CodaLab ID , Yutong Li : yutong2 ,Prateek Murgai : pmurgai

Implementation and New Variants Exploration of the Multi-Perspective Context Matching Deep Neural Network Model for Machine Comprehension

Yutong Li
Apple Inc.
yutong2@stanford.edu
lyt.pami2005@gmail.com

Prateek Murgai
Stanford University
pmurgai@stanford.edu

Abstract

This project explores the multi-perspective context matching method for the task of reading comprehension using the SQuAD data set. The original six layer model presents an interesting system for exploring deep learning architectures and their implementations on Tensorflow.The first step was to design an efficient implementation of this complex model on Tensorflow. The second step, and the aim of this project, was to devise new model variants to potentially improve the model quality and provide helpful insights to researchers.

The main variants that were experimented with were: bi-directional GRU's instead of bi-directional LSTM's, learning rate annealing, effect of gradient clipping, Tanimoto coefficient for the filter layer and model ensembles, hyper-parameter and search and regularization. Some interesting insights are gained when implementing these variants and the results and observations are described in the paper.

1 Background Work and Introduction

The task of developing machine comprehension question answering systems are compelling tasks in the field of natural language processing. The advent of state of the art deep learning architectures and the introduction of the SQuAD dataset has provided a platform for developing deep learning architectures using a big dataset like SQuad and giving rise to promising systems. One of the state of the art systems that is ranked third on the SQuAD leaderboard is the BiDAF (Bi Directional Attention Flow) network [4] that employs character level, word-level and contextual embeddings along with a bi-directional attention flow. This paper provided us a nice explanation of how to obtain a query-aware context representation and how a multi-stage architecture for a question answering system can be developed. This paper was an entry point for us to other architectures and attention methods. Another interesting system was the DCN(dynamic co-attention networks) [2] which proposes a co-attention method that attends to the question and the document simultaneously and finally uses both attention contexts. Such a co-attentive encoder can capture interesting interactions between the question and the document and gave us insights about another interesting attention network. At this point it became quite clear to us that an effective attention network is the heart of a machine comprehension system and would form an integral part of whichever system we plan to design or implement.

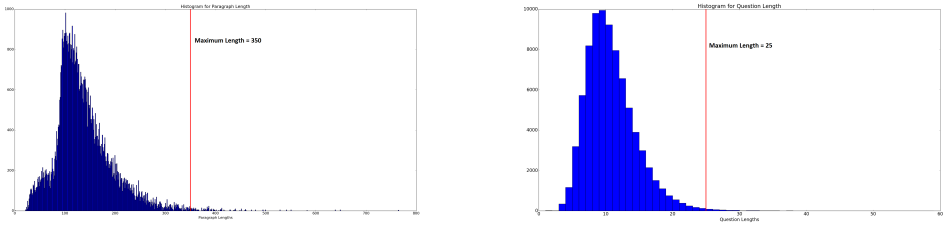
We decided to implement the Multi Perspective Context Matching [1] system which is based on the idea of predicting the answer span by matching the context of each point in the passage with the question from multiple perspectives , the assumption for doing this being that the span in the passage is more likely to be the correct answer if the context of this span is very similar to the question.

The details of the multi-perspective context matching has been briefly explained in the next section.

Along with this implementation we wanted to experiment with the algorithm and explore techniques that can aid in improving this system. Some of these experiments include using learning rate annealing, using GRU's instead of LSTM's, model ensembles, redesigning the filter layer etc. The learnings and findings of our experimentation is described in detail in Section 4.

2 Dataset Exploration: Question and Paragraph Length Distribution

We thought exploring the distribution of the question and the paragraph lengths is an important aspect for this project as it could greatly affect the maximum length parameter (for both question and context) that can affect the run time of the model itself.



(a) Paragraph Histogram (b) Question Histogram

Figure 1: Question and Paragraph Histograms

Based on the histograms shown in Figure.1 we select the **maximum sequence length for the question to be 25** and **the maximum sequence length for the paragraph to be 350**.

