Machine Comprehension Systems on SQuAD Dataset

Shantanu Thakoor, Megha Jhunjhunwala
Department of Computer Science
Stanford University
{thakoor, meghaj}@stanford.edu

Abstract

Machine comprehension is a key subfield of artificial intelligence, with the aim of building a system that can understand some unstructured information. In particular, reading comprehension is an important application of natural language processing, providing a measure of how well computers can understand textual information. For this project, we implement the Bidirectional Attention flow architecture and introduce a soft labeling technique. The resultant model achieves an F1 score of 77.091, and an exact match score of 66.286 on the SQuAD test dataset.

1 Introduction

Machine comprehension is an interesting NLP task which strives towards improving the capability of computers to understand natural languages and their description of how the world functions. Some important machine comprehension tasks include question-answering (Weston et al., 2015), text summarization (Gambhir & Gupta, 2017), and visual question answering (Hu et al., 2017). End-to-end deep learning techniques have led to significant improvement in a number of machine comprehension tasks.

The SQuAD (Rajpurkar et al. (2016)) challenge provides a reading comprehension dataset, with the task being to predict the right answer given a paragraph and a question on that paragraph. The answers are always a set of contiguous words from the paragraph, and so the model only needs to select a ”span” of text from the given context. The SQuAD data set has been created by human crowd-sourcing, and hence it is considered to be a challenging testbed to evaluate intelligent question answering systems.

2 Related Work

Many other Question Answering data sets like SQuAD have been made available recently. WikiQA (CITE) is one such dataset, which uses Wikipedia as a source of passages and answer is a sentence which can be present in any of the different documents. This is a key difference between SQuAD and other datasets as SQuAD restricts the answers to a single passage.

There has been promising work on reading comprehensions done recently. Most of these successful models use an end-to-end deep learning framework, which may be the most important reason for their success. Match-LSTM (Wang & Jiang, 2016) is one such model that uses attention from context to question along with an answer pointer network. Bidirectional attention flow (Seo et al., 2016) is a more complex model with attention flowing from question to context and vice-versa, which helps capture nuanced interactions between context and question words. Another advanced models based on similar two-way attention between context and question is the Dynamic Coattention Network (Xiong et al., 2016).

Character-level CNNs (Conneau et al., 2016) are another type of models that have gained a lot of popularity recently on NLP tasks. Their main advantage is the use of morphology in order to handle out-of-vocabulary words more effectively. These techniques are also used in popular machine reading systems, to further boost performance on a wide range of texts.

Most of the models discussed above output the start and the end of the answer span independently. However, this is a serious model limitation and attempt have been made to condition the end index based on the start (Wang & Jiang, 2016). Another approach to handle this limitation is to directly output probability of all candidate chunks up to a particular length. This appears in the Dynamic Chunk Reader model (Yu et al., 2016), although it is unclear whether it leads to an improvement in performance without heavy feature engineering.

3 Models

3.1 Baseline

The baseline model uses a 1-layer bidirectional GRU (Chung et al., 2015) to convert the input context and question embeddings (pre-trained using Glove (Pennington et al., 2014) into context and question hidden states. The forward and backward hidden states are concatenated to produce the resultant hidden state.

$$\{\overrightarrow{c_1}, \overleftarrow{c_1}, ..., \overrightarrow{c_N}, \overleftarrow{c_N}\} = biGRU(\{x_1, ..., x_N\})$$
$$\{\overrightarrow{q_1}, \overleftarrow{q_1}, ..., \overrightarrow{q_N}, \overleftarrow{q_N}\} = biGRU(\{y_1, ..., y_N\})$$
$$c_i = [\overrightarrow{c_i}, \overleftarrow{c_i}] \in \mathcal{R}^{2h} \forall i \in \{1, ..., N\}$$
$$q_j = [\overrightarrow{q_j}, \overleftarrow{q_j}] \in \mathcal{R}^{2h} \forall j \in \{1, ..., M\}$$

The GRU layer is followed by the attention layer where a dot-product attention is applied between the context hidden states attending to the question hidden states. The attention output is then produced as a weighted sum of the question hidden states.

$$e^i = [c_i^T q_1, ..., c_i^T q_M] \in \mathcal{R}^M$$
$$\alpha^i = softmax(e^i) \in \mathcal{R}^M$$
$$a_i = \sum_{j=1}^{M} \alpha_j^i q_j \in \mathcal{R}^{2h}$$

The attention outputs are concatenated to the context hidden states in order to produce the blended representations. Each of the blended representations are then fed through a fully connected layer followed by a non-linearity. A final linear layer produces two logits (one for start and the other for end) corresponding to each context word. Finally, the softmax function is applied over the logits to obtain a valid probability distribution. A cross-entropy loss function was used along with the Adam Optimizer.

