SQuAD with SDNet and BERT

Derek A. Huang
huangda@stanford.edu

Taresh K. Sethi
tareshsethi@stanford.edu

Eli J. Pugh
epugh@stanford.edu

Abstract

Many have proposed novel adaptations of BERT [3] to Question Answering, yet most do so by appending an output layer to the transformer network and fine-tuning to produce output spans for answers. SDNet [12], a model designed for CoQA [8], achieved state-of-the-art results by incorporating BERT contextual embeddings as features and computing a weighted sum of BERT layer outputs for each of its tokens, before locking its parameters. For this project, we adapt the SDNet architecture to work on the SQuAD 2.0 dataset. We then tuned parameters and settings to better fit squad, and introduced varied convolutional channels before the recurrent architecture to improve performance.

1 Introduction

As contextualized embeddings from BERT-centered models dominate the leaderboards in Question-Answering, there is a lot of discussion about what the direction of future models will be. Many have hypothesized and experimented with novel ways of fine-tuning BERT towards the Question Answering task by adding a classifier layer on top of BERT's transformer network that outputs start and end spans, as in [4]. Most recognize that large, architectural models like QANet will have trouble performing better than these simpler classifiers on top of BERT. However, quite recently, SDNet achieved state-of-the-art on the CoQA dataset as a large architectural model with lots of self and inter attention sequential mechanisms that uses BERT contextualized embeddings as a feature. We are very intrigued by this approach, especially since BERT has found to be such a difficult model to expand upon and adapt to receive better results. In this paper, we explore utilizing BERT as a feature, and attention mechanisms that take advantage of a more complex feature space.

We also explore another aspect that has received much attention in recent years. There are currently many machine reading comprehension challenges, some of the most popular including the Stanford Question Answering Dataset 2.0 (SQuAD) [7], Conversational Question Answering Dataset (CoQA) [8], and Question Answering in Context (QuAC) [1]. There are many differences in the structure of these datasets, each designed to incite diverse behaviors characteristic of human language. Yet, out of box models that perform well on one challenge are unlikely to perform well on other datasets without quite a bit of adjustments. In our project, we transform the SDNet model [12] that achieved state-of-the-art on the CoQA dataset to the SQuAD 2.0 dataset. We hypothesize that SDNet's complex attention mechanisms and its novel use of BERT as feature embeddings will perform well on the SQuAD 2.0 dataset, but also hope to explore the complexities of transforming a model for another dataset, and touch upon multi-task learning.

2 Related Work

The most popular approaches to reading comprehension tasks before contextual embeddings include FusionNet [6], QANet [11], and BiDAF++ [9]. These networks have different structures, and we use elements from each of them.

FusionNet [6], published in February 2018, used many types of multi-level attention, as well as a new concept, history-of-word. History of word represents both low and high-level semantic meaning to capture as much relevant information about each word as possible. In order to utilize this, they use "fully aware attention" between high level and low level embeddings, as well as bi-directional RNN layers.

QANet [11] was published in April 2018, and is a very impressive model without any recurrent architecture. QANet uses repeated encoder blocks that consist of position encoding, a convolutional layer, a self attention layer, and a fully-connected layer. This is then fed through context-query attention, more encoder blocks, and then softmax classifiers for start/end tokens.

Published in June 2018, Bi-Directional Attention Flow (BiDAF) [9] was the first to use attention from the context onto the question as well as from the question to the context. This directed attention was used with multi-layer LSTM

encoders and decoders, and the featurization included both GLOVE and Char-CNN embeddings. BiDAF achieved state of the art when published, and is still highly regarded as one of the best models not using contextual embeddings.
Our approach incorporates many ideas from FusionNet and BiDAF, including bi-directional RNN encoders and decoders with self-attention and bi-directional attention flow. We also took inspiration from QANet's use of convolutional layers. In order to build upon these techniques, our model also used bidirectional encoder representations from transformers (BERT) [3].
A very new and revolutionary embedding method, BERT uses unsupervised training and bidirectional encoding in order to generate three types of embeddings, position, token, and segment. These are contextual and depend on other words in context. It obtains industry-leading results on eleven NLP benchmarks, notably pushing the GLUE benchmark to 80.4% and SQuAD benchmarks to 84.292 accuracy and 86.967 F1. Almost every result on the SQuAD 2.0 leaderboard contains BERT in its description.
BERT works by first pre-training the model to correct words in the corpus that have been randomly replaced, therefore learning about words from their surrounding context on each side. The second pre-training step is next sentence prediction. Given input sentences, BERT is trained to predict whether input 2 immediately follows input 1 in context. This allows BERT to learn about relationships between sentences, which is very useful in many NLP tasks, including question answering. By using unsupervised methods to learn context, BERT is able to create embeddings that preserve more accurate information than static word vectors. This is an important achievement, since most words in many major languages have multiple meanings.

