Understanding Multi-Perspective Context Matching for Machine Comprehension

Cindy Catherine Orozco Bohorquez
Institute for Computational and Mathematical Engineering (ICME)
Leland Stanford Junior University
Stanford, CA 94305
orozcocc@stanford.edu
CodaLab: orozcocc
Leaderboard: cindyrella
Contribution: 100%

Abstract

Question-answer prediction has been one of the most desired tasks in a machine since the beginning of artificial intelligence. We study a recent model for reading comprehension called Multi-Perspective Context Matching. Given a question, we find the answer span in a context paragraph. To do so an encoder applies attention of the relevance of the context words with respect to the query and process with 2 different bidirectional LSTMs question and answer words. Later a decoder uses attention to identify different perspectives of the context words, and uses a bidirectional LSTM to aggregate the information. Lastly, we combine all the hidden layers in order to predict using 2 independent softmax the start and end of the answer. We tested 2 different set of parameters. Although we do not achieve state of the art or even standard performance, the analysis of the models show two extremely common pitfalls in machine learning that are worth studying: over-fitting and over-training. A discussion and ways to avoid this phenomenon in future work is presented.

1 Introduction

One of the most persistent challenges in creating automated machines is Question answering tasks. It is crucial to have entities that can comprehend a paragraph at the point that for any related question, they can infer its closest answer. It not only requires deeply understanding of the meaning of each word, but also inference of their relations with the rest of the text body. Early developments, such as Quarc [1], involve a set of lexicographical and semantical rules to identify the answer. Therefore, depending on the question word, Why, How, Who, Where, What and so on, it applies a different heuristic analysis of each word. As we might expect, this approach requires a lot of human intervention and therefore it is really difficult to generalize. Nevertheless it shows that different types of questions relate with different parts of speech in the text.

In 2010, the reading comprehension group at University of Washington [2] presented a set of expectations for Question answering algorithms, given its enormous importance in the evaluation of any Natural Language Processing methodology. These criteria reflect the unacceptable performance of most of the methods at the time, such as not a totally automated process without relying in complicated subroutines or human tunning, lack of adaptability for different datasets genres and sizes and absence of learning from the experience and feedback of users.

With the explosion of machine learning applications using neural networks and deep learning in the last decade, the concerns about machine comprehension seemed an incredible fit for the application

of this complex learning algorithms. Nevertheless, in order to have trustworthy results, neural networks require a vast amount of training examples, and as we could expect, it is extremely difficult to find a reliable dataset that correspond to meaningful question, context paragraph and answer span triples. One of the most remarkable datasets was presented in 2015 [3] based on "93k articles from the CNN and 220k articles from the Daily Mail website"[1], generating over a million of queries. Their approach uses the bullet points that summarize each news article, and transforms them into questions removing one word at a time.

Nevertheless, the CNN-Daily Mail dataset does not come from a human questioning process, and therefore the queries do not cover the richness and difficulty of the real life examples. To overcome these difficulties, a year later it was presented the Stanford Question Answering Dataset (SQuAD) [4] , which based on 536 Wikipedia articles, utilized crowd workers to generate over 100 000 queries. Although its genre diversity and size are not as impressive as the CNN-Daily Mail dataset, the complexity of its questions require a model to develop more structured inferences and therefore it is a better evaluation set.

After the spread of datasets as CNN-Daily Mail and SQuAD, multiple research groups have trained NLP models based on neural networks with extraordinary results. Notice that determining the span of the answer in a context paragraph is a much difficult task than just returning one-word answer or choosing among multiple possibilities. If the length of the context is n, the former has complexity $O(n^2)$ whereas the latter has complexity $O(n)$. Looking for the solution span requires comprehend each of the words in the context and the question and relate them. Therefore most of the models lately proposed replicate the encoder - decoder or Seq2seq models in machine translation [5]. Basically, it uses recurrent neural networks(RNN) to encode both question and context paragraph, applies attention to weight them and gives the span using an additional RNN. What makes each approach different is the type of attention used. [6] introduces a dynamic co-attention, where both context and question words are weighted with respect to its scope in the contiguous set of words. [7] creates chunks or subsets of the context data as answer candidates and tends to use part of speech patterns seen in training time to guess the best candidate.

