Question answering on the SQuAD dataset with bidirectional attention flow

Brahm Capoor
Department of Symbolic Systems
brahm@stanford.edu

Varun Nambikrishnan
Department of Computer Science
varun14@stanford.edu

Abstract

Our project tackles the challenging task of question answering, which must account for intricate dependencies between question and context. To tackle this problem, we employed an RNN model using Bidirectional Attention Flow, GRU Cells, Custom Feature Engineering and Intelligent Answer Span selection at test time. Our model achieves an F1 score of 0.67 and an EM score of 0.56 on the test set.

1 Introduction and dataset

Reading comprehension is an extremely difficult task for Artificial Intelligence systems. It relies both upon comprehension of a passage of text, but frequently on background knowledge as well. The SQuAD dataset [2] is a means of sidestepping some of these confounds by providing approximately 100,000 (context, question, answer) triples $< c, q, a >$ such that a is an excerpt of c that answers q. Thus, the task of question answering is reduced to finding a span of words in c that answer the question, rather than having to generate original answer text. Context paragraphs in SQuAD are sourced from Wikipedia, and questions and ground truth answers are crowdsourced from Amazon Mechanical Turk. Below is an example of such a triple:

Question: Why was Tesla returned to Gospic?
Context: On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught a large class of students in his old school, Higher Real Gymnasium, in Gospic.
Answer: not having a residence permit

Thus, the problem that a Question Answering system on the SQuAD dataset seeks to solve is identifying the *location* of the answer in the the context paragraph. Specifically, a model must output a predicted beginning index and end index of the answer in the context.

Developing our model involved iteratively building on top of previous versions in response to limitations we identified. Our processes of identifying and determining resolutions for these challenges are outlined, as well as our results at various stages of the process, are found in section 4.

2 Background and related work

There has been a vast swathe of work regarding question answering on the SQuAD dataset, exploring wide ranges of different model architectures and other techniques. We noticed no clear trend in models that scored highly on the public leaderboard. This suggested to us that there was no unambiguously correct strategy, and instead that any of a variety of approaches and improvements could result in high performance. Of particular interest to us was Seo et al.'s paper on Bi-Directional Attention Flow for Machine Comprehension [3]. The paper proposes a modified attention layer wherein instead of summarizing the context paragraph into a single feature vector, attention flows both from context to question and question to context without early summarization, thereby reducing the detrimental effects of information loss. Of additional interest was Chen et al.'s paper on Reading Wikipedia to Answer Open-Domain Questions [4]. Their model consists of additional features than just word embeddings, including binary features as to whether or not the regular, lowercase and lemmatized forms of the context words can be found in the question words. We employ a variant of such features in our model, and additionally match question words to the context to enhance our bidirectional attention flow.

