Question Answering using Bidirectional Attention Flow and Co-Attention

Parth Shah
Stanford University
parth95@stanford.edu

Apoorva Dornadula
Stanford University
apoorvad@stanford.edu

Abstract

Machine comprehension is an active field of research in the Natural Language Processing community. In this paper, we explore different models to perform question answering given a context (a passage of text) and a question about the context. We used parts of the Bidirectional Attention Flow architecture [1], Co-Attention Architecture [2], feature engineering, and other performance optimizations to create our final model architecture. Our best performing single model achieves 74.4% F1 score on the SQuaD dev set. Our best performing ensemble model achieves a 76.01% F1 score on the SQuaD dev set and a 76.79% F1 score on the SQuaD test set.

1 Introduction

The rise of neural networks has enabled and advanced many tasks in Natural Language Processing. One such task is machine comprehension. Machine comprehension aims to predict the correct answer to a question about some context (a passage of text). A popular dataset that consists of corresponding questions, context, and answers is the Stanford Question Answering Dataset (SQuaD) [3]. In this paper, we use this dataset and implement parts of the Bidirectional Attention Flow [1] and Co-Attention [2] models to achieve results on the Machine Comprehension task. In addition, we include additional input features; some taken from work done by Chen et al. [4] and some of our own. Ensembling our models also helped boost our performance.

In Section 2, we elaborate on the SQuaD dataset and present our analysis. Section 3 outlines the different architectures we explored and work done related to our paper. Section 4 contains experiments we ran on our models, input features, and an analysis of our results. Lastly, in Section 5, we conclude by stating future work that can be done as an extension to the work done in this paper.

2 Dataset

The dataset used in this paper is the Stanford Question Answering Dataset [3], released in 2016. Since the release of this dataset, there has been countless models and architectures striving to perform well on machine comprehension using this dataset. It consists of over 100,000 question and answer pairs on 500 contexts. An example of a question, context, and triplet is show below.

- Context: ... Denver linebacker Von Miller was named Super Bowl MVP, recording five solo tackles, ...
- Question: Who was the Super Bowl 50 MVP?
- Answer: 32 - 33 (start and end indices corresponding to the phrase Von Miller)

In order to understand this dataset in more detail and motivate future design and hyperparameter tuning decisions, we analyzed the dataset further. We first observed the length of the contexts,

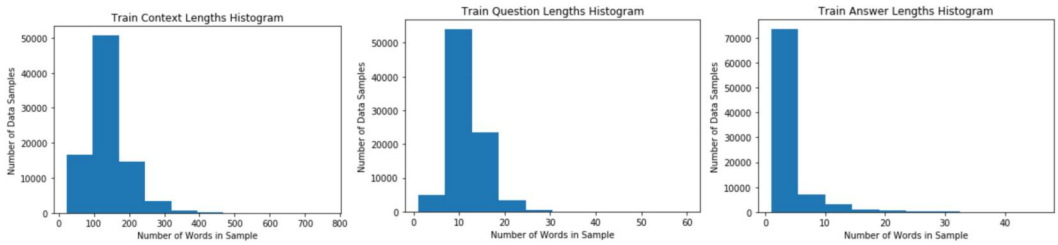


Figure 1: The histogram on the left shows the distribution of context lengths in the dataset. The middle histogram shows the distribution of question lengths in the dataset. Lastly, the right histogram shows the distribution of answer lengths in the histogram.

questions, and answers in the training dataset shown in Fig. 1. Since most of the contexts have less than 300 words, we reduced the context length from 600 (original length set in the baseline model) to 300. This change gave us memory and speed optimizations. Most questions have less than 30 words and most answers have less than 10 words. We also found that most answers started in the first quarter of the context.

3 Approach

In this section, we describe the architectures we used in our models as well as input features and other performance boosters that we implement.

3.1 Baseline

Our baseline model was implemented and designed by the CS224N (Stanford NLP Course) staff. This architecture is shown in Fig. 2. Following is a brief description of each layer in the model:

- Word Embedding Layer: This layers encodes the context and question tokens using GLoVe [5] word embedding resulting in context word embedding $(x_1, .., x_N)$ and question word embedding $(y_1, .., y_M)$
- Contextual Embedding Layer: This layers takes the word embedding generated and runs a bi-directional RNN to generate the context embeddings. The RNN share weights so that the question and the context word embedding are both encoded in the same space.
- Attention Layer: The baseline uses a simple context-to-question attention where each word in the context attends to all the question states. These attended vectors are appended to each context embedding to result in a blended representation which is used in subsequent layers.
- Output Layer: The blended representation are passed through a fully connected layer with a ReLU non linearity and passed through separate softmax layer to predict start and the end positions.

