Question Answering System with Question Type Modelling

Ksenia Ponomareva
Codalab username: kp260
kp260@stanford.edu

Abstract

Recently there has been significant progress in applying deep learning neural network based models for solving question answering tasks. In this paper a new model is presented where different types of questions are modelled in the end-to-end training. In particular, the system learns question type embedding and employs a feed forward layer for question adaptation.

1 Introduction

Rapid progress has been made in reading comprehension since the release of the SQuAD dataset [1] less than two years ago as can be seen from the associated leaderboard webpage. Neural network-based models, in particular, demonstrate effectiveness on the SQuAD dataset.

The SQuAD dataset challenge is as follows. Given a paragraph and a question about that paragraph, the model has to answer the question correctly. The unique feature of this dataset is that the answer is contained within the paragraph and as such the system has to identify the span of words, start and end, in a context that answers a given question. The ability to read text and then answer questions about it, is a challenging task for machines, since systems have to be able to model the complex interactions between the context and the question and to 'understand' the text. It has also been reported that there is some sort of lexical or syntactic divergence between the question and passage present in the majority of SQuAD dataset, see [1] and [3].

A simple baseline model has been provided by the course instructors to tackle the SQuAD challenge and the aim is to improve its performance based on the exact match (EM) and F1 score metrics. A number of enhancements to the baseline model have been made, where improvements implemented were inspired by [2], [5] and [3]. In the first two papers, it is believed that there are common features shared by all questions and specific features dependent on the type of the question. Unlike [2] and [5] the updated model learns the question type specific embedding to capture common and discriminitative features of different types of questions, rather than modelling these explicitly. Direct matching of the question aware passage representation against itself, in other words self-attention in [3], has also been implemented.

2 Dataset and Feature Analysis

Before considering any model improvements, histogram plots of the lengths, in tokens, of the context, question and answer in the training data had been produced and statistics collected. These plots are presented in Figures 1-3. From this analysis, it was concluded that 99% of data had question length of less or equal to 23 and context length of less or equal to 325. Span of the answers in the training data had also been considered with special attention paid to where the end of the

answer was positioned in the context, please see Figure 4 for details. It turned out that 99.9% of data had the length between the start of the paragraph and end span of the answer of less or equal to 318 (less than 325) and for 99% of data it was less than 216. Based on these observations, context length was truncated to 325 and question length to 23 in the model.

Answer lengths have been analyzed and it has been observed that 99% of data has answer of less or equal to 21 and 97.5% of data has length of less or equal to 15. This final result was utilized in the predicted span calculation at test time and is further described in section 4.6. The main motivation for limiting the span prediction follows from the analysis of the dev set data predictions, where a common error is that predicted answer is too long.