3 Approach

3.1 Model architecture

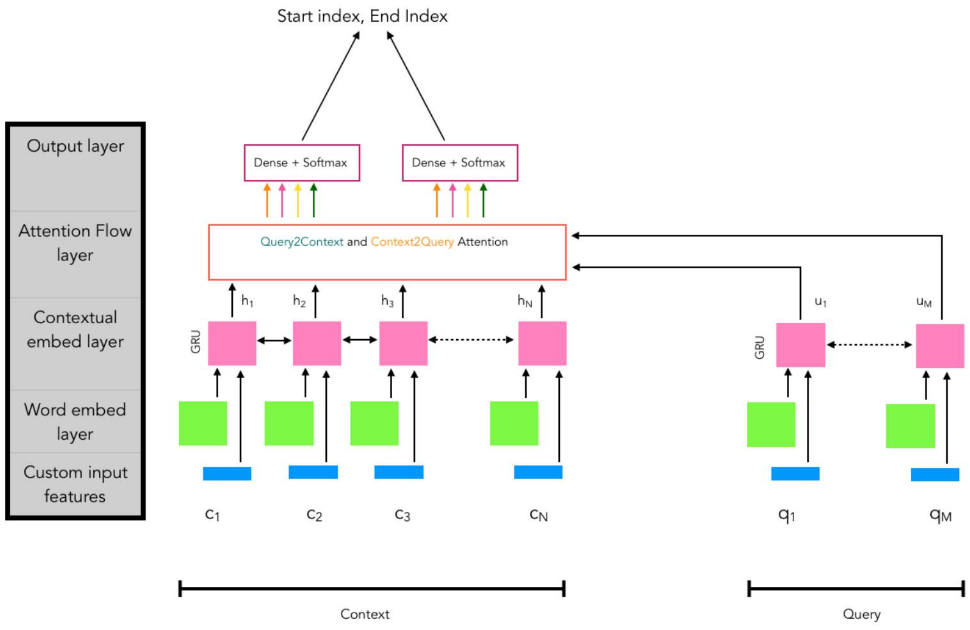


Figure 1: Model Architecture

Our model consists of three main layers: An RNN contextual embed layer, an attention flow layer, and an output layer. We describe each of these layers below.

In the RNN contextual embed layer, each context paragraph is represented by an N-length sequence of word embeddings in $\mathbb{R}^d$, and each question is represented by an M-length sequence of embeddings, also in $\mathbb{R}d$. Embeddings are represented by the green boxes in Figure 1. The model – like the baseline – uses GLoVe embeddings. While word2vec embeddings could also be used, GloVe embeddings use global co-occurrence statistics and are also much faster to train than word2vec embeddings. We then concatenate additional input features (represented by the blue boxes in Figure 1) to these embeddings, as discussed in Section 4.1. These embeddings are fed into a 1-layer bidirectional RNN (using GRU cells) to provide forward and backward hidden states for both the context and question. These hidden states are then concatenated to produce the context hidden states h_i and question hidden states u_j, shown in pink in figure 1.

In the attention layer, we use bidirectional attention flow. This means that not only do the context states attend to the question hidden states (like in basic attention implented in the baseline) but also that the question hidden states attend to the context hidden states. Both these attentions are derived from a similarity matrix, S, where $S_{i,j}$ represents the similarity between the i-th context word and the j-th question word.

The final blended representation is a vector in $\mathbb{R}^{8h}$ (where $2h$ is the dimension of a hidden state), and is the concatenation of the following vectors:

1. `context_hiddens` (pink arrow in Figure 1)
2. `c2q_attention_output` (orange arrow in Figure 1)
3. `context_hiddens * c2q_attention_output` (yellow arrow in Figure 1)
4. `context_hiddens * q2c_attention_output` (dark green arrow in Figure 1)

where 3 and 4 are Hadamard products.

In the output layer, the blended representations are fed through a fully-connected (dense) ReLU activation layer, and then a score is assigned to each context location using a downprojecting linear layer. Lastly, the softmax layer creates a probability distribution accross the context locations. This downprojection and softmax is done twice, to get probabilities for answer start position and answer end position, respectively. In our model we choose the start and end positions s and e to maximize the joint probability of $p(s) \times p(e)$, while meeting the condition that $s \leq e \leq e + 15$. The loss function we use while training is the sum of cross entropy loss for the start and end locations.

4 Experiments

For all our experiments, we trained models for 14,000 iterations and recorded losses, F1 scores and EM scores for both the train and dev sets.

4.1 Model development and Quantitative Results

We began with the baseline model provided in the starter code . The first layer is an bidirectional GRU encoder layer that outputs context and query hidden states. These hidden states are fed into the attention layer which uses basic dot product attention to produce blended representations. These are then processed by the output layer which outputs a probability distribution over context logits. To make a prediction, we take the argmax of each of these distributions to identify start and end indices [1].

While nothing was egregiously wrong with this architecture, its performance was at best mediocre:

Table 1: Baseline results

	Loss	F1 Score	EM Score
Train	3.92	0.62	0.53
Dev	4.69	0.40	0.28

Thus, we began modifying this architecture and examining the effects our improvements had on model performance.

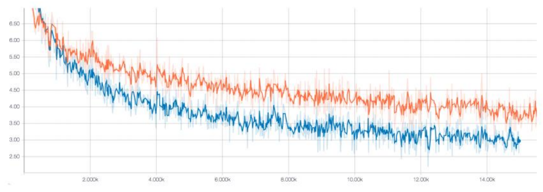
4.1.1 Bidirectional attention flow

In order to address these limitations, as well as to increase performance of our model, we modified the attention layer to use bidirectional attention flow, as discussed in section 3. This led to significant a boost in F1 and EM scores for both the train and dev sets:

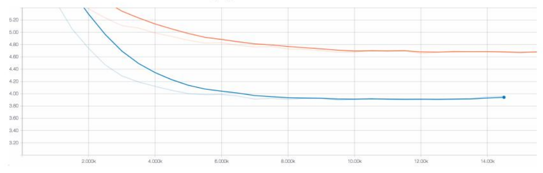
Table 2: BIDAF results

	Loss	F1 Score	EM Score
Train	3.25	0.76	0.66
Dev	3.95	0.53	0.39

We also noticed that the loss for train and dev sets reduced far more quickly.



