Advanced Natural Language Processing: Information Retrieval, Question Answering

Assaraf David

May 11, 2021

Abstract

In this report, we will try to dive deep into the Machine Question Answering Problem. More specifically, we will consider here the problem of extractive question answering, meaning that we expect the answer to be contained in the context and we need to develop techniques that extract these answers from the context. In order to train our classifiers to extract the positions of the span in the context, we will use the Stanford Question Answering Dataset [Raj+16] and compare the performances of our implementation with their baseline, in terms of exact match (EM) and F1 score.

1 Our Approach

In order to address this problem, several architectures have been proposed. In [WJ16], a Seq2Seq architecture has been used in order to produce contextualized embeddings of the different words in the context. However, one particularity of the SQuAD is that they removed fairly 'short' contexts (of ¡ 500 characters). Therefore, we only have long contexts and it might be difficult for RNN based Encoder Decoder structure to keep track of the influence of these large contexts. This is why we decided to use an architecture relying only on Attention [Vas+17]. Extending on [Dev+18] BERT architecture, leveraging the Encoder from the Transformer architecture allows to leverage a framework for a large number of tasks (this is being done with a very specific way of preparing the data, in order to be able to leverage global Seq2 Seq tasks). Here, we will use DistillBERT [San+19], which has proven to perform similarly to BERT on SQuAD and be 60% lighter.

1.1 Data Processing

Like BERT, DistillBERT uses BPE in order to account for subword information (specially the WordPiecs library). Therefore, we tokenized our text using the specific tokenizer creating those subwords and inserting the special characters required in BERT-framework training ([CLS] and [SEP]). Last, in order to leverage the Seq2Seq framework of BERT, our data is constituted the following way: Question + [SEP] + Context.

1.2 Modelling

1.2.1 Building a representation for the question and context

Feeding this data into the DistillBERT, we get as output representations from each word in the question + context. This representation is a contextualized embedding since it is constructed for each word based on the context. These embeddings might be created in several ways:

- Get the representation of the last layer in the architecture (our baseline model)
- Combining the embeddings from the hidden states in a static way (mean pooling)
- Combining the embeddings from the hidden states in a dynamic way (using Attention)

Later on, we will also try to analyze the influence of using contextualized embeddings vs word embeddings.

1.2.2 Predicting the locations of the answer

As we have already mentioned in the abstract, one advantage of SQuAD is that the answer is located in the context. Therefore, from our embedding representation of the question and context, we only

need to predict two integers in order to locate the answer: the predicted location of the start and end of the answer span. We do so connecting a Linear layer on top of the output of DistillBERT model. Leveraging a pre-trained DistillBERT model, we train this end-to-end for 1 epoch, with a relatively small learning rate (suitable for Transfer Learning) , L2 regularization and 0.2 dropout. Our optimizer is AdamW with an Exponential learning rate scheduler. Last, in order to provide stability in the training, we used weight clipping. Here is the loss dynamics we got from training this model, using Cross Entropy as the loss function:

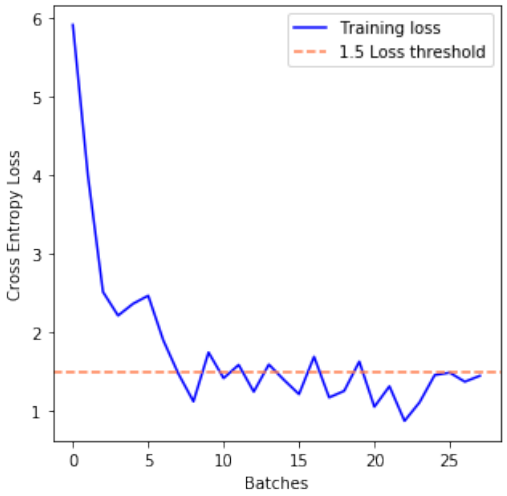

Figure 1: Training loss dynamics of our end-to-end contextualized model. One could notice that the model is beginning to overfit.