In this project we focus on Multi-Perspective Context Matching [8], where 2 attention algorithms are installed. The first one is a filter layer to reduce all the redundant information on the passage using a relevance similarity between the context and the question word vectors. This is with the aim of that the answer remains in a small span with respect of the whole paragraph. The second attention method is applied after encoding the context and question words using a RNN, and in this case we compare the context with the question under multiple perspectives, i.e. multiplying each component of the hidden vectors by learned weights and then computing similarity between question and answer. Different sets of learned weights imply different perspectives to see each interaction.

On the following sections, we will define the problem to solve and the methodology used. Then we will explain step by step the Multi-Perspective Context Matching approach. Afterwards, we present results of the 2 models we have learned and lastly we analyze the causes of its poor performance and lessons to avoid it in the future.

2 Problem

We can re-frame the problem as follows: given a pair of string sequences (Q, C), where Q is a question and C is a context paragraph, find the span $start \leq end \in [0, |C|)$ such that the chain $[C_{start}, C_{start+1}, ..., C_{end}]$ answers Q.

In general, the answer can be in any subinterval $k_i \leq k_j \in [0, |C|)$ therefore there are $O(|C|^2)$ options. Estimating the conditional probability for each combination of $(start, end)$ given a set of Question -Context (Q, C) is a difficult task since it requires a training dataset where each combination is probable and the number of possibilities to handle is enormous.

Therefore, in this case we simplify the problem assuming that the position of $start$ and end are independent. Then, in order to choose $start$ and end properly, we estimate the probability of any context-word index $k \in [0, |C|)$ being the start or the end (independently) of the answer and we

[1] [3]

assign

$$start_{pred} = \arg \max_{k \in [0,|C|)} P(start = k|(Q, P)) \tag{1}$$

$$end_{pred} = \arg \max_{k \in [0,|C|)} P(end = k|(Q, P)) \tag{2}$$

Since it is possible $start_{pred} > end_{pred}$, we return as answer for the user $start = \min(start_{pred}, end_{pred})$ and $end = \max(start_{pred}, end_{pred})$

3 Dataset

As we discussed in the introduction, having an appropriate dataset is one of the most recurrent challenges in Question answering tasks. In this case, we used a subset of Stanford Question Answering Dataset (SQuAD) [4] provided by the teaching staff of CS224N. This dataset include tuples composed by question, context paragraph, answer and $(start, end)$-pair that defines span of indexes we are looking for. From the original training set we have 85670 question-answer tuples, and we divide them into 95% training and 5% validation. The original validation set is taken as test set, with the difference that it contains 3 possible answers (not necessarily different) for the same question, making a looser evaluation of the result.

- Max. length context paragraph ≈ 750 words but answer is in the first 250 word for 99.6% of the cases
- Max. length question ≈ 58 words.

3.1 Embedding

The word embeddings were provided by the CS224N teaching staff. We use GloVe representations pretrained on Wikipedia 2014 and Gigaword 5[2], trimmed to smaller file to enable its use in our model. We use a dimension = 100. In the two models we decided to keep the embedding as a constant, in order to preserve the similarity properties between words that have been seen, and unseen words in the test set. This is particularly important since our method depends highly in those dependencies.

If we were sure of the generality of the training data, we could consider retraining the word vectors. As a side note, notice that we only use word-vectors, as difference as using both word and character embedding as in the original paper [8].

3.2 Tokening and Masking

In order to find the word vectors for a certain context or paragraph, first we create a vocabulary that maps strings into integers, that after can be identify with word vectors. In this process, if a word is not identified, it is assign as unknown token, and therefore all its similarity properties are lost. In those cases, it makes a difference to have an additional character embedding, and therefore the meaning of the word is not totally lost. This is an option we can explore in future work.