3 Approach

3.1 SDNet: Contextualized Attention-Based Deep Network for Conversational Question Answering [12]

The canonical method of utilizing (BERT) contextual embeddings in question-answering tasks is by appending an output layer to the BERT architecture that maps the contextual embeddings of the input tokens answer spans [3]. SDNet approaches modern question-answering in a new way, instead by locking BERT's parameters and incorporating it as a feature by computing a weighted sum of its outputs [12]. SDNet uses the contextual embeddings outputted by BERT downstream as inputs to a conventional question-context integration layer and output layer, and presents a novel approach for utilizing the newest breakthrough in BERT.
SDNet targets the conversational question answering problem, where given a passage C_t and a history of question-answer pairs $(Q_{t-k}, A_{t-k}), ..., (Q_{t-1}, A_{t-1})$, the model's task is to predict A_t for Q_t [12]. This model achieved state-of-the-art results when it was published on the leaderboard (Nov 29, 2018). Its single and ensemble models currently placed at $10th$ and $8th$ on respectively [12]. The two models achieve $F1$ scores of 76.8 and 79.3, as compared to the human gold standard of 88.8 [12]. Notably, SDNet was the first model ever to surpass an $F1$ score of 80 $F1$ on CoQA's in-domain dataset [12].
The model architecture of SDNet is illustrated below in (Figure 1). Here, we present an overview of the most novel ideas in SDNet's architecture that we believe are applicable to the SQuAD 2.0 dataset.
We begin with the encoding layer, specifically the way in which SDNet leverages the contextualized embeddings from BERT. In a normal BERT architecture, BERT generates L layers of hidden states for each of its BPE tokens, produced as outputs from its transformer network [12]. The novel idea in SDNet is employing a weighted sum of these hidden states to produce contextual embeddings for a word [12]. Specifically, let w be the word split into BPE tokens $s_1, ..., s_k$. Then, the contextual BERT embedding E_w is defined as

$$E_w = \sum_{l=1}^{L} \alpha_l \frac{\sum_{j=1}^{k} h_j^l}{k}$$

where a_l are the weights for each layer, and h_j^l is the hidden state for s_j in layer l. Ablation analysis showed that this method of weighting the outputs of all of the layers of the transformer network let to a 1.75% increase in $F1$ score compared to only using the last layer's output, verifying their hypothesis that outputs from each layer is useful downstream [12]. Ablation analysis also showed that larger BERT architectures led to better $F1$ scores [12].
SDNet also uses a collection of attention mechanisms, each for different tasks. After producing BERT contextualized word-embeddings, SDNet first employs conventional word-level inter-attention from question to context based on the BERT contextualized word-embeddings and GloVe word embeddings [12]. Specifically, word-level inter-attention computes an attention score a_{ij} based on context and question embeddings, and then uses these attention scores to compute a linear combination of the question embeddings [12]. More interestingly, SDNet then employs self-attention techniques to both the context and question, in an attempt to establish direct direct correlations between the words in each [12]. Last and arguably the most interesting, SDNet also employs a multilevel inter-attention mechanism over different levels of word understandings the model makes through a forward pass [12]. These varieties of attention models present novel methods of obtaining the most information out of each recurrent layer in a model.