One thing to note here is that in order to achieve such performances, we used a complex setting: the DistillBERT model created contextualized embeddings for every word based on this context, which allowed to give some useful information about every word. Moreover, the embedding generation has been fine-tuned in regards to the downstream classification. Therefore, one legitimate question would be: *how will we perform in a simpler setting*?

1.2.3 Simpler setting: Static Embeddings [QUESTION 1]

In this experiment, we are going to train the model using non contextual embeddings: we will use only word embeddings, ie use the embeddings as if it were a lookup table. Here is the loss dynamics that we got from these experiments, using static embeddings.

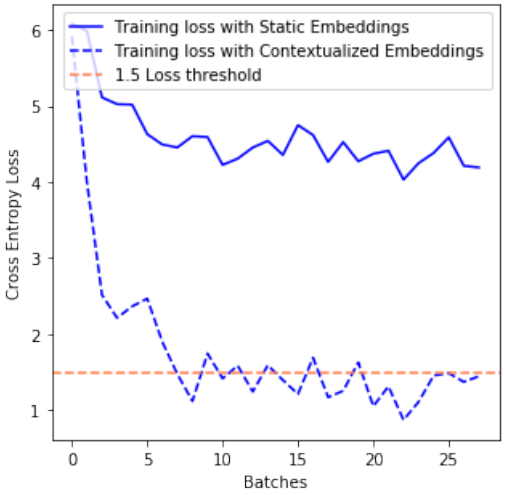

Figure 2: Comparison of the Loss dynamics when using Contextualized versus Non-Contextualized Embeddings

here, we reason in terms of training loss because the further EM and F1 scores will heavily depend upon the decoding strategy employed, and will not depend on the capacity of our model to learn different relationships. Therefore, we can see here the crucial importance of building deep contextualized representations for very word in the context, since it allows to include some context and meaning inside the representation of one word. This is the fundamental difference between LSA/Word2VEC/GloVE techniques vs ELMO/BERT-based techniques.

2 Decoding [QUESTION 2]

Given the output from the end-to-end Neural Network, we need to decode the answer from the logits being output. A decoding strategy is defined as follows: given the context and the logits (of size [2, len(context)]), output the span of the answer in the context. We identified three strategies in order to do so:

- Left to right: select $i = argmax_i \quad logit_{start}$ and then $j = argmax_j \quad logit^{j:}_{end}$

- Right to left: select $j = argmax_j \quad logit_{end}$ and then $i = argmax_i \quad logit^{:j}_{start}$
- Joint: select $i, j = argmax_{i,j} \quad logit^{i}_{start} + logit^{j}_{end}$

A priori, I would expect the left to right and right to left strategies to have similar results. The 'failure cases' for both would be symmetric and both are greedy approaches. However, in the joint strategy, we are not making any greedy decision so the results should be better. Indeed, our choices are based upon an exhaustive search, without reducing the search space due to a specific choice. The metrics used in order to assess the quality of the decoding strategy are EM and F1 scores (the same ones have been used in [San+19] in order to compare the performances of DistillBERT vs BERT on SQuAD Dataset). Here are the results I got for my Baseline model:

Strategies	EM	F1
Left to Right	0.659	0.808
Right to Left	0.657	0.804
Joint	0.664	0.811

Table 1: Baseline Results according to different strategies

We can see that these results make sense with our intuition: left to right and right to left strategies provide similar results whereas the joint decoding provides some improvement over both.

3 Other Experiments

3.1 A failure mode of our decoding strategies [QUESTION 3]

	two	actors:	Tom	and	Jerry
P(start)	0	0	0.51	0	0.49
P(end)	0	0	0.49	0	0.51

Table 2: Failure Mode

3.1.1 Setting

Here, the question tat is asked is: *Name one actor.* The only two right answers are Tom or Jerry. The issue is that our decoding strategies could select the entire spans Tom and Jerry since the decisions are local and do not take into account statistics of the entire span selected.

