Robust Answer Selection

Benjamin Geyer Aditya Vetukuri Phillip Pascal Amish Papneja

George Mason University

Abstract

Deep Learning has been very successful for solving NLP related problems recently because of its power to incorporate information from long sequential data. However, the Question Answering domain in NLP is still an active research area. Answer selection is one of the related tasks which requires a model to rank a set of candidate answers from best to worst for a given question. Commercial applications like Siri, Alexa, and IBM Watson implement this task using information retrieval-based methods which are strongly dependent on knowledge bases to answer user queries. Neural network based QA systems can significantly improve the performance by encoding text into vector representations. In this project, we evaluate various extensions on previous state-of-the-art implementations of answer selection and machine reading comprehension.

1. Introduction

1.1 The Problem

The problem of question answering can be divided into various subfields including answer selection and machine reading comprehension. The task of answer selection can be generally formulated as follows: given a question q and an answer candidate pool {a1, a2, · · · , an} select the best answer candidate. The goal of Machine Reading Comprehension (MRC) is to provide an answer to a question based on context given in the form of a paragraph. Examples of the different compositions can be seen in Figure 1 and Figure 2 showing examples of answer selection and MRC respectively.

Question	Answer
What does the Peugeot company manufacture?	One company, Yeu Tyan Machinery Manufacturing, assembles both Daihatsu and Peugeot vehicles.

Figure 1: General format example for Answer Selection (TREC-QA)

Context	Question	Answer
In early 2012, NFL Commissioner Roger Goodell stated that the league planned to...	Who was the NFL Commissioner in early 2012?	Roger Goodell

Figure 2: General format example for Machine Reading Comprehension (SQuAD 1.1)

1.2 Challenges

Given that these subproblems are very different, the techniques and datasets utilized and the challenges facing each are unique. The general goal of question answering systems is to determine the semantic meaning of each question and answer and compare them to find the answer that fits the best. However, the challenge with question answering is that the question and answer can be different lexically but encode the same semantic meaning. This is especially relevant with long sequences of text where the question and answer lengths are very imbalanced.

The task of answer selection can be applied to many different scenarios where the form of the answer can vary greatly. Some questions are looking for short, specific answers whereas other questions are more open-ended with the answers being quite long. The issue with these long answers is that determining the semantic meaning of a piece of text can become harder as the length increases. This is due to noise and a large amount of unrelated information included in the answer to give details to the user. This was the case in the InsuranceQA dataset where the questions were open ended such as “My mother needs health insurance. How does she get that?”

InsuranceQA also has a very unique vocabulary which means that when encoding the question and answer text using a pretrained word embedding, there is often no match in the pretrained vocabulary resulting in an empty embedding value for that word. Additionally, certain words in the domain of insurance can have different semantic meaning that is not encoded in a pretrained word embedding. This can result in the meaning of the embedding of a word not matching the context that it’s being used in.

Since natural language is highly flexible, a sequence of words that you are looking for might not show up word-for-word in the passage which is a challenge in MRC tasks. Another challenge is to build a model which is able to distinguish the portions of a text that are relevant to the query and focus on those portions. It is challenging to learn contextual representations of objects in each passage due to the long length.

SQuAD 2.0 includes numerous questions which are unanswerable given a context. It was a challenge to build a model that could identify those and only answer a question if the confidence level is high. Due to this, an MRC model not only needs to answer questions when possible, but also identify when there are no correct answers and abstain from answering.

1.3 Approach

For answer selection, we propose a variety of extensions on previous works including concatenating word embeddings, learning a new word embedding based on the InsuranceQA dataset, performing data augmentation to increase the number of examples in the TREC-QA dataset, and applying a transformer architecture.

We experimented on two versions of the SQuAD dataset for the MRC task. First, we implemented a seq2seq model as the baseline with a single attention layer. This was followed by an extension which included an attention flow mechanism alongside a Bidirectional LSTM for SQuAD 1.1. We experimented using learned embeddings and various pretrained embeddings such as glove, word2vec etc. For SQuAD 2.0 we implemented and fine tuned the Distil-BERT model in order to better classify unanswerable questions.

