Question Answering with SQuAD v2.0

Vladislav Savchuk
DS-01 / Innopolis University

Vyacheslav Shults
DS-01 / Innopolis University

Farit Galeev
DS-01 / Innopolis University

Abstract

Machine reading comprehension is a challenging problem in NLP, where the model should answer a given query using some context. In some datasets model not only need to find the correct answer, but also to distinguish between answerable and unanswerable questions. This document represents our research, findings and analysis when doing the project for the question answering problem in NLP using the SQuAD v2.0 dataset. In the beginning you may find a small survey of some of the solutions that were proposed previously by the researchers and later we introduce some of the concepts we have tried when trying to find a solution and some analysis about theirs performance.

1 Introduction

Question answering is one of the fields in NLP related to text comprehension. Given some context text and query, model needs to give the answer as the span in the context (or determine that the question is unanswerable given the context).

This report consists of the following parts:

- **Background**. Here you will find background information about the problem of question answering task and related works
- **Method**. Here you will find description of the dataset we have chosen as well as description of the baseline methods, analysis about it and some proposed hypothesis that we have tried to test
- **Analysis**. In this section we will have some analysis and conclusions we can make from our research
- **Conclusion**. This section wraps up everything that have been described in this report

2 Background

2.1 Related Works

Questions answering task has been widely used as a metric to measure language comprehension for your NLP model. Different models were suggested that are able to work directly in this fields, such as Syntax Guided approaches by Zhang et al., (2019) that tries to use linguistic information of the text when performing attention, BiDAF (Bi-Directional Attention Flow) by Seo et al., (2016) which tries to use context to query and query to context attentions (that's why it is called bidirectional) to 'highlight' important parts needed for question answering in both query and context texts.

There are also some models that are heirs to the BERT that is very powerful on many NLP tasks and are trying to improve upon it's success with some changes to model architecture or training process, e.g. ALBERT by Lan et al., (2019) which decreases amount of parameters by some interesting sharing techniques and loss, RoBERTa by Liu et al., (2019) which tries to improve on BERT training and some other small aspects.

One of the latest advances in this area has been made by Zhang et al., (2020). This approach is inspired by how human solve this task. Firstly, he reads the text and question sketchy extracting basic information from the text and decides about possibility of answer, and after that rereads intensively, verifies the answer and gives the final prediction.

3 Method

3.1 Dataset Decription

For the project we have chosen SQuAD v2.0 dataset as it is very popular and widely used as an evaluation metric for the language comprehension models. The dataset consists of 536 sampled top Wikipedia articles covering a wide range of topics, in which one should read the context paragraph and answer related questions. The dataset was built by crowdsourced workers who were asked to come up with the questions (wording may be or should be different from the one used in the paragraph) and mark the span with the answer. Answer can be a date, other numeric value, person name, location or any other text given by the context. All samples are split into train set (80%), development set (10%) and test set (10%). For the training we have 442 articles, with total of around 19K paragraphs and total of 130K questions (30% of them can't be answered using given context). In the development test there are 35 articles, with total 1.2K paragraphs and total of 11K questions (50% of which can't answered). We may notice that there is different amount of number of unanswerable question in dev set, compared to train set. Although their ratio is not that different this still may result in some amount of false positives. For more thorough description on what dataset may or may not have and how it is have been constructed, one should refer to Rajpurkar et al. (2016). For our exploratory data analysis one should refer to the code.

For evaluation two metrics are used: Exact Match (EM), which measures percentage of span predictions that matched the answer exactly and macro-averaged F1 score, which measures average overlap between prediction and ground truth answer.

3.2 Baseline

For the baseline we have created a very simplistic model: it is a just a transformer consisting of two encoder and decoder layers with 4 attention heads.

The input is firstly transformed using embedding layer, we then feed the transformer the question (input to the encoder) and then the context (input to the decoder). It can be seen as someone who firstly reads the question and then tries to find an answer for it in the text.