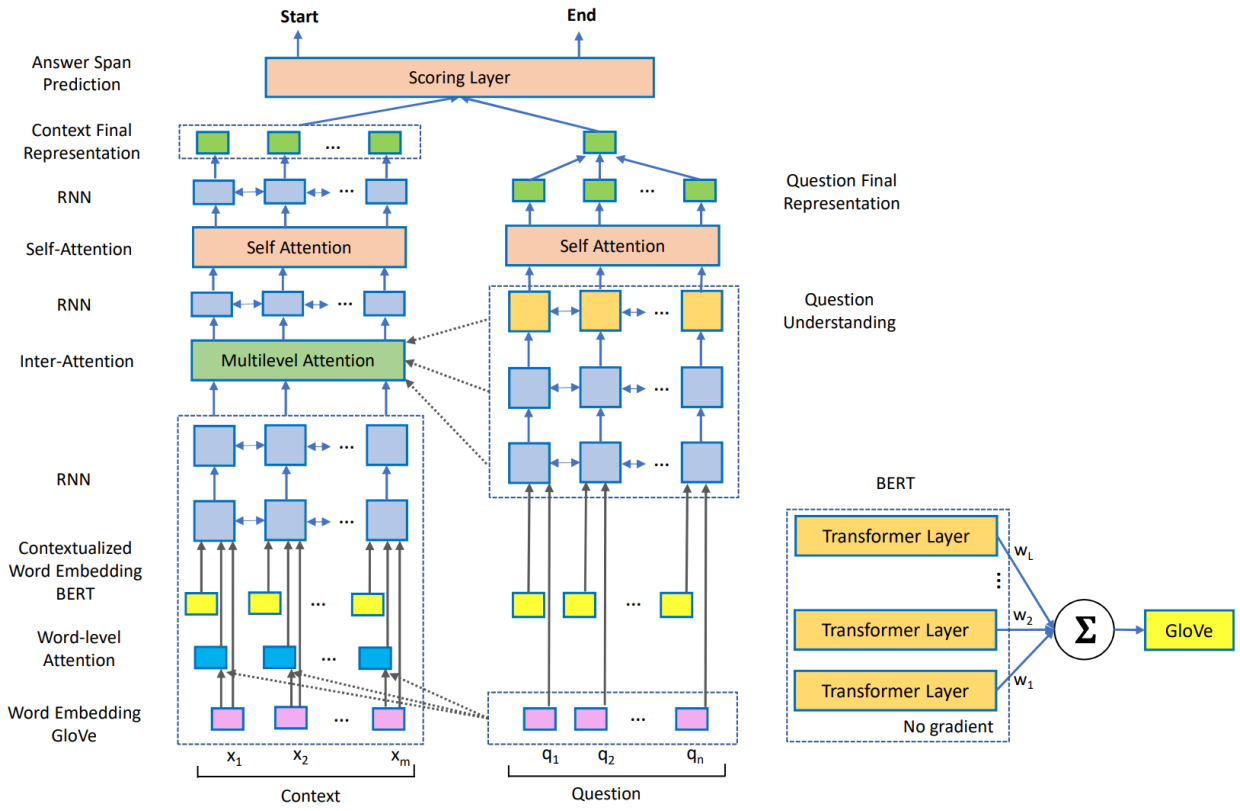

Figure 1: SDNet model architecture

3.2 Conversion between CoQA dataset and SQuAD v2.0 Dataset

Despite the apparent similarities of CoQA and SQuAD, their datasets have many differences. The conversational nature of CoQA means that SDNet uses past question and answer pairs as input at inference time, whereas this is not a feature of the SQuAD challenge. Furthermore, in addition to the no answer questions that were new to SQuAD 2.0, CoQA also contains questions that can be answered by a simple "yes" and "no". Since this project also requires a different format for submission, where unique data IDs are associated with questions, we added another field in our questions containing this ID.

In order to perform well on the SQuAD dataset, we had to change some elements of the model.

First, SDNet returns probabilities for the various answer options. SDNet incorporates these four options into its loss function as follows:

$$\mathcal{L} = \sum_k I_k^S(\log(P_{i_k^s}^S) + P_{i_k^s}^E)) + I_k^Y \log P_k^Y + I_k^N \log P_k^N + I_k^U \log P_k^U$$

where i_k^s, i_k^e refer to the ground truth start and end span positions for the kth question, $I_s^K, I_k^Y, I_k^N, I_k^U$s are indicator functions for span, yes, no, unknown, ground truths, and $P_{i_k^s}^S, P_{i_k^s}^E P_k^Y, P_k^N, P_k^U$ refer to probabilities of start and end position, "yes", "no", and "unknown".

For SQuAD to CoQA conversion, we categorized all the "impossible" questions as "unknown". Thus our loss function is as follows

$$\mathcal{L} = \sum_k I_k^S(\log(P_{i_k^s}^S) + P_{i_k^s}^E)) + I_k^U \log P_k^U$$

Next, we also needed to remove the model of its power of access previous dialogue history. The SDNet sourcecode provided an upstream method to control how many previous questions and how many previous answers to include in the current question query. However, we found that the model had performance to indicate that some form of contextual history was still being used, as we will explain in a later section. To further ensure that we were limiting inference between each question, we altered the code to prevent the addition of other passages to the current question query. We also shuffled the train and dev questions with a context.

