A Modular Architecture for Machine Comprehension

William A. Clary
Department of Electrical Engineering
Stanford University
Stanford, CA, 94305
wclary@stanford.edu

Abstract

This paper uses a modular architecture which answers machine comprehension questions on the Stanford Question Answering Dataset. The model described achieves respectable performance on the SQuAD leaderboards (F1: 73.673, EM: 63.831) and analysis of its performance highlights the necessity of certain restrictions on the way the model makes its final predictions.

1 Introduction

Recently in the field of natural language processing there has been significant interest in the problem of machine comprehension. Problems within the realm of machine comprehension typically attempt to mimic the ability to 'understand' text in a method similar how a human might. Typically this entails answers questions based on input text. Within this particular problem framework is performing well on the Stanford Question Answering Dataset (SQuAD). This dataset includes problems where the algorithm is given question and a context with which to answer the question. In this particular dataset the question is constrained to be within the context text, specifically within a continuous span of text from the context as demonstrated below:

- Question: what were the main influences of the official language of the empire ?
- Context: ottoman turkish was the official language of the empire . it was an oghuz turkic language highly influenced by persian and arabic . the ottomans had several influential languages : turkish , spoken by the majority of the people in anatolia and by the majority of muslims of the balkans except in albania and bosnia ; persian , only spoken by the educated ; arabic , spoken mainly in arabia , north africa , iraq , kuwait , the levant and parts of the horn of africa ; and somali throughout the horn of africa . in the last two centuries , usage of these became limited , though , and specific : persian served mainly as a literary language for the educated , while arabic was used for religious rites .
- Answer: persian and arabic

This simplifies the problem by reducing it to selecting text from the context. The algorithm must select the a start and end position for the answer and is evaluated by closely its predictions coincides with human decision.

This paper demonstrates the implementation of a SQuAD model (Figure 1) that uses a multi-stage architecture which performs reasonably well on the SQuAD dataset. Core to almost all high performing SQuAD models is an effective way of associating different parts of the question with different parts of the context. This implementation uses the Bi-Directional Attention Flow layer from (Seo et al., 2017), an attention method which has been demonstrated to produce highly performing models. In fact, the highest performing architecture demonstrated in this paper closely follows the implementation demonstrated in (Seo et al., 2017). With an initial encoding layer, the BiDAF attention layer, a modeling layer to perform synthesis, and an output layer to finally select the start and end of the question, this architecture progressively encodes the context and question together to select the correct answer.

This paper present multiple different experiments which investigate the impact that different modeling choices and hyper-parameters have on the final squad performance. Specifically, this paper finds minimal difference between using a GRU and a LSTM RNN cell, and between using a simple dense layer to predict the start and stop locations instead of a LSTM to condition the end location on the start location. Finally this paper presents a SQuAD model which achieves the scores of (F1: 73.673, EM: 63.831) on the official SQuAD test leaderboard.

2 Related work

Due to the popularity of the SQuAD challenge and the competitiveness of the SQuAD leaderboards, there already exists an abundance of literature devoted specifically to the SQuAD challenge. Almost all existing SQuAD models employ some form of RNN encoding Layer to develop a more useful representation of the context and the question. This is evident for even relatively early work on this topic such as (Yu et al., 2016). The question and context encodings are generated by a RNN, with the specifics of the RNN layer varying slightly between different implementations, such as choosing to use an LSTM or a GRU cell.

Perhaps the most significant difference between SQuAD models is the exact form of attention used. One effective attention implementation is the bidirectional attention flow method demonstrated in (Seo et al., 2017), which unlike normal attention methods allows for attention to flow from the context to the question and from the question to the context, instead of most simple attention mechanisms which only include attention flow from the question to the context. Other attention flow mechanisms employ 'coattention' methods which involve multi-step attention mechanisms (i.e. attention representations on top of other attention representations), such as (Xiong et al., 2018). Other attention mechanisms exist which allow for attention within representations instead of merely between representations have also demonstrated significant success, being a core element of the highly performing r-net model (Natural Language Computing Group, Microsoft Research Asia 2017)

A variety of other tools have been employed to improve the performance of different SQuAD models. The utilization of character level embeddings, as demonstrated by (Yang et al., 2017), have demonstrated the ability to improve the quality of the initial representations of the context and questions yielding higher performance on SQuAD tasks. Different kinds of answer mechanisms exist, such as those which directly compute the probability of different spans instead of independently predicting start and stop locations (Yu et al., 2016), and using multi-step reasoning methods (Liu et al., 2017) have been shown to improve the performance of the output modules of SQuAD models.