2. Related Work

Previous answer selection research such as Tan et al. generally don't investigate the usage of word embeddings, simply opting to use only one pretrained method such as word2vec. The work of Tan et al. looks at various deep learning models while utilizing only word2vec, possibly missing out on some meaning that could be provided by other embeddings or perhaps multiple embeddings being utilized at once. Multiple embeddings have been shown to be effective in other problem domains as those highlighted by Zhang et al.

There have been numerous approaches to deal with MRC tasks and attention mechanisms are a significant part of many of them. Researchers have experimented with different strategies to implement attention mechanisms, one approach is to update the attention weights dynamically given the query and the context as well as the previous attention. Furthermore, Min et al. implemented an attention flow model in which the attention vector is allowed to flow through the subsequent layers. We chose to re-implement this particular approach. Also, we can incorporate hops of attention layers to the model as proposed by Weston et al.

Many research papers have used BERT with some architecture changes for MRC tasks based on previous success of BERT in the NLP domain, eg: Ramnath et al and Yuwen Zhang et al. Most of the state of the art MRC tasks based on the SQuAD dataset have leveraged BERT and have achieved close to human-level performance as said by Ramnath et al.

3. Datasets

3.1 Answer Selection

Answer Selection focuses on choosing the correct sentence that contains the exact answer and can sufficiently support the answer choice. Selection is based on relative rankings of answers using a cosine similarity evaluation metric.

3.1.1 InsuranceQA

InsuranceQA is the first dataset which comes from the insurance domain consisting of questions from real world users and high quality answers composed by professionals. The dataset contains 17,487 questions in total having one or more answers with the number of answers totaling 24,981. Every question has an associated pool of potential answers (100-1000 per question). The data is already tokenized, and comes with a premade vocabulary. Data is then formatted in the manner of an identifier for the question/answer number followed by a set of idx_# to indicate words from the vocab. This means that we have to rebuild the sentences in order to create word embeddings. Beyond this to preprocess the data the sentences would also have to be rebuilt and then the vocabulary would need to be updated.

The metrics that will be used are common ranking metrics of top-1 accuracy and mean reciprocal rank (MRR). Top-1 accuracy is the idea of checking if the model's highest ranked answer is a correct answer for the question, and then taking the total number of correctly answered questions divided by the total number of questions. MRR on the other hand follows the formula:

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}.$$

Where Q is some sample of queries.

3.1.2 TREC-QA

This dataset comes from Text REtrieval Conference's QA track which was the first large-scale evaluation of domain-independent question answering. TREC-QA focuses on questions which are more popular culture and news based using participants in a study to answer the questions. TREC-QA consists of questions, answers, and binary labels. Not all questions have positively labeled answers in the dataset. Therefore there are actually two different leaderboards tracking the raw version (the one with all questions) and the clean version (tracking only questions which contain some positively labeled answers) of TREC-QA. For this paper's purposes only the cleaned version of TREC-QA was examined. This set contains 65 questions in dev, 68 in test, and 1,229 in the training set.

Again the metrics used for TREC-QA will be top-1 accuracy and MRR. These were chosen due to their common usage in ranking problems seen in the literature.

3.2 Machine reading comprehension

Machine Reading Comprehension (MRC) scans context or documents and tries to extract meaning from the text, just like a human reader. MRC is an active research area due its potential usage in various enterprise applications. The tasks of Machine Reading Comprehension and Question Answering have gained significant popularity over the past few years. Reading Comprehension (RC) or the ability to read a context and then answer the questions based on the context is a challenging task for machines. It requires both the underlying knowledge of the language and some general knowledge about the world itself. Consider the question "What is an off-side in a football game?" In order to answer the question one might first locate and read a passage about soccer rules, reason the exact context which is specific to the question from the passage and determine the correct answer.

3.2.1 SQUAD 1.0

SQUAD 1.0 is a popular MRC dataset, the dataset consists of questions presented by crowd workers on a set of Wikipedia articles, where for every question you have an answer which is a segment of text or a span, from the corresponding reading passage. It consists of 100,000+ question-answer pairs on 500+ articles.