Finally, the transformation includes representing the multiple possible answers for a SQuAD question as "additional" answers to capture the fact that there is variance in human answers.

3.3 Adding a Context and Question Understanding Convolution

As displayed in Figure 1, SDNet concatenates BERT contextualized word embeddings with glove word embeddings to feed as input into the question and context understanding multi-layer LSTMs. Yet, as we dove into the model structure, we found that these input dimensions were exceptionally large, and hypothesized that a more higher-level, simplified expression of these inputs could result in more information learned through the LSTMs. Therefore, taking motivation from [11], but also from previous papers that combined feature extraction with recurrent architectures, we built a custom 1-layered CNN structure to extract higher level features from our embeddings.

We describe our CNN below. Notice that we compute three convolutions on our input data that is of size *BATCH x MAX-SENTENCE-LENGTH x EMBEDDING-DIMENSION*, two that perform undilated convolutions with kernel sizes of 3 and 7, and another that performs a dilated convolution with kernel size 5. We note that we maintained a stride of 1 for each of the convolutions and selected padding to keep the dimensions the same, and to keep as much information as possible. As a last step, we average the three layers along with the initial input x, and feed that as input to the LSTM. We intended for our CNN understanding mechanism to learn higher-level local relations amongst the feature spaces as well as amongst each of the tokens of the sentence. Although we add parameters by doing so, we suspect that this actually reduces complexity and overfitting as the inputs are better normalized and more higher-level going into the recurrent architecture and downstream attention layers.

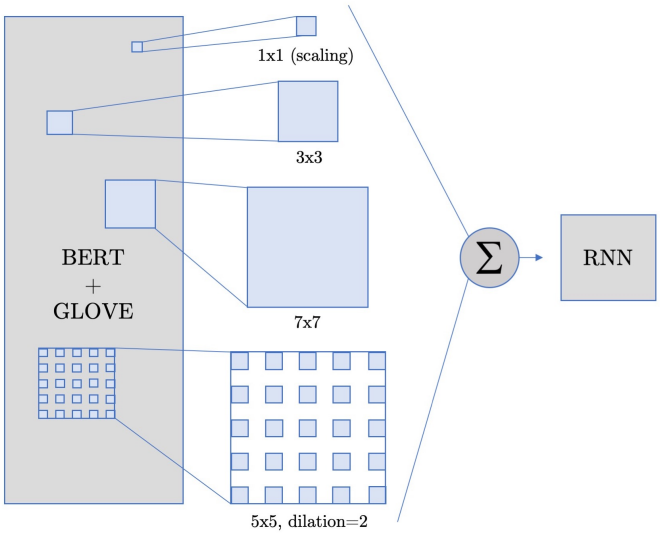

Figure 2: Convolutional Understanding Layer

3.4 Expanding upon using BERT as a feature

We played around with BERT in two ways.

The first method was by unlocking some of the layers in the BERT model. We noticed that one of the features of SDNet was that it locked BERT's internal weights. However, we thought that the model could benefit from learning the weights of the dense pooler layer before the layers get summed into an output. We wanted the BERT model to learn raw data specific to SQuAD.

We then tried to fine-tune the BERT model, then transplant it into SDNet and lock all the weights. In order to fine-tune, we used HuggingFace's BERT implementation and SQuAD fine-tuning script. This takes the pretrained BERT model and adds a linear layer on top that that computes logits for the start and end positions in a span. After running the provided script and training it on SQuAD 2.0 data, we then removed the linear layer and passed the model back into SDNet.

3.5 Multi-task training on SQuAD and CoQA

Inspired by Dr. Socher's lecture on multi-task learning, and since we are adapting a pre-existing CoQA architecture, we decided to train our model on both SQuAD and CoQA data simultaneously. We followed his guidelines of on how to perform this task. Specifically, we trained iteratively, alternating between CoQA data and SQuAD data per batch. We

made an architectural decision to use two loss functions–both the original SDNet loss function as well as our own loss function. We decided to do this because we wanted to emulate real multi-task training and have the model be able to perform on both CoQA queries as well as SQuAD queries. However, we decided to leave previous quesiton and answer history at 0 because we did not want the model to make assumptions about the SQuAD dataset.

