Question Answering System on SQuAD Dataset

Dharna Shukla [*1] **Shubham Nishad** [*1]

Abstract

Machine comprehension (MC) is one of the difficult problems in natural language processing(NLP) because answering a question about a given context paragraph requires modeling complicated interactions between the context and the question. Before SQuAD dataset, Machine Comprehension (MC) datasets were either too small to train deep learning models, or not complex enough to evaluate the ability of current Machine comprehension techniques. SQuAD dataset diminishes these drawbacks and gives a chance to develop more efficient models. The recent introduction of attention mechanisms in Machine comprehension has vastly improved the performance of the models. In this study, we discuss different types of deep neural architectures with attention mechanisms and compare their performance based on EM and F1 score.

1. Introduction

Machine comprehension of the text is difficult and next to the unattainable problem in natural language processing for a decade. The prime objective of the NLP is for machines to be able to understand and process texts as accurately as humans. In the Machine Comprehension task, the model reads a piece of text and attempts to answer a question based on given a text. Many research has been dedicated to this problem, for example, information extraction, relation extraction, semantic role labeling, and recognizing textual entailment over the years but the outcome of these study have not been very successful due to lack of well-structured datasets and machines unsuitable to support heavy neural networks. Over the years as the computation power of machines increased tremendously, neural approaches gained prominence in the field of natural language processing. The field machine comprehension took an extraordinary turn when SQuAD Dataset was developed by Pranav Rajpurkar and team in the year 2016 (Rajpurkar et al., 2016). SQuAD offers a large number of real questions and answers created through crowdsourcing from human-curated data. The SQuAD provides a demanding test environment for measuring the performance of machine comprehension algorithms because compared with previous datasets, answers in SQuAD cannot be derived from a fixed set of candidate answers.

SQuAD actively maintains a public leaderboard where researchers share their neural architectures attaining the state of the art results on the dataset. The latest deep networks have been able to surpass human performance based on EM and F1 scores. Analyzing the leaderboard shows a common set of architectures behind the competitive results. In our study, we plan to examine the architecture behind these models and provide a comparative view of the models based on factors such as evaluation metrics, training time, complexity, etc.

2. Related Work

The public leaderboard on the SQuAD website displays many deep learning models built for this dataset. Since the initial paper by (Rajpurkar et al., 2016), many researchers have come up with different architectures. (Seo et al., 2016) introduced the Bi-Directional Attention Flow (BiDAF) network, a multi-stage hierarchical process that represents the context at different levels of granularity and uses bidirectional attention flow mechanism to obtain a query-aware context representation without early summarization. BiDAF dominates most of the research in the machine comprehension field. A recent trend of using pre-trained models has been effective in many natural language tasks. Bidirectional Encoder Representations from Transformers (BERT) is one of the most successful architecture for this purpose. (Devlin et al., 2018) use the pre-trained BERT model and fine-tune it with just one additional output layer to achieve the state of the art results in the machine comprehension task. (Yu et al., 2018) proposed a new architecture called QANet which exclusively consists of convolution and self-attention in place of recurrent neural networks, this leads to 13x faster training while achieving equivalent performance to recurrent models. With the different types of networks available for machine comprehension ranging from RNN to transformers, in our work, we plan to train and evaluate some of these deep

[1]EECS, University of Ottawa, Ottawa, Canada. Mail to: Dharna <dshuk066@uottawa.ca>, Shubham <snish035@uottawa.ca>.

learning models and compare their performance on different factors.

3. Methodology

The reading comprehension task considered in this study, is defined as follows. Given a context paragraph with n words L = {l1, l2, ..., ln} and the query sentence with m words G = {g1, g2, ..., gm}, output a span S = {li, li+1, ..., li+j} from the original paragraph C.

3.1. Machine Comprehension Architecture

There has been a common architecture for neural machine comprehension, almost all the models applied on the SQuAD dataset have an embedding layer, followed by encoding layer, context-query attention, modeling layer, and output layers.

- **Embedding Layer :** Given some input word indices {w1, w2, ..., wk}, the embedding layer performs an embedding lookup to convert the indices into word embeddings {v1, v2, ..., vk}. This is done for both the context and the question, producing embeddings {c1, c2, ..., cN} for the context and {q1, q2, ..., qM} for the question.