3.2.2 SQUAD 2.0

Squad 2.0 took the 100,000 question-answer pairs from the previous SQUAD 1.0 version and added 50,000+ unanswerable questions.

Metric : The metrics that are used are F-1 score and EM score. F-1 score is the harmonic mean of precision and recall. Precision is the number of true positive results divided by the number of all positive results. Whereas recall is the number of true positive results and the number of results which are expected to be positive. EM is nothing but an exact match. It measures the percentage of predictions that match any of the ground truth results exactly.

Formula

$$F_1 = 2 \cdot \frac{\mathrm{precision} \cdot \mathrm{recall}}{\mathrm{precision} + \mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \frac{1}{2}(\mathrm{FP} + \mathrm{FN})}$$

4. Methods

4.1 Answer Selection

4.1.1 Multiple embeddings with QA LSTM/CNN

At the input, instead of using a single embedding multiple embeddings have been shown to be effective in some cases. A single pre trained word embedding can sometimes miss context that other embeddings are better able to represent. This can occur when the dataset the embeddings were trained on is lexically dissimilar to the dataset the embedding is being utilized on. By combining multiple embeddings trained on different datasets, we can hopefully mitigate this issue. Some common manners in which to use multiple embeddings include concatenation or retrofitting. In concatenation each embedding is created independently from common pretrained or learned methods such as word2vec or glove, and then these embeddings are combined together. This will effectively increase the embeddings size by a factor of the number of embeddings being concatenated.

4.1.2 Bi-LSTM

Recurrent Neural Networks (RNNs) are a common approach to solving sequence problems with

variable-length input due to their hidden vector dependence on the previous hidden vector. This allows information to be passed along when analyzing future tokens in an input. LSTMs are an extension of RNNs that help solve the vanishing gradient problem by having more control over the hidden information in the form of input, forget, and output gates that augment the cell memory vector. These allow the network to preserve long-term dependencies in sequences without the information being lost to the effect of more recent tokens. The bidirectional extension on this is allowing information to flow in both directions along the sequence by having an LSTM for each direction and combining their future and past dependent vectors to get an output for each token. These Bi-LSTMs have been shown to have great success in many different sequence learning problems that benefit from long-term as well as short term context.

4.1.3 CNN/Bi-LSTM

Another model architecture used in Tan et al. is CNNs which are applied to the outputs of the Bi-LSTM layers. CNNs slide and apply a learned filter of a certain size along the Bi-LSTM outputs and generate single output values for each application of the filter. Then a max-k pooling layer is applied to those filter outputs to condense the outputs to an even denser representation of k outputs (typically k=1). There are multiple CNN filters called channels being applied to the Bi-LSTM outputs concurrently with different filters being learned to extract and recognize different lexical patterns from the questions and answers.

4.1.4 Triplet Loss

The model was trained using a triplet loss function with a question, known correct answer (positive) for the question, and known incorrect answer (negative). If there is no labeled incorrect answer for the question, then an answer to a different question is randomly sampled from the dataset.

The model computes the cosine similarities between the question and positive answer as well as between the question and the negative answer. The triplet loss gives the difference between these similarities with a baseline margin (typically m<=0.2). The goal of the model is to maximize these differences over time. This means that even if a correct answer is considered similar to the question, the model must learn to recognize that the negative answer is incorrect.

The formula for triplet loss is as follows:

$L(r_q, r_p, r_n) = \max(0, m + d(r_q, r_p) - d(r_q, r_n)$

Where r_q represents the question, r_p is the positive answer, r_n the negative answer, and m is the margin.

During inference, the model outputs the cosine similarity between each candidate answer and the question. The best answers are selected by sorting by the cosine similarities.

The overall structure of our model can be viewed in Figure 3 below.

