Few-Shot Extractive Question-Answering

Michael Sehaik sehaik@mail.tau.ac.il
Roi Bar-On roieliyaub@mail.tau.ac.il
Yariv Levy yarivlevy@mail.tau.ac.il

1 Introduction

Few-shot extractive question-answering is an exercise project given in Advanced Natural Language Processing course at Tel-Aviv University. Our goal is implementing a generative version of Splinter (Ram et al., 2021) based on T5 (Raffel et al., 2020). To do so, we pretrain a model for extractive question answering, and fine-tune it to target styles and domains of questions with a small number of examples (*few-shot*).
Our model results were disappointing - we further discuss possible causes in this paper. Nevertheless, finetuning a pretrained model showed great results and were similar to those reported by the non-genrative model (Splinter). Those results are promising, given the fact that the generative version of the task is complex.

2 Implementation

The project was written in Python using both PyTorch and Tensorflow.
Development was based on Splinter GitHub repository, with extensions and modifications that fit our goals. We modified Splinter's training to make our pretrained models generative.
Our models were trained in 2 phases - pretraining and finetuning. The pretraining is the core of the training and as such, we used bigger dataset and trained it for longer time.

Pretraining At this phase, we initialized a random model and pretrained it for ten hours on a single GPU, using the hyperparameters repoted in T5 paper.

Finetuning At this phase we used different configurations, on different data-split size and different datasets.

3 Experiment Setup

3.1 Datasets

For simplicity, we selected a subset of the benchmark from Splinter's repository: SQuAD (Rajpurkar et al., 2016) and Natural Questions (Kwiatkowski et al., 2019).
Although required by the instructions document, TextbookQA (Kembhavi et al., 2017) wasn't used since the context of each question was too long to fit in a 512 size vector, and led to memory issues, this part wasn't an integral part of our project. As required in the instruction, we examined different dataset subsets, each of unique size - 0, 32, 128, 512.
The datasets were generated using Huggingface[1] pretrained tokenizer. Each training example and label were padded into a vector of length 512.

Pretraining Our training dataset was generated from a large amount of Wikipedia articles in a similar manner to Splinter, by randomly masking the most common n-grams. The dataset consists of around $5,000,000$ examples, where random samples of about $15,000$ examples are used in the development dataset.
The labels contain the missing span tokens in the format expected by Huggingface's[2] T5ForConditionalGeneration model. A demonstaration is given in Table 1.

Benchmark To evaluate how our pretrained model performs, we finetuned it using small subsets of training examples taken from larger datasets. We used QA mrqa-few-shot benchmark from Splinter's GitHub repository[3].
We train and evaluate our model on SQuAD (Rajpurkar et al., 2016), and naturalquestions

[1] https://huggingface.co/transformers/main_classes/tokenizer.html
[2] https://huggingface.co/transformers/main_classes/tokenizer.html
[3] https://github.com/oriram/splinter

Context: $\langle extra\_id\_1 \rangle$ ... .$\langle extra\_id\_11 \rangle$ $\langle pad \rangle$ ranks tenth of all Indian States in the Human Development Index scores with a score of 0.416. The National Council of Applied Economic Research district analysis in 2001 reveals that Krishna,$\langle extra\_id\_12 \rangle$ $\langle pad \rangle$ ... **Label**: $\langle extra\_id\_1 \rangle$ ... $\langle extra\_id\_11 \rangle$ Andhra Pradesh $\langle extra\_id\_12 \rangle$ West Godavari $\langle extra\_id\_13 \rangle$ ...

Table 1: Training example generated from a wikipedia paragraph by masking the most common n-grams. $\langle extra\_id\_x \rangle$ is called a sentinel token, and is used to match between the masked span, and the labels.

(Kwiatkowski et al., 2019) datasets, where each dataset was divided into three splits of 32, 128, or 512 examples.
In addition, we used $1,000$ examples from each split by using its development set, in order to evaluate our fine-tuned model.
The dataset for the evaluation slightly differed from the training dataset. We used examples of the form:

$$context\ \langle sep \rangle\ question\ \langle extra\_id\_1 \rangle$$

Where the label was of the form:

$$\langle extra\_id\_1 \rangle\ answer$$

3.2 Evaluation Metrics

In order to evaluate our performance, we generate question-answer pairs, that were used for testing.
We preprocess each answer by applying standard normalization, and finally we calculate F1 score for each example[4].
We report the average F1 score we got on the entire test set on each split.

