Generating Good In-Context Samples for Conversational Question Answering with GPT-3

NLP class, April 2022

Ilan Motiei (212649701), Jan Jirman (913082368)

Abstract

We investigated the ability of neural models to understand and answer questions with respect to some context, i.e., with respect to previously asked questions and their answers about a given passage. Then we used the learnt model to feed the bigger GPT-3 model (350M parameters) with in-context samples from the training set when queried about a new one. We propose a method for inferencing with GPT-3 in the conversational question answering task in a single-shot manner, and show that the best results are obtained mostly when using the model that was fine-tuned for that task. Our code is made publicly available at: https://github.com/ilanmotiei/One-Shot-CQA-with-GPT3

Introduction

Conversational question answering is a challenging task as it requires the model not only to understand the current question and the given passage it relates to, but it also requires the model to pay attention to the previous asked questions and the answers it gave on them.

In this work we investigate several ways to inference with GPT-3 (introduced in [1]) for that task. Specifically, we train a model that is fine-tuned specifically for the task of conversational question answering and compare its embeddings to the embeddings of a simple, pretrained (and not finetuned) BERT, while combined with the KATE algorithm introduced in [2], for generating the in-context examples for GPT-3 to learn from.
We also compare those methods to 2 other methods. The first is randomly choosing an example from the training data as the example for GPT-3 to learn from, while the second is similar, but at this time we randomly choose an example from the same source as the queried sample, while a source can be one of the 5 sources in the CoQA dataset we've used in our experiments, which are news, children's stories, Wikipedia, middle and high school English exams, and literature from Project Gutenberg.

Related Work

In 2020 an auto-regressive language model called GPT-3 was introduced [1]. It was trained on a very large corpus of data from diverse sources from the internet, including Wikipedia and Reddit. It has shown promising abilities to perform a task by specifying it a command (also called zero-shot), which improve when also given a few examples to learn from (called few-shot learning), as the context. In our work, we focus on the single-shot technique (where only one example is given a context), as will be explained soon.

Since then, works such as [2], explored how the model's accuracy improves with different algorithms for choosing the context for a given sample. The methods are basic - first choose a smaller model that given a query, gives you a representing embedding vector for it (such as BERT). Then, encode all the samples in your training data, and when queried about a new sample (i.e. from the test set), encode it, and take some amount of samples from the training set whose embeddings have the highest similarity with that of the queried one, according to some similarity metric (e.g. cosine-similarity), and use them as the context to the GPT-3 model.

Specifically, [2] have shown that using embeddings created by models that were fine-tuned for the same task as the tested task is always superior to using models that were fine-tuned for another tasks.

In this paper, we examine the conversational QA abilities of the GPT-3 model, guided by in-context samples chosen from another "small" model we've trained specifically for that task, compare it with other methods for choosing those samples, and show its superiority among them.

Dataset

We've used the CoQA dataset, introduced in [4] by Stanford. It contains stories with questions and answers regarding them, each taken by a conversation between 2 people, asking short questions while relating to previously asked ones and their answers.
Each answer can be a span of the story, or simply 'yes', 'no' or 'unknown'.

The dataset differs from other QA datasets not only by its conversational style, but also by significant lengths of the passages in it, and small length of the questions and answers in it, which simulate a real-world scenario, as humans tend to ask conversational and short questions.

Another attribute of this dataset is the diversity of its questions. A comparison between it and the SQUAD dataset is shown in table 1 and in figure 6 at the appendix for illustrating that fact.

Each bunch of story + questions + answers, in the dataset, is called a 'challenge'. The training set contains 7199 challenges, and the test set contains 500 challenges. The challenges come from 5 sources - Wikipedia, children's stories, news, middle and high school English exams, and literature from Project Gutenberg.
Each challenge includes 15 questions and answers in average.