3.2 Bidirectional Attention Flow

The BiDAF model (Seo et al., 2016) uses the idea that attention should be allowed to flow in both directions, i.e, from the context to the questions and vice versa. Figure 1 shows the neural network architecture for the Bidaf model we implemented. It is an improvement of the basic attention layer and produces much better accuracy because of this addition flexibility in the modeling.

A similarity matrix $S \in \mathcal{R}^{N \times M}$ is computed between the context and question hidden states as follows.

$$S_{ij} = w_{sim}^T [c_i; q_j; c_i \cdot g_j]$$

The Context-to-Question attention is similar to the baseline attention layer. Taking row-wise softmax of S yields a probability distribution and the attention outputs are obtained by a weighted sum of question hidden states.

$$\alpha^i = softmax(S_{i,:}) \in \mathcal{R}^M \forall i \in \{1, ..., N\}$$

$$\alpha_i = \sum_{j=1}^{M} alpha_j^i q_j \forall i \in \{1, ..., N\}$$

Now we perform Question-to-Context Attention. We take the max of the corresponding row of S for each context location. Then, we take the softmax over the resulting vector to obtain a probability distribution over context locations. The question-to-context output is obtained by taking the weighted sum of context hidden states using the above probability distribution.

$$m_i = max_j S_{ij} \in \mathcal{R} \forall i \in \{1, ..., N\}$$

$$\beta = softmax(m) \in \mathcal{R}^n$$

$$c^{'} = \sum_{i=1}^{N} \beta_i c_i \in \mathcal{R}^2 h$$

The final blended representation for each context word is obtained by combining the context hidden states c_i, the C2Q attention outputs a_i and the Q2C attention output $c^{'}$.

$$b_i = [c_i; a_i; c_i \cdot a_i; c_i \cdot c^{'}]$$

3.3 Dynamic Chunk Reader

Instead of predicting two independent probability distributions corresponding to the start and the end of the span, DCR directly predicts the probability of the chunk. We only consider the chunks up to a certain length L, a hyper-parameter in our experiments. The blended representation from our Bidaf model is used to construct the representation of resultant chunks.

For any chunk $c_{ij} = w_i, w_{i_1}, w_{i+2}, ..., w_j$, the representation of $c_{ij} = [h_b(i); h_f(j)]$ where h_f and h_b are the forward and backward hidden states respectively.

After the final representation of each chunk is obtained, a simple fully connected softmax layer is used to predict the probability of the chunk being the answer. The main advantage of this model is that it outputs a joint distribution of start and end indices of the correct answer, which is an inherent limitation of the bidirectional attention model.

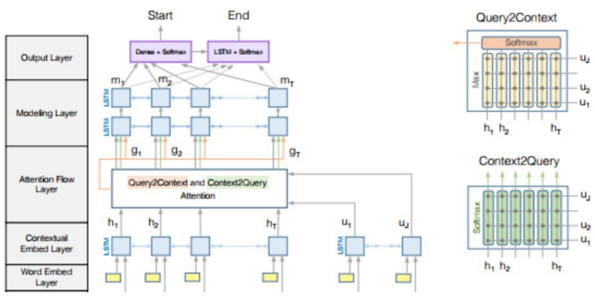


Figure 1: Neural network architecture of the BiDAF model

4 Experiments

4.1 Dataset

All our experiments are on the SQuAD dataset (Rajpurkar et al., 2016). The train set consists of over 86k examples, while the dev and test sets consist of over 10k examples.

Preliminary analysis of the training data shows that most answer spans are usually quite short, with most being less than 5 words long, as seen in Figure 3. Similarly, we see in Figures 4 and 5 that the answer usually rarely lies near the end of the context. Finally, in Figure 2 we see that most contexts are of length less than 250. This insight in particular is helpful, as it allows us to train on only examples of length less than 250, leading to a model with fewer parameters that can be trained much more efficiently.