Our subsequent models are built over the baseline by changing the attention layer, adding additional layers or adding input features to improve the accuracy. The following sections discuss different approaches that we used to improve over the baseline performance.

3.2 Bidirectional Attention Flow Approach

Our best single model uses bidirectional attention flow (BiDAF) presented in [1]. The architecture is as shown in (3a). Most layers in the BiDAF model are similar to the baseline however the approach uses a different attention model and the passes the blended vectors through a modelling layer before predicting the start and end position.

BiDAF involves attention flow from both question to context and context to question compared to just uni-directional attention flow in baseline. Fig 3a describes the computation of the bi-directional

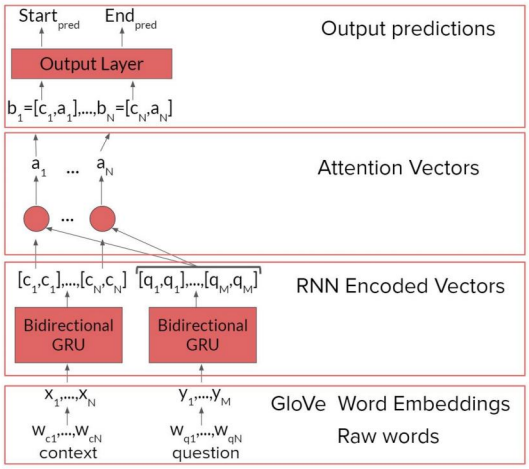
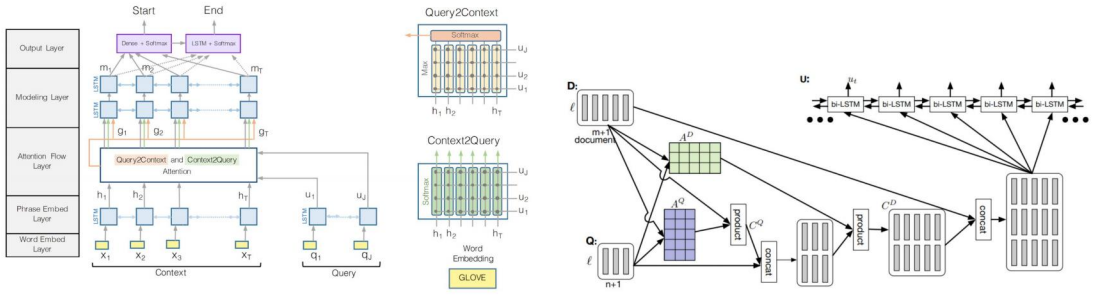


Figure 2: Baseline model architecture

attention. These attended vectors are the appended with the original context embedding and passed through 2 LSTM layers which comprise the modelling layer. The final hidden states are used along with blended representation to then predict the start and end position. Further details about the implementation are described in the original paper [1].

3.3 Co-Attention

Another attention approach that we implemented was Co-Attention [2]. Similar to the BiDAF model[1] the attention layer in the Co-Attention model includes both context-to-question attention and question-to-context attention, however the computation is very different. Fig 3b describe how the attention flow is computed for this model. The original paper also implements a Dynamic Pointer Encoder which iteratively improves upon its start and end prediction but we do not use that for our implementation and experiments and just incorporate the attention module presented in the paper.



(a) (b)

Figure 3: The figure on the left shows the modified BiDAF architecture that we implemented. The figure on the right shows the attention mechanism taken from Co-Attention that we incorporated in our model.

3.4 Feature Engineering

We also experimented with additional input features, added to each word embedding before passing to the encoder RNN. Several of these features were introduced in [4]. The four features that we use in our models are explained below:

- **Exact Match Context**: A feature value of 1 is added to the context word embedding if that word is also present in the question. If the word is not present, a feature value of 0 is added to the context word embedding. This is done for each word in the context.
- **Exact Match Question**: A feature value of 1 is added to the question word embedding if that word is also present in the context. If the word is not present, a feature value of 0 is added to the question word embedding. This is done for each word in the question.
- **Aligned Question Embedding**: The word embedding for a context word is attended to by the word embeddings for the question. The resulting attention output is appended to the context word embedding as additional features. This is done for each word in the context.
- **Word Classification**: For each word in the context and question, an additional feature is added to each word embedding depending on the type of word it is. A feature value of 1 is added if the word is a punctuation (ex: , . ' " ! ? ; ‘ : -). A feature value of 2 is added if the word is a numeric value (consists of numbers). A feature value of 3 is added if the word is alphabetic (consists of letters of the alphabet).

