ASKiT: Search and Ask Model

Stanford CS224N {Custom}

Adam Klein and **Matthew Villescas**
Department of Computer Science
Stanford University
`{aklein4,mattjv22}@stanford.edu`
Mentor: Yuan Gao

Abstract

Often in life, the answers to our questions evade us not because the answers aren't there but because we're not asking the right questions. To this end, We seek to construct a model for the evidence retrieval portion of the multi-hop question-answering (QA) task. Our model takes a question, searches an open corpus for information to answer it, and–if unsatisfied–produces a new sub-question whose answer will help solve the original, repeating until all the necessary information is found. In this way, we provide a novel framework that is analogous to that of a lawyer asking questions in order to answer them: That is, using the inverse of the original QA task to solve the task itself. In practice, the model used for answering questions and the model used for asking questions both have their own respective merits; however, when combined, their performance left much to be desired, but the divorced results imply that a few modifications in strategy are in order for an ideal question answering model that plays the role of cross-examiner. And perhaps that in a different context with enough training time, the initial idea will prove fruitful.

1 Introduction

Question-Answering is a very broad task. It is often impossible for a model to simply "know" the answer to every question you tell it; in many formulations, a context is provided for a model to search for an answer. However, this betrays the spirit of question answering in a colloquial sense since the answers are already readily available to us in a context we know of. This, funnily enough, begs the question: how do we question-answer if the machine can't "know" the answer to every question and isn't provided with a specific context? A model that has a large knowledge base (such as Wikipedia) "knows" the answers to many questions, but it needs to how to search. This lends itself to a new formulation for question answering.

1.1 Multi-hop Question Answering

In the multi-hop question-answering task, the model is given a question, and it must search through a large open corpus to find evidence that can be used to answer that question. Multi-hop reasoning questions include Type I questions where an intermediate entity is requires to find the answer, and Type II questions where information must be retrieved about 2 separate entities in order to compare them, and Type III questions where an intermediate entity is required to infer information about the original subject. Yang et al. (2018)

While multi-hop question-answering usually involves a step in which the model returns an answer to the users question, we decided to focus on the other portion of the task: collecting evidence. Here, we do not return a final answer string. Instead, our model acts more like a search engine that returns all of the evidence. Many times the evidence is sufficient to answer the question; however, it is very possible that the answer does not appear. That being said, the evidence is relevant to the question, otherwise it wouldn't have been produced. We theorize that details are there to ask another, better question in hopes of finding a sufficient answer. This motivates constructing a model to generate new questions based on the evidence.

1.2 Question Approximation

The question generation (QG) task is usually defined as the inverse of the standard QA task. Given an answer-containing context, the traditional task is to produce a question that would fit the context. However, this formulation doesn't quite fit the goals of our model; if the model already had the answer, then it wouldn't need to generate another question. We, therefore, modify the task to be more open-ended: in particular, given some evidence that's related to the answer, we want the model to produce a sensible question that it could then recursively answer with the hopes of answering the original question. This open-ended approach introduces a new problem though.

If the evidence was "George Washington was the first president of the United States," then the model could generate one of multiple valid questions, such as "Who was the first president of the United States?", "Who was George Washington?", "George Washington was the president of what country?", etc. More concretely, it's impossible to construct a perfect question inverter–that is, a model can't perfectly predict the original question from relevant evidence. This seems like a limitation, but in our context, this is a strength. Our model would already know the original question; we're aiming for it to generate a different but related question. The guiding assumption is then that a related question is an approximation of the original question: A question that is close but not quite the same. This motivates our question approximation task (QAp): Using evidence generated to answer an original question, produce the original question. Since it's impossible to perfectly produce the original question, the resultant question would be its question approximation.

1.3 Sub-Question Generation

Question approximation, however, is not enough. Ideally, the model would utilize the original question and its evidence to inform a new, more useful question. The sub-question generation task (SQG) takes in, as input, the original question and evidence and produces a different but related question. Sub-question generation mimics the role of a cross-examiner, stringing past questions and responses to ask more intimately related but different questions.

