Exploring Question Understanding and Adaptation in Neural-Network-Based Question Answering

Junbei Zhang[1], Xiaodan Zhu[2], Qian Chen[1], Lirong Dai[1], Si Wei[3] and Hui Jiang[4]
[1]University of Science and Technology of China
[2]National Research Council Canada
[3]iFLYTEK Research, China
[4]York University
{zjunbei,cq1231}@mail.ustc.edu.cn, xiaodan.zhu@nrc-cnrc.gc.ca
lrdai@ustc.edu.cn, siwei@iflytek.com, hj@cse.yorku.ca

Abstract

The last several years have seen intensive interest in exploring neural-network-based models for machine comprehension (MC) and question answering (QA). In this paper, we approach the problems by closely modelling questions in a neural network framework. We first introduce syntactic information to help encode questions. We then view and model different types of questions and the information shared among them as an adaptation task and proposed adaptation models for them. On the Stanford Question Answering Dataset (SQuAD), we show that these approaches can help attain better results over a competitive baseline.

1 Introduction

Enabling computers to understand given documents and answer questions about their content has recently attracted intensive interest, including but not limited to the efforts as in [Richardson et al., 2013, Hermann et al., 2015, Hill et al., 2015, Rajpurkar et al., 2016, Nguyen et al., 2016, Berant et al., 2014]. Many specific problems such as machine comprehension and question answering often involve modeling such question-document pairs.

The recent availability of relatively large training datasets (see Section 2 for more details) has made it more feasible to train and estimate rather complex models in an end-to-end fashion for these problems, in which a whole model is fit directly with given question-answer tuples and the resulting model has shown to be rather effective.

In this paper, we take a closer look at modeling questions in such an end-to-end neural network framework, since we regard question understanding is of importance for such problems. We first introduced syntactic information to help encode questions. We then viewed and modelled different types of questions and the information shared among them as an adaptation problem and proposed adaptation models for them. On the Stanford Question Answering Dataset (SQuAD), we show that these approaches can help attain better results on our competitive baselines.

2 Related Work

Recent advance on reading comprehension and question answering has been closely associated with the availability of various datasets. Richardson et al. [2013] released the MCTest data consisting of 500 short, fictional open-domain stories and 2000 questions. The CNN/Daily Mail dataset [Hermann et al., 2015] contains news articles for close style machine comprehension, in which only entities are removed and tested for comprehension. Children's Book Test (CBT) [Hill et al., 2015] leverages named entities, common nouns, verbs, and prepositions to test reading comprehension. The Stanford Question Answering Dataset (SQuAD) [Rajpurkar et al., 2016] is more recently released dataset, which consists of more than 100,000 questions for documents taken from Wikipedia across a wide range of topics. The question-answer pairs are annotated through crowdsourcing. Answers are spans of text marked in the original documents. In this paper, we use SQuAD to evaluate our models.

Many neural network models have been studied on the SQuAD task. Wang and Jiang [2016] proposed *match LSTM* to associate documents and questions and adapted the so-called *pointer Network* [Vinyals et al., 2015] to determine the positions of the answer text spans. Yu et al. [2016] proposed a *dynamic chunk reader* to extract and rank a set of answer candidates. Yang et al. [2016] focused on word representation and presented a fine-grained gating mechanism to dynamically combine word-level and character-level representations based on the properties of words. Wang et al. [2016] proposed a *multi-perspective context matching* (MPCM) model, which matched an encoded document and question from multiple perspectives. Xiong et al. [2016] proposed a dynamic decoder and so-called *highway maxout network* to improve the effectiveness of the decoder. The *bi-directional attention flow* (BIDAF) [Seo et al., 2016] used the bi-directional attention to obtain a question-aware context representation.

In this paper, we introduce syntactic information to encode questions with a specific form of recursive neural networks [Zhu et al., 2015, Tai et al., 2015, Chen et al., 2016, Socher et al., 2011]. More specifically, we explore a tree-structured LSTM [Zhu et al., 2015, Tai et al., 2015] which extends the linear-chain long short-term memory (LSTM) [Hochreiter and Schmidhuber, 1997] to a recursive structure, which has the potential to capture long-distance interactions over the structures.

Different types of questions are often used to seek for different types of information. For example, a "what" question could have very different property from that of a "why" question, while they may share information and need to be trained together instead of separately. We view this as a "adaptation" problem to let different types of questions share a basic model but still discriminate them when needed. Specifically, we are motivated by the ideas "i-vector" [Dehak et al., 2011] in speech recognition, where neural network based adaptation is performed among different (groups) of speakers and we focused instead on different types of questions here.