3.3 Baseline

For baseline, we used a pretrained T5-small model from Huggingface while finetuning was done in the same way as on the suggested model. The average F1 score was compared between our model and the finetuned Huggingface model.

3.4 Hyperparameter Settings

Pretraining For pretraining, we initialize a model randomly and train it for 10 hours or up to $500,000$ steps (whichever came first). We used Adam (Kingma and Ba, 2017) optimizer and batches of $2,048$. Learning rate starts at $5 \cdot 10^{-5}$ and warmups for $50,000$ steps, after which it decays linearly.

Finetuning For finetuning, we train all models for $1,500$ steps using AdamW (Loshchilov and Hutter, 2019) optimizer and batches of 32. We used a constant learning rate of $1 \cdot 10^{-3}$ for 1200 steps, after which the learning rate decays exponentially. We update all parameters.

4 Results

Pretraining Our model results were disappointing, the model failed to converge, even after 10 hours of pretraining. Under close study it was apparent that our model generated as output sentinel tokens without the missing spans, which is a local minimum solution. Using different settings of hyperparameters, did not yield any improvement. In figure 3 we can see that the best evaluation loss was not as close to zero as we would like. We further elaborate on the possible reasons in the analysis section.

Finetuning Figure 1 and table 2 demonstrate our results on the SQuAD, and naturalquestions benchmark. We can see that our generative model results were very similar to those reported in the Splinter paper. The F1 score was around 70 for SQuAD, and 50 for naturalquestions when training on 512 examples.
We believe that the small difference in the score between vanilla Splinter and our generative implementation is partly due to different phrasing of the answer, e.g. predicting "two" instead of "2" or detailing gaps as seen in table 2.

Model	**SQuAD**	**NQ**
0 examples		
Baseline	0	0
16 examples		
Baseline	47.98	30.24
128 examples		
Baseline	62.2	43.47
512 examples		
Baseline	69.38	50.38

Table 2: Average F1 score of the baseline model after finetuning in each setting.

[4] https://github.com/mrqa/MRQA-Shared-Task-2019

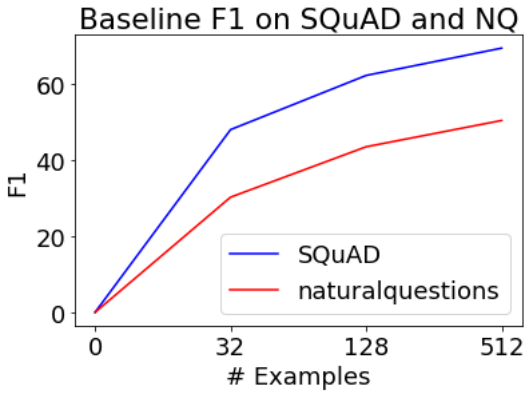

Figure 1: Plot of the average F1 score of the baseline model after finetuning in each setting.

5 Analysis

5.1 Pretraining

We note possible reasons for non-convergence of the pretraining:

- Training duration - the resources defined for this project allowed us to reach only $700k$ steps on single sequences (batches of size 1). Much less than Splinter which was trained for $2.4M$ training steps with batches of 256 sequences.
- Train loop implementation - our training loop was tested with finetuning and over-fitting on a small dataset with success (F1 maximum score of 100).
- Dataset - our dataset was inspected visually by us, and tested on the model. We did not try many different hyperparameters for the dataset generation (probabilities, length, etc.) as it was very time consuming. In addition, due to memory limitations, our model could only handle short sequences with up to 256 tokens.
- Hyperparameters - We trained our model using the hyperparameters suggested in T5 paper. In addition, we also performed a small search on important parameters such as batch size, learning rate, scheduling, optimizers. A possible difference between our training and T5's could be related to the scheduling rates, and the different dataset.
- Initialization - We wrote our train loop with PyTorch, since it provided us with freedom in the implementation. This means we did not use the Trainer[5] class from the Huggingface library, and therefore our model random initialization could be sub-optimal.

5.2 Results

We analyze the outputs of the finetuned model in order to inspect potential caveats. Most errors were one of the following:

- One of many possible answers - in our evaluation script we took into consideration only the first available answer labels. When evaluating a question answering model with F1, one should compare the generated text with all possible answers and sentence phrasing.
- Wrong answers - A small set of answers were incorrect. This raises questions on what exactly did our model learn: Did it learn to understand natural language? Or does it pick any date in the context when asked upon a time period of an event? i.e. the model might be falling to simple heuristics, as oppose to doing complex thinking.