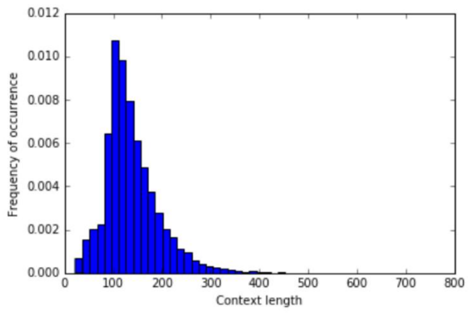


Figure 2: Frequency of context lengths

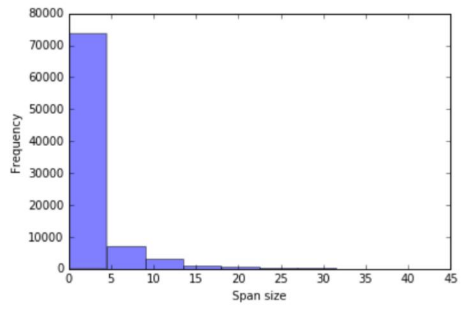


Figure 3: Frequency of answer lengths

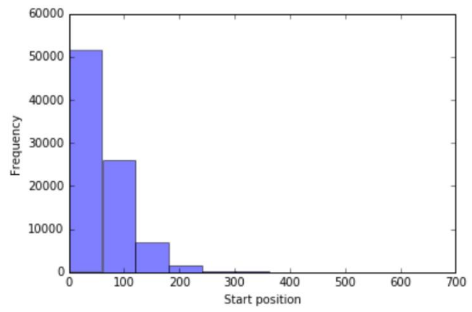


Figure 4: Frequency of answer start position

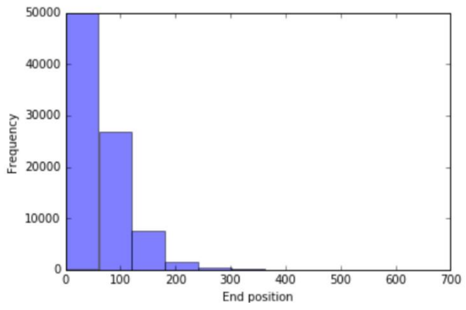
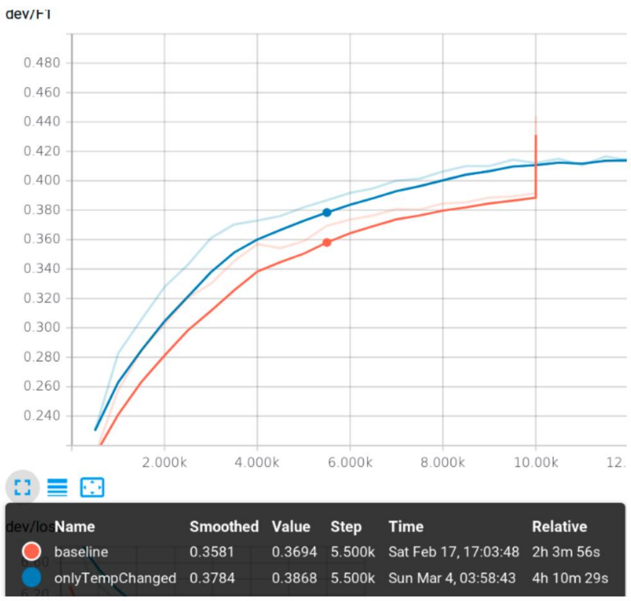


Figure 5: Frequency of answer end position

4.2 SOFT LABELING

Although the task of predicting the optimal start and end positions is being treated as a classification task, it does not take into account the similarity between classes. In our experiments, we find that without heavy feature engineering, the Dynamic Chunk Reader model does not perform as well as our baseline. We hypothesize that this is due to it learning using an overly harsh metric - it penalizes the model unless it predicts both the start and end positions perfectly, which may be difficult. On the other hand, even the baseline does not take into account inter-class similarity - it penalizes the model even if its prediction is very close to the correct position. We propose a novel method we call soft labeling, to take advantage of this inter-class similarity. We define our a distribution $\hat{p}$ where $\hat{p}_i \propto exp(-\tau|i-t|)$, where τ is a temperature hyper-parameter and t is the original target position. We thus train our model to learn to output $\hat{p}$. Note that our optimization step is still natural - the cross-entropy error corresponds to minimizing the cross-entropy between our outputted distribution and the desired distribution $\hat{p}$, which is exactly what we want. Figure 6 demonstrates the significant advantage of this method, even on the baseline model.