3 Multi-Perspective Context Matching Approach

Here we briefly explain the algorithm, the system architecture and each of the layers according to our understanding.

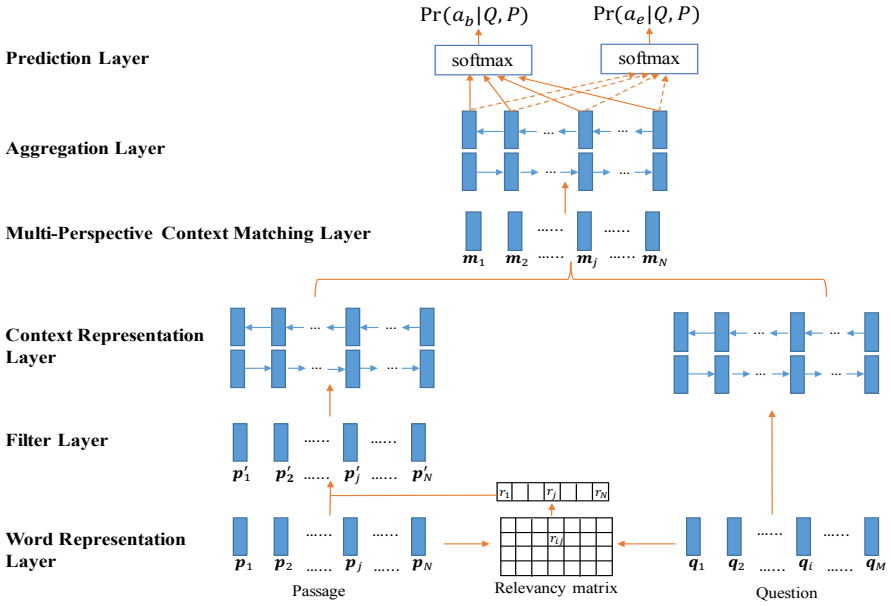


Figure 2: System Architecture

Figure.2 shows the architecture of the system and the different layers employed for this machine comprehension task. Now we will briefly explain each layer

Word Representation Layer. The output of this layer is word vector sequences for question Q : $[\boldsymbol{q}_1, ..., \boldsymbol{q}_M]$, and passage P : $[\boldsymbol{p}_1, ..., \boldsymbol{p}_N]$. Where M is the maximum sequence length defined for the question (25 in our system) and N is the maximum sequence length for the paragraph (350 for our system)

Filter Layer. This layer filters out redundant information from the passage. A relevancy degree r_j is estimated for each word $\boldsymbol{p}_j$ in passage P. First we compute the relevancy degree $r_{i,j}$ between each word pair $\boldsymbol{q}_i \in Q$ and $\boldsymbol{p}_j \in P$ by calculating the cosine similarity $r_{i,j} = \frac{\boldsymbol{q}_i^T \boldsymbol{p}_j}{\|\boldsymbol{q}_i\| \cdot \|\boldsymbol{p}_j\|}$, and get the relevancy degree by $r_j = \max_{i \in M} r_{i,j}$. Then each word is filtered by $\boldsymbol{p}'_j = r_j \cdot \boldsymbol{p}_j$, and pass $\boldsymbol{p}'_j$ to the next layer.

Proposed Change : Instead of using cosine similarity for estimating the relevancy matrix we employ the Tanimoto coefficients for calculating the similarity.

Below is the new formula for calculating the new relevancy matrix :

$r_{i,j} = \frac{\boldsymbol{q}_i^T \boldsymbol{p}_j}{\|\boldsymbol{q}_i\|^2 + |\boldsymbol{p}_j\|^2 - \boldsymbol{q}_i^T \boldsymbol{p}_j}$

There are two main reasons for using such a coefficient for the filter(attention) layer :

- It gives an attention boost to the question, paragraph pair that have more similarity by subtracting the intersection of the two in the denominator.
- It pushes down the attention of the question, paragraph pair that have less similarity.