4 Experiments

For our experiments, we tested our approaches on the datasets provided, and evaluated our mode;ls with F1 and EM scores. Specifically, we evaluate based on the Answer vs. No Answer F1 score, the F1 score on answerable questions, and the aggregate F1 score. We are minimizing the same cross-entropy loss function used in the original SDNet [12]. We conducted several experiments, moss notably including adding a convolutional layer before the encoding bi-LSTM, fine tuning BERT, and training for multi-task using CoQA data as well.

4.1 Our Final Results

Dev Scores	EM	F1	Answerable F1	No Answer F1
BiDAF Baseline	56.298	59.920	52.439	66.430
SDNet Baseline	63.968	65.488	66.432	59.89
+ Convolutional Understanding Layer	57.963	59.270	47.984	69.633
+ BERT Fine Tuning Pooling Layer	58.418	60.040	54.532	63.594
+ Multi-tasking on both SQuAD and CoQA	59.175	60.852	64.437	56.359

Figure 3: Results

Our best model was SDNet baseline, which scored 64.531 EM and 66.274 F1 on test, which was an improvement over dev.

Because of resources and time, we were only able to fine-tune bert-small, which still took 10 hours. We noticed a significant dip in F1 and EM score, but we are unable to determine if it is due to the use of a smaller BERT model, or if the fine-tuning harmed performance. Given more time and more resources, we would like to run this experiment but on bert-large, so that we can directly compare the results.

4.2 Our Best Model - the SDNet Baseline Model

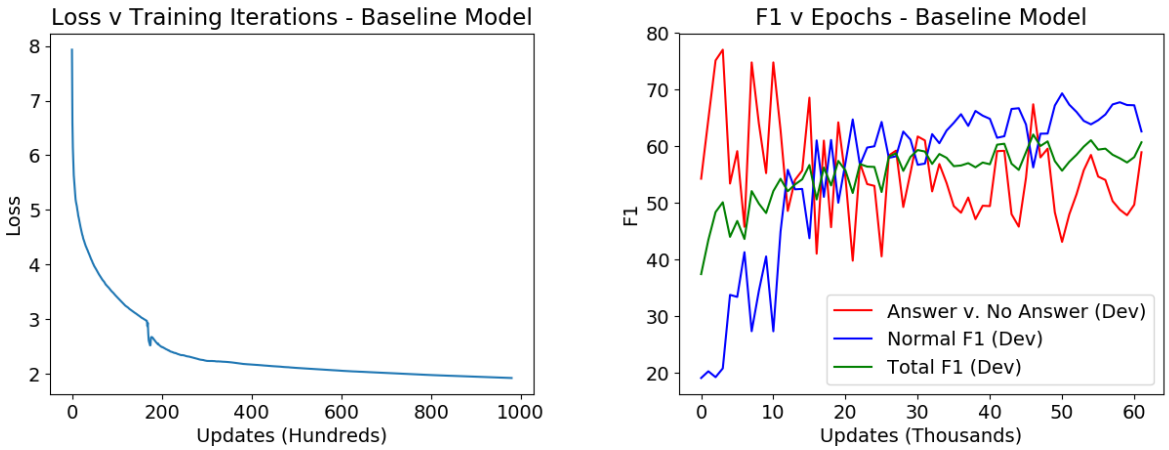

Figure 4: Best Model Loss and F1 v Epochs

It's interesting to see how when training, our model first attempts to quickly improve by boosting no answer accuracy, since this is arguably an easier task. After a while of training, it's impressive to see the answerable F1 improve quickly and actually become more accurate than the no-answer F1.

The fact that our model is accurately predicting answer spans is probably heavily influenced by the complexity and multitude of attention models that were used.
Another interesting note is the slight cliff in the loss. This came at a time when our VM was shut off, and we were forced to restart training. Since we were using ADAMAX optimization, our momentum was restarted and thus the updates were different than they would've been without restarting.

