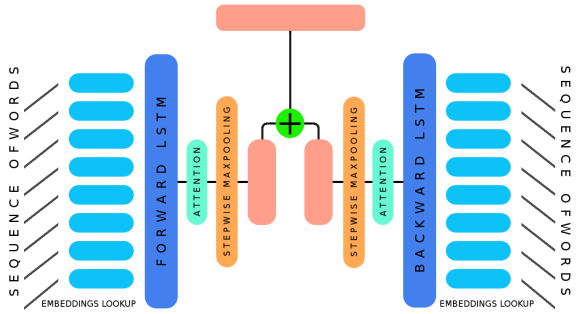

Figure 1: The encoding unit. The Embedding lookup uses pre-trained Glove word vectors (Pennington et al., 2014) and updates them through training. The output is the concatenation of max-pooled vectors of LSTM encoders.

$$W^A_{i,t} = E^\top W^A_k \quad (8)$$
$$\overrightarrow{h_{i,t}} = LSTM(\overrightarrow{h_{i,t-1}}, W^A_{i,t}) \quad (9)$$
$$\overrightarrow{h^{A_Q}_{i,t}} = ATT(\overrightarrow{h_{i,t}}, \mathbf{enc^Q_i}) \quad (10)$$
$$\overrightarrow{h^{A_Q}_{i}} = SWMP(\overrightarrow{h^{A_Q}_{i,t}}) \quad (11)$$
$$\overleftarrow{h^{A_Q}_{i,t}} = LSTM(\overleftarrow{h_{i,t+1}}, W^A_{i,t}) \quad (12)$$
$$\overleftarrow{h^{A_Q}_{i,t}} = ATT(\overleftarrow{h^{A_Q}_{i,t}}, \mathbf{enc^Q_i}) \quad (13)$$
$$\overleftarrow{h^{A_Q}_{i}} = SWMP(\overleftarrow{h^{A_Q}_{i,t}}) \quad (14)$$
$$h^{A_Q}_i = [\overrightarrow{h^{A_Q}_i}; \overleftarrow{h^{A_Q}_i}] \quad (15)$$

$$W^A_{i,t} = E^\top W^A_k \quad (16)$$
$$\overrightarrow{h_{i,t}} = LSTM(\overrightarrow{h_{i,t-1}}, W^A_{i,t}) \quad (17)$$
$$\overrightarrow{h^{A_S}_{i,t}} = ATT(\overrightarrow{h_{i,t}}, \mathbf{enc^S_i}) \quad (18)$$
$$\overrightarrow{h^{A_S}_{i}} = SWMP(\overrightarrow{h^{A_S}_{i,t}}) \quad (19)$$
$$\overleftarrow{h^{A_S}_{i,t}} = LSTM(\overleftarrow{h_{i,t+1}}, W^A_{i,t}) \quad (20)$$
$$\overleftarrow{h^{A_S}_{i,t}} = ATT(\overleftarrow{h^{A_S}_{i,t}}, \mathbf{enc^S_i}) \quad (21)$$
$$\overleftarrow{h^{A_S}_{i}} = SWMP(\overleftarrow{h^{A_S}_{i,t}}) \quad (22)$$
$$h^{A_S}_i = [\overrightarrow{h^{A_S}_i}; \overleftarrow{h^{A_S}_i}] \quad (23)$$

Figure 2: Question-aware (equations 16-23) and sentence-aware (equations 8-15) encoding

the distance between questions and their true and false constituents. GESD linearly combines two other measures called L2-norm and inner product. L2-norm is the forward-line semantic distance between two sentences and inner product measures the angle between two sentence vectors.

$$DIS(Q, A) = \frac{1}{1+exp(-(Q.A))} * \frac{1}{1+||Q-A||}$$

Formula 1: The distance between Question (Q) and Answer (A) vectors.

Now everything is ready to train the model. The overall system architecture is illustrated in Figure 3. We use Hinge loss function (Formula 2) to estimate the loss on each question-answer combination. Hinge function increases the loss with the distance between a question and its true constituents while decreases it with the distance between a question and its false constituents. In Equation 2, enc^{QS} is the joint vector representation of questions and their answer sentences, $h_i^{A^-_{QS}}$ is the vector of false answers, $h^{A^+_{QS}}$ is the vectors of true answers. Finally, m is the margin between positive and negative answers. It makes a trade-off between the mistakes in positive and negative classes.

$$\mathcal{L} = \sum_i \max(0, m + DIS(enc^{QS}_i, h_i^{A^+_{QS}}) - DIS(enc^{QS}_i, h_i^{A^-_{QS}}))$$

Formula 2: Hinge function. m is the margin, A^-_{QS} are false and A^+_{QS} are true answers.