Context Representation Layer. To incorporate contextual information into the each time step in the passage and the question bi-directional LSTM (BiLSTM) are used to encode contextual embeddings for each question word.

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_i^q &= \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{i-1}^q, \boldsymbol{q}_i) \qquad i = 1, ..., M \\ \overleftarrow{\boldsymbol{h}}_i^q &= \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{i+1}^q, \boldsymbol{q}_i) \qquad i = M, ..., 1 \end{aligned} \tag{1}$$

Same is done with the passage,

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}_j^p &= \overrightarrow{\text{LSTM}}(\overrightarrow{\boldsymbol{h}}_{j-1}^p, \boldsymbol{p}'_j) \qquad j = 1, ..., N \\ \overleftarrow{\boldsymbol{h}}_j^p &= \overleftarrow{\text{LSTM}}(\overleftarrow{\boldsymbol{h}}_{j+1}^p, \boldsymbol{p}'_j) \qquad j = N, ..., 1 \end{aligned} \tag{2}$$

Multi-Perspective Context Matching Layer. In this layer each contextual embedding of the passage is compared with the question with three perspectives. The three perspectives are as follows :

First, dimensional weighted matchings with

$$\boldsymbol{m} = f_m(\boldsymbol{v}_1, \boldsymbol{v}_2; \boldsymbol{W}) \tag{3}$$

where $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ are two d-dimensional vectors, $\boldsymbol{W} \in \Re^{l \times d}$ is a trainable parameter, l is the number of perspectives, and the returned value $\boldsymbol{m}$ is a l-dimensional vector $\boldsymbol{m} = [m_1, ..., m_k, ..., m_l]$. Each element $m_k \in \boldsymbol{m}$ is a matching value from the k-th perspective, and it is calculated by the cosine similarity between two weighted vectors

$$m_k = cosine(W_k \circ \boldsymbol{v}_1, W_k \circ \boldsymbol{v}_2) \tag{4}$$

where $\circ$ is the elementwise multiplication, and W_k is the k-th row of $\boldsymbol{W}$, which controls the k-th perspective and assigns different weights to different dimensions of the d-dimensional space.

Second, on the orthogonal direction of f_m, we define three matching strategies to compare each contextual embedding of the passage with the question:

(1) Full-Matching: each forward (or backward) contextual embedding of the passage is compared with the forward (or backward) representation of the entire question.

$$\begin{aligned}\overrightarrow{\boldsymbol{m}}_j^{full} &= f_m(\overrightarrow{\boldsymbol{h}}_j^p, \overrightarrow{\boldsymbol{h}}_M^q; \boldsymbol{W}^1) \\ \overleftarrow{\boldsymbol{m}}_j^{full} &= f_m(\overleftarrow{\boldsymbol{h}}_j^p, \overleftarrow{\boldsymbol{h}}_1^q; \boldsymbol{W}^2)\end{aligned} \tag{5}$$

(2) Maxpooling-Matching: each forward (or backward) contextual embedding of the passage is compared with every forward (or backward) contextual embeddings of the question, and only the maximum value is retained.

$$\begin{aligned}\overrightarrow{\boldsymbol{m}}_j^{max} &= \max_{i \in (1 \ldots M)} f_m(\overrightarrow{\boldsymbol{h}}_j^p, \overrightarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^3) \\ \overleftarrow{\boldsymbol{m}}_j^{max} &= \max_{i \in (1 \ldots M)} f_m(\overleftarrow{\boldsymbol{h}}_j^p, \overleftarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^4)\end{aligned} \tag{6}$$

(3) Meanpooling-Matching: This is similar to the Maxpooling-Matching, but we replace the $\max$ operation with the $mean$ operation.

$$\begin{aligned}\overrightarrow{\boldsymbol{m}}_j^{mean} &= \frac{1}{M} \sum_{i=1}^{M} f_m(\overrightarrow{\boldsymbol{h}}_j^p, \overrightarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^5) \\ \overleftarrow{\boldsymbol{m}}_j^{mean} &= \frac{1}{M} \sum_{i=1}^{M} f_m(\overleftarrow{\boldsymbol{h}}_j^p, \overleftarrow{\boldsymbol{h}}_i^q; \boldsymbol{W}^6)\end{aligned} \tag{7}$$