- **Encoder Layer :** The encoder layer takes the embedding layer's output as input and uses a recurrent neural net to allow the model to incorporate temporal dependencies between timesteps of the embedding layer's output. The encoded output is the RNN's hidden state at each position.

- **Attention Layer :** Attention mechanisms such as Dot Product Attention or Context-Query Attention are applied to the encoder output so that the decoder can identify important words in the context.

- **Modelling Layer :** The modeling layer refines attention output. Since we use the modeling layer after the attention layer, the context representation already has question information. Thus the modeling layer is used to establish the temporal relationship between context and question. Similar to the Encoder layer, we use a recurrent neural network.

- **Output Layer :** The output layer is responsible for producing the *pstart* and *pend*, which represent the answer span start probability and end probability respectively. Output layer takes as input the outputs of the attention layer and the modelling layer.

3.2. Relevant Neural Models

In this section we describe the different architectures that we selected for this study.

3.2.1. Baseline: Context to Question Attention

The baseline model implements the five layers of the basic architecture for Machine comprehension. First, the context and question are converted to vector form using pre-trained glove embedding. These vectors are then applied to the bidirectional encoder layer. The encoder layer consists of Bidirectional Gated Recurrent Units (GRUs) that allows the model to learn the temporal relation between the question and the context. We concatenate the output for both the context and the question before applying context to question attention. Here we calculate the attention distribution of the context attending to the question. Attention matrix is calculated by applying softmax over the product of context embeddings and the transpose of question embeddings, then this matrix is concatenated with the original context embedding. The modeling layer downsizes the attention output and this output is used to generate the start span and end span individually using the softmax activation function. In our study we use the model implemented by (Zia, 2019), this model replaces the RNN networks with BiLSTM for better performance.

3.2.2. BiLSTM architecture using BiDAF attention

Bidirectional LSTM (BiLSTM) networks are used for the encoder and modeling layer. The main strength of this model is the Bidirectional Attention Flow (BiDAF) mechanism. This model uses learned character-level word embeddings in addition to the word-level embeddings. The model consists of 5 layers, Embedding Layer, Encoder Layer, Attention Layer, Modeling Layer, and Output Layer respectively as mentioned in section 3.1. Fig 1. shows layers of model implementing BiDirectional Attention Flow.

The main part of the BiDAF model is the bidirectional attention flow layer. The prime concept is that attention should flow both directions, from the context to the question and from the question to the context.

Let us assume context hidden states $(c_i, ..., c_N) \epsilon R^{2H}$ and question hidden states $(q_1, ..., q_M) \epsilon R^{2H}$. Similarity Matrix can be calculated as $S \epsilon R^{NXM}$, which contains similarity score $S_{i,j}$ for each pair (c_i , q_j) of context and question hidden states. We could deduce $S_{i,j} = w_T^{sim}[c_i, q_j; c_i \circ q_j] \epsilon R$ .Since the similarity matrix S contains information for both the question and context, authors (CS224n, 2019) have used it to normalize across either the row or the column in order to attend to the question or context,respectively. First Authors have performed Question-to-Context(Q2C)