3 Methodology

The primary model described in this paper (Figure 1) is similar to the model utilized in (Seo et al., 2017) with a few differences. The overall structure of the model is very modular with a definitive sequence of compartmentalized layers which cumulatively form the model output.

3.1 Word Embedding Layer

Utilizing pre-trained Glove vectors a series of word embeddings are generated for both the question up to a length T for the context and M for the question. This results in the context embeddings $x_i \in R^{100}$ for $i = \{1, 2, ..., T\}$ and question embeddings $q_i \in R^{100}$ for $i = \{1, 2, ..., M\}$. Passages are either zero padded or truncated to fit this length.

3.2 RNN Encoding Layer

With these embeddings for the context and the question encodings for both the context and question are generated by a bidirectional LSTM layer. Specifically

$$\{\overrightarrow{c_1}, \overleftarrow{c_1}, ...\overrightarrow{c_T}, \overleftarrow{c_T}\} = biLSTM(\{x_1, ..., x_T\}) \tag{1}$$

$$\{\overrightarrow{u_1}, \overleftarrow{u_1}, ...\overrightarrow{u_M}, \overleftarrow{u_M}\} = biLSTM(\{q_1, ..., q_M\}) \tag{2}$$

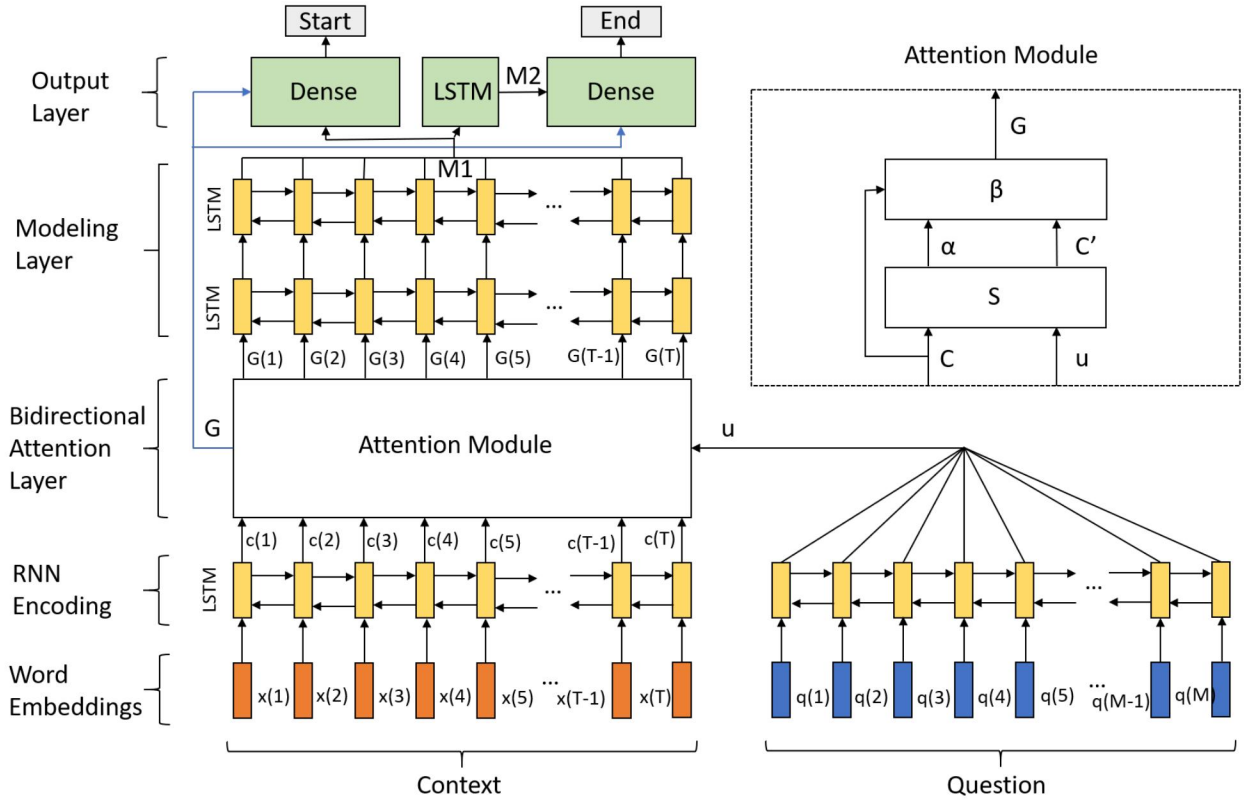

Figure 1: SQuAD Model with BiDirectional Attention Module

$$c_i = [\overrightarrow{c_i}, \overleftarrow{c_i}] \in R^{200} \tag{3}$$

$$u_i = [\overrightarrow{u_i}, \overleftarrow{u_i}] \in R^{200} \tag{4}$$

3.3 BiDirectional Attention Layer

This paper employs a bidirectional attention layer which relates the context and question encodings(c and u) into a attention aware output G. This is described mathematically below:

$$S_{ij} = w^T[c_i, u_j, c_i \circ u_j] \tag{5}$$

$$a_i = softmax(S_{i,:}) \in R^M \tag{6}$$

$$\alpha_i = \sum_{j=1}^{M} a_j^i u_j \tag{7}$$