3 The Approach

3.1 The Baseline Model

Our baseline model is composed of the following typical components: *word embedding*, *input encoder*, *alignment*, *aggregation*, and *prediction*. Below we discuss these components in more details.

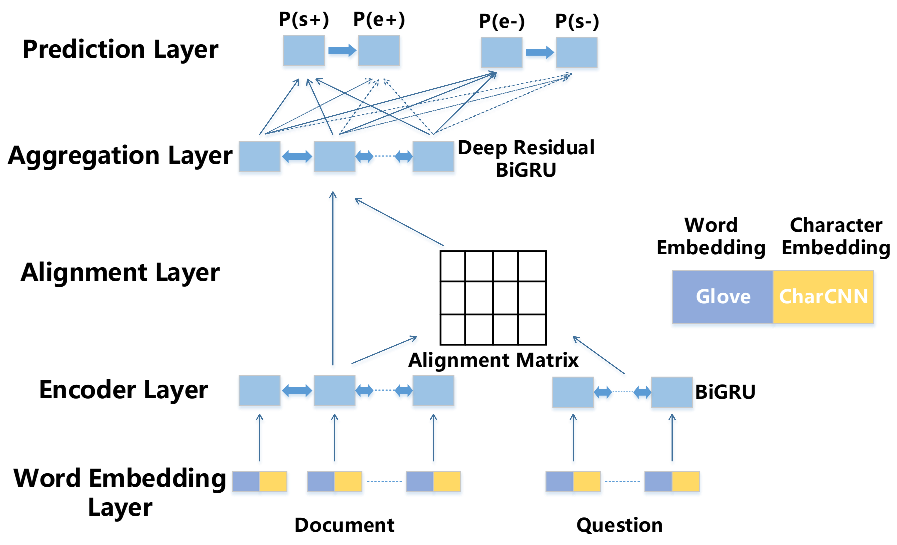


Figure 1: A high level view of our basic model.

Word embedding We concatenate embedding at two levels to represent a word: the character composition and word-level embedding. The character composition feeds all characters of a word into a convolutional neural network (CNN) [Kim, 2014] to obtain a representation for the word. And we use the pre-trained 300-D GloVe vectors [Pennington et al., 2014] (see the experiment section for details) to initialize our word-level embedding. Each word is therefore represented as the concatenation of the character-composition vector and word-level embedding. This is performed on both questions and documents, resulting in two matrices: the $\mathbf{Q}^e \in \mathbb{R}^{N \times d_w}$ for a question and the $\mathbf{D}^e \in \mathbb{R}^{M \times d_w}$ for a document, where N is the question length (number of word tokens), M is the document length, and d_w is the embedding dimensionality.

Input encoding The above word representation focuses on representing individual words, and an input encoder here employs recurrent neural networks to obtain the representation of a word under its context. We use bi-directional GRU (BiGRU) [Cho et al., 2014] for both documents and questions.

$$\mathbf{Q}^c_i = \text{BiGRU}(\mathbf{Q}^e_i, i), \forall i \in [1, \ldots, N] \tag{1}$$

$$\mathbf{D}^c_j = \text{BiGRU}(\mathbf{D}^e_j, j), \forall j \in [1, \ldots, M] \tag{2}$$

A BiGRU runs a forward and backward GRU on a sequence starting from the left and the right end, respectively. By concatenating the hidden states of these two GRUs for each word, we obtain the a representation for a question or document: $\mathbf{Q}^c \in \mathbb{R}^{N \times d_c}$ for a question and $\mathbf{D}^c \in \mathbb{R}^{M \times d_c}$ for a document.

Alignment Questions and documents interact closely. As in most previous work, our framework use both soft attention over questions and that over documents to capture the interaction between them. More specifically, in this soft-alignment layer, we first feed the contextual representation matrix $\mathbf{Q}^c$ and $\mathbf{D}^c$ to obtain alignment matrix $\mathbf{U} \in \mathbb{R}^{N \times M}$:

$$\mathbf{U}_{ij} = \mathbf{Q}^c_i \cdot \mathbf{D}^{c\mathrm{T}}_j, \forall i \in [1, \ldots, N], \forall j \in [1, \ldots, M] \tag{3}$$

Each $\mathbf{U}_{ij}$ represents the similarity between a question word $\mathbf{Q}^c_i$ and a document word $\mathbf{D}^c_j$.