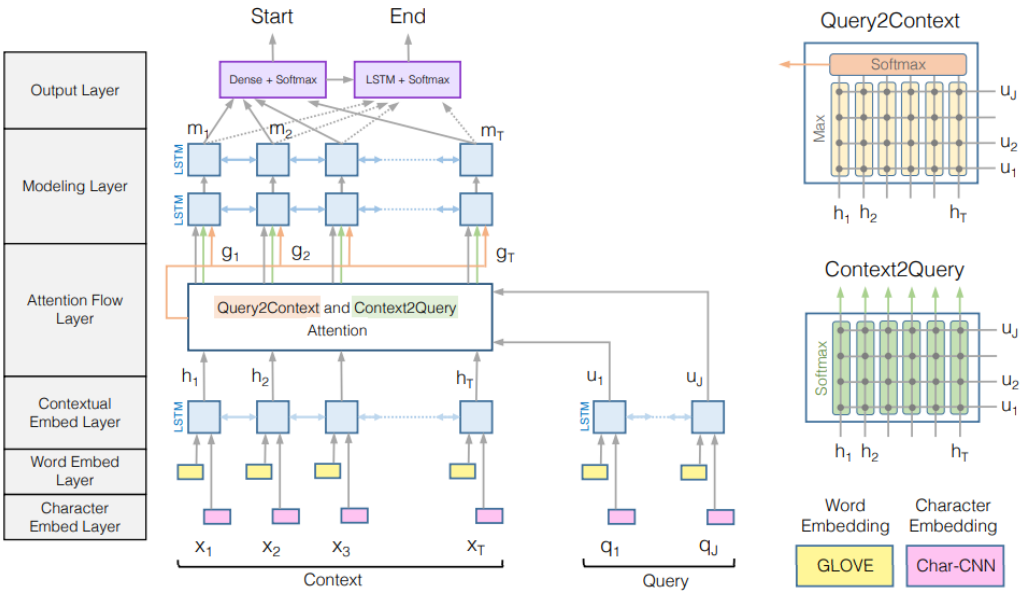


Figure 1. BiDirectional Attention Flow Model. Img src : (Seo et al., 2016)

Attention.In this authors have taken the softmax of the columns of S to get $\bar{\bar{S}} \epsilon R^{NXM}$, where each column is an attention distribution over context words. Then it was multiplied with $\bar{S}$ into $\bar{\bar{S}}$ and used the result to take weighted sums of the hidden states $c_j$to get the Q2C attention output:

$$\bar{\bar{S}} = softmax(\bar{S} :, j \epsilon R^N) \forall_j \epsilon (1, ...M)$$
$$S' = \bar{S}\bar{\bar{S}} \epsilon R^{NXN}$$
$$b_i = \sum_{j=1}^{N} S'_{i,j} c_j \epsilon R^{2H} \forall_i \epsilon (1, ...N)$$

Lastly, for each context location $i \epsilon 1, ..., N$ they obtain the output g_i of the bidirectional attention flow layer by combining the context hidden state c_i, the C2Q attention output a_i, and the Q2C attention output b_i:

$$g_i = [c_i; a_i; c_i \circ a_i; c_i \circ b_i] \epsilon R^{8H} \forall_i \epsilon (1, ...N)$$

where $\circ$ represents elementwise multiplication. In our study we train the BiDAF implementations provided by (Dwivedi, 2018) and (Lee, 2019).

3.2.3. QANET BASED ENCODER MODEL

Based on the QANET paper by (Yu et al., 2018), this is an RNN-free, attention-based model that achieves a high score on the SQuAD leaderboard. This model trains 4.5x times faster compared to the comparable RNN model. The authors of the model take inspiration from transformers and develop component called encoder block. Similar to the transformer, the encoder block uses positional encoding, residual connections, layer normalization, self-attention sublayers, and feed-forward sublayers. Fig 2. shows the architecture of the Encoder model.

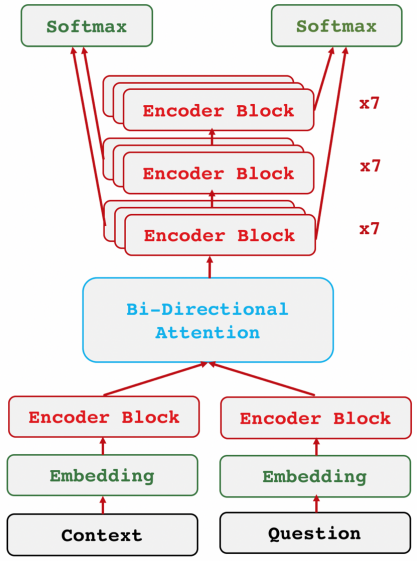


Figure 2. Five-layer Encoder block model inspired from QANet. Image source

The embedding layer uses Glove 300-dim embeddings to convert the context and query words into 200-dim character embeddings which in turn are used to make 200-dim character-level word embeddings. The encoding layer consists of a single encoder block to which we apply downsized