2 Related Work

Multi-hop question answering has seen success from many different approaches. An early method was to use supervised learning to train recurrent neural networks to identify which evidence statements belonged to the question Yang et al. (2018). Heuristic-based methods have also been successful in chaining evidence statements together through shared entities et al. (2021). Now, transformers have become the standard, and are able to retrieve evidence from all of wikipedia with up to 67% exact match accuracy Zhu et al. (2021).

Question generation is a developing field with many approaches. Lopez et al. (2020) details a transformer-based model that can construct various appropriate questions from a context paragraph without the use of answer metadata. Our question approximation and especially sub-question generation tasks have this spirit with the evidence acting as the context paragraph, however, we want our questions to be informed by previous questions that we have and know to exist. Further, our questions have the goal of explicitly leading to some unknown answer of a known question, making our variation of question generation merely a subtask of a more robust question answering machine.

3 Approach

3.1 Multi-hop Question Answering

We decided to treat multi-hop question answering as a reinforcement learning problem. In this framework, we define the notion of states, actions, and rewards. Our 'state' is defined as the question we are trying to answer, as well as the evidence that we have already collected. An 'action' is defined as adding a new evidence statement to our state, or deciding that we have enough evidence already and stopping the search. The reward is given at the end of the search, and is higher when the evidence is more correct.

Running the entire corpus through a model at every step of the search is computationally infeasible, so we needed a way to find relevant statements from the corpus without actually reading all of them every time. Therefore, we broke our retrieval model into 2 components: course retrieval and fine retrieval Yin et al. (2022). The course retrieval component quickly searches over the corpus, and narrows down a small set of candidate statements that it believes to be good choices The fine retrieval model then chooses its action from those candidates.

3.1.1 Course Retrieval

For course retrieval, we started with the bert-based "sentence-transformers/multi-qa-MiniLM-L6-cos-v1" model, which was pretrained for the semantic search task. This model encodes each statement from the corpus into a 384 dimensional vector. At search time, the question is also encoded into a vector, and statements from the corpus are evaluated in relevance using the dot-product similarity between the statement vector and the query vector.

Inspired by single-hop information retrieval Lee et al. (2019), we fine-tuned this model on multi-hop reasoning using the labelled evidence statements in the HotpotQA dataset Yang et al. (2018). At each training step, we generated a set of example states that the model may encounter at search time. Each example state was composed of the question, as well as an incomplete subset of the target evidence set, with the question seperated from the evidence using the <sep> token. To increase robustness, some of the example states were also given incorrect 'noise' statements in the evidence. We then trained the model using the cross-entropy loss between the predicted dot-product similarities of the state encoding and the corpus encodings, with the missing correct evidence given 1 labels and the rest of the corpus with 0. It should also be noted that the corpus encodings did not change during training, with each question's corpus was pre-encoded before training using the pretrained model. This was done to prevent the encoder from overfitting the corpus encodings to their respective questions.

3.1.2 Fine Retrieval

The fine-retrieval model takes in the candidates produces by course evidence retrieval and decides whether to take one of them as the next piece of evidence, or submit the current evidence as the final answer. To handle this decision, we combined the predictions of 2 separate models: one to evaluate the candidates, and one to evaluate the current state for submission. For evidence evaluation, we use bert-mini Turc et al. (2019); Bhargava et al. (2021), which pretrained on the next sentence prediction task. We used this model because the 'next evidence' prediction task is conceptually similar to next-sentence prediction. To decide when to submit the current state, we used "deepset/tinyroberta-squad2", which was pretrained on the SQUADv2 Rajpurkar et al. (2018) to predict whether a question is answerable based on the context. In order to extract a reinforcement policy, we feed each possible action through its corresponding model to get an evalutation logit. The logits are then passed through the softmax to get the probabilistic policy $\pi(a|s)$ that describes the probability of taking each action given the current state.