3.5 Additional Performance Optimization

We used several additional techniques to improve performance of our model which are described below:

- **Smart Span**: Given the answer span is a single continuous span in the context, the start answer location cannot appear later than the end answer location. The baseline model, does not uphold this invariant when selecting the start and end locations because they are selected independently. We implemented smart span selection which selects the best start and end location pair where the start location appears at or before the end location. We define "best" as the greatest product of the start and end probabilities.
- **Conditioning End on Start**: Smart span explicitly models the prediction such that the end lies after the start. However, we want our model to learn this on its own. To achieve this, we try to condition the end prediction on the start prediction. BiDAF model [1] uses a separate LSTM encoding layer for the end prediction. We pass initial hidden states to the LSTM based on the start probabilities. The idea is that the encoder would itself learn the relation between the end and the start positions. By conditioning end on start we were able to get comparable results to that of smart span which shows that the approach was able to learn the constraint of end position being after start position.
- **Ensemble**: We also ensemble our models to improve the overall performance on the machine comprehension task. Ensembling involves combining multiple models and using the probability distribution of their individual predictions to make a final prediction. A diagram of one of the ensembling experiments we performed is shown in Fig. 4. This experiment involves ensembling our bidirectional attention flow and co-attention models. We also tried ensembling various co-attention models and various bidirectional attention flow models independently. The results of our ensembling experiments are presented in the Experiments section of this paper.

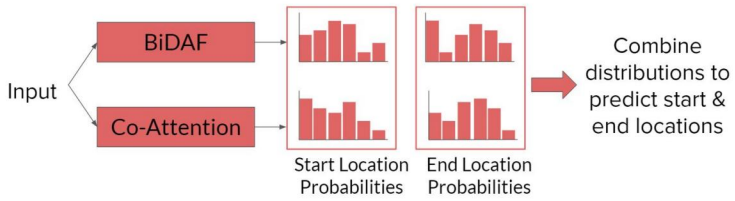


Figure 4: Ensembling Diagram

There are many different ways of combining the probability distributions from different models when ensembling. We ensemble by predicting start and end for each model and assign a confidence score to this prediction equal the product of the probability of the start and the probability of the end prediction. For each example we choose the start and end pair corresponding to the highest confidence score.

4 Experiments

We performed several experiments on the models that we had implemented. In this section we describe these experiments and explain the observations. We also analyze the performance of different features that we added to the embeddings which are described in 3.4

4.1 Input Features

We included 4 additional input features as described in Section 3.4. Figure 5 shows the performance of each input features individually when added to the baseline model.

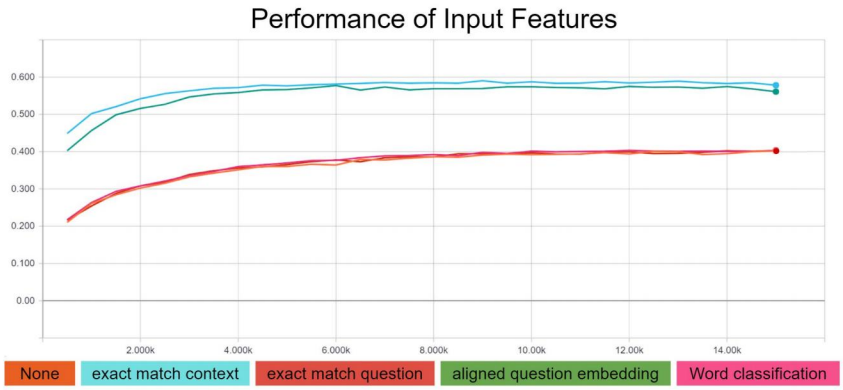


Figure 5: F1 scores of each input feature used individually on the baseline model.

Figure 5 shows that the exact match context and aligned question embedding input features improve the performance (F1 score) by almost 20% over the baseline which is a very significant improvement. The other features, namely the exact match question and word classification features, do not improve the performance of the baseline model.