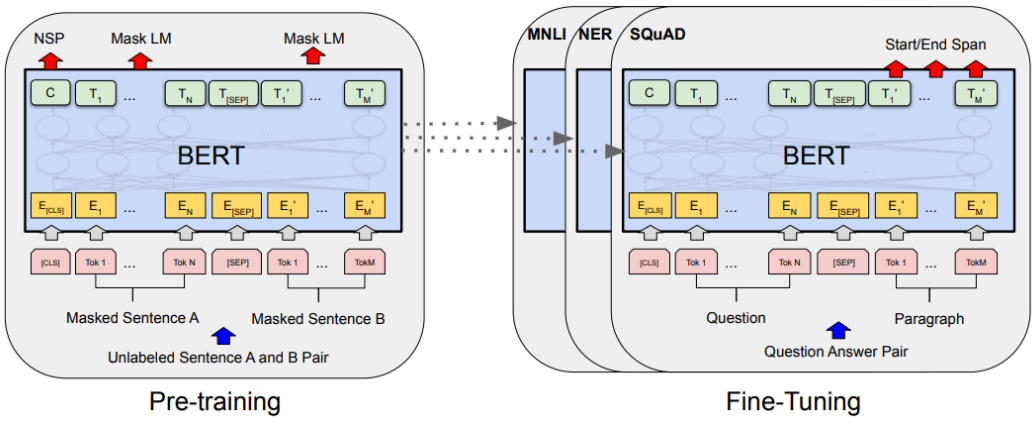


Figure 3. The pre-training and fine-tuning procedures for BERT. Only the output layer differs between both the procedures, rest of the architecture is same. Parameters from the pre-training phase are used to initialize models for fine tuning. [CLS] is a special symbol added in front of every input example, and [SEP] is a special separator token (e.g. separating questions/answers). Img src: (Devlin et al., 2018)

word embeddings of 128-dim. Self-attention is implemented with multi-head, scaled dot-product attention. The BiDAF layer follows the standard implementation of the context-to-query (C2Q) and query-to-context (Q2C) layers. The modeling layer uses an encoder block similar to the encoder layer. Single Encoder block is applied three times to produce outputs M1, M2, M3. The output layer uses the softmax function. The start *p_start* is predicted using [M1, M2] while the end *p_end* is predicted using [M1, M3]. The predicted answer is the span (i,j) where *p_start(i)* * *p_end(j)* is maximised. In our study, we use the encoder model implemented by (Chute, 2018).

3.2.4. BERT

Bidirectional Encoder Representations from Transformers (BERT) is designed to pre-train deep bidirectional representations from the unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering (Devlin et al., 2018). According to fig 3. BERT has two phases, the pre-training phase, and the fine-tuning phase. BERT's model architecture is a multi-layer bidirectional Transformer encoder where we pass the question and the context as a single padded sequence during the pre-training phase. The start and the end vectors are only applied during the fine-tuning phase. The probability of a word being the start or end of the answer span is computed using dot product attention followed by softmax function over all the words in the paragraph. The score of a candidate span (i,j) is defined as S·Ti + E·Tj, where S is the start word of answer, Ti is the hidden vector for i; similarly for E is end word of answer and Tj is a hidden vector for j. Finally, the maximum scoring span where j is greater or equal to than i is used as a prediction. The training objective of this model is the sum of log-likelihoods of the correct start and end positions of the answer. For BERT implementation in our study, we use the pre-trained BERT model provided by the HuggingFace Transformer library (HuggingFace, 2019).