Thus, the matching vector for each position of the passage is the concatenation of all the matching vectors $\boldsymbol{m}_j = [\overrightarrow{\boldsymbol{m}}_j^{full}; \overleftarrow{\boldsymbol{m}}_j^{full}; \overrightarrow{\boldsymbol{m}}_j^{max}; \overleftarrow{\boldsymbol{m}}_j^{max}; \overrightarrow{\boldsymbol{m}}_j^{mean}; \overleftarrow{\boldsymbol{m}}_j^{mean}]$.

Aggregation Layer.

This layer is employed to aggregate the matching vectors, so that each time step of the passages can interact with its surrounding positions. We incorporate the matching vectors with a BiLSTM, and generate the aggregation vector for each time step.

Prediction Layer. Two different feed-forward neural networks are employed to calculate the two span probabilities $\Pr(a_{\mathrm{e}}|Q, P)$ and $\Pr(a_{\mathrm{b}}|Q, P)$

We feed the aggregation vector of each time step into the feed-forward neural network individually, calculate a value for each time step, then apply softmax non-linearity across the entire passage.

4 Two Big Folds : Our Learning and Findings

4.1 Performance Wise Findings and Tuning(Memory and Speed):

4.1.1 Three implementations of the same algorithm: The story of 22 hours to 14 to 2.4 hours

The core context matching layer takes the most time for model training. We rewrote our algorithm three times by different analysis methods for improving the training speed. Below is the core findings from this analysis.

1. Since the context matching layer which includes **fullmatching**, **maxpooling**, **meanpooling** requires many high dimensional tensor computations, in the first version, we used significant amount of tiling operations in our implementation to make sure we use vectorized computation as much as we can. However, one epoch run took **twenty two hours** because large tensor computation requires a lot of memory and we could only take **batch size = 4** on GPU.

2. Looking at this the memory bottleneck, we rewrote the algorithm by employing the following two criteria:
   (a) Replace **'tensor tiling'** wherever possible with **'unfolded batch-wise time step pack'**.
   (b) Maximize the reuse of all the variables, such as the crossing variables shared by max pooling and mean pooling and full matching.

   This way, loops are added and tiling operations are reduced. We found GPU favors this mode because it accepts larger **batch size(=6)** to run models. We reduced our model training time to 14 hours per epoch, 30 percent speed up compared with the initial version mentioned above. However, interestingly, we observed it takes much longer time for the same job to run on CPU machine.
3. 14 hours was still a big number, it stops us from verifying experiments and further tuning. This led us to revisiting the code and think further. Instead of using 'unfolded batch-wise time step pack' we rewrote significant part of the code to maximize the broadcasting and unbelievably, we reduced the epoch running time **from 14 hours to 2.4 hours**.

4.1.2 Would Replacing BiLSTM With GRU Improve the Training Time.

We were able to achieve about 22 percent parameters reduction by using GRU compared with BiLSTM, however we did not see an obvious running time reduction with GRU. Also, we could not o increase the batch size by the 22 percent parameters saving, since we ran into out of memory once we increased the batch size.

4.1.3 Disable The Global Norm Computation and Logging

Logging norm is very helpful at the initial stage of model training but once we are confident about the model correctness and hyper-parameters, to disable the norm calculation and logging will be a wise choice. In our experiments, we see a visible gain for the training efficiency(20-40 percent) after disabling norm computation and logging.

4.1.4 Would Retraining Embeddings Badly Increase The Training Time

Our experiments show the answer is no. Because training time is not a function of a single factor.

1. **Memory Wise** : Embeddings as trainable variables actually does not affect too much on memory. Thus, it will not affect much on the batch size.
2. **Computation Wise**: Certainly on the other hand, retrainable embeddings adds backpropagation computation for each minibatch in general, however in our experiment the GPU has been capable enough to make up the extra computation cost while not comprising the training speed in general.