Another detail to realize is that the length of context and question varies in every tuple, and most of the RNN approaches require a fix length statement. Therefore we need to apply masking (with '0's)the set of context word vectors. If the context paragraph is to large, we should trimmed to a considerable size. This information should be additional input for the model.

4 Model

We replicated the model presented in [8] in order to estimate the conditional probability of each of the indexes of being start / end given a question and a context paragraph. Let $C = [c_1, c_2, ..., c_k]$ the set of context-word vectors and $Q = [q_1, ..., q_s]$ the set of question-word vectors

[2]Distributed Word Representations in Assignment 4 handout

4.1 Relevance

We modify the context-word vectors by multiplying the highest significance with respect to the question-word vectors, i.e. for all $i \in [0, |C|)$

$$\text{relevance}(c_i) = \left(\max_{j \in [0,|Q|)} \frac{c_i^\top q_j}{||c_i||\ ||q_j||} \right) c_i \tag{3}$$

This will deprecate the words in the paragraph that are not closely related with the question words and therefore we assume that they are not part of the answer (although this could exclude future candidates).

4.2 Encoding

In order to use contextual information into the word-vectors we apply two different bidirectional LSTMs to question and passage. We decide to apply bidirectional in order to collect the information from left and right of the word and LSTMS because of the length of the paragraph, we can preserve information for longer. In the original paper, they used the same LSTM cell for both RNN, in our case we defined them independently (with different parameters), with the intuition that the type of information we want to extract from the question is different than the one in the context. At the end we concatenate the hidden vectors coming from both directions.

4.3 Multi-perspective Context Matching

This is the core of the method proposed in [8]. The idea is to compare under different perspectives the word vectors from the context and the question. To do this we construct a weight matrix $W \in \mathbb{R}^{H \times M}$ where H is the dimension of the hidden state and M is the number of perspectives we want to compute. For each context-hidden vector h_i^c we will compute a vector $p_i \in \mathbb{R}^M$ of perspectives using the question-hidden vector h_j^c as follows: For $m \in [0, M)$

$$\textbf{Full Matching} \quad p_i^m = \frac{(W_m \circ h_i^c)^\top (W_m \circ h_{|Q|}^q)}{||W_m \circ h_i^c||\ ||W_m \circ h_{|Q|}^q||} \tag{4}$$

$$\textbf{Max-pool Matching} \quad p_i^m = \max_{j \in [0,|Q|)} \frac{(W_m \circ h_i^c)^\top (W_m \circ h_j^q)}{||W_m \circ h_i^c||\ ||W_m \circ h_j^q||} \tag{5}$$

$$\textbf{Mean-pool Matching} \quad p_i^m = \frac{1}{|Q|} \sum_{j \in [0,|Q|)} \frac{(W_m \circ h_i^c)^\top (W_m \circ h_j^q)}{||W_m \circ h_i^c||\ ||W_m \circ h_j^q||} \tag{6}$$

Notice that this three types of matching represent: the similarity with r the whole concept of the question (**full matching**) represented in the last hidden vector; the maximum similarity among each partial representation of the question; and an aggregated similarity along we walk through the question. Each of these matching types tend to identify activations that become important to different types of question: direct (What, when, where) v.s. inference (why, how).

In practice we will train a weight matrix for each matching type and direction of the hidden vector. Notice that for other approaches, we could replace the Hadamard product by a matrix-vector product, and therefore each perspective would mean a projection in a different subspace and taking the energy norm in such. This could be an interesting feature to analyze in the future.

4.4 Decoder

To extract the $start$ and end predictions, first we apply an additional bidirectional LSTM to the perspective vectors, in order to aggregate the information from the context, more specifically, make a smoothing with respect to its neighbors.

At last, to predict the conditional probabilities according to each position, we concatenate all the hidden states, apply a linear transformation and then a softmax. We repeat this process for both $start$ and end probabilities, using different parameters.