While addition of input features to the baseline leads to drastic improvement in the performance, we noticed that adding them to more complex BiDAF [1] and Co-Attention [2] does not result in similar large improvements and adding these features on these model lead only about 3-4% increase in the performance. The major reason behind this is that the larger models are able to learn these relations on their own and hence adding explicit feature does not result in similar improvement.

4.2 Hyperparameter Tuning

Each of the implemented model uses several different hyper-parameters. Changing these parameters result in different models and in this section we describe several different experiments that we did for tuning some of these hyper-parameters namely the dropout, the hidden size and the embedding size. Table 1 summarizes the experiments that we did for the 2 models. The base parameters were: Dropout(0.25), Embedding Size(100), Hidden Layer size(100) and we ran all our experiments for 15k iterations.

In-general we see that the lower value of dropout performed slightly better than the higher values and reaches the peak accuracy faster. However, it also begins to overfit on the train data because of the lower regularization. The performance number of higher dropout improved slightly when the model was trained for more time but the overall performance was still poorer compared to dropout value of 0.15.

The original BiDAF paper[1] uses Adadelta compared to Adam. We also experimented with adadelta using the same parameters as mentioned in the paper. The resulting model achieves similar performance compared to the adam but it took lot more iterations to reach the performance. Using Adam optimizer took around 15k iterations to reach peak accuracy while we had to train Adadelta model for over 30k iterations.

Hyper Parameter	F1 Score	EM Score
Dropout (0.15)	**69.20**%	**53.78**%
Dropout (0.25)	68.66%	52.86%
Dropout (0.35)	64.78%	49.28%
Embedding Size (100)	**67.89**%	**52.33**%
Embedding Size (200)	66.85%	51.20%
Embedding Size (300)	66.24%	50.66%
Hidden Layer Size (100)	69.20%	53.78%
Hidden Layer Size (150)	**69.77**%	**54.51**%
Hidden Layer Size (200)	69.62%	54.50%

(a) BiDAF model

Hyper Parameter	F1 Score	EM Score
Dropout (0.15)	**66.07**%	**50.06**%
Dropout (0.25)	63.62%	49.95%
Dropout (0.35)	62.65%	47.41%
Embedding Size (100)	62.40%	47.20%
Embedding Size (200)	62.50%	47.12%
Embedding Size (300)	**63.03**%	**47.70**%
Hidden Layer Size (100)	**62.71**%	**47.15**%
Hidden Layer Size (150)	52.5%	36.79%
Hidden Layer Size (200)	62.01%	46.70%

(b) Co-Attention Model

Table 1: Hyper-parameter tuning results

Our best single model was bidirectional attention flow model with a final F1 score of 74.83% on the SQuaD[3] dev set. This model used a dropout of 0.15, embedding size of 100, hidden size of 150, and two input features (exact match context and exact match question). The values of dropout, embedding size, and hidden size were determined after tuning. The training graphs for tuning dropout, embedding size, and hidden size can be seen in Appendix A (Fig. 10)

4.3 Ensemble Results

The below table contains the F1 and EM performance of ensembling different models. First, we ensembled two of our best performing bidirectional attention flow models. Second, we ensembled two of our best co-attention models. Lastly, we ensembled multiple BiDAF and co-attention models together to achieve our best results on both the dev and test set. On the SQuaD Test set, an ensemble of bidirectional attention flow and co-attention models give us an F1 score of 76.586% and an EM score of 66.894%.

Table 2: Ensemble Results

Models Used	F1 Score	EM Score
BiDAF Models	75.80%	65.62%
Co-Attention Models	73.59%	62.88%
Best BiDAF and Best Co-Attention Models	76.01%	66.05%