It is worth noting that our model achieved a worse score on the naturalquestions dataset, than on SQuAD. When we examined the dataset we noticed that some question in the naturalquestion dataset, have a more structured and complex answer. The answer is constructed from multiple parts of the context with conjunction words in between. See example at table 4.

6 Related Work

This project is based on *Splinter* (Ram et al., 2021). We replace the QASS layer with a decoder in order to produce answer tokens, instead of predicting the spans. In order to do that we reformulate the pretraining task, by using a different sentinal token for each masked span (and not a uniform [QUESTION] token).

[5] https://huggingface.co/transformers/main_classes/trainer.html

7 Appendix A: Pretraining Learning Graphs

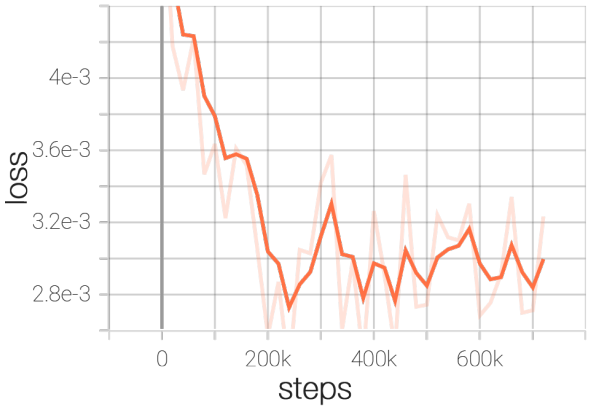

Figure 2: Pretraining train loss (scaled by 1/1024).

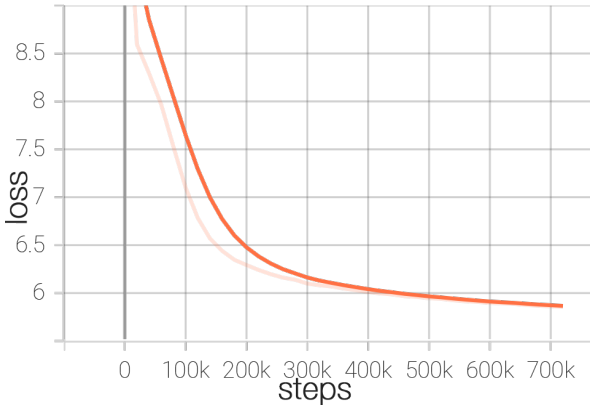

Figure 3: Pretraining evaluation loss.

8 Appendix B: Finetuning Results

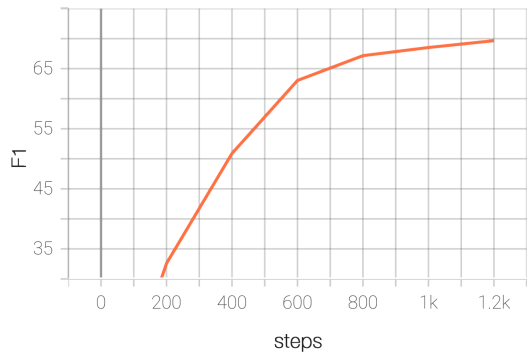

Figure 4: F1 score for the SQuAD dataset when training on 512 examples.

9 Appendix C: Generated Answers

Question: What is the AFC short for? **Label**: American Football Conference **Generated Answer**: American Football Conference
Question: What day was the game played on? **Label**: February 7, 2016 **Generated Answer**: February 7, 2016
Question: Who was first on the team in total tackles? **Label**: Linebacker Brandon Marshall **Generated Answer**: Brandon Marshall

Table 3: Examples of generated answers on SQuAD (Rajpurkar et al., 2016) test dataset.

Question: nba record for most double doubles in a season **Label**: Tim Duncan leads the National Basketball Association (NBA) in the points - rebounds combination with 840 **Generated Answer**: Tim Duncan
Question: who was the war of 1812 fought between **Label**: the United States, the United Kingdom, and their respective allies **Generated Answer**: the United States, the United Kingdom, and their respective allies

Table 4: Examples of generated answers on naturalquestions (Kwiatkowski et al., 2019) test dataset. We can see that the model prefers to generate short spans, and not long sentences.

References

Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. 2017. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5376–5384.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. 2019. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer.

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

Ori Ram, Yuval Kirstain, Jonathan Berant, Amir Globerson, and Omer Levy. 2021. Few-shot question answering by pretraining span selection. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3066–3079, Online. Association for Computational Linguistics.