Since the rewards in the reinforcement environment are so sparse (very few of the sentences in the corpus are actually relevant to a given question), it would be difficult to train a policy from scratch, even with the pretrained models. Therefore, we first trained our retrieval model using supervised learning on the HotpotQA datasetYang et al. (2018), which has labelled evidence. Treating the labelled evidence as the 'correct actions' set C, we minimized the loss function below. This loss was chosen because it is analogous to the REINFORCE policy gradient algorithm Sutton et al. (1999), where choosing a correct action yields a reward of 1 and an incorrect action yields 0.

$$\mathrm{L}_{supervised} = -\textstyle\sum_{a \in C} log\pi(a|s)$$

For for reinforcement learning fine-tuning, we used the Proximal Policy Optimization (PPO) policy gradient algorithm, with the loss described below. The reason that we are using this algorithm compared to an off-policy one like Q-learning is that state value functions might be difficult to predict when operating on an open corpus. For our rewards, we are using f1-smoothed Yin et al. (2022) discounted terminal rewards based on how accurately the retrieved evidence matches the labelled ones from HotPotQA, with a +1 bonus for an exact match. This reward method was chosen as a balance between optimizing for exact matches, while also incrementally improving on incorrect outputs. Instead of calculating the baseline value $b(s)$ at each state using a separate network as is usually done with PPO, we sampled a set of possible trajectories stemming from that state, and estimated the baseline value as the expectation over those trajectories. This was done due to the afformentioned difficulty in existimating state values in an corpus.

$$\mathrm{L}_{PPO} = -\sum_{t=0}^{T} min\left[r_t A_t, clip(r_t, 1+\epsilon, 1-\epsilon)A_t\right]$$
$$\mathrm{A}_t = R_t - b(s_t),\ r_t = \frac{\pi_{new}(a_t|s_t)}{\pi_{old}(a_t|s_t)}$$

In an attempt to further improve model performance, we leverage the fact that after PPO fine-tuning, we have maximized the expected reward from sampling a trajectory τ (a sequence of actions that takes the state from only the question to the solution). Therefore, we would expect that more rewarding trajectories would be more likely to be sampled by our learned policy. This hypothesis is confirmed in our experiment section where we show that trajectory probability and reward are positively correlated. In order to get more likely trajectories than would be selected by either sampling from our policy or selecting actions greedily, we implemented a beam search over different trajectories. Here, we sample multiple trajectories at once, keeping track of the log probability, $p(\tau)$, of each which is updated using the equation below. Our beams are the trajectories that have the highest log probability at each step.

$$\mathrm{p}(\tau) = \sum_{t=0}^{T} \log(\pi(a_t|s_t))$$

3.2 The T5 Model

The Text-to-Text Transfer Transformer (T5) model, first introduced by Raffel et al. (2020), is an encoder-decoder model pretrained on a large variety of unsupervised and supervised text-to-text tasks. It's very versatile, performing well on many tasks such as summarize, traditional question-answer, and translation just by prepending the task at the beginning of the input string. In the supervised setting, it trains by corrupting tokens and teacher forcing. This is an incredibly robust model architecture that was used as the main basis for all of the question generation tasks.

3.3 Question Approximation Model

For our question-generation model, we wanted to start with a pretrained model that was already very used to question generation in the traditional setting of having access to an answer and a context. The idea was that input evidence essentially acted as both the context and the answer since it was the agent model's best guess at a solution to the original question. We thus trained a T5 model specialized in Conditional Generation by Simonini (2021), which was originally finetuned on a modified version of the SQuAD dataset Rajpurkar et al. (2018) that made the answers the input and questions the target. We adapted question-generation training scripts by Suraj (2022) and Simonini (2021) to further finetune Simimoni's model to take in evidence the agent produced to answer a HotPot question and use it to reproduce the original question, thus yielding an approximation of the question. This model was not used at inference time; instead, it was used to generate a dataset for the sub-question generation model.

3.4 Sub-Question Generation Model

Using our question approximation model, we generated several approximations for HotPotQA questions with varying amounts of evidence produced by the agent model. For our question-refinement model, we once again trained Simimoni's model to generate the approximated questions, using the original question and its corresponding evidence as input. The training script was a slightly modified version of the script used for the question approximation model. This model thus create questions while being informed by the evidence and the original question.