4. Experimental Setup

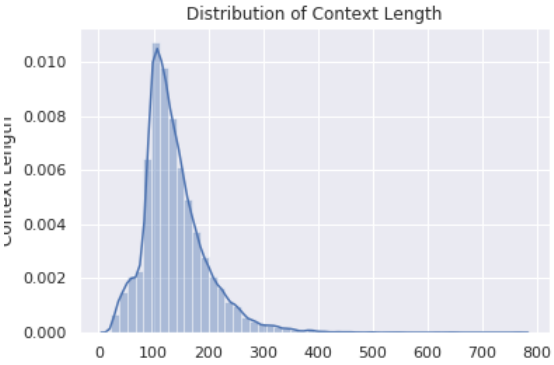
4.1. Dataset

Stanford Question Answering Dataset (SQuAD), is the largest reading comprehension dataset consisting of 100,000+ questions posed by crowd workers on a set of Wikipedia articles, where the answer to each query is a chunk of text from the corresponding reading passage. The SQuAD contains 107,785 question-answer pairs on 536 articles. In contrast to prior available datasets, SQuAD does not provide a list of answer choices for each question. Rather, models must select the answer from all possible spans in the passage, thus needing to cope with a fairly large number of candidates (Rajpurkar et al., 2016). The authors randomly partitioned articles into a training set (80 percent), a development set (10 percent), and a testing set (10 percent). The testing set is not yet publicly available (Li et al., 2018). An example of the dataset is shown below.

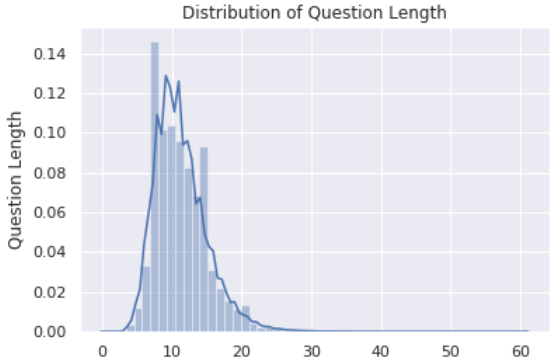
Context Paragraph: *On 24 March 1879, Tesla was returned to Gospic under police guard for not having a residence permit. On 17 April 1879, Milutin Tesla died at the age of 60 after contracting an unspecified illness (although some sources say that he died of a stroke). During that year, Tesla taught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

Question: Why was Tesla returned to Gospic?

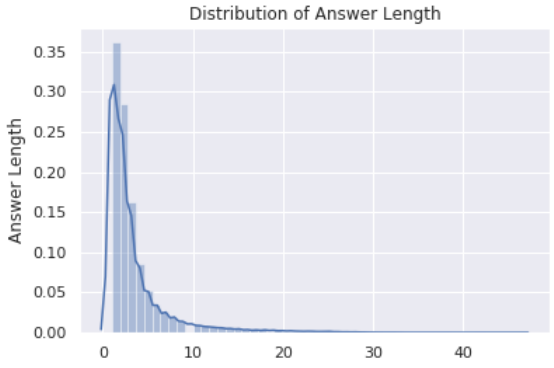
Answer: Not having a residence permit



(a) Context Length distribution on the Train Set



(b) Question Length distribution on the Train Set



(c) Answer Length distribution on the Train Set

Figure 4. Exploratory Data Analysis on SQuAD

Fig 4a. shows the average distribution of context length. We use this to devise the ideal length of context to be given for training. If we select a large value the training time will increase. For our study we select values between 300 to 400 words based on model architecture.

Fig 4b. shows the average distribution of context length. We use this to devise the ideal length of context to be given for training. If we select a large value the training time will increase. For our study we select values between 300 to 400 words based on model architecture.

4.2. Model Parameters

For the Baseline model (Zia, 2019), we use 100 hidden GRU Units with a word embedding dimension of 100. The context length was set to 325 while the question length was set to 22. The batch size is set to 128. The learning rate is 0.00001 while the loss function is categorical-crossentropy. For the first implementation of the BiDAF model (Lee, 2019), we use the hidden unit size of 100 with a word embedding dimension of 300. The context length was set to 400 while the question length was set to 50. The batch size is set to 64. The learning rate is 0.5 while we try to minimize negative log-likelihood. For the second iteration of the BiDAF model (Dwivedi, 2018), we use 150 hidden GRU Units with a word embedding dimension of 100. The context length was set to 300 while the question length was set to 30. The batch size is set to 60. The learning rate is 0.001 while we try to minimize negative log-likelihood. For the encoder based model (Chute, 2018), we use a hidden unit size (d-model) of 128 with a word embedding dimension of 300. The context length was set to 400 while the question length was set to 50. The batch size is set to 32. The learning rate is 0.001 while we try to maximize the F1 score. For the BERT based model (HuggingFace, 2019), word embedding dimension of 300. The context length was set to 384 while the question length was set to 44. The batch size is set to 12. The learning rate is 0.00003 while we try to maximize the F1 score.
The models were trained on an Amazon EC2 P2 instance which had an Nvidia Tesla K80 GPU with 4 vCPUs and 61 GB of RAM. In total, we ran the instance for 80 hours.