$$m_i = \max_j S_{ij} \tag{8}$$

$$\beta = softmax(m) \tag{9}$$

$$c' = \sum_{i=1}^{T} \beta_i c_i \tag{10}$$

Finally the output is formed:

$$G_i = [c_i; \alpha_i; c_i \circ \alpha_i; c_i \circ c'] \in R^{800} \quad \forall i \in \{1, .., T\} \tag{11}$$

3.4 Modeling Layer

The output of the attention layer is then fed into a modeling layer which encodes the information the context gained of the question from the attention layer:

$$\{\overrightarrow{M_1^0}, \overleftarrow{M_1^0}, ...\overrightarrow{M_T^0}, \overleftarrow{M_T^0}\} = biLSTM(\{G_1, ..., G_T\}) \tag{12}$$

$$M_i^0 = [\overrightarrow{M_i^0}, \overleftarrow{M_i^0}] \in R^{200} \tag{13}$$

$$\{\overrightarrow{M_1^1}, \overleftarrow{M_1^1}, ...\overrightarrow{M_T^1}, \overleftarrow{M_T^1}\} = biLSTM(\{M_1^0, ..., M_T^0\}) \tag{14}$$

$$M_i^1 = [\overrightarrow{M_i^1}, \overleftarrow{M_i^1}] \in R^{200} \tag{15}$$

3.5 Output Layer

The output layer first predicts the start of the answer and then the end. Mathematically the distribution over start locations is predicted:

$$p^{\text{start}} = \text{softmax}(w_1^T[G; M^1]) \tag{16}$$

Then the distribution over the end is predicted by:

$$\{\overrightarrow{M_1^2}, \overleftarrow{M_1^2}, ...\overrightarrow{M_T^2}, \overleftarrow{M_T^2}\} = biLSTM(\{M_1^1, ..., M_T^1\}) \tag{17}$$

$$M_i^2 = [\overrightarrow{M_i^2}, \overleftarrow{M_i^2}] \in R^{200} \tag{18}$$

$$p^{\text{end}} = \text{softmax}(w_2^T[G; M^2]) \tag{19}$$

With these distributions over start and end locations the loss is computed as loss $= -\log p^{\text{start}}(i_{\text{true start}}) - \log p^{\text{end}}(i_{\text{true end}})$. Final predictions are $\ell^{\text{start}} = \text{argmax} \quad p^{\text{start}}$ and $\ell^{\text{end}} = \text{argmax} \quad p^{\text{end}}$.