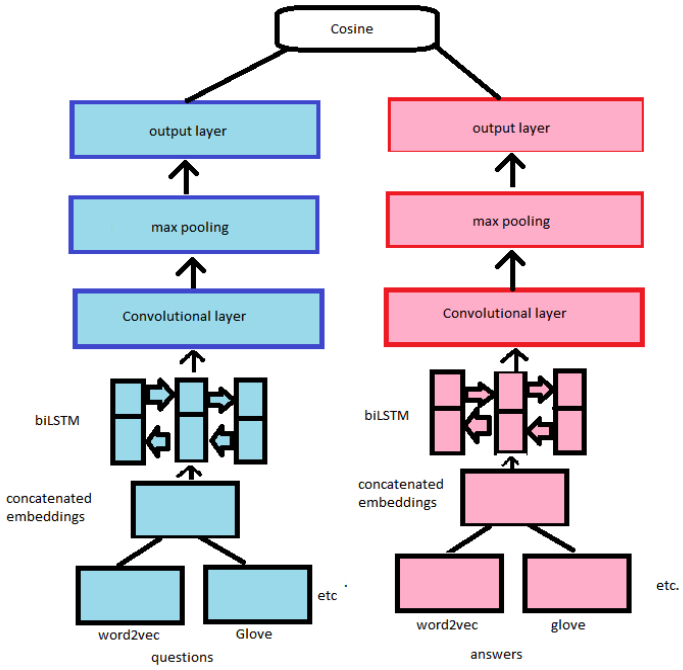

Figure 3: Bi-LSTM/CNN with multiple embeddings

4.2 Machine Reading Comprehension

We have two separate models for both SQUAD 1.0 and SQUAD 2.0. However we explored SQUAD 1.0 with various model architectures. The best performing model turned out to be the Bi-Directional LSTM with Attention Flow. For SQUAD 2.0, we used the Distil-BERT pretrained model for Question Answering and fine-tuned it on SQUAD 2.0.

4.2.1 MRC on Answerable Questions

Seq2Seq Bi-LSTM with Attention

The structure of the model is a network of two encoders and a decoder, all implemented in Bi-Directional LSTMs.

At first, the question Q and paragraph P are encoded in two independent Bi-LSTMs which produce corresponding hidden states at each word position as H_p H_q. Then we send the encoded question and paragraph matrices H_q and H_P are put into another encoder with sequence to sequence attention. For each hidden state vector we calculate the scores. Then the score matrix of each question over the whole paragraph is multiplied by H_P. Now we send the product to another layer of LSTM to generate the weight of context H_C for a specific question. We concatenate H_c with H_p into a state matrix to have focus on the parts which are important rather than the parts which are not. The final step is to feed matrix into an encoder which consists of two bidirectional lstm which gives their respective output vectors and we used softmax activation and cross-entropy loss to get the outputs.

Bidirectional Attention Flow

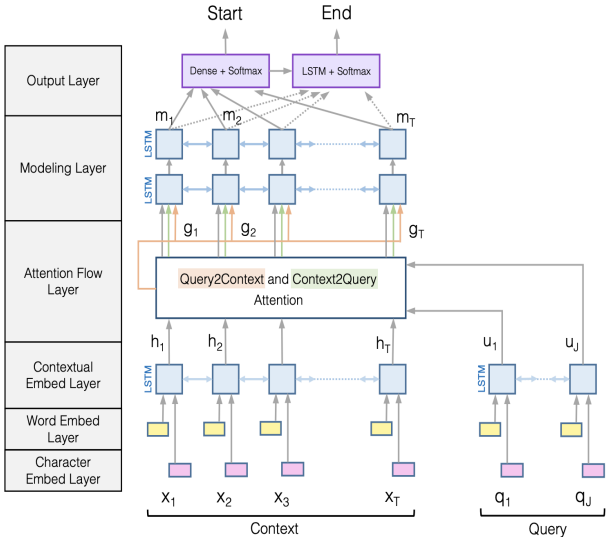

Figure 4: BIDAF diagram (Minjoon Seo et al.)