Word-level Q-code Similar as in [Seo et al., 2016], we obtain a word-level Q-code. Specifically, for each document word w_j, we find which words in the question are relevant to it. To this end, $\mathbf{a}_j \in \mathbb{R}^N$ is computed with the following equation and used as a soft attention weight:

$$\mathbf{a}_j = softmax(\mathbf{U}_{:j}), \forall j \in [1, \ldots, M] \tag{4}$$

With the attention weights computed, we obtain the encoding of the question for each document word w_j as follows, which we call *word-level Q-code* in this paper:

$$\mathbf{Q}^w = \mathbf{a}^\mathrm{T} \cdot \mathbf{Q}^c \in \mathbb{R}^{M \times d_c} \tag{5}$$

Question-based filtering To better explore question understanding, we design this *question-based filtering* layer. As detailed later, different question representation can be easily incorporated to this layer in addition to being used as a filter to find key information in the document based on the question. This layer is expandable with more complicated question modeling.

In the basic form of question-based filtering, for each question word w_i, we find which words in the document are associated. Similar to $\mathbf{a}_j$ discussed above, we can obtain the attention weights on document words for each question word w_i:

$$\mathbf{b}_i = softmax(\mathbf{U}_{i:}) \in \mathbb{R}^M, \forall i \in [1, \ldots, N] \tag{6}$$

By pooling $\mathbf{b} \in \mathbb{R}^{N \times M}$, we can obtain a question-based filtering weight $\mathbf{b}^f$:

$$\mathbf{b}^f = norm(pooling(\mathbf{b})) \in \mathbb{R}^M \tag{7}$$

$$norm(\mathbf{x}) = \frac{\mathbf{x}}{\sum_i x_i} \tag{8}$$

where the specific pooling function we used include max-pooling and mean-pooling. Then the document softly filtered based on the corresponding question $\mathbf{D}^f$ can be calculated by:

$$\mathbf{D}^{f_{max}}_j = b^{f_{max}}_j \mathbf{D}^c_j, \forall j \in [1, \ldots, M] \tag{9}$$

$$\mathbf{D}_j^{f_{mean}} = b_j^{f_{mean}} \mathbf{D}_j^c, \forall j \in [1, \ldots, M] \tag{10}$$

$$\mathbf{D}^f = [\mathbf{D}^{f_{max}}, \mathbf{D}^{f_{mean}}] \tag{11}$$

Through concatenating the document representation $\mathbf{D}^c$, word-level Q-code $\mathbf{Q}^w$ and question-filtered document $\mathbf{D}^f$, we can finally obtain the alignment layer representation:

$$\mathbf{I} = [\mathbf{D}^c, \mathbf{Q}^w, \mathbf{D}^c \circ \mathbf{Q}^w, \mathbf{D}^c - \mathbf{Q}^w, \mathbf{D}^f, \mathbf{b}^{f_{max}}, \mathbf{b}^{f_{mean}}] \in \mathbb{R}^{M \times (6d_c+2)} \tag{12}$$

where "$\circ$" stands for element-wise multiplication and "$-$" is simply the vector subtraction.

Aggregation After acquiring the local alignment representation, key information in document and question has been collected, and the aggregation layer is then performed to find answers. We use three BiGRU layers to model the process that aggregates local information to make the global decision to find the answer spans. We found a residual architecture [He et al., 2016] as described in Figure 2 is very effective in this aggregation process:

$$\mathbf{I}_i^1 = \text{BiGRU}(\mathbf{I}_i) \tag{13}$$

$$\mathbf{I}_i^2 = \mathbf{I}_i^1 + \text{BiGRU}(\mathbf{I}_i^1) \tag{14}$$

$$\mathbf{I}_i^3 = \mathbf{I}_i^2 + \text{BiGRU}(\mathbf{I}_i^2) \tag{15}$$

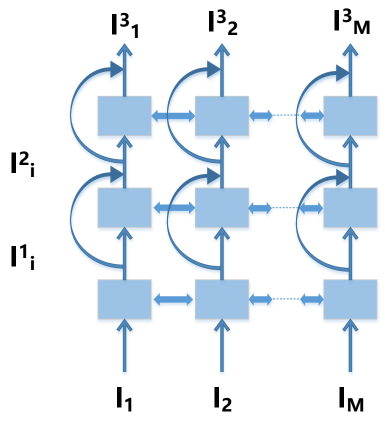


Figure 2: The inference layer implemented with a residual network.