Table 1: Parameters used in the Model

PARAMETER	VALUE
Question size	58
Context size	$M_1 = 750/M_2 = 250$
Embedding size	100
Hidden layer in Encoder	120
Hidden layer in Decoder	60
Number of perspectives	50
Dropout in all the layers	$M_1 = 0.15/M_2 = 0.5$
Gradient clipping	> 10
Learning rate	0.0001
Batch size	$M_1 = 10/M_2 = 40$
Training time	$M_1 = 10hs/M_2 = 1.5hs$

We evaluate our model using cross-entropy loss, in order to replicate our optimization goal. In this last step we need to take into account two factors: masking and $start \leq end$. For masking, since we know the original length of the paragraph, we can neglect the effect of the padding by masking the result of the softmax computation. This means that we make zero all the entries of the result of the linear transformation before applying the softmax. Therefore the predictions of $start$ and end will always be among the real corpus of the paragraph. Nevertheless, we still can face the problem of d $start > end$. For practical matters we just adjust each of the predictions such that they satisfy this constrain, but this is not enforced in the loss function as a constraint. For future development we can consider scenarios to include this constraint in the optimization process, for example add to the loss a term dependent on $start_{pred} - end_{pred}$

5 Experiments

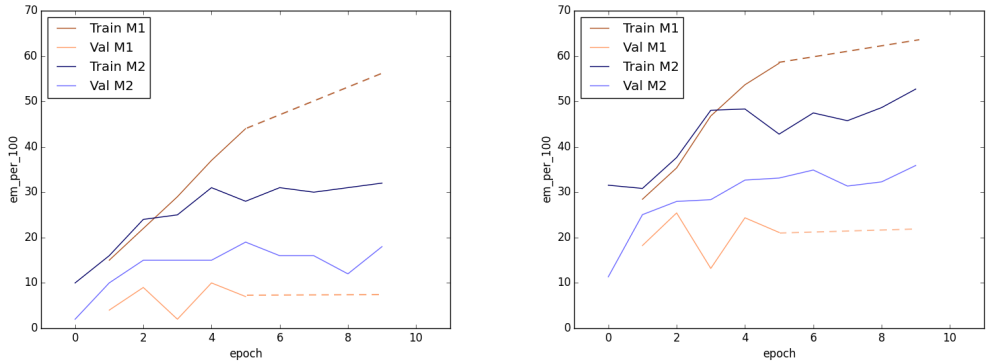
Having an unbelievable complex model as this one where there are a lot of different parameters to tweak, test and improve makes us have an enormous set of initial expectation. Nevertheless, this number of ideas and experiments reduced exponentially once we faced a vast number of challenges in the implementation of the model, specially setting up a baseline that was executable using Tensorflow + GPUs.

First we got a runnable implementation using only full matching perception, dimension of hidden layers = 20 and dropout = 1 (i.e. not dropout). The results of this model stayed around 5% F_1 error even for training data, after multiple epochs of training. This method was unable to learn the complexity of this task, but moves us to have a starting point. We would not include further analysis of this model, because it corresponds to only a toy implementation of the method.

After this model we implemented 2 other versions, both of them described the structure explained in the model, including the three types of multi-perspective matching. The difference between the two of them corresponds to change in the dropout rate and the size of the paragraph (see table 1).

Initially, we started with the maximum context size of 750 words, in order to capture all the information from the paragraph, but this resulted in a huge amount of parameters, that originated two problems: large amount of training time (10 h per epoch) because of the GPU running out of memory (batch size small = 10); and over fitting. As we can see in figure 1, the model M_1 after 6 epochs is around 60% in the train dataset, whereas is only 10% accurate in the validation set. The overfitting is also clear when we see the fast decay of the loss function (figure 2) in each epoch, and the small values it achieves.