The performance of this is model is, as expected, very poor (70% F1 with no answers with 0.2%F1 on questions with answers more descriptive results can be found in **Table 1**), but even with this small model as an example we can make some interesting and useful conclusions about the problem we are solving and properties of the models that should be able to solve it:

- The baseline is able to predict if the question can be answered nearly perfectly, but is still unable to learn proper understanding on how to answer to the questions after some epochs (even though f1 on questions with answer is raising a little bit, we lose f1 on unanswerable questions). This means that it doesn't have the capacity to comprehend what it should answer with (e.g. we need a smarter model

- The paragraphs of some context are very long, therefore the model should use a proper RNN, so that our model wouldn't forget what happened 200 words ago if this part was important for answering the question

- It is very hard for model to understand the some concepts about language structure without proper pretraining of some of the parts of the model on other language comprehension tasks (e.g. embeddings, feature extractor). When constructing the baseline it was seen that using GLoVE embeddings instead of the training new ones is advantageous and yields better result. It seems that context text itself isn't enough for model to learn proper words representations.

In the next subsections one may find our research on using, building and trying to improve other models.

3.3 Modules on Top of ALBERT

This section of the research was inspired by Yuwen et al., (2018) For the first tryout we have taken ALBERT model as a feature extractor model and build several modules on top of ALBERT module.

The idea of this kind has come from Computer Vision, as there exists feature extractor module and head for specific task, so we have decided to try and see if using some other additional modules on top will improve our predictions.

3.3.1 Encoder and Decoder blocks

Bidirectional Long Short-Term Memory Layer Encoder/Decoder LSTM model on top of ALBERT may help to interpret temporal dependencies between time-steps of the output tokenized sequence better, so this is one of the improvements we have tried.

AutoEncoder There is a second hypothesis that the output of the Albert features is noisy, so AutoEncoder may help to more generalize ALBERT tokenized sequence before task specific QA layer (which is a simple Linear layer). AutoEncoder was designed by using Convolution Layers.

ALBERT Output layer A linear output layer that converts the dimension of the output sequence from `(batch_size, seq_len, hidden_state)` to `(batch_size, seq_len, 2)`. Further, we split it into two parts get the start and end logits.Then we compute the cross-entropy loss with the start and end position vectors.

In **Figure 1** one may see how the architecture of the model works. We take ALBERT features, run them through either BiLSTM or AutoEncoder and then through the output.

During the training we have freezed ALBERT weights to drastically decrease the training time (although this and the fact that we trained only for 2 epochs may have decreased the final quality of the model). The results we've got can be seen in **Table 1**. They seem strange (we also added just ALBERT model for the comparison of the obtained result), but after doing hours of debugging we didn't find any possible errors that could affect the model, so we may only suggest that freezing albert and/or doing little epochs drastically affected the performance of the model. Analysis of them will be done below.

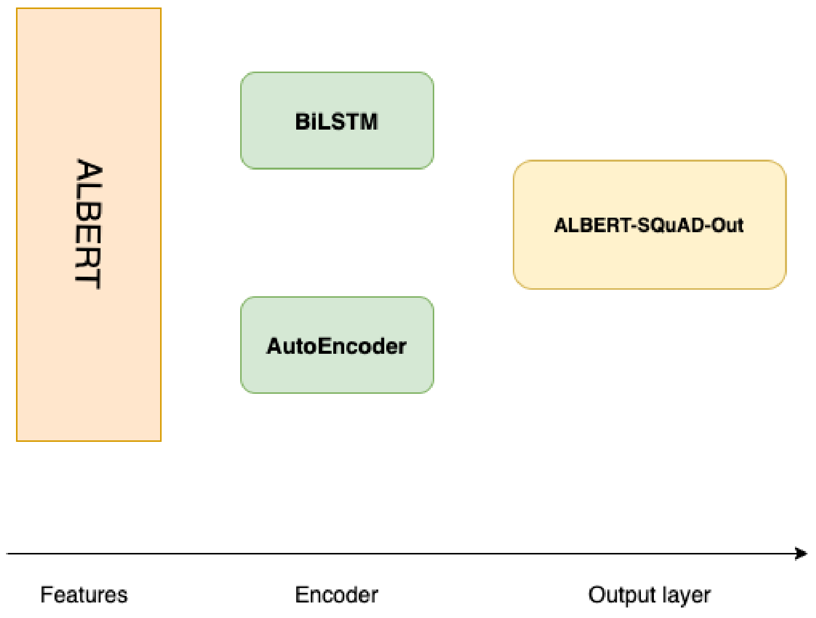

Figure 1: Architecture of the model

3.4 Retrospective Reader

After some failures we have decided to look at the top of the SQuAD 2.0 leaderboard and implement one of the best algorithms from it. We stopped at Retrospective Reader (2020) algorithm that split MRC into two separate tasks: determining whether a question can be answered or not and prediction of the answer. Each task is performed by separate network and after that output are are merged to create a final answer.