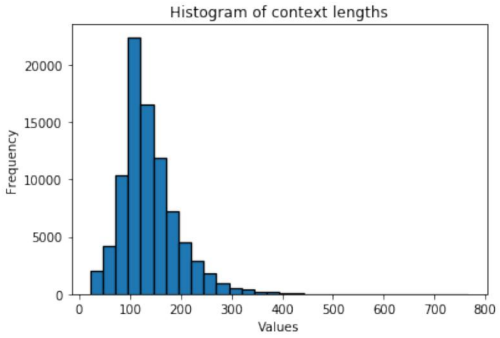

Figure 1: Context length

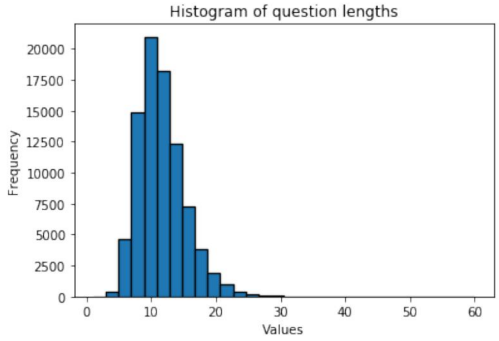

Figure 2: Question length

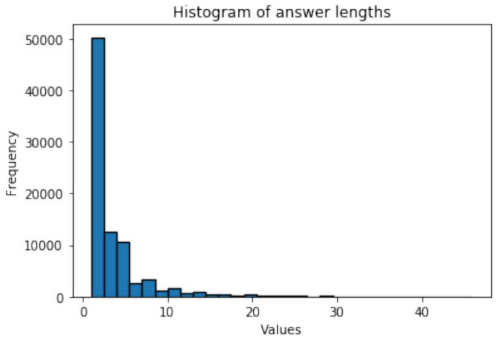

Figure 3: Answer length

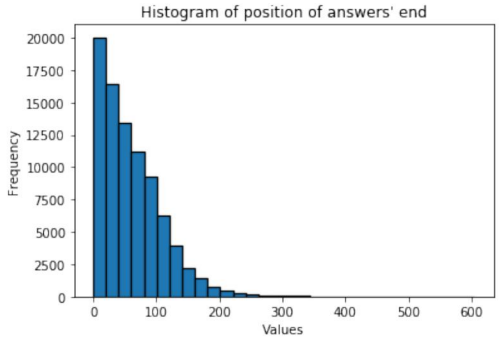

Figure 4: Answer end position

Besides context, question and answer lengths, additional features, such as question types were examined. As is discussed in [2], different types of questions and corresponding answers could potentially have different distributional regularity. Since a ”where” query pays attention to a different type of information in the paragraph as compared to a ”when” query, incorporating question type abstraction in the expression for queries could provide additional clues for searching the answer. Figure 5 provides information on the different question types in the trainable questions data based on the key word frequency. Eleven top query types have been identified, with over 50% of data falling into ”what” category. Here category ”be” refers to any combination of ”is”, ”was”, ”have been” and etc.

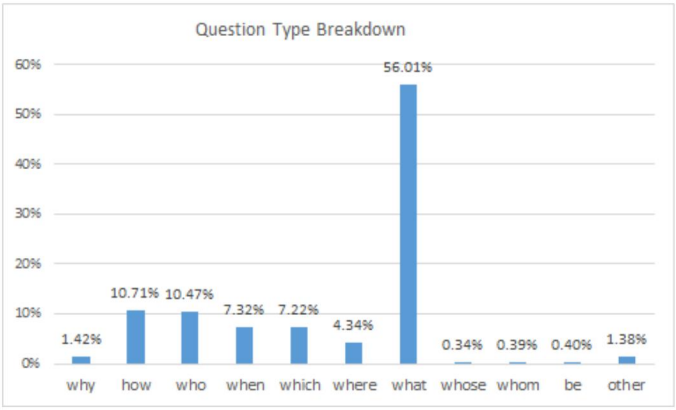

Figure 5: Question types

3 Model Architecture

The general architecture of the new deep learning model is shown in Figure 5. The following sections describe in detail each stage and layer as well as discuss incremental enhancements applied to the provided baseline model.

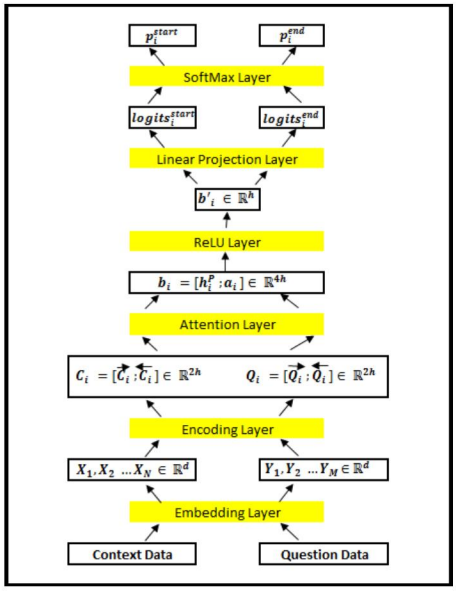

Figure 6: Model architecture

3.1 Embedding layer

Pre-trained GloVe vectors have been used to represent the words in the context and the question. Embedded vector dimensionality, g, has been increased from 100 used in the baseline model to 300 in the updated model.