4.2 Quality Wise Findings And Tuning

4.2.1 First 100 Samples Not Enough for Selecting Best Model

In our analysis we found that selecting the best model based on the F1 and EM for the first 100 samples is not the right way of selecting models (as suggested on Piazza many times). Models that had a higher F1 and EM on the first 100 samples gave a lower F1 and EM on the dev set as compared to the models that had a lower F1 and EM for the first 100 samples.

Solution : Random sampling instead of picking up the first 100 samples does the job and is a better predictor of the best model. We back this solution up in the results section.

4.2.2 Fix loss out of bound issue

Since we are using cross entropy with sparse logits for loss calculation, we ran into out of bound exception and 'nan' problem. we then filtered the answer span for both 'val' and 'train' set where index is out of bound of the predefined max length of context paragraph. it's critical for computing the loss for both training set and validation set.

4.2.3 Loss Reduction Efficiency Discussion

1. **With and Without Some Layers** : For instance, we've observed without meanpooling the loss reduction will become worse than with meanpooling. Our practical results for the first epoch run with lr=0.01 and no gradient clip showed meanpooling can benefit the loss reduction to a lower number 8.30, than 9.10, the case without meanpooling. (the numbers provided are only based on our experimental settings.
2. **The first epoch is important** :The first epoch determines the first landing position for the model in its computational space. with good hyper-parameters, the loss should go down very fast; also, the following epoch training will also benefit greatly from this starting point. Figure.3 from the CS231n class wiki shows plots for different types of learning rates

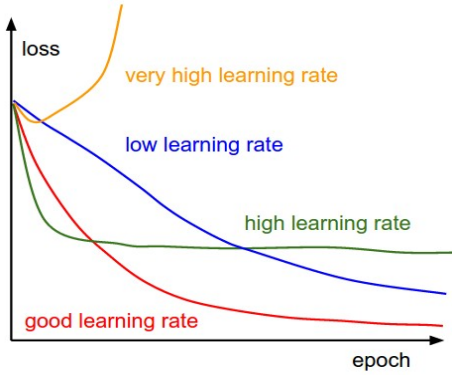


Figure 3: Loss curves with different learning rates

In our case, we had our initial epoch trained with a good learning rate 0.01 and no gradient clip. we continued the 2nd epoch training based on this epoch, and ran into two struggles:

   (a) Keeping the same hyper-parameters: The model either had slow loss reduction or a lot of fluctuation or even increased loss.
   (b) Using smaller learning rate and enabling gradient clip rate: It only eliminated the increased loss, the slow loss reduction and fluctuation still exist.

3. **Hyper-parameters** : Learning rate, gradient clip rate and dropout all three are important.
   (a) Reducing the learning rate by 0.5-0.8 every half epoch aided us in getting a healthy learning curve. Without such annealing we faced either loss saturation or loss increase.
   (b) Gradient clipping of 3 to 5 also yields a good learning curve as compared to a higher clipping value or no clipping which led to loss saturation or too much loss fluctuation.
   (c) Increasing dropout from 0.15 to 0.6 gave a boost in the EM of 10 percent (from 40 to 50 percent) on the validation set (100 samples selected randomly)
4. **GRU v.s. LSTM** : GRU saves memory, however in experiments we have observed multiple times that LSTM usually reduces loss faster with running the same amount of batches than GRU does.

5 RESULTS

5.1 Embedding Visualization

It is interesting to view the embeddings before and after training to see how these transform from the initial epoch to the final epoch. Figure.2 shows embeddings before training and after four epochs of training. The embeddings have been reduced from 100 dimensions to 3 dimensions using the T-SNE method.