	SQuAD	CoQA
Passage Length	117	271
Question Length	10.1	5.5
Answer Length	3.2	2.7

Table 1: Average number of words in passage, question and answer in SQUAD and CO-QA

Method

At first, following the work of [3], we train a network using the pretrained BERT model, while adding a few layers on top of it for making the final classification, i.e., for predicting whether the answer is a span of the story. (or one of 'yes', 'no' 'unknown'). If it's a span, we predict the answer start and end indexes at the context. An illustration of the model is given in figure 1.

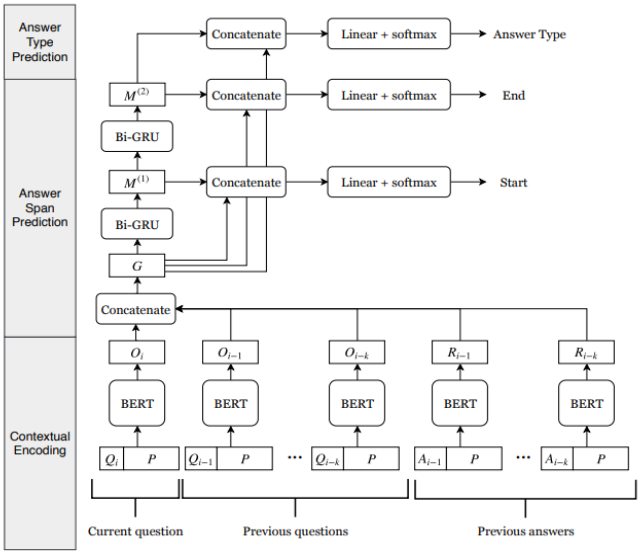


figure 1: illustration of the CQA model

That model is designed specifically for the conversational QA task, as it involves also questions and answers from the past for answering the current question.
We specifically used a history-window size (amount of questions and answers from the past that we relate to when answering on a new one) of 3, as the authors of [3] have shown that a bigger window size improves the accuracy only negligibly. This results in a model with 149M parameters.

The underlying fine-tuned BERT (109M parameters) will be used as an encoder as will be explained soon.

As already mentioned, we define a sample in the dataset as a story + questions regarding it + their answers. A sample will also be called a 'challenge'.

Following the work in [2], we use the KATE algorithm, but with another method for encoding challenges, designed specifically for our task.

Using an encoder model, that encodes a given text to a vector of a fixed size, we encode each challenge in the training set with the following procedure:

- Denote the encoder function as $E: Texts \rightarrow \mathbb{R}^d$.
- Denote the challenge story as C, and its questions and answers as $\{Q_1, \ldots, Q_n\}$ and $\{A_1, \ldots, A_n\}$ respectively.
- We take the first k-questions, and for each question Q_i, we encode the text $C \mid\mid Q_i$ to the vector $E(C \mid\mid Q_i) \in \mathbb{R}^d$.
- Finally, we average over all those k-encodings to get the final encoding of the whole challenge.

Find the span containing the answer for the following questions regarding the story:

story: <C'>
question: <$Q_1{}'$>
answer: <$A_1{}'$>
.
.
.
question: <$Q_m{}'$>
answer: <$A_m{}'$>

Find the span containing the answer for the following questions regarding the story:

story: <C>
question: <Q_1>
answer: <A_1>
.
.
.
question: <Q_j>
answer: <BLANK>

figure 2: The input GPT-3 is fed

k is a hyperparameter that we tune.

Note that for efficiency, we create an index of all the training set samples encodings, instead of creating them from scratch for every given queried test-set sample.

When we want to answer on the j'th question of a challenge at the dev-set,
we first encode it by following the same procedure described above, but instead of using the first k-questions, we use the **last** $min(k,j)$ questions, as they reflect more about the current question, and as we don't have more than j questions at that time-step.

Now, we take the sample at the training-set which's encoding has the highest cosine-similarity with the test-set sample's current encoding.

Denote that chosen training-set challenge as $\{C', Q_1', \ldots, Q_m', A_1', \ldots, A_m'\}$, and denote the test-set sample as $\{C, Q_1, \ldots, Q_n, A_1, \ldots, A_n\}$ (where C represents the story of the challenge, Q represents a question, and A represents an answer).