Following the question type analysis described in section 3 an additional trainable embedding layer has been added to the baseline model for the question vectors. The goal is to capture each query category, however instead of matching each of the eleven query types explicitly as is done in [2], system is left to learn on its own by comparing question vectors to cluster centroids. Each cluster

here models a question type. In order for this calculation to be differentiable, a cosine distance is used instead of matching to the nearest cluster. The details of the process followed are described next.

Given K types of question queries, for every question vector, $Y \in R^{M \times g}$, and each centroid vector of K clusters, $\bar{Y} \in R^{K \times M \times g}$, cosine distance is used to compute similarity weight, w^a, as follows:

$$w_k^a = softmax(cos\_dist(Y, \bar{Y}_k), \alpha), \forall k \in [1, ..., K], \tag{1}$$

$$cos\_dist(u, v) = \frac{< u, v >}{||u|| * ||v||}, \tag{2}$$

$$softmax(Y_i, \alpha) = \frac{exp(\alpha Y_i)}{\sum_j exp(\alpha Y_j).} \tag{3}$$

In the updated model, $\alpha = 50$ is used to ensure that only the closest question type will have a high weight while calculation remains differentiable. Largest w_k^a for each question vector provides a key, k, to the extra embedding matrix row to be used to encode the question type information into extra dimension, e, that is then concatenated with the rest of the question vector. Since in the next layer a shared RNN encoder is used, context embedded vectors are appended with zeros to match the dimension, $d = g + e$, of the new question vectors.

Cluster centroids are updated after w^a has been calculated as follows:

$$\bar{Y}_k' = (1 - \beta w_k^a)\bar{Y}_k + \beta w_k^a Y, \forall k \in [1, ..., K], \tag{4}$$

where $\beta = 0.01$ controls the update amount. This update modifies the center vectors of the K clusters in order to fit each cluster to model different type of questions. Cluster centroids are trainable and are initiated using Xavier initialisation.

3.2 Encoder layer

The baseline model uses a bidirectional GRU RNN as a shared encoding layer between context and query vectors. The enhancement made here is to use bidirectional LSTM instead of GRU to better capture the long-term context of longer paraghraphs.

3.3 Question adaptation

Each question, $Y \in R^L$, can be decomposed into two parts: the systematic component driven by the cluster the query belongs to and the idiosyncratic component that is specific to each question. Following methodology described in [2], the cluster information is encoded in a vector $\bar{Y}^c$ and the unique question information is encoded in δ_Y. Both of these components, as well as the original query vector are fed to the feed-forward layer to obtain a new representation, Y', of the question vector. Figure 7 provides a visual summary of the question adaptation utilized in the updated model.

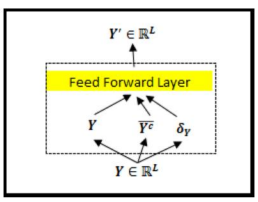

Figure 7: Question adaptation

Question adaptation follows equations (1)-(3), detailed in section 4.1. As soon as similarity weight vector, w^a has been computed, cluster centroids are updated using equation (4) and $\bar{Y}^c$ and δ_Y are

calculated as follows:

$$\bar{Y}^c = \sum_{k}^{K'} w_k^a \bar{Y}_k,$$
$$\delta_Y = Y - \bar{Y}^c.$$

As a final step, the adapted representation of the query is obtained by using a feed forward layer with Relu:

$$Y' = Relu(W[Y, \bar{Y}^c, \delta_Y]).$$

In the updated model, Y is the question hidden state Q that is the output of the encoder layer.

3.4 Attention layer

The baseline model uses a basic dot-product attention where context hidden states attend to the question hidden states. In the updated model, question hidden states are adapted first before being attended to. Self-attention for question aware context, as instroduced in [3], has also been implemented. Hidden states concatenated with the attention output are passed on to the output layer.