5 Datasets

The Stanford Question Answering Dataset (SQuAD) (Rajpurkar et al., 2016) is a dataset for sentence-level (i.e. answer selection) and word-level (i.e. answer extraction) QA. It includes 107,785 question-answer pairs synthesized by crowd workers on 536 Wikipedia articles. The dataset is randomly shuffled and divided into training (80%), development (10%) and test (10%) sets. Due to its large number of questions compared to previous datasets (Hirschman et al., 1999; Richardson et al., 2013), it is considered a good testbed for data-intensive QA methods. The answers in SQuAD are categorized into ten types including Person, Date, Location, etc (Rajpurkar et al., 2016). However, there are no statistics available on the constituent type of each answer.

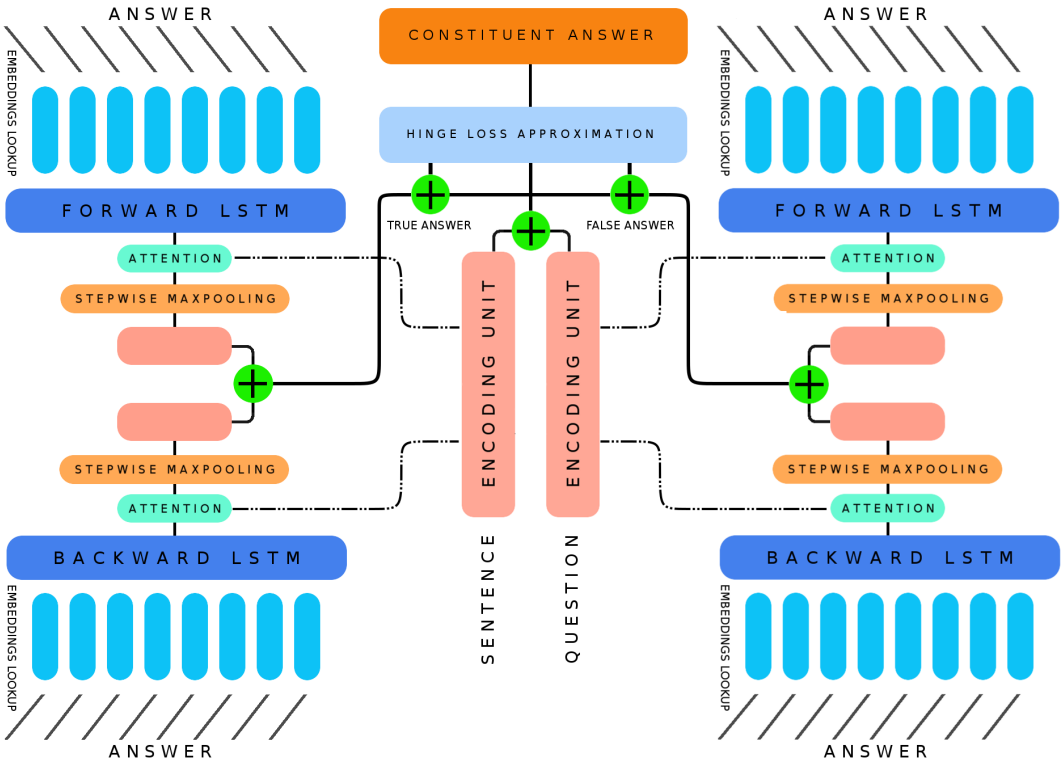

Figure 3: The system architecture. Two answer modules, one with attention on questions and the other on sentences, provide a joint representation containing all required information with respect to questions and sentences for making inference on true constituents.

To control the vocabulary size we needed to eliminate redundant numeric values, but at the same time, we wanted to parse the contents, and we needed to keep the semantic values of numeric tokens. Hence to preprocess the questions and sentences in the dataset, we removed all non-alphanumeric characters from all contents and then replaced numeric values with '9'. Then we used CoreNLP tool (Manning et al., 2014) to tokenize and to perform constituency parsing on the contents.