Prediction The SQuAD QA task requires a span of text to answer a question. We use a pointer network [Vinyals et al., 2015] to predict the starting and end position of answers as in [Wang and Jiang, 2016]. Different from their methods, we use a two-directional prediction to obtain the positions. For one direction, we first predict the starting position of the answer span followed by predicting the end position, which is implemented with the following equations:

$$P(s+) = softmax(W_{s+} \cdot I^3) \tag{16}$$

$$P(e+) = softmax(W_{e+} \cdot I^3 + W_{h+} \cdot h_{s+}) \tag{17}$$

where $\mathbf{I}^3$ is inference layer output, $\mathbf{h}_{s+}$ is the hidden state of the first step, and all $\mathbf{W}$ are trainable matrices. We also perform this by predicting the end position first and then the starting position:

$$P(e-) = softmax(W_{e-} \cdot I^3) \tag{18}$$

$$P(s-) = softmax(W_{s-} \cdot I^3 + W_{h-} \cdot h_{e-}) \tag{19}$$

We finally identify the span of an answer with the following equation:

$$P(s) = pooling([P(s+), P(s-)]) \tag{20}$$

$$P(e) = pooling([P(e+), P(e-)]) \tag{21}$$

We use the mean-pooling here as it is more effective on the development set than the alternatives such as the max-pooling.

3.2 Question Understanding and Adaptation

3.2.1 Introducing syntactic information for neural question encoding

The interplay of syntax and semantics of natural language questions is of interest for question representation. We attempt to incorporate syntactic information in questions representation with TreeLSTM [Zhu et al., 2015, Tai et al., 2015]. In general a TreeLSTM could perform semantic composition over given syntactic structures.

Unlike the chain-structured LSTM [Hochreiter and Schmidhuber, 1997], the TreeLSTM captures long-distance interaction on a tree. The update of a TreeLSTM node is described at a high level with Equation (22), and the detailed computation is described in (23–29). Specifically, the input of a TreeLSTM node is used to configure four gates: the input gate $\mathbf{i}_t$, output gate $\mathbf{o}_t$, and the two forget gates $\mathbf{f}_t^L$ for the left child input and $\mathbf{f}_t^R$ for the right. The memory cell $\mathbf{c}_t$ considers each child's cell vector, $\mathbf{c}_{t-1}^L$ and $\mathbf{c}_{t-1}^R$, which are gated by the left forget gate $\mathbf{f}_t^L$ and right forget gate $\mathbf{f}_t^R$, respectively.

$$\mathbf{h}_t = \text{TreeLSTM}(\mathbf{x}_t, \mathbf{h}_{t-1}^L, \mathbf{h}_{t-1}^R), \tag{22}$$
$$\mathbf{h}_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t), \tag{23}$$
$$\mathbf{o}_t = \sigma(\mathbf{W}_o\mathbf{x}_t + \mathbf{U}_o^L\mathbf{h}_{t-1}^L + \mathbf{U}_o^R\mathbf{h}_{t-1}^R), \tag{24}$$
$$\mathbf{c}_t = \mathbf{f}_t^L \circ \mathbf{c}_{t-1}^L + \mathbf{f}_t^R \circ \mathbf{c}_{t-1}^R + \mathbf{i}_t \circ \mathbf{u}_t, \tag{25}$$
$$\mathbf{f}_t^L = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f^{LL}\mathbf{h}_{t-1}^L + \mathbf{U}_f^{LR}\mathbf{h}_{t-1}^R), \tag{26}$$
$$\mathbf{f}_t^R = \sigma(\mathbf{W}_f\mathbf{x}_t + \mathbf{U}_f^{RL}\mathbf{h}_{t-1}^L + \mathbf{U}_f^{RR}\mathbf{h}_{t-1}^R), \tag{27}$$
$$\mathbf{i}_t = \sigma(\mathbf{W}_i\mathbf{x}_t + \mathbf{U}_i^L\mathbf{h}_{t-1}^L + \mathbf{U}_i^R\mathbf{h}_{t-1}^R), \tag{28}$$
$$\mathbf{u}_t = \tanh(\mathbf{W}_c\mathbf{x}_t + \mathbf{U}_c^L\mathbf{h}_{t-1}^L + \mathbf{U}_c^R\mathbf{h}_{t-1}^R), \tag{29}$$

where σ is the sigmoid function, $\circ$ is the element-wise multiplication of two vectors, and all $\mathbf{W}$, $\mathbf{U}$ are trainable matrices.

To obtain the parse tree information, we use Stanford CoreNLP (PCFG Parser) [Manning et al., 2014, Klein and Manning, 2003] to produce a binarized constituency parse for each question and build the TreeLSTM based on the parse tree. The root node of TreeLSTM is used as the representation for the whole question. More specifically, we use it as TreeLSTM Q-code $\mathbf{Q}^{TL} \in \mathbb{R}^{d_c}$, by not only simply concatenating it to the alignment layer output but also using it as a question filter, just as we discussed in the *question-based filtering* section:

$$\mathbf{Q}^{TL} = \text{TreeLSTM}(\mathbf{Q}^e) \in \mathbb{R}^{d_c} \tag{30}$$
$$\mathbf{b}^{TL} = norm(\mathbf{Q}^{TL} \cdot \mathbf{D}^{c\mathrm{T}}) \in \mathbb{R}^M \tag{31}$$
$$\mathbf{D}_j^{TL} = b_j^{TL}\mathbf{D}_j^c, \forall j \in [1, \ldots, M] \tag{32}$$
$$\mathbf{I}_{new} = [\mathbf{I}, repmat(\mathbf{Q}^{TL}), \mathbf{D}^{TL}, \mathbf{b}^{TL}] \tag{33}$$

where $\mathbf{I}_{new}$ is the new output of alignment layer, and function $repmat$ copies $\mathbf{Q}^{TL}$ for M times to fit with $\mathbf{I}$.