4 Experiments

4.1 Data

4.1.1 HotPotQA Dataset

The HotPotQA dataset is unique as its over 113k Wikipedia based question-answer pairs boast four desirable features: questions searching and reasoning over several documents; questions are not constrained to pre-existing knowledge bases; there are sentence-level supporting facts for reasoning, which allow strong supervision and explanations for predictions; and comparison questions that allow models to extract relevant facts in order to compare. The dataset thus provides questions complex enough to allow for a multiple-question strategy to be viable, has been empirically proven to be difficult for state-of-the-art QA models, and provides enough data for our model to reason with the additional questions that it asks. Yang et al. (2018)

4.1.2 Question Approximation Dataset

The goal was to approximate the original question based off the evidence the agent model found to try and answer it. We thus constructed a new dataset with target questions deriving from HotPotQA with corresponding evidence generated from the agent model for each target question. A datapoint would be $(e_1 \dots e_n, Q)$ for the n pieces of evidence the agent model generated based on the original question Q. The task is to work backwards and try to generate the precise original question with just the evidence, yielding a question approximation.

4.1.3 Sub-Question Generation Dataset

Using our Question Approximation Dataset, we generated new questions with our question approximation model taking, as input, subsets of the full evidence. That is, for question Q with agent model-generated evidence pieces $e_1 e_2 \dots e_n$ we used our question approximation model to generate new questions $Q'_1, \dots, Q'_n$ where Q'_1 was generated from evidence e_1, Q'_2 was generated from evidence e_1 and e_2, up to Q'_n which was generated with evidence $e_1, \dots e_n$. Each of these new questions would be the target for a respective datapoint containing the original question Q as well as the evidence used to generate it. In the above example, we'd have n data points: $(Q, e_1, Q'_1), (Q, e_1 e_2, Q'_2), \dots, (Q, e_1 \dots e_n, Q'_n)$. The idea behind this was that more or less of the evidence may or may not be helpful for question approximation. This was repeated with every original question. The task would be to use the original question Q and evidence $e_1 \dots e_k$ (for $k \leq n$) provided to predict the approximate question Q'_k.

4.2 Evaluation method

4.2.1 Evidence Retrieval Evaluation

To evaluate our evidence retrieval model, use the metrics that are compared on the HotPotQA leaderboard. These metrics are the exact match percentage (EM) and F1 score for the returned evidence set, compared to the labelled gold evidence from the dataset. As a matter of logistics, we used the 'distractor' dev set as our test set, and seperated a small validation set out of the training set.

For a baseline comparison, we turned to the original model published by the creators of the dataset Yang et al. (2018). This model achieved an EM of 20.32 and F1 of 64.49.

4.2.2 QAp and SQG Evaluation

It was unclear to us how to quantitatively evaluate both the QAp and SQG tasks. For QAp, we opted to linguistically compare many of the question approximations to the original question, determining how similar/helpful of question it was. For SQG, this could also be done with the training data, but since it's ultimately a subcomponent of the question-answering process, the evaluation of generated questions could be determined by how much use the agent model had for it in future RL experiments.

4.3 Experimental details

Training graphs from our experiments can be seen in the appendix.

4.3.1 QAp and SQG Details

The question approximation model was trained on 60,000 HotPotQA questions with corresponding evidence generated by the agent model for 7 epochs. The learning rate was 0.0001 with the AdamW optimizer. The sub-question generation model was trained with the same learning rate, number of epochs, and optimizer over 40,000 datapoints from its respective generated dataset. For token embeddings, both models used the default T5 base encodings.

4.3.2 Course Retrieval

To fine-tune our course retrieval model, we trained for 60,000 steps on the labelled HotPotQA evidence questions. We used a batch size of 24, a learning rate of 1e-6, and the AdamW optimizer. We also used a half-cosine learning rate schedular with a 10,000 step linear warm-up followed by 50,000 steps in which the learning rate decayed back to zero as a half-cosine wave. At the end of training, our validation loss reached -0.456. In the validation set, our highest rated statement was gold evidence on 75.9% of questions, and the best-rated gold evidence was on average in the 99.2 percentile of all statements.

4.3.3 Fine Retrieval

To train our course retrieval model, we trained for 60,000 steps on the labelled HotPotQA evidence questions. We used a batch size of 24, a learning rate of 1e-6, and the AdamW optimizer. We also used a half-cosine learning rate schedular with a 10,000 step linear warm-up followed by 50,000 steps in which the learning rate decayed back to zero as a half-cosine wave. At the end of training, our validation loss reached -0.456. In the validation set, our highest rated statement was gold evidence on 75.9% of questions, and the best-rated gold evidence was on average in the 99.2 percentile of all statements.