After extracting constituents from the tree of sentences and comparing them with gold answers, we realized that 72% of the answers are constituents. Other 21% of the answers had slight divergences from a constituent, like lacking or having a determiner or punctuation mark which were eventually going to be disregarded in the official evaluation script. The remaining 7% was a combination of two smaller constituents or a part of a larger one.

In the training set, to use constituents as answers, we replaced non-matching answers with the smallest and most similar constituents. Since at the evaluation time, we needed the gold answers and not their replaced constituents, we did not change the answers in the development set.

We extracted a total number of 48 different constituents types including both terminal and non-terminal ones from SQuAD. The percentage of each constituent type in training and development sets are presented in Table 1. The figures for development set are computed only based on exact match answers.

We used SQuAD's development set for testing the system and reporting the results. To prepare the dataset for training and evaluating our system we used a state-of-the-art answer sentence selection system (Aghaebrahimian, 2017a) to extract the best answer sentences. The system provides us the best sentence with 94.4 % accuracy given each question. After pre-processing the sentence as explained above, we extracted its constituents and trained the model using the correct constituents as true and other constituents as negative samples.

At test time, we used the same procedure to extract the constituents, but we used gold answers as they

are without substituting non-matching answer-constituents. Then, we added other constituents as negative samples.

For evaluation purpose, we used SQuAD's official evaluation script which computes the exact match and the F1 score. The exact match is the percentage of predictions which exactly match the gold answer and the F1 (Macro-averaged) score is the average overlap between the prediction and ground truth answer while treating them both as bags of tokens, and computing their F1.

To experiment our model furthermore, we used MS-MARCO dataset (Nguyen et al., 2015). As a machine comprehension dataset, MS-MARCO has two fundamental differences with SQuAD. Every question in MS-MARCO has several passages from which the best answer should be retrieved. Moreover, the answers in MS-MARCO are not necessarily sub-spans of the provided contexts so that BLEU and ROUGE are used as the metrics in the official tool of MS-MARCO evaluation. During training we used the highest BLEU scored constituent as the answer and in the evaluation, we computed the BLEU and ROUGE scores of the constituents selected by the system. As the results in Table 7 show, our system obtained competitive results to another state-of-the-art system trained on the same dataset.

6 Experiment

To evaluate our new architecture and to see how integrating linguistic constituents affects its performance we set up two settings. We designed one setting for evaluating the effect of using constituents instead of words (constituent-base vs. word-base) and another to evaluate the effect of using attention mechanism on top of vector training modules (uni- vs. bi-attention). Therefore we conducted four experiments on both datasets or eight experiments in total.

In the constituent-base setting, we generated the training and test data as described in Section 5. In word-base however, we replaced constituents with the tokens in answer sentences for both train and test sets and trained our model to compute two scores for initial and final positions of answer chunks. In the constituent-base setting at test time, we directly used the predicted constituent as the final answer. In the word-base setting, however, we got the final answer using the highest-scored words for initial and final positions.

We also investigated the effect of bilateral attention on the model performance. In the bi-attention model, we used the model as described in Section 4. In the uni-attention model, we eliminated the attention on sentences and only used the module for attention on questions.

For training our model, we used 300-dimensional pre-trained Glove word vectors (Pennington et al., 2014) to generate the embedding matrix and kept the embedding matrix updated through training. We used 128-dimensional LSTMs for all recurrent networks and used 'Adam' with parameters learning rate=0.001, $\beta_1 = 0.9$, $\beta_2 = 0.999$ for optimization. We set batch size to 32 and dropout rate to 0.5 for all LSTMs and embedding layers. We performed the accuracy check only on the first best answer.

7 Result

The results of our experiments are summarized in Table 2. By contrasting the results of uni-/bi-attention and word/constituent-base models, we can see that the proposed bi-attention mechanism with linguistic constituents integrated into it makes a significant improvement on answer extraction. Another interesting observation is that the Exact Match metric benefits from restriction to constituents as answers. Concerning the MS-MARCO dataset, the results are competitive to a state-of-the-art system tested on the same dataset (Wang et al., 2017).

8 Ablation and Error Analysis

In this section, we analyze the SQuAD concerning answer distribution over different query types. Table 1 shows that the NP type constituents are the most prominent type among all other answers. However, to investigate the importance of other types in overall system performance, we performed an ablation study where we studied the influence of each constituent type on overall accuracy. The results are presented in Figure 4. We also studied how much the model succeeded in retrieving answers from each type. The results are presented in the same figure. This table also shows how often does the method succeed in cases where the correct answer is, in fact, a constituent span.