3.1.2 Results using left to right greedy strategy

The answer will be deterministic here since the greedy decision does not involve any sampling decision. **Result: Tom and Jerry** and we have a probability 1 of being wrong

3.1.3 Results using a sampling based method

Here, we will samples uniformly from $start \sim P(start)$ and $end \sim P(end)$. The correct outputs should be Tom, Tom or $Jerry, Jerry$. Therefore $P(correct) = P(Tom, Tom) + P(Jerry, Jerry)$ where P is a joint probability over P^{start} and P^{end}. By independence of these two probabilities, the result is $P(correct) = P^{start}(Tom)P^{end}(Tom) + P^{start}(Jerry)P^{end}(Jerry) = 2 * 0.51 * 0.49 = 0.4998$.

3.1.4 Towards correctness and ambiguity balance

In order to get a right solution roughly 50% of the time, one easy fix to our decoding strategy would be to restrict the potential lengths of the decoding strings to one. Therefore, the strategy could be to sample the initial word from $P(tom, jerry)$ and then output them directly as result from the decoder. Therefore, we could get a 51% probability of getting Tom and a 49% probability of getting Jerry.

3.2 Improving our model performances[QUESTION 4]

3.2.1 Improving the loss performances

In the [Dev+18] model it was said that combining the embeddings from all of the encoder hidden states might improve performances. Therefore, this is what I am going to do. I will create a representation from every word that will be the output of some attention mechanism between all of the previous hidden states. The rational in this idea is that, in every one of the 6 heads of the BERT Encoder, the MultiHead Self Attention performed is between

different words of the context, in the same embedding setting. However, one could potentially infer that the different heads build different semantic relationships between word inputs, some are more related to the structure whereas others are more related to the meaning. This motivates combining the different representations using Attention. Moreover, training these attention weights in an end-to-end fashion with respect to the downstream classification task might provide good results. Once again, here we will only consider performances in terms of loss function, since the EM And F1 score will rely on the downstream decoding strategy. The results we got are actually better than our previous results in terms of loss dynamics:

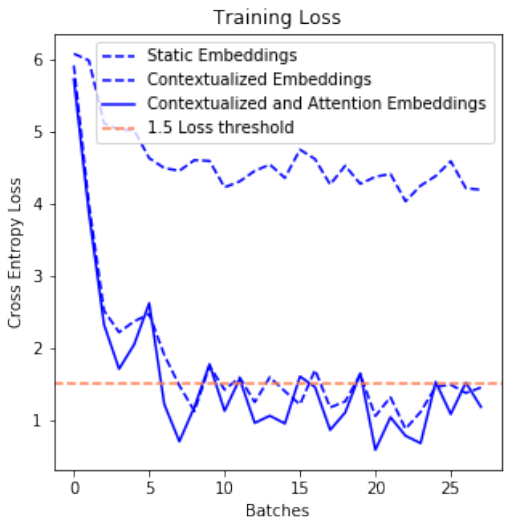

Figure 3: Comparison of different implementations of the Information Retrieval Problem

In terms of EM and F1 scores, the scores I got were:

Strategies	EM	F1
Left to Right	0.664	0.810
Right to Left	0.661	0.809
Joint	0.683	0.824

Table 3: Attention Results according to different strategies

Therefore, we can see that this strategy of Attention between different hidden states provided a significant improvement in terms of F1 and EM Scores.

References

[Raj+16] Pranav Rajpurkar et al. "Squad: 100,000+ questions for machine comprehension of text". In: *arXiv preprint arXiv:1606.05250* (2016).

[WJ16] Shuohang Wang and Jing Jiang. "Machine comprehension using match-lstm and answer pointer". In: *arXiv preprint arXiv:1608.07905* (2016).

[Vas+17] Ashish Vaswani et al. "Attention is all you need". In: *arXiv preprint arXiv:1706.03762* (2017).

[Dev+18] Jacob Devlin et al. "Bert: Pre-training of deep bidirectional transformers for language understanding". In: *arXiv preprint arXiv:1810.04805* (2018).

[San+19] Victor Sanh et al. "DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter". In: *arXiv preprint arXiv:1910.01108* (2019).