For PPO fine-tuning, we sampled and trained on 10,000 trajectories. We used a learning rate of 1e-6, batch size of 6, clpping epsilon of 0.2, and the model was given 6 actions to choose from by the course-retrieval model.

4.4 Trajectory-Reward Correlation

As described earlier, we wanted to confirm that trajectory probabilities are positively correlated to their rewards. To quantify this correlation, we measured the Spearman Rank Correlation Coefficient ($\rho_{spearman}$) between the reward of a trajectory and its probability. This value ranges from -1 to 1, with a positive value indicating positive correlation.

$$\rho_{spearman} = \frac{cov(R(p(\tau)), R(r_\tau))}{cov(R(p(\tau)))cov(R(r_\tau))}$$

To calculate it, we sampled 10 trajectories each for 1,000 different questions from our validation set, and rank each trajectory in relation to the others for its question. This gave us $\rho_{spearman}(r, p(\tau)) = 0.42$, which is a reasonably positive correlation.

4.5 Results

4.5.1 Information Retrieval

	EM	F1
Baseline	20.3	64.49
Supervised	30.2	63.9
PPO	33.9	67.4
PPO+beam	**34.4**	**69.0**
PPO+sub-questions	0.0	40.8

As seen in the table above, our model outperformed the baseline in both metrics. It can also be seen that each step of our method improves the model, with PPO fine-tuning outperforming supervised training, and beam search with a beam size of 4 outperforming greedy action selection. We also see that using our sub-question generator to improve the question greatly degrades performance.

4.5.2 Question Approximation Examples

- **Original Question** What rock band formed in New York in 1981 worked with Martin Bisi?
- **Evidence** Sonic Youth, Founding members Thurston Moore (guitar, vocals), Kim Gordon (bass guitar, vocals, guitar) and Lee Ranaldo (guitar, vocals) remained together for the entire history of the band, while Steve Shelley (drums) followed a series of short-term drummers in 1985, and rounded out the core line-up. Sonic Youth, Part of the first wave of American noise rock groups, the band carried out their interpretation of the hardcore punk ethos throughout the evolving American underground that focused more on the DIY ethic of the genre rather than its specific sound. Sonic Youth, In their early career Sonic Youth were associated with the no wave art and music scene in New York City.
- **Approximated Question** Who was a founding member of the band that was associated with the no wave art and music scene in New York City?

- **Original Question** Which city has a larger population, Yangzhou or Chengdu??
- **Evidence** Chengdu, At the time of the 2010 census, Chengdu was the 5th-most populous agglomeration in China, with 10,484,996 inhabitants in the built-up area including Xinjin County and Deyang's Guanghan City. Yangzhou, Its population was 4,414,681 at the 2010 census and its urban area is home to 2,146,980 inhabitants, including three urban districts, currently in the agglomeration.
- **Approximated Question** What is the population of the 5th-most populous agglomeration in China, Chengdu or Yangzhou?

4.5.3 Sub-Question Generation Example

- **Original Question** Which school allows students to cross-register at Brandeis University, Rutgers University or Wellesley College?
- **Evidence** Wellesley College, The college is also known for allowing its students to cross-register at Massachusetts Institute of Technology, Brandeis University, Babson College and Franklin W. Olin College of Engineering. Wellesley College, Wellesley College is a private women's liberal arts college located west of Boston in the town of Wellesley, Massachusetts, United States.
- **Sub-Question** What is the name of the private women's liberal arts college located west of Boston in the town of Wellesley, Massachusetts?

- **Original Question** What synthetic stimulant is sometimes called flaka or gravel?
- **Evidence** Alpha-Pyrrolidinopentiophenone, Colloquially it is sometimes called flakka or gravel.
- **Sub-Question** Alpha-Pyrrolidinopentiophenone is a synthetic stimulant, it is also known as what?