For each task as encoder we used pretrained ALBERT of base size and trained them separately. In inference mode we get prediction by each module and then merge the to get final prediction. The results of the training can be seen in **Table 1**.

4 Analysis

Working with such dataset wasn't easy and lack of experience was shown even when developing a simple baseline model, although we were able to make some conclusion even when doing simple things. Some of them were already described in the method section, but here you will find description of something that wasn't included.

When building baseline it was noticed that:

1. Seq2Seq network without any pretraining wouldn't perform well on this type of task. This is because there are a lot of unknown words that could be found and it needs to generate them. If the word isn't in the vocabulary it will be mapped to UNK token which hurts performance (when predicting the answer and when trying to comprehend context). It is also very hard to learn an embedding layer as there are various topics present in the dataset which means that the model wouldn't be able to learn the proper representation of the words

2. Using a simple LSTM or GRU layers as feature extraction mechanism without any attention also doesn't result in any good result as model simply isn't able to learn what the output should be due to the long sequence length in the context text

From the results we can suggest that:

1. Although, the results of our experiments on ALBERT aren't credible enough, we can suggest that using AutoEncoder over BiLSTM after ALBERT features leads to better results.

2. Also looking at the single ALBERT model it can be seen that using any other layer (in our case, compared against adding AutoEncoder and BiLSTM) after it except just single linear layer increases overall performance.

4.1 Results

Method	**All**		**HasAns**		**NoAns**	
	EM	**F1**	**EM**	**F1**	**EM**	**F1**
Baseline	39.43	39.72	0.05	0.62	78.7	78.7
Just ALBERT	5.55	8.6	0.18	6.27	10.91	10.91
ALBERT + AutoEncoder + ALBERTOut	20.94	22.04	0.25	2.45	41.58	41.58
ALBERT + BiLSTM + ALBERTOut	17	18.15	0.47	2.67	33.59	33.59
Retro-Reader over ALBERT	78.3	81.62	74.3	80.99	82.25	82.25

Table 1: Results (%) on Dev set with different methods for Question Answering task.

From the results it can be seen that the for the models it is easier to determine if the question has answer given current context or not. This fact and also the fact that there are about 50% of questions that can't be answered in the dev set makes some of the models look good, although they might struggle to predict good in when the question might have an answer.

5 Conclusion

In the end, this turned out to be a very challenging task for the beginners. Even building the baseline wasn't as trivial as one would have thought as there are many design decision that come into play when trying to create even the simplest model for this complex dataset. Nevertheless, this activity provided a lot of experience, we have gained some insights into the field of question answering, problems and challenges one can face with when approaching this problem, as well as got introduced to several interesting architectures and concepts from different papers (although some of them are still hard to comprehend).

Unfortunately, due to a time constraint we had, as well as the heaviness of some of the models (too long to see the results and make changes), we couldn't check more hypothesis that we actually had and maybe produce better results on some of the models we have tried (e.g. by finding out the reason why the scores on ALBERTs are so low).

References

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. 2016. Squad: 100,000+ questions for machine comprehension of text.

Minjoon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. 2016. Bidirectional attention flow for machine comprehension.

Zhaozhuo Xu Yuwen Zhang. 2018. Bert for question answering on squad 2.0.

Zhuosheng Zhang, Yuwei Wu, Junru Zhou, Sufeng Duan, Hai Zhao, and Rui Wang. 2019. Sg-net: Syntax-guided machine reading comprehension.

Zhuosheng Zhang, Junjie Yang, and Hai Zhao. 2020. Retrospective reader for machine reading comprehension.