4.3. Evaluation Metric

The original SQuAD study recommends two metrics, Exact Match, and F1 score to compare the span level performance of the models with respect to the ground truth answer spans. We used the same metrics to evaluate our models.

- **Exact Match (EM)** is a metric that measures the percentage of predictions that match one of the ground truths answers exactly.If the predicted text answer and the ground-truth text answer are exactly the same, then the EM score is 1, otherwise 0 (Li et al., 2018).

- **F1 Score** is a metric that measures the average overlap between the prediction and ground truth. The F1 score is based on precision and recall. Precision is the percentage of words in the predicted answer existing in the ground-truth answer, whereas the recall is the percentage of words in the ground-truth answer also

appearing in the predicted answer(Li et al., 2018).

5. Results

Table 1. shows the best results we obtained on each model on the SQuAD dev set. BERT performs better than all the other models and almost matches human level F1 scores. All the models were trained with the focus on maximizing the F1-score.

Model	Embedding size	F1	EM
Baseline	100	24.9	23.8
BiDAF	300	60.35	57.55
BiDAF-Best	100	64	48.8
Transformer	300	76.46	65.92
BERT	300	**87.3**	**77.98**
Human Performance		89.45	86.83

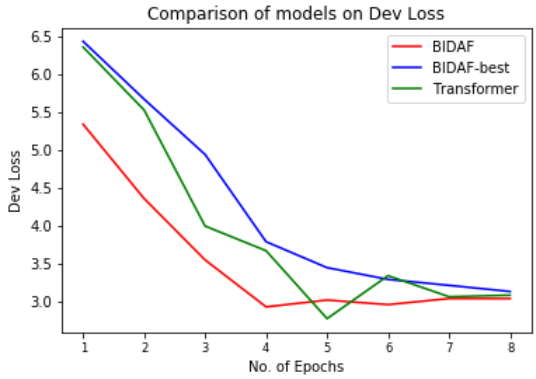
Table 1. Evaluation of Different Models on SQuAD

The Baseline model gets low scores because it only applies dot product attention. This single layer of attention is not able to capture all the temporal relationship between the context and the question. Per epoch time was very high for the baseline model, it took around 3 hours for one epoch.

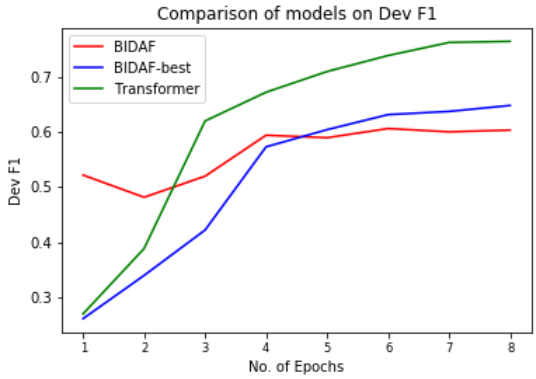
The BiDAF and BiDAF-best models are based on the same principles but differ in their implementation. This leads to a difference in their performance. We can see the performance improvement provided by the Bidirectional nature of BiDAF. When compared to the baseline model we achieve almost 2.5x the scores. The BiDAF-best model had the epoch time of around 1 hour 15 minutes.

The Transformer based Encoder model is able to perform better than BiDAF models because the BiDAF model uses BiLSTM which is not able to capture all the latent information between context and question, while the Transformer is able to perform parallel operations thereby capturing more latent information. The F1 score of 76.6 achieved by the Encoder model is really competitive in the SQuAD leaderboard especially when we consider how lightweight the model is. We found the training time for the Transformer to be 2.5x - 3x times faster than that of BiDAF Models.