5 Analysis

5.1 Multi-hop Results

Our reinforcement-based evidence retrieval model showed promising results. It was able to achieve relatively impressive scores when it solely answered without employing the question asking mechanisms. However, the improves of the PPO training were a bit disappointing; they improved the exact match score slightly, but the F1 score remained relatively unchanged, and with a stronger implementation of the question asking system, having a higher exact match score wouldn't be as significant. We believe that this lack of performance gain can be contributed to the lack of training time dedicated to PPO - while it trained for 48 hours, sampling trajectories is very slow. It is likely that if we were to resume training, we would continue to see improvement. Furthermore, in many of the cases that the model got wrong, it seemed to go down the 'wrong path', finding one bad piece of evidence and then finding others that were related to it. This shows a need to improve robustness, which may come with more training.

5.2 QAp and SQG Results

The question approximation and sub-question generation results are, unsurprisingly, a mixed bag.

For question approximation, sometimes the model produces very faithful recreations at the original questions (a good approximation). Sometimes it asks somewhat non-senseical questions such as the second QAp question example. However, even in these non-sensical questions, there's still information semantic search may make use of. Sometimes QAp produces very off questions; when both a date and a name are in evidence, it often defaults to asking when they were born. This probably because those two appear very commonly in such questions.

For SQG, it sometimes introduces very interesting bridge questions such as in both examples. The structure of these questions are very interesting because they would help answer the original quesiton, but they are redundant since many times th evidence is already sufficient to answer the original question. This means that when SQG messes up and asks a question (it also has a taste for asking birthdays when there's a date and name in the evidence–most likely a remenant of the training data), it would lead the model astray rather than being helpful.

5.2.1 Synthesized Results

A run of the agent model with the sub-question generator model yielded rather disastrous results, dropping the scores significantly. This is likely due to a naive addition of the mechanism, and the model probably needs more time reinforcing with question asking in order to better learn how to utilize it. Further, there were several issues with the sub-question generating. Many times, the answers were already in the evidence, making the SQG model more destructive than anything else. We think this is due to the fact that HotPotQA has very specific, well-defined questions that are relatively easy to search for. Perhaps the question-asking strategy is best used for vaguer or "worse formulated questions. This might call for the use of a dataset such as Natural Questions or more diverse data overall.

6 Conclusion

The multi-hop question answering showed some promising results divorced from the SQG model, while the SQG model learned how to ask different, related, and sometimes interesting question. The issue is that these were often times redundant, making it more harmful than helpful. Our evidence retrieval model showed promising early results, but would definitely benefit from further training. We also realize that the incorporation of a true question-answering component to our framework could make the model more useful, and could be leveraged to improve evidence retrieval also. Interestingly, this could also allow us to open our reinforcement framework up to training on datasets that don't contain labelled evidence sets, instead giving rewards based on whether the correct answer was given - as is often used in single-hop question answering Lee et al. (2019).

Finally, we believe that performance of our combined information retrieval and sub-question generation system would benefit greatly from joint training. Therefore, the most pertinant continuation of this work would be to incorporate the sub-question generator into the PPO training loop, so that it learns to ask questions that are directly beneficial for search.

References

Jifan Chen et al. 2021. Multi-hop question answering via reasoning chains.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in nli: Ways (not) to go beyond simple heuristics.

Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. 2019. Latent retrieval for weakly supervised open domain question answering.

Luis Enrico Lopez, Diane Kathryn Cruz, Jan Christian Blaise Cruz, and Charibeth Cheng. 2020. Transformer-based end-to-end question generation. *CoRR*, abs/2005.01107.

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, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

Simonini. 2021. t5-end2end-question-generation.

Suraj. 2022. question-generation.

Richard S Sutton, David McAllester, Satinder Singh, and Yishay Mansour. 1999. Policy gradient methods for reinforcement learning with function approximation. In *Advances in Neural Information Processing Systems*, volume 12. MIT Press.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. 2018. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600.

Zhangyue Yin, Yuxin Wang, Yiguang Wu, Hang Yan, Xiannian Hu, Xinyu Zhang, Zhao Cao, Xuanjing Huang, and Xipeng Qiu. 2022. Rethinking label smoothing on multi-hop question answering.

Yunchang Zhu, Liang Pang, Yanyan Lan, Huawei Shen, and Xueqi Cheng. 2021. Adaptive information seeking for open-domain question answering.