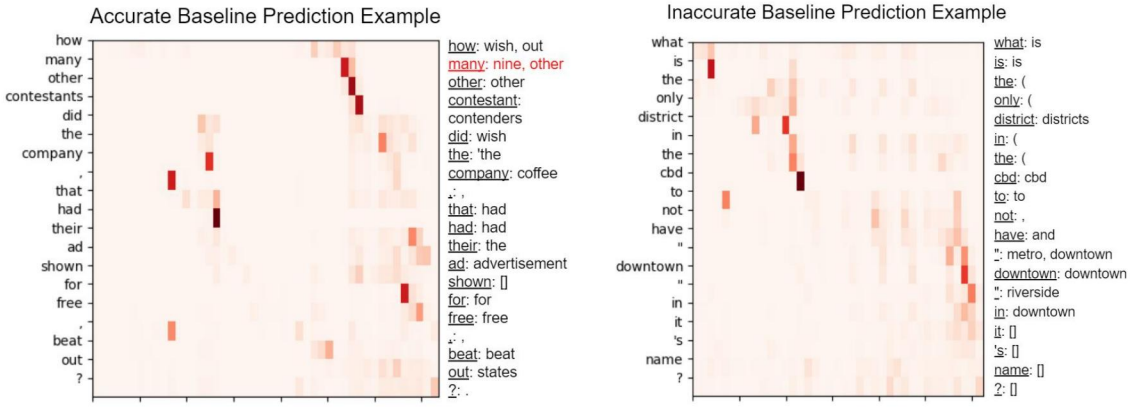
4.4 Attention Analysis

The major difference in our different models in the attention module. In this section we analyze the attention flow for some examples for Baseline and BiDAF model and try to reason why these lead to better or poorer performance.

To understand the start and end predictions made by our models, we observed which words in the context were most attended by the question tokens(given higher weightage). This tells us which words in the context were considered important which is very important for predicting correct answer span. We present the analysis for the baseline model and the bidirectional attention model in Fig. 6 and Fig. 7, respectively.

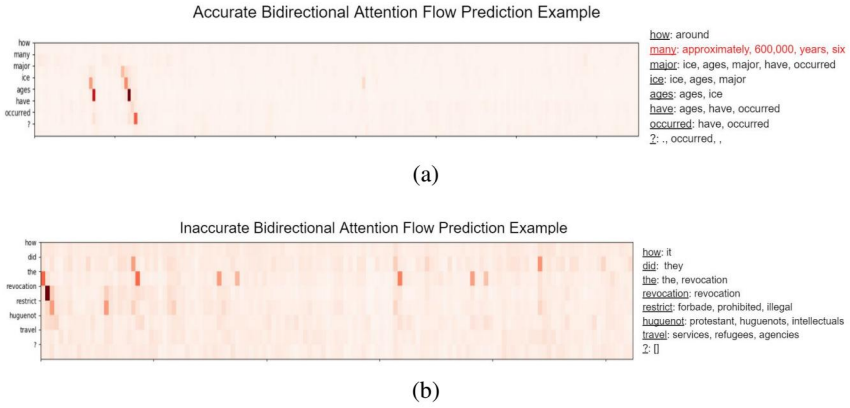
In Fig. 6a, we observe that the word ”many” in the question correctly attends to numerical words in the context, such as ”nine”, which was the answer to the question. Fig. 6b is an example where the baseline model predicted the answer incorrectly. The model predicted the answer to be ”south coast metro” while the answer is ”southern california”. None of the words in the question attend to the words ”southern” or ”california” in the context, so it makes sense that the model got this prediction incorrect.

In Fig. 7a, we observe that the word ”many” in the question attends to numeric words and words that describe quantities. This highlights the answer to the question, which is the word ”six”. In Fig.



(a) (b)

Figure 6: Attention analysis for Baseline



(a)

(b)

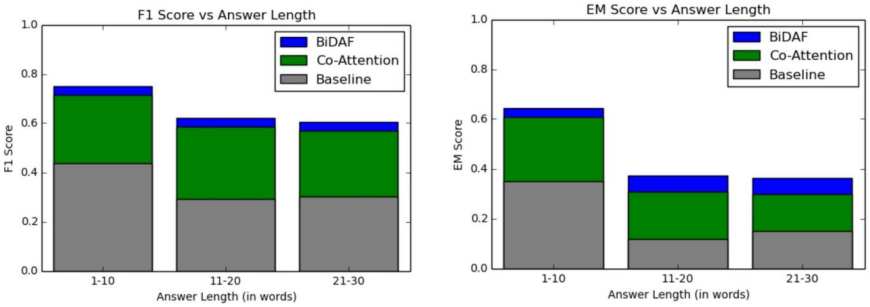
Figure 7: Attention analysis for BiDAF

7b, the true answer, ”prohibited emigration”, is part of the predicted answer, ”forbade protestant services , required education of children as catholics , and prohibited emigration”. Many of the words in the question attend to words that appear in the predicted answer, such as ”protestant” and ”forbade”. We also notice that words in the question attend to words that are close in meaning in the context. For example, ”restrict” is similar to ”forbade”, ”prohibited”, and ”illegal”. Although the F1 score in this example was low, the attention output is understandable and interpretable.

