Training Question Answering Models from Synthetic Data (Research Paper Summary)

Can a Q/A model trained on Synthetic Data beat SOTA??

Prakhar Mishra May 28, 2021 · 5 min read ⋆

Image from Source

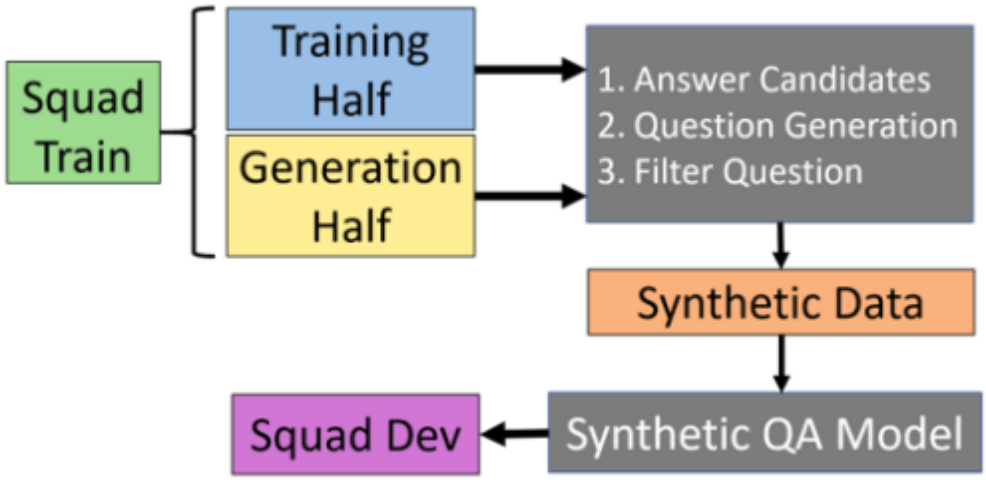
same!

Problem Statement

This research focuses on improving question answering models by generating synthetic questions and answers when dealing with the limited amount of human-annotated data. Author's achieve better performance on SQUAD1.1 question answering task solely with synthetic data compared to human-annotated questions from the training set.

Proposed Method

The authors propose a *3-step pipeline* comprising of **un-conditional answer extraction**, **question generation** and **question filtration**. The data that they use for training all three components in the pipeline is part of SQUAD's train split and use the other half to generate synthetic data from the trained components. Once the synthetic data is generated they train the Q/A model and test its performance on SQUAD's development set. *Below fig. show the entire pipeline —*



Synthetic QA model pipeline flow | Image from Source

Answer Generation — a' ~ p(a|c)

Under this step, they propose a BERT model that learns to extract answer spans over a given context. They train this model by just observing context and not the question tokens *(which is a little unlikely to how typical q/a models are trained)*. This way they learn a prior distribution of the answers in the dataset. As discussed above, they use (context and answer) pairs from SQUAD's train split for training this model.

that models start and end tokens independently. *Mathematically it can be represented as* —

$$p(a|c;\theta_A) = \frac{e^{f(a,c;\theta_A)}}{\sum_{a''} e^{f(a'',c;\theta_A)}}$$

$$f(a,c;\theta_A) = \text{MLP}(\text{CONCAT}(\text{BERT}(c)[s], \text{BERT}(c)[e]))$$

Answer generation model

Here, a=(s, e) and s, e, c are start tokens, end tokens and context tokens respectively. Firstly, we pass context through the BERT model and grab the start and end token embeddings at the output. Both the embedding representations are then concatenated and passed to a multi-layered perceptron model followed by softmax over possible start and end spans. The aim is to have this softmax probability tilt more towards 1 for actual ground truth spans and diminishing to 0 for all other spans. So, the better the skewness, the better the model.

Question Generation — q' ~ p(q|a', c)

As the part of next steps, they train a question generation model using the pre-trained GPT-2 language model. They concatenate context tokens, answer tokens, and question tokens into a large single sequence, separated by the end of sequence tokens. They also prepend and append ***:question:*** token near the question as question markers. Also to make sure the model knows different input segments, they define three types of segment embeddings one for each data type. Additionally, they also infuse answer segment embeddings to context embeddings helping the model to locate answer span in the context. *As can be seen in the below figure—*

During inference, they provide context and answer tokens followed by ***:question:*** as the trigger word and sample till the next occurrence of the same trigger word.

Roundtrip Question Filtration — a'?a* ~ p(a|c, q')

The idea of Roundtrip filtration is built on the motivation of over-generating and then filtering things, wherein, the goal is to validate and select the final question from the candidate set. So for this, we first train a question answering model p(a|c, q) on the labelled data from SQUAD, here a, c, q are the answer, context and question respectively. Then during inference, we pass in generated question along with context to generate candidate answer. And based on if the candidate answer matches the generated answer*(step-1),* the question is selected or rejected.

Inference Flow

After the training phase is done, we proceed with the inference phase, wherein, let's say we have a new context now, we first pass this through the answer generation model that generates an answer. We use the generated answer and the pre-defined context and pass it to the Question generation model. Once we have candidate question and answer, we use context and generated question to get possible candidate answer. If the answer that we generate as a part of this step matches the generated answer*(part of step 1)* we consider the generated question to be valid whereas all others were rejected. *Below fig. shows the entire inference flow —*

Inference Flow | Image by Author

My thoughts

I found this to be a pretty interesting read and I feel that the idea of training the Q/A model on synthetic data not only a good step to be taken in case of less data but can also be used as a part of the pre-training step followed domain-specific Q/A fine-tuning. Also, it would be a good experiment to try out encoder-decoder models (like T5, etc) for the question generation system rather than just using the decoder only model.

> If you wish you can also checkout other research paper summaries that i have written.

So yeah, that's it for this blog. If you like consuming video content over textual (just like me :D), make sure to check out —

Watch more videos like these

Feel free to read the entire paper and say "**Hi**" to the authors and appreciate their contribution.

> **Paper Title:** *Training Question Answering Models from Synthetic Data*

> **Authors:** *Raul Puri, Ryan Spring, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro*

Also, in case you enjoyed reading this article, you can choose to ***buy me a "chai"*** *on https://www.buymeacoffee.com/TechvizCoffee — because I don't actually drink coffee :) Thank you very much! It's totally optional and voluntary :)*

Question Answering Machine Learning Data Augmentation NLP Research