4 Experiments

For this project the dataset used was the aforementioned SQuAD. The dataset was split into 3 parts, the training set, the dev set, and the final secret test set. The training set was used to actually train the model, the development set was used evaluate the models performance as a function of hyperparameters, and the final secret test set was used to determine the model's 'official' performance. With the final goal of this project being to create a model that performs well on the SQuAD, initial investigation entailed finding the optimal model structure for this problem. After finalizing the model structure, hyperparameters where selected which performed best for this model structure. With this model and these hyperparameters training was continued to achieve the best possible performance.

Table 1: Hyperparameter Table

PARAMETER	VALUE
learning rate	0.001
max gradient norm	5.0
dropout	0.20
batch size	80
hidden size	100
context len	350
question len	30
embedding size	100

4.1 Structure Experiments

An investigation finding the optimal structure for this problem began with determining whether the LSTM output layer described above was truly necessary or whether similar performance could be attained by a simpler layer. Specifically the default output layer:

$$b'_i = \text{ReLU}(W_3 b_i + v) \in R^h \tag{20}$$

$$\text{logits}_i^{\text{start}} = w_{\text{start}}^T b'_i + u_{\text{start}} \tag{21}$$

$$\text{logits}_i^{\text{end}} = w_{\text{end}}^T b'_i + u_{\text{end}} \tag{22}$$

$$p^{\text{start}} = \text{softmax}(\text{logits}_i^{\text{start}}) \tag{23}$$

$$p^{\text{end}} = \text{softmax}(\text{logits}_i^{\text{end}}) \tag{24}$$

was compared to the implemented layer by running both models for 5000 iterations and presuming that that the relative performance between these two output layers after this relatively low training time would be indicative of performance after a more substantial training time. In this experiment the default layer achieved the tensorboard scores: (F1: 0.6596, EM: 0.5077) and the LSTM output layer achieved the scores: (F1: 0.6589, EM: 0.5070). These scores were considered to be so similar that no significant differentiation could be made between the two output layers. While this likely indicates that complicated output layers of many squad models are not too helpful towards achieving high performance, the LSTM output layer was selected for the final model due to its popularity among other SQuAD models.

Additionally, this paper investigated the difference between using an LSTM vs GRU cell for all the recurrent networks utilized in the model. Similar to the investigation of output layers the model was ran for 10000 iterations and the two models were compared at this point. The LSTM cell achieved the scores: (F1: 0.6604, EM: 0.5129) while the GRU cell achieved the scores: (F1: 0.6558, EM: 0.5041). While the scores were once again very similar, the LSTM cell was chosen for its minutely better performance compared to the GRU cell.

4.2 Hyperparameter Experiments

The hyperparameters in table 1 where chosen for the final model. A somewhat informal investigation was conducted into the hyperparameters to be used for this project. This procedure began with determining reasonable values for the context and question length. Upon examining the test set questions it was quickly determined that only a trivial number of samples and a question length greater than 30 or a context length of 350, and thus these values where chosen to be the minimum length that encompassed the vast majority of samples. With the given model, context length and question length, the hidden size and batch size were chosen to reduce the model size below 8 gb, a small enough

size to fit on the gpu used for training. The hyperparameter that was most iterated upon was the dropout rate. Initial development began with this parameter set to 0.15. While this resulted in good performance on the training set, the difference between the loss on the training set and the loss on the dev set grew significantly while the dev loss stopped decreasing, indicating that the model was overfitting the training set. For example at 8000 iterations the smoothed loss on the training set was 2.499 while on the dev set it was 2.981. After changing the dropout rate to 0.20, this gap went away and 0.20 was the dropout rate chosen for the final model.

4.3 Final Results

Having finalized the model and hyperparameters, this model was let train for approximately 14 hours or for 15000 iterations and achieved its peak performance on the dev set at 14000 iterations, with tensorboard dev scores: (F1: 0.6740, EM: 0.5272). The official codalab test scores were: (F1: 73.673, EM: 63.831). The significant improvement between the tensorboard dev scores and the codalab test scores can be attributed to the different way the scores are computed, with the codalab test comparing the model's answers to 3 human answers and taking the best result. While this performance is impressive, it is still significantly off from the reported BiDAF scores reported in (Seo et al., 2017) of (F1: 77.5, EM: 68.4), a paper which utilized a model very similar to the one here. This could be attributed to either their usage of character level features or different hyperparameter selection.