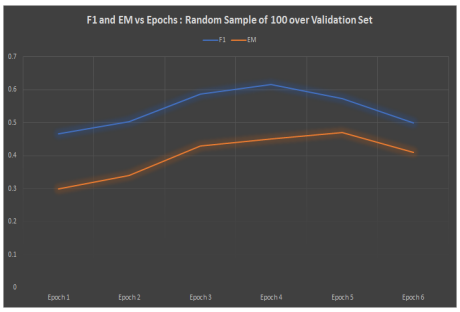
Figure 4: Embeddings before training and after four epochs of training

5.2 Performance Plots

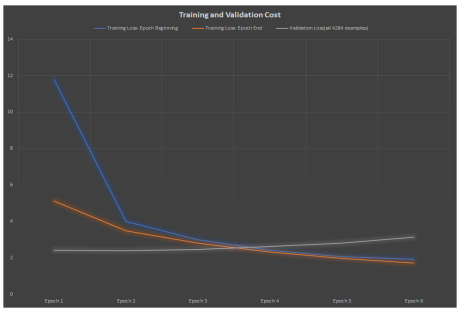
Figure 5. displays some performance metrics changing over epochs. We can see from 5 (a) that the epoch 4 has the highest F1 and EM and the thus the model from this epoch is selected as our best model. The F1 and EM that we achieved for the dev set using this model are **F1 : 48.62559 , EM : 35.846736** .

We also find that the validation cost doesn't reflect the performance of the model in this case as you can see it has an increasing trend. One possible reason is that this validation cost is dependent on the minibatch size as well as the initialization of some of the hidden states in the BiLSTM's.

From the training loss we can observe that this complex model learns too much within the first epoch and after that the learning saturates over epochs though there is decrease in loss.



(a) F1 and EM for the best run



(b) Train Loss and Validation Loss for the best run

Figure 5: Performance Plots

5.3 F1 for the First 100 samples is not a good indicator of the best model

In the section we want to present some analysis we did to prove that selecting the best model based on the F1 for the first 100 samples of the validation set is not the right way to select the best model. Instead randomly selecting 100 samples from the validation set and calculating F1 and EM for those samples is the correct way.

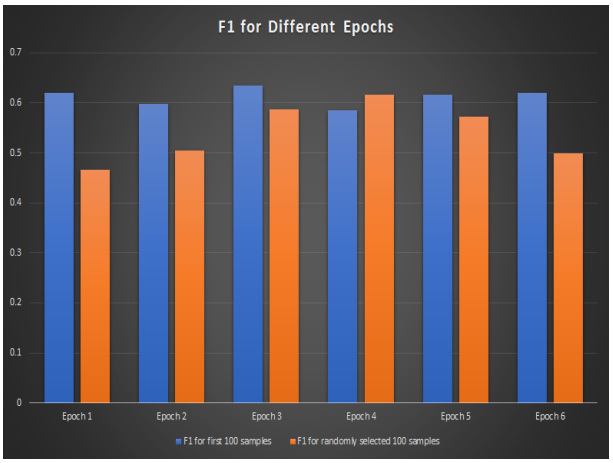


Figure 6: F1 calculated using First 100 samples and Randomly Selected Samples from Validation Set

Figure.6 is the F1 calculated during training over epochs using the two methods. The best model according to the first method (Selecting model based on the first 100 samples) is **Epoch 3 model** whereas best model according to the second method(Selecting model based on random sampling) is **Epoch 4** model. Below are the dev set F1 and EM calculated for both the models:

Random Sampling Method (Epoch 4 Model) : F1 : 48.625 , EM :35.846

First 100 Sample Method (Epoch 3 Model) : F1 : 44.441 , EM :27.277

Clearly the best model selected using the random sampling method yields a higher F1 and EM on the dev set. This trend is seen in almost all the runs and model versions we have tried.Thus we have come to the conclusion that random sampling on validation set to select the best model is a better predictor of the best model.

5.4 Prediction Analysis

In this section we present examples where our model failed where it passed and the reasons why it failed/passed and possible solutions.