3.5 Output layer

This layer has not been changed from the baseline.

3.6 Prediction

In the baseline model predictions for the answer text are done independently of each other, which can sometimes result in an empty string. To obtain predicted in this updated model, results of the answer span analysis described in section 3 are used here. Following an implementation in [4], start and end location pair (i,j) are chosen with i$\leq$ j $\leq$ i+15 that maximizes $p^{start}(i)p^{end}(j)$.

4 Experiments

This section provides further details on how experiments were run including model configurations and hyperparameter tuning. Starting from the baseline the following investigations and changes have been done for this updated model.

Model 1:

- Pretrained word vectors: GloVE.6B.300d was used instead of GloVE.6B.100d in the baseline model.
- Context length truncated to 325, question length truncated to 23 word tokens.
- Updated answer span as in section 3.6 with maximum answer length limited to 15 word tokens.
- Encoder layer RNN was updated to LSTM instead of GRU.
- Dropout values have been experimented with in the range of [0.15, 0.5] in order to avoid early overfitting, with $dropout = 0.25$ providing the best scores.

In the official dev evaluation in codalabs model 1 has achieved a relative improvement over baseline model with 46.45 and 36.79 as F1 and EM scores respectively, tensorboard results are also shown on figure 8 below.

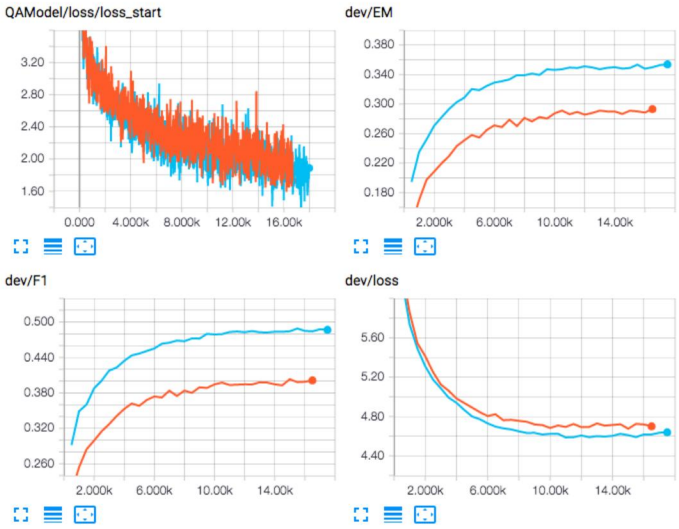

Figure 8: Model1(blue) vs Baseline (orange)

It can be observed that limiting answer span in the predicted text has improved both EM and F1 scores, see figures 9-10 below for comparison between baseline and model1.

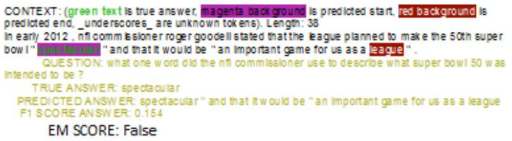

Figure 9: Dev set example baseline

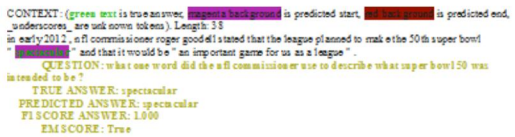

Figure 10: Dev set example model1

After completing model 1 updates, self-attention was introduced to the attention layer in the model, section 4.6.

Self_attn:

- Pretrained word vectors: GloVE.6B.300d was used instead of GloVE.6B.100d in the baseline model.
- Due to OOM errors, context length was further truncated to 200.
- Updated answer span as in section 3.6 with maximum answer length limited to 15 word tokens.
- Encoder layer RNN was updated to LSTM instead of GRU. Self-attention hidden RNN was GRU for computation efficiency.
- Hidden size for encoder and self-attention layer had to be reduced to 75 due to OOM issues.
- Dropout values have been experimented with in the range of [0.15, 0.5] in order to avoid early overfitting, with $dropout = 0.25$ providing the best scores.
- Different optimizers were used with Adadelta with $\alpha = 1$, $\rho = 0.95$ and $\epsilon = 1e - 6$ providing a slight improvement over Adam.
- Batch size had to be reduced from 100 to 25 due to OOM issues.