As seen in Figure 4, the answers are mostly singular noun phrases after which with a significant difference are proper nouns, verb phrases, and prepo-

	SQuAD Development set		MS-MARCO Evaluation set	
	Exact-match(%)	F1(%)	BLEU	ROUGE
Logistic Regression (Rajpurkar et al., 2016)	40.00 %	51.00 %	-	-
Uni-Attention Word-base (this work)	55.12 %	57.98 %	35.6	35.1
Bi-Attention Word-base (this work)	59.84 %	63.08 %	38.1	38.4
Uni-Attention Constituency-base (this work)	73.82 %	77.43 %	39.6	39.9
TreeLSTM (Zhang et al., 2017)	69.10 %	78.38 %	-	-
BIDAF (Seo et al., 2016)	72.6 %	80.7 %	-	-
CCNN (Xie and Eric, 2017)	74.1 %	82.6 %	-	-
R-net (Wang et al., 2017)	75.60 %	82.80 %	42.2	42.9
Bi-Attention Constituency-base (this work)	**80.72 %**	**83.25 %**	42.1	42.7
Human Performance (Rajpurkar et al., 2016)	82.30 %	91.22 %	-	-

Table 2: The performances of different models in the exact match and F1 metrics for SQuAD and BLEU and ROUGE for the MS-MARCO dataset.

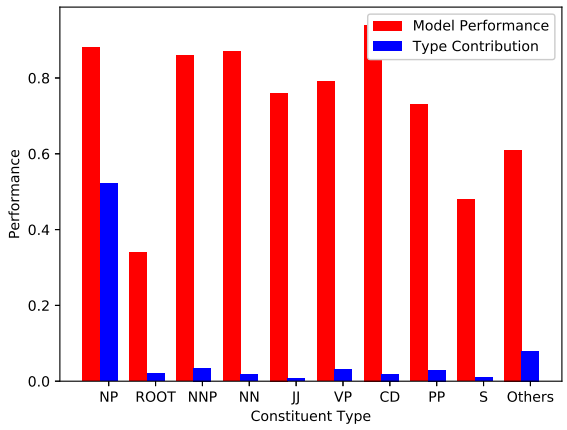

Figure 4: Blue lines are the contribution of each type in the overall system performance using the best model and at the convergence time. Red bars represent the performance of the model in retrieving each constituent type when the model is converged. Performance is expressed in the exact match metric (%). As a guide to how to read the chars, the first blue line for NP type says that 50% of all correctly extracted answers by the system are NP type-answers. The red line of the same type says that our system managed to retrieve about 87% of all NP-type answers in the dataset.

sitional phrases. We can also see how the model performed for each constituent. It seems that extracting cardinal numbers is much easier for the model than retrieving roots or full sentences.

An analysis of the errors shows that false answer sentence, non-constituent answers, parsing errors, overlapping constituents and unknown words are the primary reasons for the mistakes made by our system. The sentence selection process brought about six percent incorrect answers. The next primary reason for making mistakes is the constituents which contain other smaller constituents. While in all cases we extract the smallest constituent, in about three percent of overlapping constituents the more extended ones are the correct answer. Parsing errors where the constituents are not retrieved correctly and unknown words where the embeddings are not trained properly are responsible for other four percent of the errors. Finally, non-constituent answers led to around eight percent false answers in the system output.

9 Conclusion and Future Work

We described a new linguistically-based end-to-end DNN for Question Answering from unstructured data. This model is a neural formulation in which linguistic constituents are explicitly modeled. It operates an LSTM over the constituents and uses the resulting hidden states to attend both to question and to the encompassing context sentence, thereby enriching the constituents representation with both. The use of constituents instead of an arbitrary string of words in answers improves the system performance in three ways.

First, it increases the precision of the system. By looking at the small gap between the F1 and the exact match metrics in our system and compare it to the ones for the other systems, we can see that the ratio of exact-match answers in our system is higher than that of the other ones.

Second, it helps an answer to look more like a human-generated one. Considering prediction and ground truth as bags of tokens, the F1 (Macro-

averaged) metric computes the average overlap between the prediction and ground truth answer. While a predicted answer may have a full overlap with the ground truth hence gains a high F1 score, due to the irrelevant words it contains, it poses an incoherent answer to users. The longer the gap between exact match and F1 measures, the more inappropriate words appear in answers. This is primarily an essential factor in the overall quality of dialogue QA systems where users expect to receive a natural and human-generated-like answer. Last but not least, imposing constraints on the candidate space, limit errors and make the system more efficient by decreasing the search space and weeding out non-relevant answers. In the future, we plan to integrate dependency relations into the model by designing a larger model and evaluating it on other QA datasets.