We note that soft labeling may improve our score even further on the official SQuAD evaluation. The official SQuAD evaluation compares our answer to the answers of three human annotators, and takes the best of the three scores. Only one of these three answers is available to our model during training. In the event that our model fails to output the exact same output as the answer, we would like to predict an output close to that one answer, as the other two human annotators have also presumably chosen answers similar to the first one - in other words, our model would benefit from taking advantage of inter-label similarity.



Name	Smoothed	Value	Step	Time	Relative
🟥 baseline	0.3581	0.3694	5.500k	Sat Feb 17, 17:03:48	2h 3m 56s
🟦 onlyTempChanged	0.3784	0.3868	5.500k	Sun Mar 4, 03:58:43	4h 10m 29s

Figure 6: Dev F1 scores of baseline model, with and without soft labeling

ADDITIONAL FEATURES

In addition to using pre-trained word embeddings for each context and question word, we augment our inputs by adding in extra features known to be useful. These include:

- For each token, whether it is purely alphabetical, alphanumerical, or other
- For each context token, whether or not it is present in the question
- For each question token, whether or not it is present in the context

4.3 HYPER-PARAMETER DESCRIPTION

In our experiments, we obtain the following optimal values for hyper-parameters and training choices after a wide range of trials:

- We use 200-dimensional GloVe word embeddings, and embeddings are not trainable (as this leads to high overfitting)
- The batch size is 256
- The recurrent network used is an LSTM, with a hidden state size of 150
- We use the Adam optimizer with an initial learning rate of 0.001
- Learning rate is decayed by a factor of 0.98 every 500 iterations

We note that in our experiments, using layer normalization (Lei Ba et al., 2016) greatly decreases generalization error, but this comes at the cost of a 5-fold increase in the training time. For this reason, our best performing models are not trained using layer normalization, as training for more epochs has shown better results.

RESULTS

In Table 1, we provide a summary of the results obtained and a limited ablation study of our various features.

Our best performing model, the ensemble of 3 architecturally similar Bidaf models trained with differing hyper-parameters, achieves a test F1 score of 77.09 and test EM = 67.40, placing us at the 8th position on the leaderboard.

Model	Dev F1	Dev EM
Baseline	39.15	30.90
Baseline + Soft Labeling	41.15	29.54
Dynamic Chunk Reader	44.75	31.58
Bidaf with layer norm	64.81	48.93
Bidaf without features	71.61	59.80
Bidaf with features	73.88	60.19
Ensemble of 3 models	74.86	63.20

Table 1: Results of our models on SQuAD dev set

5 ANALYSIS

In order to understand the predominant type of errors that our best model (ensemble of 3) was making, we computed separate F1 scores on the dev set on each of the different question types. Fig 7 shows the performance of our model on different question types. Our model performs the best on ”Where” and ”Who” question types which makes sense as answers to these questions are usually proper nouns for which enough information is usually captured in the word embeddings. In sharp contrast to such questions are the ”When” and ”How” (usually ”how many”) questions whose answer types include mostly numbers and since we did not train a character level embedding, most of such numbers would be UNK tokens. Our model has very little signal to do good in such question types.

We also plotted heat maps corresponding to the context-to-question attention outputs of our model. The heat maps helped us to analyze probable reasons the model was making errors in predicting the correct answer. For example, whenever the model correctly predicts the answer, it is successful in identifying the most important words in the question that it needs to attend. On the contrary, whenever it does badly, it fails to give attention to some key word which was crucial to extracting the correct answer.

In Fig 8, the example is as follows:
Context: during this period , the island enjoyed increased revenues through the sale of flax , with prices peaking in 1951 . however , the industry declined because of transportation costs and competition from synthetic fibres . the decision by the british post office to use synthetic fibres for its mailbags was a further blow , contributing to the closure of the island ’s flax mills in 1965 .
Question: what did the island sell for increased revenue during this period ?
True answer: flax
Predicted answer: flax
Here, our model pays attention to important question words like ”sell”, ”increase”, and ”revenue”.

In Fig 9, the example is as follows:
Context: the forests play a vital role in harbouring more than 45,000 floral and 81,000 faunal species of which 5150 floral and 1837 faunal species are endemic . plant and animal species confined to a specific geographical area are called endemic species. **Question**: how many endemic floral species do forests harbor ?
True answer: 5150
Predicted answer: 45000
Our model is only giving attention to question words like ”many”, ”floral” and ”species” while not giving enough attention to ”endemic”. Also, since no character level embeddings were used, our model is not able to distinguish between different numeric strings.