A Appendix

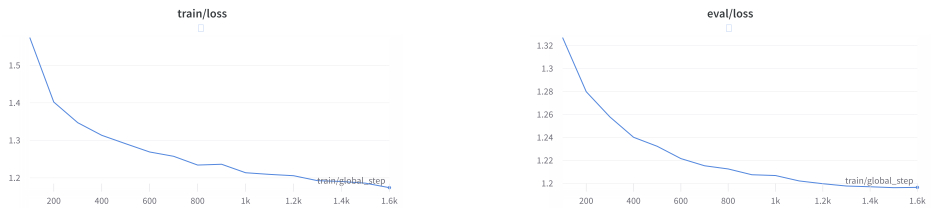

Figure 1: Train and Val Loss for QAp

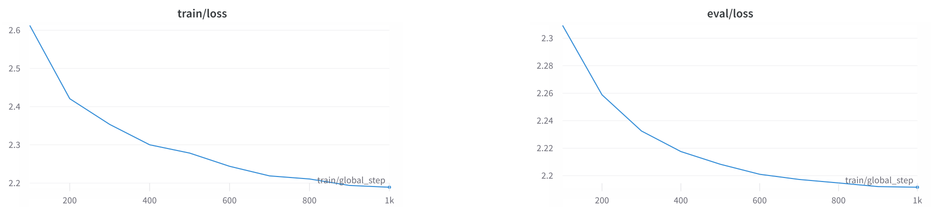

Figure 2: Train and Val Loss for SQG

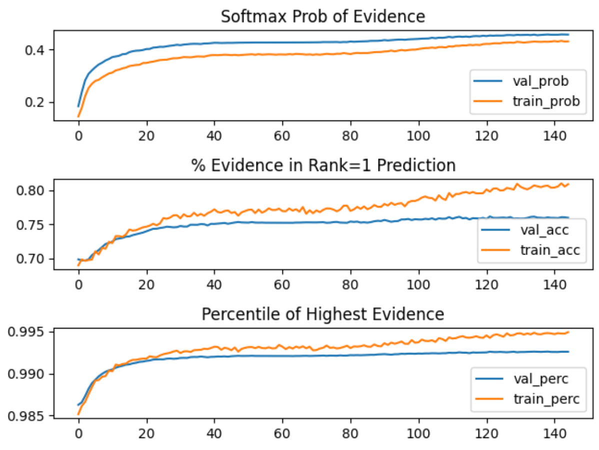

Figure 3: Course retrieval performance over Epochs

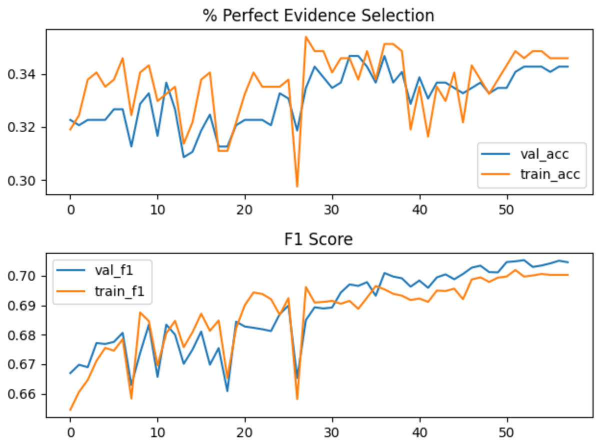

Figure 4: PPO fine-tuning performance over epochs.

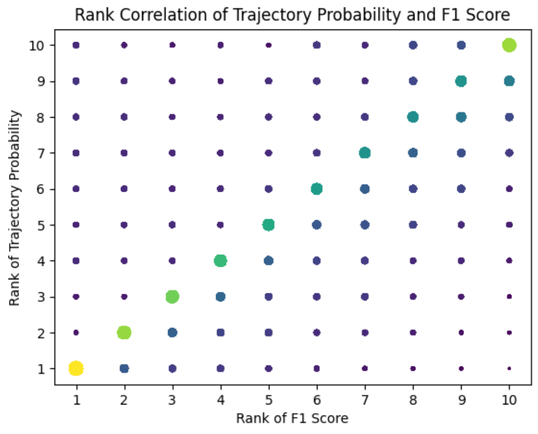

Figure 5: Relation between F1 and trajectory probability ranks. Larger dot and hotter color indicate higher frequency