We then experimented on SQUAD 1.0 re-implementing Minjoon Seo et al. The model has several stages and consists of six layers. Each word is mapped to a vector space using a character-level CNN and each word is embedded to a vector space using Word2Vec. We then concatenate both the embeddings and they are sent to the contextual embedding layer. This layer utilizes contextual clues from the paragraph and fine tunes the embeddings. Then we pass on the embedding matrix to an attention flow layer which couples the question and paragraph vectors and produces a set of query-aware feature vectors for each word in the paragraph. This feature vector is passed to the network along with the embedding matrix and is present at each time step and is allowed to flow through to the subsequent modeling layer. The modeling layer is a Recurrent Neural Network which scans the context. The loss function which is the sum of negative probabilities of the true and end indices averaged over all examples.
We apply the softmax activation function over the probabilities and we get the resultant answer.

4.2.2 MRC on Unanswerable Questions

For the SQUAD 2.0, we used the DistilBERT model from Huggingface transformers for QuestionAnswering as the baseline and evaluated its performance on SQUAD 2.0. Later, we fined-tuned the model on SQUAD using transfer-learning. We experimented the model on different learned and pretrained embeddings. In order to know which words contribute more to the context, we used Self Attention on top of BERT.

5 . Experiments

5.1 Answer Selection

Combining word embeddings
Our main experiment for Answer Selection was the implementation of combining multiple word embeddings via concatenation. The embeddings we chose to investigate were word2vec, fasttext, and glove.

Learned vs Pretrained Embeddings
For InsuranceQA, we experiment with training our own word embeddings to compare against the pretrained embeddings. We performed this experiment using word2vec and fasttext embeddings as glove had a different embedding format.

Data augmentation for TREC-QA
One issue with TREC-QA is that it only has ~1000 training questions (82 test questions) which is a very small amount compared to the ~30,000 that InsuranceQA has. Each question only has a few corresponding answers for the model to learn from. This is an issue for the model because it can't learn the true meaning of a question with only a few examples. The solution to this is using data

augmentation to create more training examples for each question. Additionally, having more test questions allows our results to be more robust to noise. Data augmentation works by slightly modifying a piece of text using a variety of methods including back-translation, synonym replacement, random insertion, random swap, and random deletion of characters and words. We used the NLPAug to perform all of these except back-translation on TREC-QA. For this experiment, we simply used the model architecture that we found performed best in our other experiments.

BERT for InsuranceQA/TREC-QA

We also attempted to implement a BERT model for use on InsuranceQA and TREC-QA even though there has been relatively little research into the application of BERT to these datasets. Due to this lack of knowledge, time constraints, and the difficulty of modifying the data format of both of the datasets, we were not able to complete our implementation for InsuranceQA/TREC-QA.

5.2 MRC

Concatenation of Word Embeddings and Character Embeddings

We concatenated the vector space of word embeddings and character level embeddings. We experimented on word2Vec, fasttext and glove.

Learned and Pretrained Embeddings

We experimented and evaluated our baseline learned embeddings with pre trained embeddings like glove, fasttext and Word2Vec.

Attention Flow

Instead of summarizing the query and context into single feature vectors. We passed the attention vector to flow through the following modelling layer.

Fine-tune and transfer learning on BERT.

We used the DistilBERT model from hugging-face transformers and fine-tuned it on our task for SQUAD.

Optimizing BERT with task specific layers

After fine tuning BERT on SQUAD, we decided to add task specific layers on top of BERT but due to time constraint we failed to obtain any results.

6. Results

6.1 InsuranceQA

Initially for InsuranceQA we attempted to utilize various learned embeddings using the methods of word2vec and fasttext as well as their concatenation. After doing this, pretrained embeddings were tested with the results being mediocre. These experiments can be seen in Figure 5.

Learned Embeddings	Top-1	MRR	Time per epoch(s)
Word2Vec	0.522	0.63783	47
Fasttext	0.518	0.63405	48
Word + Fast	0.531	0.64547	60
Pre-trained Embeddings			
Word2Vec	0.431	0.56605	47
Fasttext	0.406	0.53606	48
Glove	0.461	0.57951	49
Word + Fast	0.439	0.57613	60

Figure 5: InsuranceQA results

As noted above, the learned embeddings did better and the concatenation of both did the best while increasing runtimes by only ~20%. However, the data preprocessing could be improved. This preprocessing issue likely explains the difference in results between our research and Tan et al. In their work, a Bi-LSTM/CNN model with pretrained word2vec was able to achieve a top 1 score of 64.6 on the test set, though they did not record their MRR scores for us to compare against. Thankfully, this issue with the preprocessing doesn't carry over to TREC-QA which had much better results.