Analyzing the model outputs individually also proves to be interesting.

- Question: what did luther consider christ 's life ?
- Context: on the other hand , luther also points out that the ten commandments when considered not as god 's condemning judgment but as an expression of his eternal will , that is , of the natural law also positively teach how the christian ought to live . this has traditionally been called the " third use of the law . " for luther , also christ 's life , when understood as an example , is nothing more than an illustration of the ten commandments , which a christian should follow in his or her vocations on a daily basis .
- True Answer: illustration of the ten commandments
- Predicted Answer: nothing more than an illustration of the ten commandments
- F1 Score: 0.727
- EM Score: False

In the example above we see that the model is generally outputting reasonable answers to questions, even when it achieves relatively lower F1 and EM scores than in expectation. This example also demonstrates how the model seems to have learned some nontrivial English usage, such as how 'nothing more than...' is a noun dependent on subsequent information. Examining these predictions directly does illustrate the problem with this model that it sometimes outputs invalid answers:

- Question: what has successfully dealt with ozone depletion ?
- Context: the ipcc process on climate change and its efficiency and success has been compared with dealings with other environmental challenges (compare ozone depletion and global warming) . in case of the ozone depletion global regulation based on the montreal protocol has been successful , in case of climate change , the kyoto protocol failed . the ozone case was used to assess the efficiency of the ipcc process . the lockstep situation of the ipcc is having built a broad science consensus while states and governments still follow different , if not opposing goals . the underlying linear model of policy-making of more knowledge we have , the better the political response will be is being doubted .
- True Answer: the montreal protocol
- Predicted Answer: $\emptyset$
- F1 Score: 0.000
- EM Score: False

While the output layer should produce start and stop locations which result in a valid sapn, it is not impossible under the current model for it to predict invalid spans. Informally, this appears to account for a fair portion of examples,

indicating that a modification that forces the model to choose the end point to be after the start point, such as a span representation, would be very productive improving performance. In conjunction with results from the structure experiments, this suggests that forcing the model to choose from valid answers with a possibly simpler model is more productive than using a complicated output layer, such as the LSTM output layer, to generate potentially invalid answers.

5 Conclusion

In total, this project demonstrates and evaluates a SQuAD model that performs reasonably well by objective measures. While the demonstrated scores (F1: 73.673, EM: 63.831) are not state of the art, they are close and validate this papers approach towards solving this problem. The most significant conclusion that can be drawn form this project is the necessity of implementing a output method which only allows the model to output valid spans of text (the end location comes after the start location). The LSTM output method demonstrated in this paper does not seem to adequately ensure that the output is a valid span from the text. Future work for this project would be to implement some output method, such as a span representation, which ensure that the output is valid. Additionally, it would be interesting to investigate whether using some additional features, such as character embeddings or parts of speech tags, would result in significant improvement for this model.

References

Caiming Xiong, Victor Zhong, Richard Socher (2016) Dynamic Coattention Networks For Question Answering. *ICLR 2017*

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hannaneh Hajishirzi (2016) Bidirectional Attention Flow for Machine Comprehension. *ICLR 2017*

Natural Language Computing Group, Microsoft Research Asia (2016) R-NET: MACHINE READING COMPREHENSION WITH SELF-MATCHING NETWORKS

Xiaodong Liu, Yelong Shen, Kevin Duh and Jianfeng Gao (2017) Stochastic Answer Networks for Machine Reading Comprehension

Yang Yu, Wei Zhang, Kazi Hasan, Mo Yu, Bing Xiang, Bowen Zhou (2016) End-to-End Reading Comprehension with Dynamic Answer Chunk Ranking. *AAA*

Zhilin Yang, Bhuwan Dhingra, Ye Yuan, Junjie Hu, William W. Cohen, Ruslan Salakhutdinov (2017) Words or Characters? Fine-grained Gating for Reading Comprehension *ICLR 2017*