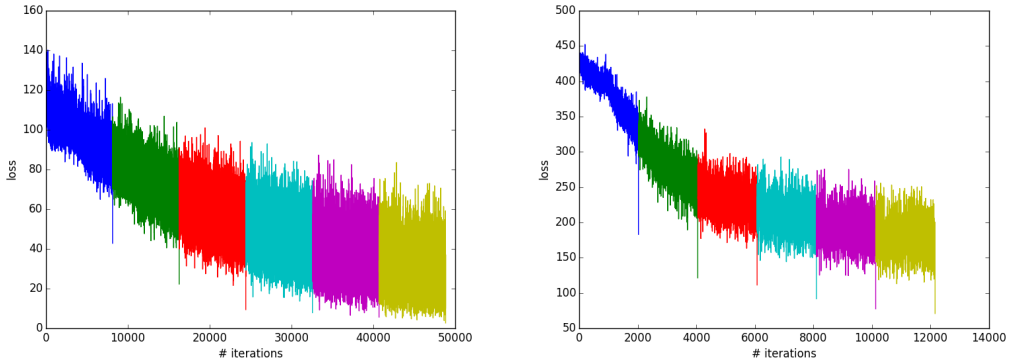
After realizing this fact, and expending around 60 hours in computations, we decided to modify the model reducing the number of words we count in the context to 250 (because 99.6% of the cases the answer span will be contained in this interval). But since this could be not enough, we increase the dropout rate to 50% given that this is a standard technique to prevent overfitting in complex models. In those cases the training one epoch took only 1.5 hours (see table 1).



(a) Exact Matching

(b) F_1 measure: Overlap

Figure 1: Performance the train and validation set of the two models among different epochs.



(a) Model 1: Overfitting

(b) Model 2:over training

Figure 2: Behavior of loss function among different epochs.

As we can see in figure 1, the overfitting is removed from the model, and now both training and validation have the same behavior along each epoch. Nevertheless as we can see in figure 2, the loss remains constant after epoch 5 although we trained up to epoch 10. This means we overtrained our model. Although in the beginning we could think that overtraining the model is harmless, we would see in the following section that this confuses the method and can damage the prediction.

5.1 Evaluation

To evaluate the performance of both methods, we apply a quantitative and qualitative analysis. The first one is to measure our closest to benchmark and state of the art approach, and the second one, to analyze in which cases the method performs correctly and in which others it needs external help.

5.1.1 Quantitative

In this case we use 2 standard measures for accuracy: Exact matching and F_1:

- **Exact matching** measure when the predicted start and end totally coincide with the true values of the answer.

- $\mathbf{F_1}$ measures the overlap between prediction and ground truth.

$$overlap = \max(0, \min(end_{pred}, end_{real}) - \max(start_{pred}, start_{real} + 1) \quad (7)$$

$$precision = \frac{overlap}{end_{pred} - start_{pred} + 1} \quad (8)$$

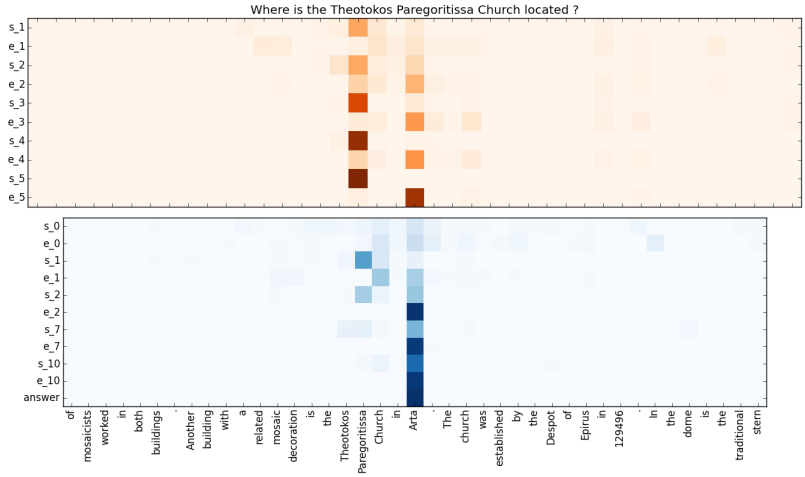
$$recall = \frac{overlap}{end_{real} - start_{real} + 1} \quad (9)$$