3.2.2 Question Adaptation

Questions by nature are often composed to fulfill different types of information needs. For example, a "when" question seeks for different types of information (i.e., temporal information) than those for a "why" question. Different types of questions and the corresponding answers could potentially have different distributional regularity.

Explicit question-type embedding The previous models are often trained for all questions without explicitly discriminating different question types; however, for a target question, both the common features shared by all questions and the specific features for a specific type of question are further considered in this paper, as they could potentially obey different distributions. In this paper we further explicitly model different types of questions in the end-to-end training. We start from a simple way to first analyze the word frequency of all questions, and obtain top-10 most frequent question types: *what, how, who, when, which, where, why, be, whose,* and *whom*, in which *be* stands for the questions beginning with different forms of the word *be* such as *is, am*, and *are*. We explicitly encode question-type information to be an 11-dimensional one-hot vector (the top-10 question types and "other" question type). Each question type is with a trainable embedding vector. We call this explicit question type code, $\mathbf{ET} \in \mathbb{R}^{d_{ET}}$. Then the vector for each question type is tuned during training, and is added to the system with the following equation:

$$\mathbf{I}_{new} = [\mathbf{I}, repmat(\mathbf{ET})] \tag{34}$$

Question adaptation As discussed, different types of questions and their answers may share common regularity and have separate property at the same time. We also view this as an adaptation problem in order to let different types of questions share a basic model but still discriminate them when needed. Specifically, we borrow ideas from speaker adaptation [Dehak et al., 2011] in speech recognition, where neural-network-based adaptation is performed among different groups of speakers.

Conceptually we regard a type of questions as a group of acoustically similar speakers. Specifically we propose a question discriminative block or simply called a *discriminative block* (Figure 3) below to perform question adaptation. The main idea is described below:

$$\mathbf{x}' = f([\mathbf{x}, \bar{\mathbf{x}}^c, \delta_{\mathbf{x}}]) \tag{35}$$

For each input question $\mathbf{x}$, we can decompose it to two parts: the cluster it belong(i.e., question type) and the diverse in the cluster. The information of the cluster is encoded in a vector $\bar{\mathbf{x}}^c$. In order to keep calculation differentiable, we compute the weight of all the clusters based on the distances of $\mathbf{x}$ and each cluster center vector, in stead of just choosing the closest cluster. Then the discriminative vector $\delta_{\mathbf{x}}$ with regard to these most relevant clusters are computed. All this information is combined to obtain the discriminative information. In order to keep the full information of input, we also copy the input question $\mathbf{x}$, together with the acquired discriminative information, to a feed-forward layer to obtain a new representation $\mathbf{x}'$ for the question.

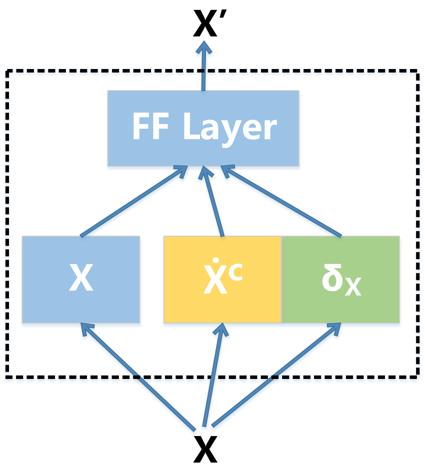


Figure 3: The *discriminative block* for question discrimination and adaptation.