4.3 A close second - the SDNet with an added layer of Convolutional Understanding

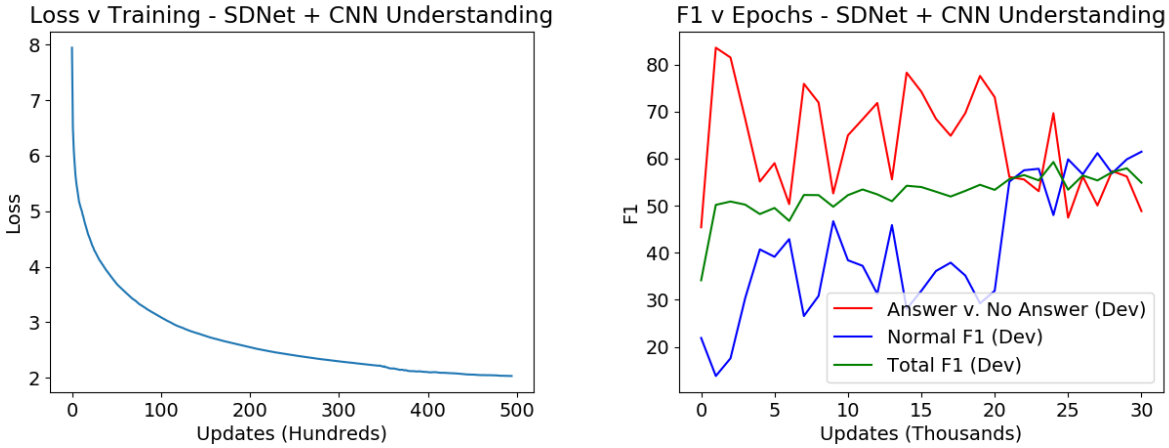

Figure 5: Best Model Loss and F1 v Epochs

The results of our CNN were impressive given the much shorter training time compared to our baseline SDNet. Since the loss is still decreasing and F1 trending upwards when we stopped training, we predict that more time devoted to training and developing this model would lead to performance gains similar to that of our best model. The convolutions only added less than 100 more parameters, so we were hoping that this would not hurt the model's gradient passing ability, but could still improve our model's performance by added interaction between BPE tokens.
One thing to note when looking at dev F1 scores is how much the normal and no-answer F1 scores compete. This is because when the model predicts more no-answer questions, then it's accuracy will improve on no-answer F1, but this takes away from the number of questions it gets a chance to predict answer spans for. It was also interesting to watch how our model valued no-answer more at the start, and then over time favored attempts to answer questions.

5 Analysis

5.1 Qualitative Evaluation

We evaluated our results qualitatively by selecting random examples where our model was failing. We summarize our findings below with a few key examples that illustrate this.
Here is an example of a question where our model incorrectly predicted answer vs no answer:

Context:
"Like many cities in Central and Eastern Europe, infrastructure in Warsaw suffered considerably during its time as an Eastern Bloc economy – though it is worth mentioning that the initial Three-Year Plan to rebuild Poland (especially Warsaw) was a major success, but what followed was very much the opposite. However, over the past decade Warsaw has seen many improvements due to solid economic growth, an increase in foreign investment as well as funding from the European Union. In particular, the city's metro, roads, sidewalks, health care facilities and sanitation facilities have improved markedly."
Question:
"What was a major failure, especially in the building of Warsaw?"
Target Response:
–no answer–
Predicted Response:
"Three-Year Plan to rebuild Poland"

We think our model incorrectly predicted this due to the attention module. Namely, the question opens with "what" and follows with "major" and "building of Warsaw", and the sentence in the context, "Three-Year Plan to rebuild Poland (especially Warsaw) was a major" contains these words. It is likely that less emphasis was put on the "success" from the attention modules, and thus the response focused on the other portions of the context to determine an answer span.

Here is an example of a question where our model incorrectly predicted the span:

> **Context**:
> "Moderate and reformist Islamists who accept and work within the democratic process include parties like the Tunisian Ennahda Movement. Jamaat-e-Islami of Pakistan is basically a socio-political and democratic Vanguard party but has also gained political influence through military coup d'état in past. The Islamist groups like Hezbollah in Lebanon and Hamas in Palestine participate in democratic and political process as well as armed attacks, seeking to abolish the state of Israel. Radical Islamist organizations like al-Qaeda and the Egyptian Islamic Jihad, and groups such as the Taliban, entirely reject democracy, often declaring as kuffar those Muslims who support it (see takfirism), as well as calling for violent/offensive jihad or urging and conducting attacks on a religious basis."
> **Question**:
> "What is the goal of Islamist groups like Hezbollah and Hamas?"
> **Target Response**:
> "abolish the state of Israel"
> **Predicted Response**:
> "democratic and political process as well as armed attacks"

This is a rather tough question that comes down to nuanced semantics. It is true that the Islamist groups participated in "democratic and political processes as well as armed attacks", and in some sense this was their goal (albeit a short term one). The model did well to predict that this question had a response, but the question is hard to answer because the text does not contain the word "goal", and the model likely draws an association between the "goal" in the question and the activities that were committed by the mentioned organizations.