$$f_1 = \frac{2 * precision * recall}{precision + recall} \quad (10)$$

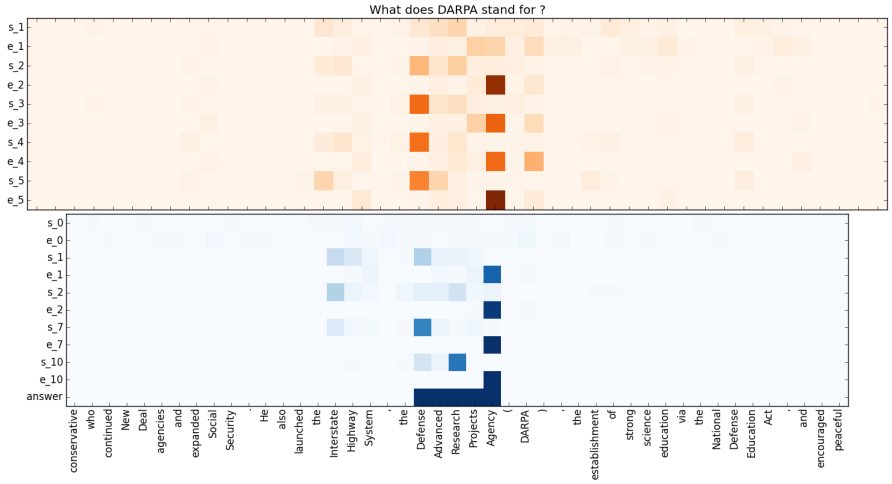
5.1.2 Quantitative

We look for the conditional probability vector for start s_i and end e_i predicted in the first 100 words of the validation set after each epoch i to see: How training affects the prediction?, Which types of questions are the easiest/hardest to solve?. M_1 is represented by orange and M_2 by blue.

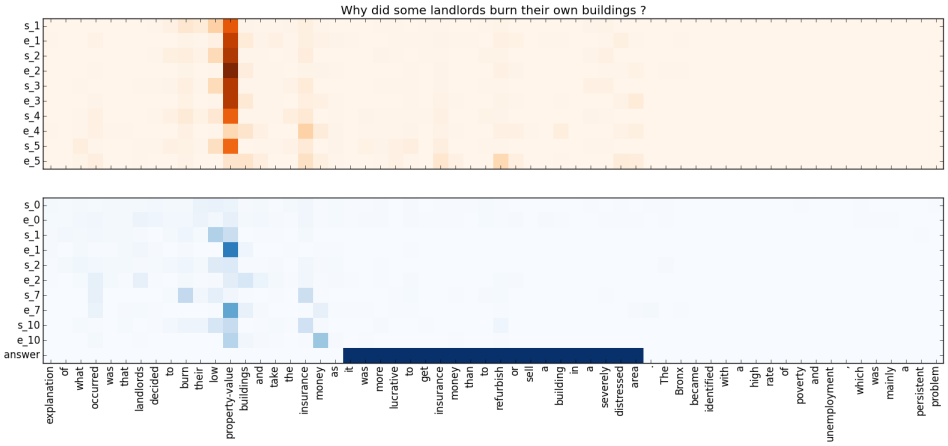
- M_2 avoids overfitting and reduces gap of uncertainty with respect of M_1. Extremely accurate with where - when - -what questions, that usually require a small answer next to the same words we find in the question.



- Overtraining of M_2 damage interval prediction, having worse results than M_1 in large answers. We can see that in epoch 7 the model finds the correct answer, but if we continue training it looses it in epoch 10.