More specifically, the adaptation algorithm contains two steps: *adapting* and *updating*, which is detailed as follows:

- **Adapting** In the adapting step, we first compute the similarity score between an input question vector $\mathbf{x} \in \mathbb{R}^h$ and each centroid vector of K clusters $\bar{\mathbf{x}} \in \mathbb{R}^{K \times h}$. Each cluster here models a question type. Unlike the explicit question type modeling discussed above, here we do not specify what question types we are modeling but let the system to learn. Specifically, we only need to pre-specific how many clusters, K, we are modeling. The similarity between an input question and cluster centroid can be used to compute similarity weight $\mathbf{w}^a$:

$$w_k^a = softmax(cos\_sim(\mathbf{x}, \bar{\mathbf{x}}_k), \alpha), \forall k \in [1, \ldots, K] \tag{36}$$

$$cos\_sim(\mathbf{u}, \mathbf{v}) = \frac{< \mathbf{u}, \mathbf{v} >}{||\mathbf{u}|| \cdot ||\mathbf{v}||} \tag{37}$$

$$softmax(x_i, \alpha) = \frac{e^{\alpha x_i}}{\sum_j e^{\alpha x_j}} \tag{38}$$

We set α equals 50 to make sure only closest class will have a high weight while maintain differentiable. Then we acquire a soft class-center vector $\bar{\mathbf{x}}^c$:

$$\bar{\mathbf{x}}^c = \sum_k w_k^a \bar{\mathbf{x}}_k \in \mathbb{R}^h \tag{39}$$

We then compute a discriminative vector $\delta_{\mathbf{x}}$ between the input question with regard to the soft class-center vector:

$$\delta_{\mathbf{x}} = \mathbf{x} - \bar{\mathbf{x}}^c \tag{40}$$

Note that $\bar{\mathbf{x}}^c$ here models the cluster information and $\delta_{\mathbf{x}}$ represents the discriminative information in the cluster. By feeding $\mathbf{x}$, $\bar{\mathbf{x}}^c$ and $\delta_{\mathbf{x}}$ into feedforward layer with Relu, we obtain $\mathbf{x}' \in \mathbb{R}^K$:

$$\mathbf{x}' = Relu(\mathbf{W} \cdot [\mathbf{x}, \bar{\mathbf{x}}^c, \delta_{\mathbf{x}}]) \tag{41}$$

With $\mathbf{x}'$ ready, we can apply Discriminative Block to any question code and obtain its adaptation Q-code. In this paper, we use TreeLSTM Q-code as the input vector $\mathbf{x}$, and obtain TreeLSTM adaptation Q-code $\mathbf{Q}^{TLa} \in \mathbb{R}^{d_c}$. Similar to TreeLSTM Q-code $\mathbf{Q}^{TL}$, we concatenate $\mathbf{Q}^{TLa}$ to alignment output $\mathbf{I}$ and also use it as a question filter:

$$\mathbf{Q}^{TLa} = Relu(\mathbf{W} \cdot [\mathbf{Q}^{TL}, \overline{\mathbf{Q}^{TL}}^c, \delta_{\mathbf{Q^{TL}}}]) \tag{42}$$

$$\mathbf{b}^{TLa} = norm(\mathbf{Q}^{TLa} \cdot \mathbf{D}^{c\mathrm{T}}) \in \mathbb{R}^M \tag{43}$$

$$\mathbf{D}_j^{TLa} = b_j^{TLa} \mathbf{D}_j^c, \forall j \in [1, \ldots, M] \tag{44}$$

$$\mathbf{I}_{new} = [\mathbf{I}, repmat(\mathbf{Q}^{TLa}), \mathbf{D}^{TLa}, \mathbf{b}^{TLa}] \tag{45}$$

- **Updating** The updating stage attempts to modify the center vectors of the K clusters in order to fit each cluster to model different types of questions. The updating is performed according to the following formula:

$$\bar{\mathbf{x}}_k' = (1 - \beta \mathrm{w}_k^a)\bar{\mathbf{x}}_k + \beta \mathrm{w}_k^a \mathbf{x}, \forall k \in [1, \ldots, K] \tag{46}$$

In the equation, β is an updating rate used to control the amount of each updating, and we set it to 0.01. When $\mathbf{x}$ is far away from K-th cluster center $\bar{\mathbf{x}}_k$, w_k^a is close to be value 0 and the k-th cluster center $\bar{\mathbf{x}}_k$ tends not to be updated. If $\mathbf{x}$ is instead close to the j-th cluster center $\bar{\mathbf{x}}_j$, w_k^a is close to the value 1 and the centroid of the j-th cluster $\bar{\mathbf{x}}_j$ will be updated more aggressively using $\mathbf{x}$.

4 Experiment Results

4.1 Set-Up

We test our models on Stanford Question Answering Dataset (SQuAD) [Rajpurkar et al., 2016]. The SQuAD dataset consists of more than 100,000 questions annotated by crowdsourcing workers on a selected set of Wikipedia articles, and the answer to each question is a span of text in the Wikipedia articles. Training data includes 87,599 instances and validation set has 10,570 instances. The test data is hidden and kept by the organizer. The evaluation of SQuAD is Exact Match (EM) and F1 score.