6 FUTURE WORK

One major limitation exposed during error analysis of our best performing model was the lack of embeddings for numeric strings. This can be effectively improved by adding character level embeddings as they would help both for numbers and out-of-vocabulary words. Also, as shown by our results, adding textual features greatly improve the performance of the model. Adding POS/NER features would thus greatly enhance the performance of the model. Another observation that we had

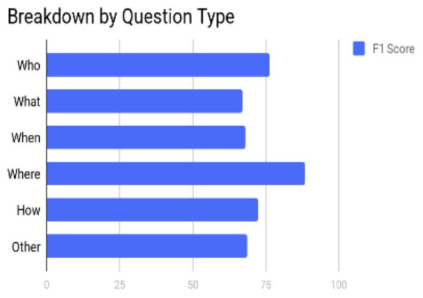


Figure 7: Breakdown of F1 score according to question type

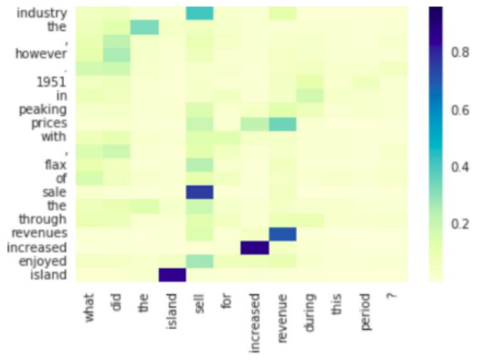


Figure 8: Visualization of attention for a correctly predicted questions

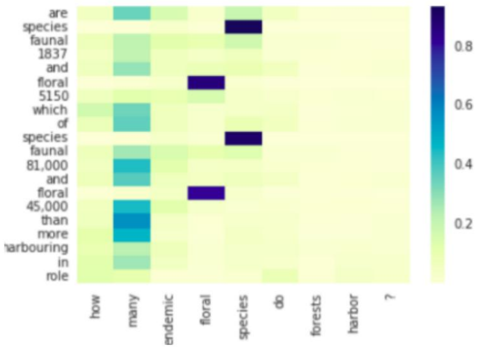


Figure 9: Visualization of attention for a incorrectly predicted question

during experimentation is that different question categories like ”When”,”Who”,”Where” have very different types of answers.We believe that classifying questions into different categories and training different models would help the model learn faster and produce better results.

REFERENCES

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Gated feedback recurrent neural networks. In *International Conference on Machine Learning*, pp. 2067–2075, 2015.

Alexis Conneau, Holger Schwenk, Loïc Barrault, and Yann LeCun. Very deep convolutional networks for natural language processing. *CoRR*, abs/1606.01781, 2016. URL `http://arxiv.org/abs/1606.01781`.

Mahak Gambhir and Vishal Gupta. Recent automatic text summarization techniques: a survey. *Artificial Intelligence Review*, 47(1):1–66, 2017.

Ronghang Hu, Jacob Andreas, Marcus Rohrbach, Trevor Darrell, and Kate Saenko. Learning to reason: End-to-end module networks for visual question answering. *CoRR, abs/1704.05526*, 3, 2017.

J. Lei Ba, J. R. Kiros, and G. E. Hinton. Layer Normalization. *ArXiv e-prints*, July 2016.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

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

Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL `http://arxiv.org/abs/1611.01603`.

Shuohang Wang and Jing Jiang. Machine comprehension using match-lstm and answer pointer. *CoRR*, abs/1608.07905, 2016. URL `http://arxiv.org/abs/1608.07905`.

Jason Weston, Antoine Bordes, Sumit Chopra, Alexander M Rush, Bart van Merriënboer, Armand Joulin, and Tomas Mikolov. Towards ai-complete question answering: A set of prerequisite toy tasks. *arXiv preprint arXiv:1502.05698*, 2015.

Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016. URL `http://arxiv.org/abs/1611.01604`.

Yang Yu, Wei Zhang, Kazi Saidul Hasan, Mo Yu, Bing Xiang, and Bowen Zhou. End-to-end reading comprehension with dynamic answer chunk ranking. *CoRR*, abs/1610.09996, 2016. URL `http://arxiv.org/abs/1610.09996`.