Unfortunately, adding self-attention layer to model 1 caused EM and F1 scores to drop signifficantly. One possible explanation is that it could have potentially been caused by having to reduce hidden state size of the encoder from 200 to 75 due to OOM issues and hence looses predictive power due

to reduction in number of parameters. Another possible explanation is that a simple dot-product attention between context and question hidden states is too basic and needs to be enhanced first before self-attention is introduced. In the dev set example below, question aware context seems to put too much attention on the word "tardis" in the paragraph since it is "aware" that this word is also repeated in the question.

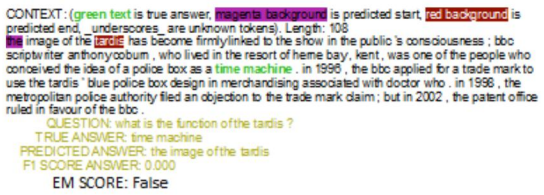

Figure 11: Dev set example for self-attention

Due to the low scores, self-attention model has not been used further. Instead, question specific embedding and question adaption are introduced to model 1.

Model2:

- Hidden size for biLSTM RNN encoder layer had to be reduced to 150 due to OOM issues.
- Dropout values have been experimented with in the range of [0.15, 0.5] in order to avoid early overfitting, with $dropout = 0.5$ providing the best scores.
- Clusters: K=11 is used for question embedding and Kc=100 is used for question adaptation part.
- Learning rate: experiments were done for a range of learning rates between 0.005 and 0.0005, with 0.0005 providing the best scores.
- Instead of initiating clusters by partitioning question vectors, all cluster centroids are initiated using Xavier initialization, are trainable and are also updated after similarity weights are calculated. Making question embedding clusters trainable has improved the EM score.
- Extra embedding dimension used in question type embedding, e, is set to 50.

Unfortunately, model2 has failed to improve baseline dramatically ending up with similar scores of 39.40 for F1 and 28.07 for EM in tensorboard and 39.28 for F1 and 31.85 for EM in the official dev evaluation in Codalabs. This could have been caused by the reduction in the hidden size of the encode from 200 to 150 due to OOM issues. Another reason could be that over 50% of the training data set consists of "what" questions and there potentially is not enough data to train other types of questions properly.

5 Conclusions

In this paper, two enhancement models have been introduced. Unfortunately due to lots of debugging issues improvements over the baseline model have not been as great as anticipated. Further work could include enhancing the baseline model with more complex attention and decoder layers, as well as further hyper parameter tuning and bug fixing. It might be also useful to consider additional training data sets with more variety of question types present and using a larger pretrained word vectors set, such as CommonCrawl.840B.300d.

Acknowledgments

I would like to thank all the CS224N staff for the great class experience.

References

[1] Rajpurkar, P., Zhang, J., Lopyrev, K. & Liang, P. (2016) SQuAD: 100,000+ Questions for Machine Comprehension of Text. *CoRR*, abs/1606.05250.

[2] Zhang, J., Zhu, X., Chen, Q., DaiL., Wei, S. & Jiang, H. (2017) Exploring Question Understanding and Adaptation in Neural-Network-Based Question Answering. arXiv:1703.04617v2.

[3] Wang, W., Yang, N., Wei, F., Chang, B. & Zhou, M. (2017) Gated Self-Matching Networks for Reading Comprehension and Question Answering. ACL2017.

[4] Chen, D., Fisch A., Weston, J., & Bordes, A. (2017) Reading wikipedia to answer open-domain questions. arXiv:1704.00051.

[5] Hu, M., Peng, Y. & Qiu, X. (2017) Reinforced Mnemonic Reader for Machine Comprehension. arXiv:1705.02798v3.