We use pre-trained *300-D Glove 840B* vectors [Pennington et al., 2014] to initialize our word embeddings. Out-of-vocabulary (OOV) words are initialized randomly with Gaussian samples. CharCNN filter length is 1,3,5, each is 50 dimensions. All vectors including word embedding are updated during training. The cluster number K in discriminative block is 100. The Adam method [Kingma and Ba, 2014] is used for optimization. And the first momentum is set to be 0.9 and the second 0.999. The initial learning rate is 0.0004 and the batch size is 32. We will half learning rate when meet a bad iteration, and the patience is 7. Our early stop evaluation is the EM and F1 score of validation set. All hidden states of GRUs, and TreeLSTMs are 500 dimensions, while word-level embedding d_w is 300 dimensions. We set max length of document to 500, and drop the question-document pairs beyond this on training set. Explicit question-type dimension d_{ET} is 50. We apply dropout to the Encoder layer and aggregation layer with a dropout rate of 0.5.

Model	EM	F1
Logistic Regression Baseline [Rajpurkar et al., 2016]	40.4	51.0
Match-LSTM with Ans-Ptr (Sentence) [Wang and Jiang, 2016]	54.505	67.748
Match-LSTM with Ans-Ptr (Boundary) [Wang and Jiang, 2016]	60.474	70.695
Dynamic Chunk Reader [Yu et al., 2016]	62.499	70.956
Fine-Grained Gating [Yang et al., 2016]	62.446	73.327
Match-LSTM with Bi-Ans-Ptr (Boundary) [Wang and Jiang, 2016]	64.744	73.743
Multi-Perspective Matching [Wang et al., 2016]	65.551	75.118
Dynamic Coattention Networks [Xiong et al., 2016]	66.233	75.896
BiLSTM [German Research Center for Artificial Intelligence(unpublished)]	68.436	77.070
BiDAF [Seo et al., 2016]	67.974	77.323
jNet(Ours)	**68.730**	**77.393**
r-net [Microsoft Research Asia(unpublished)]	70.062	78.782
Human Performance [Rajpurkar et al., 2016]	82.304	91.221

Table 1: The official leaderboard of single models on SQuAD test set as we submitted our systems (January 20, 2017).

4.2 Results

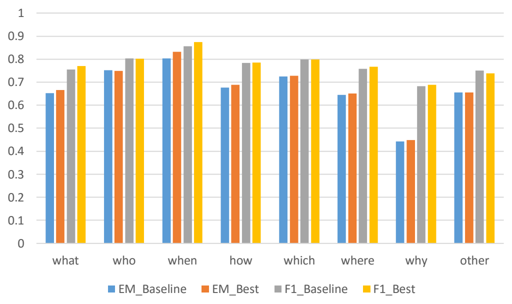
Overall results Table 1 shows the official leaderboard on SQuAD test set when we submitted our system. Our model achieves a 68.73% EM score and 77.39% F1 score, which is ranked among the state of the art single models (without model ensembling).

Model	EM	F1
Baseline	68.00	77.36
+Explicit question types (**ET**)	68.16	77.58
+TreeLSTM ($\mathbf{Q}^{TL}$)	68.29	77.67
+TreeLSTM adaptation ($\mathbf{Q}^{TLa}$, K=20)	68.73	77.74
+TreeLSTM adaptation ($\mathbf{Q}^{TLa}$, K=100)	**69.10**	**78.38**

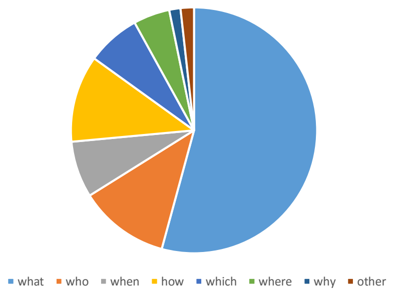
Table 2: Performance of various Q-code on the development set.

Table 2 shows the ablation performances of various Q-code on the development set. Note that since the testset is hidden from us, we can only perform such an analysis on the development set. Our baseline model using no Q-code achieved a 68.00% and 77.36% EM and F1 scores, respectively. When we added the explicit question type T-code into the baseline model, the performance was improved slightly to 68.16%(EM) and 77.58%(F1). We then used TreeLSTM introduce syntactic parses for question representation and understanding (replacing simple question type as question understanding Q-code), which consistently shows further improvement. We further incorporated the soft adaptation. When letting the number of hidden question types (K) to be 20, the performance improves to 68.73%/77.74% on EM and F1, respectively, which corresponds to the results of our model reported in Table 1. Furthermore, after submitted our result, we have experimented with a large value of K and found that when $K = 100$, we can achieve a better performance of 69.10%/78.38% on the development set.