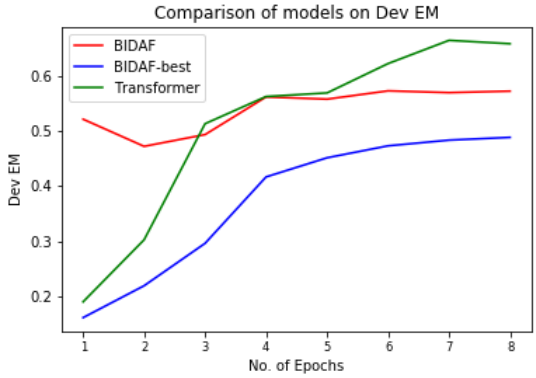
For achieving high scores with BERT we used a pre-trained model fine-tuned on the SQuAD dataset. BERT being a large model with 340 million parameters and 24 layers were not possible to train on our AWS instance. We selected the bert-large-uncased-whole-word-masking-finetuned-squad model from the BertForQuestionAnswering of the HuggingFace library. One of the main reasons a high EM score of 77.98 was possible is because we used the BertTokenizer instead of the standard tokenizer.



(a) Loss on the test (dev) set



(b) F1 score on the test (dev) set



(c) EM score on the test (dev) set

Figure 5. Comparison of Models based on Loss, F1 and EM

When comparing the loss of models during training, from Fig 5a. we find the Transformer model and the BiDAF-best model to follow a similar trajectory for loss while the BiDAF model does not see much decrease in loss after epoch 4. From Fig 5b. and Fig 5c we can infer that the performance of the Transformer model increases rapidly

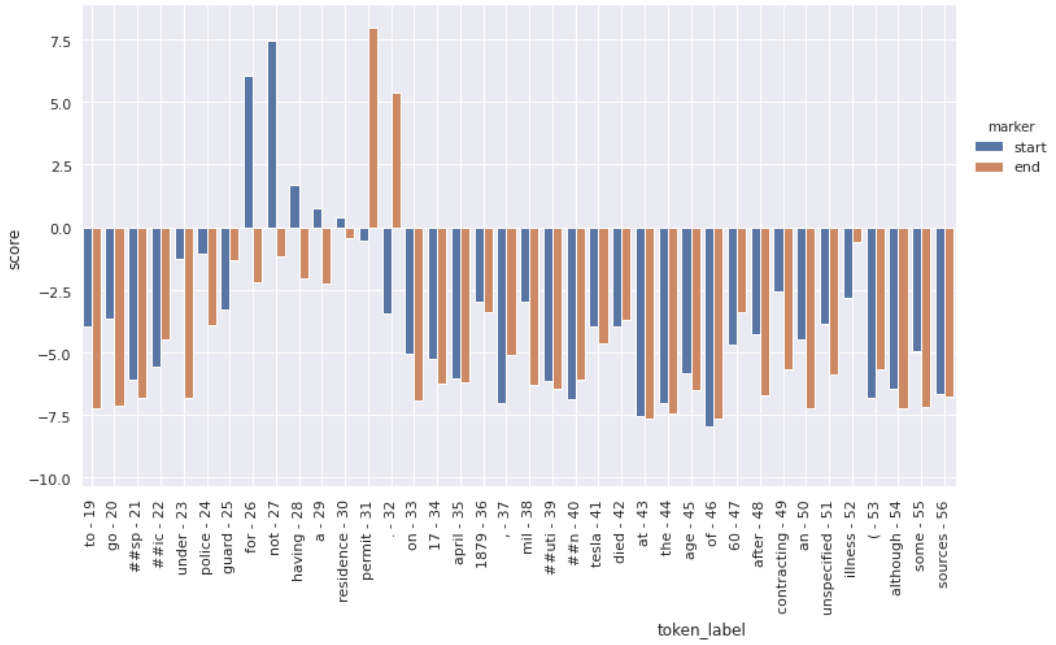


Figure 6. The start span and end span prediction of the Question and Answer mentioned in Sec 4.1

with each iteration whereas the BiDAF model achieves a high peak early but struggles to improve the score. The BiDAF-Best model also continues to improve its score but is not able to match the performance of the Transformer model.