(a) Train loss



(b) Dev loss

Figure 2: BIDAF vs Baseline loss. BIDAF loss is Blue, Baseline loss is orange.

4.1.2 Augmented input features

Following the modifications to our attention layer, we turned our gaze to incorporating additional input features. Taking a page from Chen et al.'s book, we theorized that the presence of context words in the question was important information and so we consider two indicators:

1. f_{exact}: whether a context word is found in the query

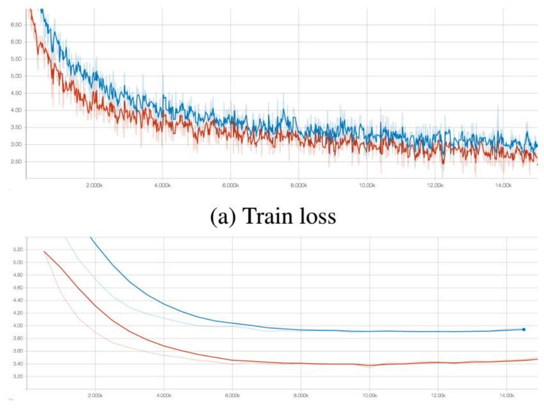
2. f_{lower}: whether the lowercase context word is found the query

Finally, we define a new feature $f_{match} = \max(f_{exact}, f_{lower})$ and append it to our input context vectors. We take the maximum of these two indicators instead of appending each as a separate feature to avoid double-counting words that are lowercase and present in the query. Unlike Chen et al., we also add a similar feature to our input question vectors where f_{exact} and f_{lower} both search for forms of a question word in the query. These additional features also led to a reasonable increase in performance:

Table 3: BIDAF with augmented features results

	Loss	F1 Score	EM Score
Train	2.99	0.81	0.71
Dev	3.46	0.59	0.45

In addition, this improvement also made loss for both train and dev sets decrease much quicker:



(a) Train loss

(b) Dev loss

Figure 3: BIDAF vs Bidaf with Augmented Features loss. BIDAF loss is Blue, Bidaf with Augmented Features loss is red.

4.1.3 Replacing GRUs with LSTMs

Following this, we experimented with replacing the GRU cells in our encoder layer with LSTM cells. Unfortunately, this did not prove fruitful and as a result of this lackluster performance, we decided to revert to using GRU cells in the encoder layer.

Table 4: LSTM Cell results

	Loss	F1 Score	EM Score
Train	3.25	0.77	0.67
Dev	3.49	0.58	0.44

4.1.4 Intelligent Answer Span Prediction

The final significant improvement we made to our model was to use more intelligent answer span selection, as discussed in section 3. This led to a slight boost in EM and F1 scores, at the expense of increased loss:

Table 5: Intelligent Answer Span prediction

	Loss	F1 Score	EM Score
Train	3.20	0.81	0.7
Dev	3.55	0.61	0.46

4.1.5 Hyperparameter tuning

We noticed a significant discrepancy between test set results and dev set results, indicating that our model was probably overfitting to the test set. To combat this, we experimented with various values of dropout. Unfortunately, none of them seemed to address the overfitting issue, only marginally the discrepencay between performance on the train and dev set. This issue might be addressed in future work by experimenting with other hyperparameters such as regularization constants and learning rate.

4.2 Qualitative Results and examples

Below we discuss two examples of our model incorrectly identifying an answer and discuss the limitations they demonstrate and how we might resolve them.