5.2 Error Analysis

In [12], Zhu et al. reported that SDNet with Question/Answer past history achieved an F1 score of 77.99, and removing the Question/Answer past history achieved an F1 score of 69.43. As the CoQA task is a harder task than SQuAD, we expected that the out-of-box model would perform better than a 69.43 F1 score on SQuAD. However, we found that it barely got better than an F1 score of 63, and only eventually after some minor manipulations did it reach the F1 score we reported. We speculate that there may be some issue we've overlooked. For one, our pre-processing step could be missing something. As CoQA is based on conversation history, it expects a specific order of question/answer pairs, so it does not shuffle the data before-hand. We manually shuffle the data before hand by editing some of SDNet's batch generation code, and suspect that may be a point of concern. We also are still not completely sure that we have ignored past question/answer pairs completely in our modified SDNet model. We are well aware that if we haven't, the model could be learning unnecessary noise from looking at random past data, and therefore not being able to learn effectively. This could be verified by our graphs of F1 scores over updates, especially since most of the learning was very jagged across updates.
We also would like to reflect on why our experiments did not produce better results than the SDNet baseline. It seems that anything we tried caused a large separation amongst Answer vs. No Answer and Normal F1 scores in the beginning portion of learning, and that the model could not learn to reconcile the two and produce a novel F1 score. It also speaks to the idea of a model being easily adaptable. We found it very difficult to add significant elements that raised the F1 score, and therefore would all like to explore the benefits of multi-task learning in the future, and perhaps getting models to work nicely across multiple datasets.

6 Conclusion

In this paper, we describe SDNet, a conversational question answering model, and transform the architecture to perform on the SQuAD dataset. In addition, we provide several experiments to analyze performance benefits of fine tuning, adding additional layers, and multi-task learning. While our results are not state of the art, we learned that adding a complex model on top of BERT does not improve over much simpler models on the leaderboard.
For future work, we would like to see a reduction of this model to have fewer parameters and complexity. It's entirely possible that SDNet benefits from this complexity on CoQA, where past questions and answers are used to improve performance, but on SQuAD, simpler models seem to be the better choice.

References

[1] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. Quac : Question answering in context. *CoRR*, abs/1808.07036, 2018.
https://arxiv.org/abs/1808.07036.

[2] Jacob Devlin and Ming-Wei Chang. Open Sourcing BERT: State-of-the-art pre-training for natural language processing. *Blog*, 2018.
https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html.

[3] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.
http://arxiv.org/abs/1810.04805.

[4] Hugging Face. The big--extending-repository-of-transformers: Pretrained pytorch models for google's bert, openai gpt gpt-2, google/cmu transformer-xl. 2019.
https://github.com/huggingface/pytorch-pretrained-BERT.

[5] Hsin-Yuan Huang, Eunsol Choi, and Wen-tau Yih. Flowqa: Grasping flow in history for conversational machine comprehension. *CoRR*, abs/1810.06683, 2018.
https://arxiv.org/abs/1810.06683.

[6] Hsin-Yuan Huang, Chenguang Zhu, Yelong Shen, and Weizhu Chen. Fusionnet: Fusing via fully-aware attention with application to machine comprehension. *CoRR*, abs/1711.07341, 2017.
https://arxiv.org/abs/1711.07341.

[7] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016.
Paper: http://arxiv.org/abs/1606.05250
Data: https://rajpurkar.github.io/SQuAD-explorer/.

[8] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042, 2018.
https://arxiv.org/abs/1808.07042.

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

[10] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.
https://arxiv.org/abs/1706.03762v5.

[11] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018.
https://arxiv.org/abs/1804.09541.

[12] Chenguang Zhu, Michael Zeng, and Xuedong Huang. Sdnet: Contextualized attention-based deep network for conversational question answering. *CoRR*, abs/1812.03593, 2018.
http://arxiv.org/abs/1812.03593.