Fig 6. shows the working of attention mechanism in BERT model on the question and answer mentioned in Sec 4.1. The start span receives maximum score at the word 'not' and the end span receives the maximum score at the word 'permit'.

6. Discussion

The F1 and EM scores should be similar to each other, our trained models achieve less EM scores compared to F1 because some answers in the dev dataset contain punctuation's and other special symbols. For the exact match to happen our generated answer should also have the same symbols. This depends a lot on the tokenizer that we use. Fine tuning of tokenizer can help us achieve better EM scores.

Recurrent Neural Networks have been the go to architecture in NLP for a long time, this study showcases the advancement of Transformer and related models. Using transformers we are able to reduce the training time significantly as well as decreasing the complexity of the model.

For all our models we experimented with Glove embeddings of different dimensions, we found that increasing the dimension from 100 to 300 gave little increase in the score but the training time was increased significantly. Using a large corpus gave us good results, the 840B corpus performed way better than the 6B corpus, this is due to the fact that less words would be missing from vocabulary. We experimented with FastText embeddings but were not able to complete the experiment because of time constraint. Following other research studies we believe that changing the embedding would not have resulted in much performance gain.

From the performance of the models we can see that the attention mechanism being used is more important than the underlying architecture. The introduction of self-attention in transformer results in significant increase in performance when compared to BiDAF. The findings of our study point towards the importance of attention in NLP. This has been further verified by (Vaswani et al., 2017).

7. Future Work

In the future, we want to focus more on the transformer-based architectures. We are interested especially in lightweight models like ALBERT, DistilBERT, and FastBert and want to compare their performance with BERT. In this study because of time constraints, we only focused on a single dataset, in the future, we also want to apply multiple questions answering datasets to measure how well the neural networks perform to different questions so as to provide a holistic evaluation on the neural machine comprehension task. It would be great to apply a system similar to AutoML to get the perfect architecture that can perform better than humans on multiple datasets.

8. Conclusion

In this study we successfully trained and tested different types of neural machine comprehension models. We were able to achieve competitive results on the SQuAD dataset. The BERT model achieved score similar to human performance. The different types of models showcase us the direction in which the field of NLP is moving. BERT and its related language models dominate the field. Attention proved to the most significant part of our study. The type of attention mechanism defines the performance you will be getting from your architecture. The usage of RNN free models enabled us to use parallelization and decrease the training time significantly, this is important since some of our models have training time of around 25 hours.

Acknowledgement

We would like to thank Professor Diana Inkpen for providing positive learning environment and developing curiosity about natural language processing among students. We would also like to thank Aftaab Zia, Mina Lee, Priya Dwivedi, Chris Chute and HuggingFace for providing implementation of some popular deep learning models.

References

Chute, C. Attention is all squad needs. https://github.com/chrischute/squad-transformer, May 2018.

CS224n, S. Cs 224n default final project: Question answering on squad 2.0. *Last updated on February*, 28, 2019.

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Dwivedi, P. Optimizing bidaf attnetion. https://github.com/priya-dwivedi/cs224n-Squad-Project, May 2018.

HuggingFace. Fine tuning bert on squad. https://huggingface.co/transformers/examples.html#squad, February 2019.

Lee, M. Basic implementation for bidaf using bilstm. https://github.com/minggg/squad, March 2019.

Li, C.-H., Wu, S.-L., Liu, C.-L., and Lee, H.-y. Spoken squad: A study of mitigating the impact of speech recognition errors on listening comprehension. *arXiv preprint arXiv:1804.00320*, 2018.

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

Seo, M., Kembhavi, A., Farhadi, A., and Hajishirzi, H. Bidirectional attention flow for machine comprehension. *arXiv preprint arXiv:1611.01603*, 2016.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. 2017. URL https://arxiv.org/pdf/1706.03762.pdf.

Yu, A. W., Dohan, D., Luong, M.-T., Zhao, R., Chen, K., Norouzi, M., and Le, Q. V. Qanet: Combining local convolution with global self-attention for reading comprehension. *arXiv preprint arXiv:1804.09541*, 2018.

Zia, A. Context-based-qna-system. https://github.com/Aftaab99/Context-based-QnA-system, December 2019.