4.2.1

CONTEXT: to remedy the causes of the fire , changes were made in the block ii spacecraft and operational procedures , the most important of which were use of a _nitrogen/oxygen_ mixture instead of pure oxygen before and during launch , and removal of flammable cabin and space suit materials . the block ii design already called for replacement of the block i _plug-type_ hatch cover with a quick-release , outward opening door . nasa discontinued the manned block i program , using the block i spacecraft only for unmanned saturn v flights . crew members would also exclusively wear modified , fire-resistant block ii space suits , and would be designated by the block ii titles , regardless of whether a lm was present on the flight or not .
QUESTION: what type of materials inside the cabin were removed to help prevent more fire hazards in the future ?
TRUE ANSWER: flammable cabin and space suit materials
PREDICTED ANSWER: flammable
F1 SCORE: 0.286
EM SCORE: False

While our model is not strictly incorrect in answering the question, it does not identify the whole answer. These imprecise boundaries are likely because the model predicts similar probabilities for each of the words in the correct answer span, and so the final selection of end index is relatively arbitrary. In addition 'flammable' is a reasonable answer to the question, which suggests the model has learned to succinctly answer questions and dispense with further detail. In order to resolve the first of these issues, we would incorporate Part-of-Speech tags into our input features. Using these features, the model would likely learn that a single adjective is rarely the correct answer. It seems far less trivial to address the second possible cause, and we are unsure if it would genuinely to lead to better performance since some answers actually are this short.

4.2.2

CONTEXT: southern california is home to many major business districts . central business districts (cbd) include downtown los angeles , downtown san diego , downtown san bernardino , downtown bakersfield , south coast metro and downtown riverside .
QUESTION: what is the only district in the cbd to not have " downtown " in it 's name ?
TRUE ANSWER: south coast metro
PREDICTED ANSWER: central business districts
F1 SCORE: 0.000
EM SCORE: False

The model fails to answer this question correctly for two reasons. Firstly, we noticed that the model generally struggled to find the answer when the answer was in a long list of candidates, indicating that it likely assigns similar probabilities to each entity in the list. This question also requires some more generalized reasoning than just semantic understanding. The first of these issues might be resolved by the use of ensemble models, which would aggregate these differences in probability and hopefully make the probability of the correct answer more pronounced, relative to the other candidates. The second would likely only be solved by a drastically different model architecture, or more human engineered features.

5 Conclusions and future work

This project was a fascinating way to engage with research into Deep Learning for Natural Language Processing. Designing a model to best tackle the challenge and weighing all the necessary considerations was a valuable way of understanding real-world contexts in which class material could be applied. Our model presents an effective way to tackle the problem of Question Answering on the SQuAD dataset, producing significantly better results than the baseline. However, it also presents several intriguing opportunities for extension, both in the short and long term.

In the short term, some of the low hanging fruit we can grasp are to explore using additional input features. For example, Chen et al. also incorporated lemmatized versions of context and query words in their definition of f_{match}. In addition, they also incorporated Part of Speech tags, Named Entity Recognition tags and normalized term frequency. As we discussed earlier, this might allow our model to learn more sophisticated syntactic and semantic rules. In addition, we could experiment with different ways to avoid overfitting, for instance by modifying our regularization techniques and learning rate. Another hyperparameter we could tune is the size of our word embeddings. Most of these changes could be made without drastically modifying our model architecture and would likely lead to small but nontrivial improvements.

In the longer term, it would be worthwhile to consider different model architectures. For instance, we toyed with the idea of introducing a character-level CNN to produce character embeddings, but ultimately abandoned it due to time constraints. This would allow us to condition on internal word structure (the morphology of words), handle out of vocabulary words better, and produce more information dense embeddings when combined with the original word embeddings. In addition, we could add more modeling layers to capture the interaction amongst context words, conditioned on the query, since our contextual embedding layer operates independent of the query. Finally, we could consider ensembling different models together and making a decision based on a majority vote to reduce the likelihood of an anomalous answer being selected.

6 Acknowledgments

We would like to thank the instructors and course staff of CS 224N for organzing the class as well as for their enthusiastic and thorough support, both on the project and throughout the quarter. In addition, we'd like to thank Microsoft for providing computing resources.

7 References

[1] CS 224n Default Project Handout

[2] Rajpurkar, J. Zhang, K. Lopyrev, and P. Liang, *"Squad: 100,000+ questions for machine comprehension of text.* arXiv preprint arXiv:1606.05250, 2016.

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

[4] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. *Reading wikipedia to answer open-domain questions.* arXiv preprint arXiv:1704.00051, 2017.