- Strong dependence of direct similarity between question words and answer words. None of two models can infer answers that do not include words from question. The similarity algorithms only work for direct similarity, and do not combine the context.



6 Discussion and Future work

We analyze the effect of choosing different parameters in the Multi-perspective context matching model, and how we can easily fall in pitfalls such as overfitting and overtraining. Our first mode M_1 suffers from extreme overfitting, caused by unnecessary large amount of parameters and low dropout rate. Therefore increasing the dropout rate and reducing the number of unnecesary words in M_2 reduces the overfitting, but then we need to be really careful to stop the training once F_1 error in validation set remains approximately constant, otherwise we can just create confusion and decrease the performance in the model.

In the qualitative studies we saw that we get a really good performance in direct similarity, (i.e. What,When,Where questions). But, although the number of parameters is large, it is not enough to train more complex features such as inference of causes and consequences in questions including Why and How. Therefore we need to redistribute the number of parameters in a smarter way, including for example additional recurrent neural networks and more powerful attention algorithms to identify this dependencies. One important point is that the last softmax layer involves a huge linear transformation where the tradeoff between number of parameters and complexity is not favorable. Therefore we propose to just apply a small transformation for each vector and compute its probability independently.

Also there is still room to tun all the hyperparameters involved: Dropout rate, number of hidden layers, learning rate, number of Perspectives. And in order to move forward we need to also analyze the effect of each layer of the model, i.e. which are the benefits of choosing a determined attention models, how to relate the $start$ and end predictions (enforcing $start < end$ in the model), choosing among LSTMs v.s. GRUs, Bi directional versus one-directional recurrent neural networks, and what about doing the process deeper and deeper, stacking RNNs one over the other.

This was an incredible experience, although the learning curve was really step at the beginning, passing from a fill out code to decipher a template and start your model from scratch. Also I discover the importance of making design decisions at the beginning instead of just running a model with nonsense parameters, because these decisions affect hardly the time that you expend training a useless vs. meaningful method.

Thanks to all the teaching staff of CS224N for this challenging course.