Finally, we feed GPT-3 with the input that's shown in figure 2. Note that because that each challenge in the dataset is long, and because of the input length limit of GPT, we can't naively feed more than 2 challenges to GPT as input, and that's why we didn't use more than one challenge as the context example in all our experiments.

Experiment

We evaluate our method on the CoQA dataset.
We focus on the case where the answers on the questions about the story are a span of it, or simply are 'yes', 'no' or 'unknown' (if not found as a span of the story), as those are the only types of question-answer shown in that dataset.

We first train the model designed in [3] on that dataset. As it has no available code or checkpoints we've restored the code and created a checkpoint which got to a score of 68.4% F1, and 57.9% EM, for 6 epochs, and take the checkpoint from the third one as it seems that the model overfits the training data after that epoch.
A plot of the training curve is shown in figure 3, as well as a plot of the test accuracy shown in figure 4.

We trained the model with the AdamW optimizer, with a batch-size of 20 (using gradient accumulation), each consists of 5 question and answer pairs (+context passage), a learning rate of 5e-05, $\beta_1 = 0.9, \beta_2 = 0.999$

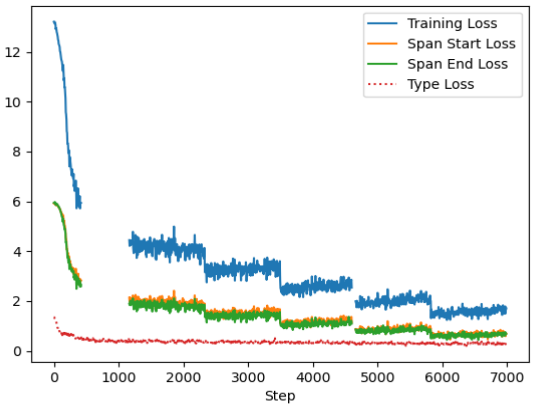


figure 3: Training loss as a function of the training steps. The curve is corrupted due to nan values in the loss during some periods at the training

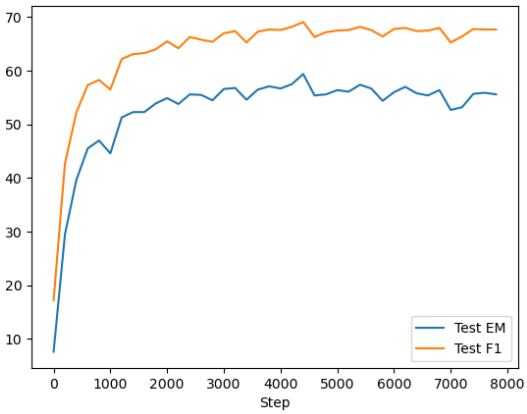


figure 4: Test accuracy as a function of the training steps. Accuracy measured by F1 and EM scores

weight-decay of 1e-02, learning rate warmup for the first 15% steps, and an exponential learning rate decay after the warmup, which cuts the learning rate by 5% from its current value after each 1000 steps.
We use the underlying BERT checkpoint of the finetuned model as our first encoder.

As our second encoder, we use the general pretrained BERT model, which wasn't fine-tuned for the same task (or for any task except language modeling).

As two other policies for choosing the samples, we use random sampling of a context sample from the training set for a given queried sample from the test set (we refer to this method as 'totally-random'), and a random sampling from the same source of the queried sample from the test set (which we refer as 'same-source-random').

We use the 'Ada' version of OpenAI's GPT-3 which has 350M parameters, through their public API. Since inferencing with the model through the API costs money, we randomly sample a subset of 50 challenges from the dev-set, 10 from each source, which we use all along these experiments.

We set the temperature parameter to 0.6, "top_p" to 1. Frequency penalty and presence penalty are both set to 0.
As evaluation metric for the GPT's predictions, we use the EM (exact match) and the F1 scores.