As demonstrated in Figure 6, the model appears to learn most of the features of the data within the first 10 epochs with further incremental gains before terminating early. This termination is due to early stopping and the score not improving from the best value at around epoch 20 (a patience of 20 was set). This particular run was an outlier as most others reached around 60 epochs before early stopping.

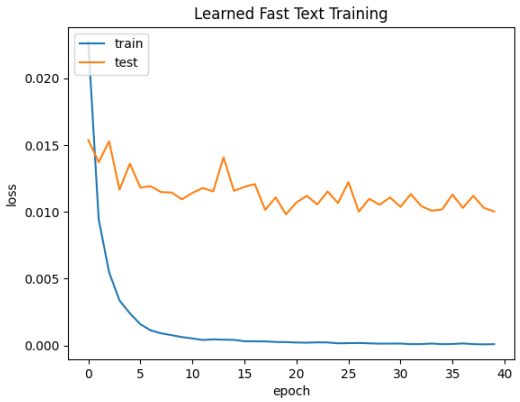

Figure 6: learned fast text word embedding with our model. The blue line is loss and the orange is validation loss.

6.2 TREC-QA

For TREC-QA, pretrained 300 dimensional embeddings were utilized. Again glove, word2vec, and fasttext were chosen for our experiments. The model was trained with embeddings from each individual embedding, each concatenation of two embeddings, and the concatenation of all 3. These results can be seen below in Figure 7. Our best results initially came from glove and word2vec concatenated; with the added method of data augmentation, we were able to achieve slightly better results. These results beat Tan et al's implementation using a similar model of Bi-LSTM/CNN in which they achieved 81.04 MRR while we achieved 81.67 MRR. Though it is of note that our base model with only word2vec appears to do slightly worse than Tan et al's with results of 0.7911 MRR compared to 81.04 MRR as Tan et al.

Embeddings	Top-1	MRR
Glove	0.67684	0.76793
Word2vec	0.71579	0.79115
Fasttext	0.70566	0.78545
Fast + Glove	0.68421	0.77476
Fast + Word	0.62105	0.74133
Glove + Word	0.73684	0.80505
Fast + Glove + Word	0.69474	0.77162
Augmentation (Glove + Word)	0.74284	81.6703

Figure 7: TREC-QA results

6.3 SQUAD

Concatenating the word embeddings and character embeddings made a difference and gave us better results. We did various experiments on pre-trained word embeddings and concatenated them with our Character embeddings and here are some of those results in Figure 8.

Model	Embeddings	Attention	F-1	EM
Seq2Seq	Learned	Single Layer	0.56132	0.48732
Seq2Seq	Word2Vec	Single Layer	0.57491	0.50122
BIDAF Model	Learned	Attention Flow	0.70463	0.64733
BIDAFModel	Word2Vec	Attention Flow	0.72431	0.65872
BIDAF Model	Word2Vec + Character	Attention Flow	0.76623	0.69238
BIDAF Model	**Fasttext + Character**	**Attention Flow**	0.77237	0.71621
BIDAF Model	Glove + Character	Attention Flow	0.76212	0.70774

Figure 8 : SQUAD 1.0 Results

For SQUAD 2.0 we used the DistilBERT model from hugging-face and fine-tuned it on the SQUAD 2.0 dataset. We experimented the model with few pre-trained embeddings and we also used SelfAttention which gave us the best results in all the experiments that we have done. Please refer to Figure 9.

Model	EM SCORE	F-1 SCORE
DistilBERT Uncase Pretrained	0.626994	0.685472
DistilBert Uncase with Fasttext	0.616882	0.679457
DistilBert FineTuned on SQUAD	0.684562	0.742515
BERTFineTuned + SelfAttention	0.716712	0.778293
BIDAF (Ours)	0.566781	0.6234512

Figure 9 : SQUAD 2.0 RESULTS

7. Conclusion

Unfortunately, the results shown achieved on InsuranceQA were not close to state-of-the-art. We were unable to match the baseline results from Tan et al., but our proposed extensions did improve our baseline implementation. The issue likely arose from our difficulty with preprocessing the unconventionally formatted data in InsuranceQA. Given more time, better results likely could have been achieved. It is of note that the model is unlikely to be the issue because, for the TREC-QA dataset, we used the same model and had no issues.