Figure 4(a) shows the EM/F1 scores of different question types while Figure 4(b) is the question type amount distribution on the development set. In Figure 4(a) we can see that the average EM/F1 of the "when" question is highest and those of the "why" question is the lowest. From Figure 4(b) we can see the "what" question is the major class.



(a) Question Type Results



(b) Question Type Amount

Figure 4: Question Type Analysis

Figure 5 shows the composition of F1 score. Take our best model as an example, we observed a 78.38% F1 score on the whole development set, which can be separated into two parts: one is where F1 score equals to 100%, which means an exact match. This part accounts for 69.10% of the entire development set. And the other part accounts for 30.90%, of which the average F1 score is 30.03%. For the latter, we can further divide it into two sub-parts: one is where the F1 score equals to 0%, which means that predict answer is totally wrong. This part occupies 14.89% of the total development set. The other part accounts for 16.01% of the development set, of which average F1 score is 57.96%. From this analysis we can see that reducing the zero F1 score (14.89%) is potentially an important direction to further improve the system.

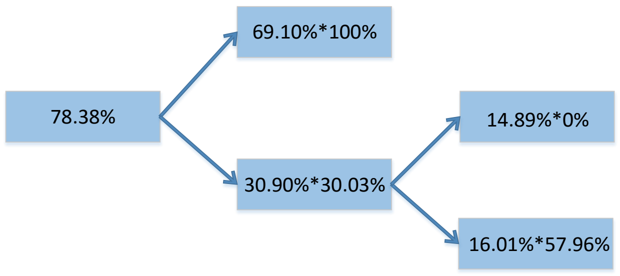


Figure 5: F1 Score Analysis.

5 Conclusions

Closely modelling questions could be of importance for question answering and machine reading. In this paper, we introduce syntactic information to help encode questions in neural networks. We view and model different types of questions and the information shared among them as an adaptation task and proposed adaptation models for them. On the Stanford Question Answering Dataset (SQuAD), we show that these approaches can help attain better results over a competitive baseline.

References

Jonathan Berant, Vivek Srikumar, Pei Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D. Manning. Modeling biological processes for reading comprehension. In *Conference on Empirical Methods in Natural Language Processing*, pages 1499–1510, 2014.

Qian Chen, Xiaodan Zhu, Zhenhua Ling, Si Wei, and Hui Jiang. Enhancing and combining sequential and tree lstm for natural language inference. *arXiv preprint arXiv:1609.06038*, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*, 2014.

Najim Dehak, Patrick J Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet. Front-end factor analysis for speaker verification. *IEEE Transactions on Audio, Speech, and Language Processing*, 19(4):788–798, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

Karl Moritz Hermann, Tomas Kocisky, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. In *Advances in Neural Information Processing Systems*, pages 1693–1701, 2015.

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

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8): 1735–1780, 1997.

Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

Diederik P. Kingma and Lei Jimmy Ba. Adam: A method for stochastic optimization. *CoRR*, abs/1412.6980, 2014.

Dan Klein and Christopher D Manning. Accurate unlexicalized parsing. In *Meeting on Association for Computational Linguistics*, pages 423–430, 2003.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David Mcclosky. The stanford corenlp natural language processing toolkit. In *Meeting of the Association for Computational Linguistics: System Demonstrations*, 2014.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. Ms marco: A human generated machine reading comprehension dataset. 2016.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. URL `http://www.aclweb.org/anthology/D14-1162`.

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

Matthew Richardson, Christopher JC Burges, and Erin Renshaw. Mctest: A challenge dataset for the open-domain machine comprehension of text. In *EMNLP*, volume 3, page 4, 2013.

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

Richard Socher, Chiung Yu Lin, Andrew Y. Ng, and Christopher D. Manning. Parsing natural scenes and natural language with recursive neural networks. In *International Conference on Machine Learning, ICML 2011, Bellevue, Washington, Usa, June 28 - July*, pages 129–136, 2011.

Kai Sheng Tai, Richard Socher, and Christopher Manning. Improved Semantic Representations From Tree-Structured Long Short-Term Memory Networks. In *Proceedings of ACL*, 2015.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700, 2015.

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

Zhiguo Wang, Haitao Mi, Wael Hamza, and Radu Florian. Multi-perspective context matching for machine comprehension. *arXiv preprint arXiv:1612.04211*, 2016.

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

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W Cohen, and Ruslan Salakhutdinov. Words or characters? fine-grained gating for reading comprehension. *arXiv preprint arXiv:1611.01724*, 2016.

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end answer chunk extraction and ranking for reading comprehension. *arXiv preprint arXiv:1610.09996*, 2016.

Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In *ICML*, pages 1604–1612, 2015.