Acknowledgments

This research was partially funded by the Ministry of Education, Youth and Sports of the Czech Republic (project LM2015071), by Charles University SVV project number 260 453 and GAUK 207-10/250098 of Charles University in Prague.

References

Ahmad Aghaebrahimian. 2017a. Constrained deep answer sentence selection. In *Proceedings of the 20th International Conference on Text, Speech, and Dialogue (TSD)*.

Ahmad Aghaebrahimian. 2017b. Hybrid Deep Open-Domain Question Answering. In *Proceedings of the 8th Language and Technology Conference (LTC)*.

Ahmad Aghaebrahimian and Filip Jurčíček. 2016a. Constraint-Based Open-Domain Question Answering Using Knowledge Graph Search. In *Proceedings of the 19th International Conference on Text, Speech and Dialogue (TSD), LNAI 9924*.

Ahmad Aghaebrahimian and Filip Jurčíček. 2016b. Open-domain Factoid Question Answering via Knowledge Graph Search. In *Proceedings of the Workshop on Human-Computer Question Answering, The North American Chapter of the Association for Computational Linguistics (NAACL)*.

Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. 2015. Large-scale simple question answering with memory networks. *arXiv preprint arXiv:1506.02075*.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. 2017. Reading wikipedia to answer open- domain questions. In *arXiv:1704.00051, 2017a*.

Noam Chomsky. 1957. *Syntactic structures*. The Hague, Paris: Mouton.

Noam Chomsky. 1993. *Lectures on Government and Binding: The Pisa Lectures*. Mouton de Gruyter.

Peter Clark and Oren Etzioni. 2016. My computer is an honor student but how intelligent is it? standardized tests as a measure of ai. *AI Magazine*.

James Clarke, Dan Goldwasser, Ming-Wei Chang, and Dan Roth. 2010. Driving semantic parsing from the worlds response. *In Proceedings of the Conference on Computational Natural Language Learning (CoNLL)*.

Yiming Cui, Zhipeng Chen, Si Wei, Shijin Wang, Ting Liu, and Guoping Hu. 2016. Attention-over-attention neural networks for reading comprehension. *arXiv:1607.04423*.

Jeffry L. Elman. 1990. Finding structure in time. *Cognitive Science*, 14(2):179–211.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, and Bowen Zhou. 2015. Applying deep learning to answer selection: a study and an open task. In *Proceedings of IEEE ASRU Workshop*.

Gerald Gazdar, Ewan H. Klein, Geoffrey K. Pullum, and Ivan A. Sag. 1994. *Generalized Phrase Structure Grammar*. Blackwell, Oxford.

Yichen Gong and Samuel R Bowman. 2017. Ruminating reader: Reasoning with gated multi-hop attention. *arXiv:1704.07415*.

Hua He, Kevin Gimpel, and Jimmy Lin. 2015. Multi-perspective sentence similarity modeling with convolutional neural networks. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing(EMNLP)*.

Hua He and Jimmy Lin. 2016. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of The North American Chapter of the Association for Computational Linguistics (NAACL)*.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. 2015. Teaching machines to read and comprehend. In *Proceedings of Advances in Neural Information Processing Systems*.

Felix Hill, Antoine Bordes, Sumit Chopra, and Jason Weston. 2015. The goldilocks principle: Reading children's books with explicit memory representations. *arXiv :1511.02301*.

Lynette Hirschman, Marc Light, Eric Breck, and John D. Burger. 1999. Deep read: A reading comprehension system. In *Proceedings of Association for Computational Linguistics (ACL)*.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural Comput.*, 9(8).

Minghao Hu, Yuxing Peng, and Xipeng Qiu. 2017. Reinforced mnemonic reader for machine comprehension. *arXiv:1705.02798*.

Tomasz Jurczyk, Michael Zhai, and Jinho D. Choi. 2016. Selqa: A new benchmark for selection- based question answering. In *Proceedings of the 28th International Conference on Tools with Artificial Intelligence*.

Rudolf Kadlec, Martin Schmid, Ondrej Bajgar, and Jan Kleindienst. 2016. Text understanding with the attention sum reader network. In *Proceedings of the Association for Computational Linguistics*.