1. **Unknown prediction(Failure)**
   ”id”: ”5737a5931c456719005744eb” : ”question”: ”What is responsible for speeding up or slowing down an object?”
   Answer: tangential
   Predicted : $< unk >$ forces
   Possible Reasons: On investigation we found that there is no word ’tangential’ in the word embeddings thus we believe the model couldn’t predict the answer.
   Solution: Character based learning in the word representation layer and a better pretrained embeddings are possible solutions.
2. **Completely Wrong Answer(Failure)**
   ”id”: ”5733ea04d058e614000b6598” : ”question”: What native chief travelled to French fort and threatened Marin?
   Answer : Tanaghrisson **Predicted**: ”5733ea04d058e614000b6598”: ”garrisoned forts . He first constructed Fort $< unk >$ Isle (near present-day Erie , Pennsylvania) on Lake Erie ’s south shore . He had a road built to the headwaters of LeBoeuf Creek . $< unk >$ constructed a second fort at Fort Le $< unk >$ (present-day Waterford , Pennsylvania) , designed to guard the headwaters of LeBoeuf Creek . As he moved south , he drove off or captured British traders , alarming both the British and the Iroquois . $< unk >$, a chief of the $< unk >$, who were remnants of Iroquois and other tribes who had been driven west by colonial expansion . He intensely disliked the French (whom he accused of killing and

eating his father) . Traveling to Fort Le $< unk >$, he threatened the French with military action , which $< unk >$ contemptuously dismissed",

Possible Reasons:

(a) The context is too complicated even for the humans to understand. It is too rich in information.

(b) The actual answer is a $< unk >$ in the context, so it could not predict the actual answer.

Possible Solution : Providing POS tags as a feature will definitely aid in predicting answers with such a complicated context.

3. **Empty Answers(Failure)** "id": "57378e311c456719005744b2", "question": "How many vector equations did Heaviside and Gibbs reformilate Maxwell's 20 scalar eqution into?"
Answer : 4
Predicted Answer : ""
Possible Reason : Predicted answer end index is less than the predicted answer start index
Possible Solution: Force the end answer index to be greater than the start answer index in the model so that such predictions are not made.
4. **Wrong Prediction due to closeness of words(Failure)**
"id": "57287c2bff5b5019007da26c","question": "What process attributes new wealth to those that already have it?"
Answer: Wealth concentration
Predicted : beneficiaries
Possible Reasons : If one reads the context related to this question, you will find that the word 'new wealth' appears only once and also next to the word "beneficiaries" thus we are assuming the model could never predict any other answer.
Possible Solution: As the model doesn't learn well in this case, we believe a better attention mechanism in the filter layer would solve the issue.
5. **Model Passed**: Our model predicted the correct answers in cases where the answers were a part of the pretrained embeddings and where the answers spans were short (1-5).

6 Conclusion and Future Work

1. Our exploration and research clearly displays the importance of learning rate annealing, gradient clipping and dropout for such a machine comprehension system.
2. From the Tensorflow implementation perspective we learnt how different tensor operations affect the training time and memory usage on a GPU.
3. **Future Work**:
   (a) Currently we are running the model with the Tanimoto Coefficient for the Filter Layer and are using models from each epoch to create an ensemble.
   (b) In addition to dropout we will add l2 regularization to prevent over-fitting
   (c) We would want to increase the output size (currrently only 200) and re-adjust the maximum length of the paragraph and the question.
   (d) We are also planning to implement another attention mechanism similar to one in [3].

References

[1] Zhiguo Wang, Haito Mi, Wael Hamza, and Radu Florian. Multi-Perspective Context Matching for Machine Comprehension. arXiv: 1612.04211, 2016

[2] Caiming Xiong, Victor Zhong, and Richard Socher. Dynamic coattention networks for question answering. arXiv:1611.01604, 2016

[3] Minh-Thang Luong, Hieu Pham, Christopher D.Manning.Effective Approaches to Attention-based Neural Machine Translation. Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, pages 14121421, Lisbon, Portugal, 17-21 September 2015. c 2015 Association for Computational Linguistics.

[4]Bidirectional Attention Flow for Machine Comprehension.arXiv:1611.01603