We have shown that our proposed improvements have achieved some results for TREC-QA that are better than those achieved with the same models in Tan et al. However, the best combination of methods has yet to be shown as Tan et al. had the best performance with an Bi-LSTM/CNN model that

included attention. We can theorize that combining attention with our extensions of multiple embeddings and data augmentation would achieve even better results.

In relation to MRC, we were unable to achieve results that matched Manjoon Tan et al., but we experimented with various approaches to improve our performance on the SQUAD dataset. Attention Flow greatly helps the model determine the context of a piece of text in the paragraph. The results we achieved for SQUAD 1.0 with BIDAF + Fasttext matched the 54th position in the SQUAD leaderboard. Previous researchers have experimented with various other learning techniques such as dying gradients, slow learning, teacher forcing and have achieved better results. To further improve our performance on SQUAD 1.0, we should, in the future, incorporate some of these approaches.

For SQUAD 2.0, we were unable to implement a calibrator to abstain the model for unanswerable questions. We were only able to use the pretrained BERT and experiment on it with various embeddings and Attention. Using SelfAttention and DistilBert Embeddings, we were able to achieve an EM score of 71% which is 76th position in SQUAD 2.0 leaderboard Given more time, we could promise better results with some data augmentation techniques and some task specific attention layers on top of BERT.

7. References

Changchang Zeng, Shaobo Li, Qin Li, Jie Hu, Jianjun Hu. 2020. A Survey on Machine Reading Comprehension: Tasks, Evaluation Metrics and Benchmark Datasets

Ellen M. Voorhees, Dawn M. Tice. 2001. The TREC question answering track. In The Text REtrieval Conference.

Ming Tan, Cicero dos Santos, Bing Xiang & Bowen Zhou. 2016. LSTM-Based Deep Learning Models for Non-Factoid Answer Selection. In conference paper at ICLR.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, Hanneh Hajishirzi. BI-Directional Attention Flow For Machine Comprehension. At ICLR 2017.

Minwei Feng, Bing Xiang, Michael R. Glass, Lidan Wang, Bowen Zhou. Applying Deep Learning to Answer Selection: A Study and An Open Task. In ASRU 2015.

Pranav Rajpurkar, Robin Jia, Percy Liang. Unanswerable Questions for SQuAD. [cs.CL] 11 June 2018.

Sahana Ramnath, Preksha Nema, Deep Sahni, Mitesh M. Khapra. n.d. Towards Interpreting BERT for Reading Comprehension Based QA

Wenqi Hou, Yun Nie. Seq2Seq Question Answering Attention Model. N.d.

Ye Zhang Stephen Roller Byron C. Wallace. 2016. MGNC-CNN: A Simple Approach to Exploiting Multiple Word Embeddings for Sentence Classification. In Proceedings of NAACL-HLT.

Yuwen Zhang, Zhaozhuo Xu. BERT for Question Answering on SquAD 2.0. 2019

Contributions

Code:
insuranceQA, TREC-QA and all answer selection related work was primarily done by phil and ben working in tandem.

All the work related to SQUAD 1.0 was done by Aditya, and Amish contributed to all the work related to SQUAD 2.0

Paper:
Abstract- all 4 worked on
Section 1- first paragraph all 4, overall & answer selection challenges ben, machine reading comprehension challenges amish/ben.
Section 1.3- aditya/amish
Section 2- first paragraph phil, second paragraph Amish
Section 3.1- phil/ben
Section 3.2- Aditya
Section 4.1- phil/ben
Section 4.2- Aditya
Section 5.1- phil/ben
Section 5.2- Aditya
Section 6.1, 6.2- phil/ben
Section 6.3- Aditya
Section 7- first two paragraphs phil/ben, rest are Aditya