We also analyzed the query-to-context attention flow used in BiDAF. The attention flow predicts a softmax probability for each word in the context which ‘represent’ how relevant it is for the given question. We noticed that in the cases when the question was able to attend to the right words in the context the EM/F1 scores were higher than in the cases the attention was attending to the wrong parts of the context. We provide 2 examples of this in the Appendix.

4.5 Analysis of Answers Predicted

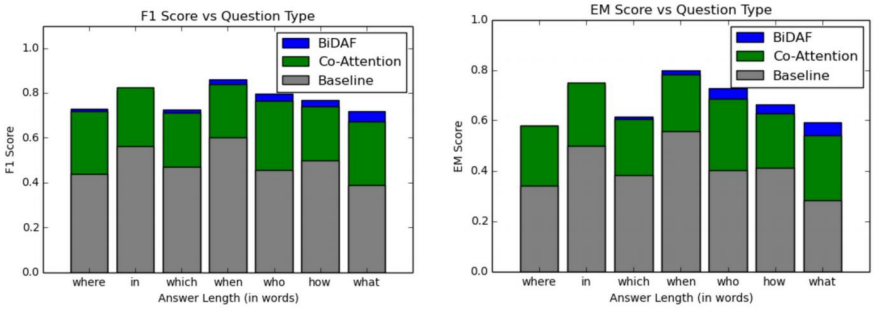
To better understand the strengths and shortcomings of our model, we analyzed the answers it was predicting. We noticed that as the true length of the answer increased, the F1 and EM scores decreased as shown in figure 8. We suspect this is because the majority of answers in the training set are 5 words or less as we can see from figure 1. During training, the model has not seen enough examples with answers with lengths more than 10, and is therefore unable to accurately predict the span.



(a) F1 scores (b) EM scores

Figure 8: Analysis over different answer lengths

We also analyzed the F1/EM scores for each type of question as displayed in figure 9. We observe that questions starting with the word ”when”, ”in”, ”who” performs the best and questions starting with ”which” or ”what” perform the worst. One possible explanation for why ”when”, ”who” questions perform well is that the corresponding answers are generally short and simple like a name of a city or a person while ”which” and ”what” would require more complex reasoning and hence the poorer performance.



(a) F1 scores (b) EM scores

Figure 9: Analysis over different question types

5 Conclusion

Using various aspects of cutting edge models for machine comprehension, we were able to create high performing models on the SQuaD challenge. We used individual bidirectional attention and co-attention models to show how they can be improved over the baseline implementation using feature engineering and other optimizations. Ensembling these models also give us a performance boost. In the future, we would like to try the iterative reasoning technique presented in past work exploring machine comprehension. We would also like to test and improve our model to adversarial inputs.

Acknowledgments

We would like to thank the CS224N course staff for conducting an amazing course this quarter. We both learned a lot and enjoyed doing this project. The baseline model and code was greatly appreciated and it was interesting to focus on the more interesting architectural/modeling aspects of creating a machine comprehension model.

References

[1] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016.

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. *CoRR*, abs/1611.01604, 2016.

[3] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.

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

[5] 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.