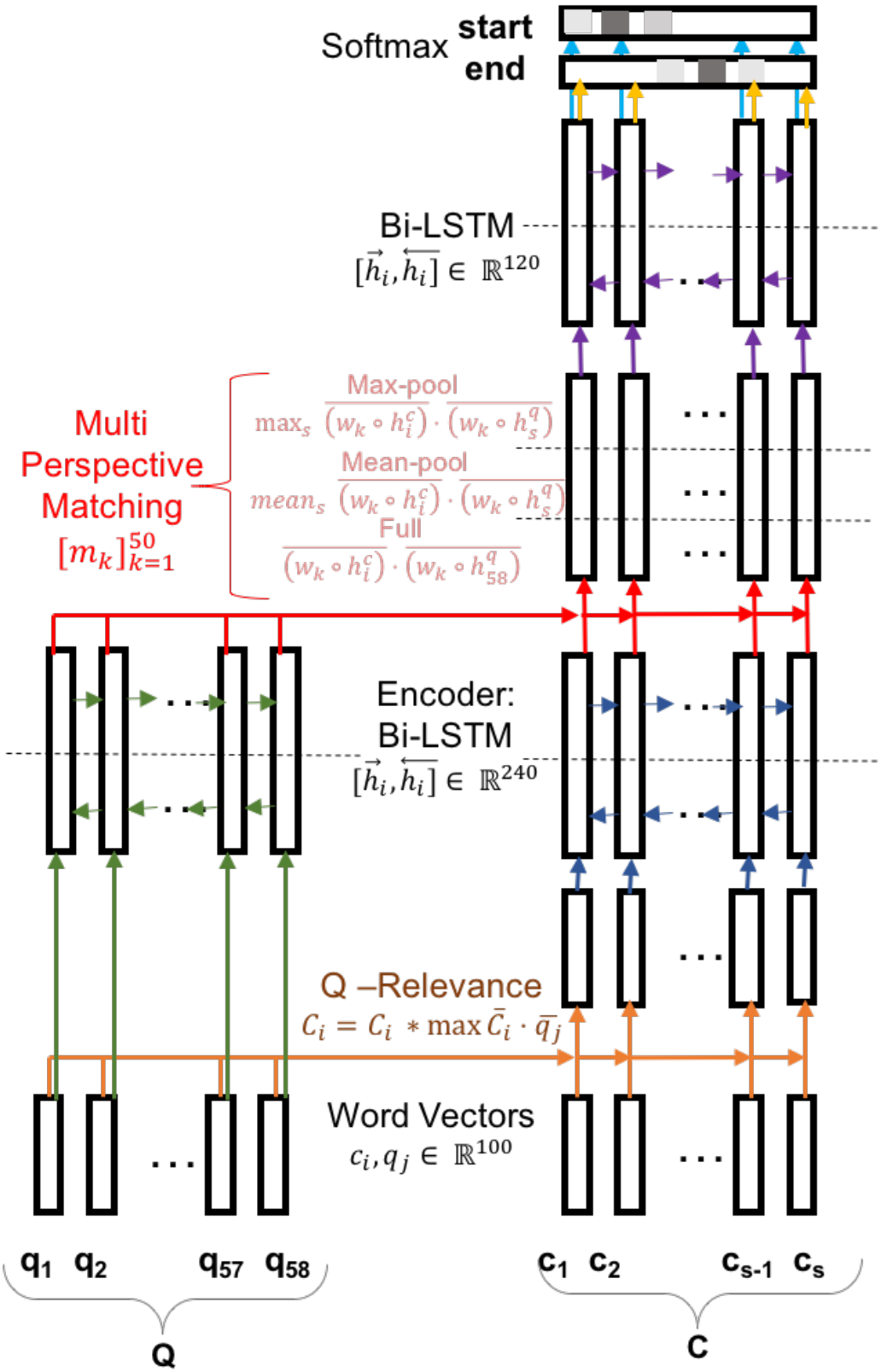


Figure 3: Encoder - Attention -Decoder model extracted from [8]

References

[1] Ellen Riloff and Michael Thelen. "A rule-based question answering system for reading comprehension tests". In Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems - Volume 6, 2000.

[2] Hoifung Poon, Janara Christensen, Pedro Domingos, Oren Etzioni, Raphael Hoffmann, Chloe Kiddon, Thomas Lin, Xiao Ling, Mausam, Alan Ritter, Stefan Schoenmackers, Stephen Soderland, Dan Weld, Fei Wu, and Congle Zhang. "Machine reading at the University of Washington". In Proceedings of the NAACL HLT 2010 First International Workshop on Formalisms and Methodology for Learning by Reading, 2010.

[3] Karl Moritz Hermann and Tomás Kociský and Edward Grefenstette and Lasse Espeholt and Will Kay and Mustafa Suleyman and Phil Blunsom. "Teaching Machines to Read and Comprehend". CoRR, abs/1506.03340, 2015.

[4] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. CoRR, abs/1606.05250, 2016

[5] Christopher Manning, Richard Socher, Guillaume Genthial, Lucas Liu, Barak Oshri, Kushal Ranjan. CS224n: Natural Language Processing with Deep Learning , Lecture Notes: Part VI, Winter 2017

[6] Caiming Xiong and Victor Zhong and Richard Socher. "Dynamic Coattention Networks For Question Answering". CoRR , abs/1611.01604, 2016.

[7] Yang Yu and Wei Zhang and Kazi Hasan and Mo Yu and Bing Xiang and Bowen Zhou, "End-to-End Reading Comprehension with Dynamic Answer Chunk Ranking". CoRR , abs/1610.09996, 2016.

[8] Zhiguo Wang , Haitao Mi, Wael Hamza and Radu Florian. "Multi-Perspective Context Matching for Machine Comprehension." CoRR, abs/1612.04211, 2016.