Tom Kwiatkowski, Luke Zettlemoyer, Sharon Goldwater, and Mark Steedman. 2010. Inducing probabilistic ccg grammars from logical form with higher-order unification. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*.

Yann LeCun, Leon Bottou, Yoshua Bengio, and Patrick Haffner. 1998. Gradient-based learning applied to document recognition. In *Proceedings of the IEEE*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *proceedings of the Association for Computational Linguistics (ACL) System Demonstrations*.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. 2015. Ms marco: A human generated machine reading comprehension dataset. *CoRR, abs/1611.09268*.

Boyuan Pan, Hao Li, Zhou Zhao, Bin Cao, Deng Cai, and Xiaofei He. 2017. Memen: Multi-layer embedding with memory networks for machine comprehension. *arXiv:1707.09098*.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *Procedings of the conference Empirical Methods in Natural Language Processing (EMNLP)*.

Carl Pollard and Ivan A. Sag. 1994. *Head-driven phrase structure grammar*. University of Chicago Press, Chicago.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text. *arXiv:1606.05250*.

Jinfeng Rao, Hua He, and Jimmy Lin. 2016. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16.

Matthew Richardson, Burges, Christopher J.C., and Renshaw Erin. 2013. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of Empirical Methods in Natural Language Processing(EMNLP)*.

Cicero dos Santos, Ming Tan, Bing Xiang, and Bowen Zhou. 2016. Attentive pooling networks. *arXiv:1602.03609v1*.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension. *arXiv:1611.01603*.

Dan Shen and Dietrich Klakow. 2006. Exploring correlation of dependency relation paths for answer extraction. In *Proceedings of the 21st International Conference on Computational Linguistics*.

Yelong Shen, Po-Sen Huang, Jianfeng Gao, and Weizhu Chen. 2016. Reasonet: Learning to stop reading in machine comprehension. In *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016)*.

Md Arafat Sultan, Vittorio Castelli, and Radu Florian. 2016. A joint model for answer sentence ranking and answer extraction. In *Transactions of the Association for Computational Linguistics*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2017. Pointer networks. In *Proceedings of NIPS*.

Shuohang Wang and Jing Jiang. 2016. Machine comprehension using match-lstm and answer pointer. *arXiv:1608.07905*.

Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. 2017. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the Association for Computational Linguistics(ACL)*.

Zhiguo Wang, Haitao Mi, and Abraham Ittycheriah. 2016. Sentence similarity learning by lexical decomposition and composition. *arXiv:1602.07019*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017a. Fastqa: A simple and efficient neural architecture for question answering. *arXiv:1703.04816*.

Dirk Weissenborn, Georg Wiese, and Laura Seiffe. 2017b. Making neural qa as simple as possible but not simpler. In *Proceedings of the Computational Natural Language Learning (CoNLL)*.

Pengtao Xie and Xing Eric. 2017. A constituent-centric neural architecture for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*.

Caiming Xiong, Victor Zhong, and Richard Socher. 2016. Dynamic coattention networks for question answering. *arXiv:1611.01604*.

Xuchen Yao and Benjamin Van Durme. 2014. Information extraction over structured data: Question answering with freebase. *In Proceedings of Association for Computational Linguistics*.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. 2013. Question answering using enhanced lexical semantic models. *In Proceedings of Association for Computational Linguistics(ACL)*.

Lei Yu, Karl Moritz Hermann, Phil Blunsom, and Stephen Pulman. 2016. Deep learning for answer sentence selection. *Proceedings of the NIPS Deep Learning Workshop*.

Junbei Zhang, Xiaodan Zhu, Qian Chen, Lirong Dai, and Hui Jiang. 2017. Exploring question understanding and adaptation in neural-network-based question answering. *arXiv:1703.04617*.

A Full Constituent types

Other constituent types include: NP, ROOT, NNP, NN, JJ, VP, CD, PP, S, NNS, ADJP, SBAR, NP-TMP, QP, ADVP, VBG, DT, VBN, IN, NNPS, VB, RB, VBD, VBZ, JJR, VBP, UCP, X, CC, FRAG, WHNP, JJS, NAC, NX, FW, TO, RBR, PDT, PRN, INTJ, PRT, WHPP, PRP, SINV, WHADJP, MD, RRC, WHADVP

For a description of each type please refer to (Manning et al., 2014)