6 Appendix A

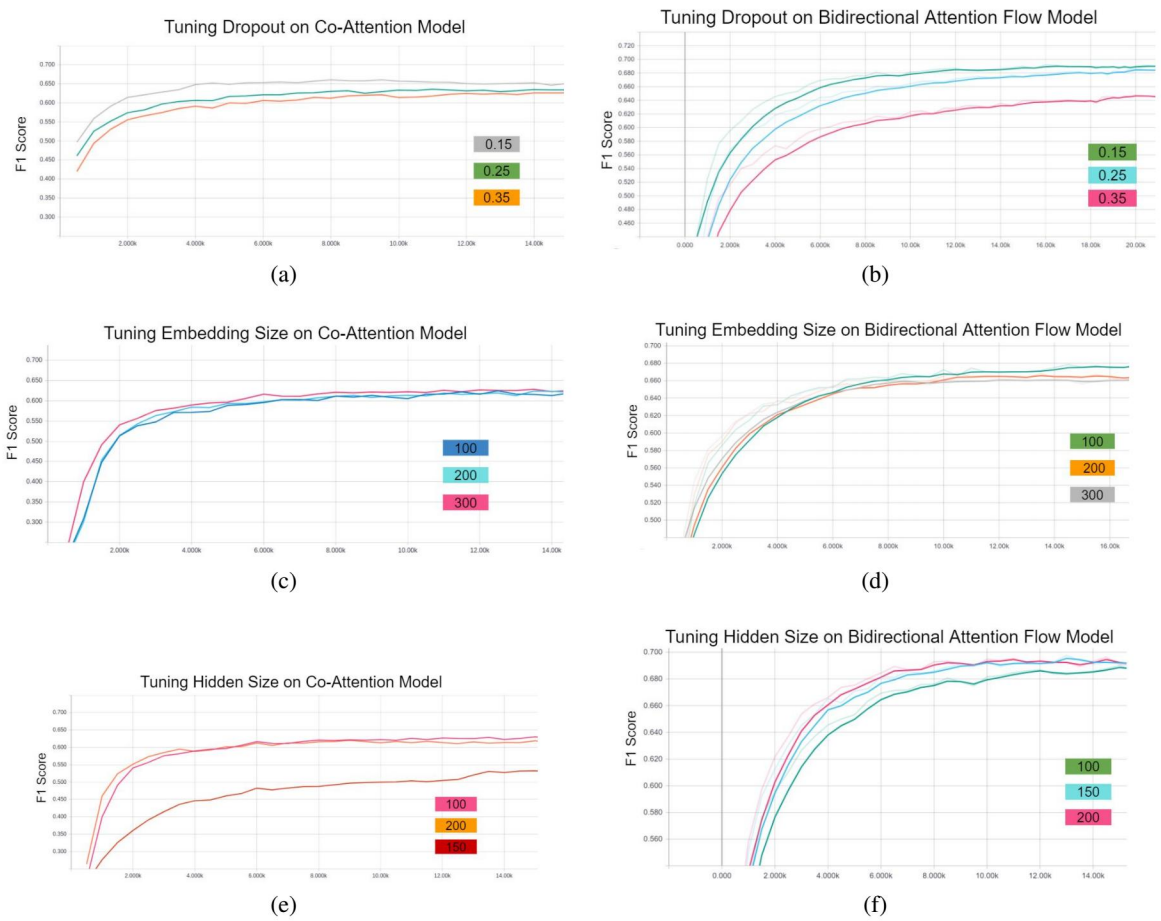


Figure 10: These plots show the effect of tuning dropout, embedding size, and hidden size hyperparameters.

Examples for top attended words in the context passage in BiDAF model:

Example 1:
Context: ”sizeable minorities of other faiths do exist (muslim 11.2 % , indigenous beliefs 1.7 %) , and nonreligious 2.4 % . sixty percent of the muslim population lives in kenya ’s coastal region , comprising 50 % of the total population there . roughly 4 % of muslims are ahmadiyya , 8 % shia and another 8 % are non-denominational muslims , while 73 % are sunni . western areas of the coast region are mostly christian . the upper part of kenya ’s eastern region is home to 10 % of the country ’s muslims , where they constitute the majority religious group . in addition , there is a large hindu population in kenya (around 300,000) , who have played a key role in the local economy ; they are mostly of indian origin .”
Question: what religion is the western region mostly ?
Answer: mostly christian
Attended words: origin, christian, hindu, muslim
Explanation: The question asks about a particular religion and we see that most of the top attended words are very relevant to the question and hence we perform really well on this example. Our model predicts it to be Christian.

Example 2:
Context: ”policies of british prime minister benjamin disraeli . it was shortly appropriated by supporters of ” imperialism ” such as joseph chamberlain . for some , imperialism designated a policy of idealism and philanthropy ; others alleged that it was characterized by political self-interest , and a growing number associated it with capitalist greed . liberal john a. hobson and marxist vladimir lenin added a more theoretical macroeconomic connotation to the term . lenin in particular exerted substantial influence over later marxist conceptions of imperialism with his work imperialism , the highest stage of capitalism . in his writings lenin portrayed imperialism as a natural extension of capitalism that arose from need for capitalist economies to constantly expand investment , material resources and manpower in such a way that necessitated colonial expansion . this conception of imperialism as a structural feature of capitalism is echoed by later marxist theoreticians . many theoreticians on the left have followed in emphasizing the structural or systemic character of ” imperialism ” . such writers have expanded the time period associated with the term so that it now designates neither a policy”
Question: what was the idealized value of imperialism ?
Attended words: of, term
Explanation: In this example none of the top attended words are relevant to the question and we see that that in this case our model does indeed perform badly for this example and we get F1/EM score of 0.