The random choosing methods (same-source-random and totally-random) are examined 5 times each, and the average F1 and EM scores are taken as an approximation for their performances.

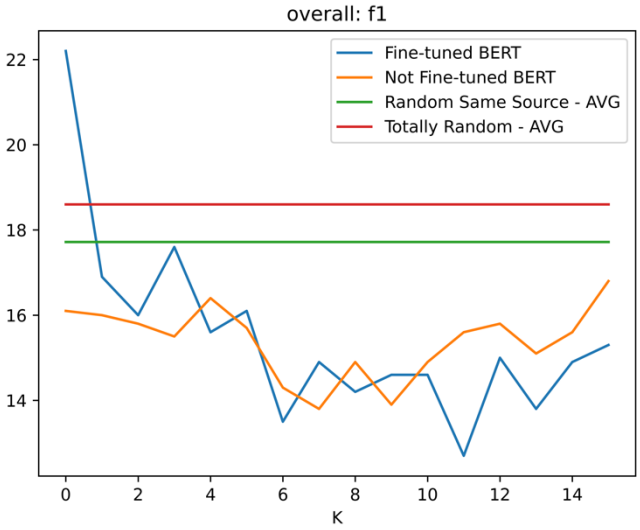
We test our method with k values ranging between 0 (only the story is used for the encoding of a challenge) to 15 (almost all of the questions in the challenge are used in its encoding).

Results

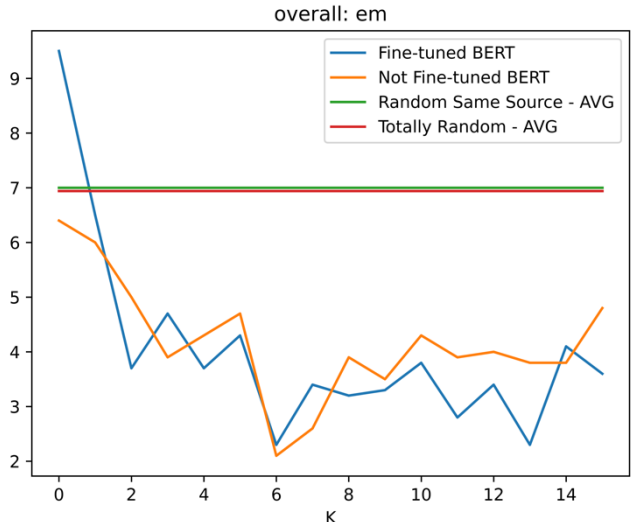
We find that our proposed method outperforms all the others when using the fine-tuned version of BERT, as can be seen in the graphs in figure 5, and at the other graphs at the appendix (figures 7-11), specifically when used with k=0, i.e., when the example challenge that's fed to GPT-3 is chosen with no respect to the questions, but only according to the passages at the challenges.

While our method for selecting the in-context examples sometime show a slightly better performance when using the pretrained BERT model when compared to the fine-tuned one, (see figures 7-8 in the appendix), we find our method when using the fine-tuned BERT model with k=0, the best of all, most of the times.

Another interesting fact is the peak at all the graphs that appears when K=3 for the fine-tuned model. This is though not very surprising, since the fine-tuned model was trained with a history window of (k=) 3, meaning it have learned to better encode the passage + 3 questions from each challenge.



(a). Overall (average across all domains) F1 score as a function of k



(b). Overall (average across all domains) Exact Match score as a function of k

figure 5: The overall accuracies of all the proposed methods. Our method, using the finetuned model outperforms all the others

Our method also surpasses the naive random choosing methods, including the same-source-random method, showing its usefulness.

Retriever & Web API

We have implemented a retrieval method, to automatically find relevant stories given a single question (first question in a conversation). We are using the standard TF-IDF retrieval method with cosine similarity as a similarity metric.

We retrieve top n stories (n is a hyperparameter, in our implementation set to 3) and run our question answering model (Reader) on each, to search for the answer. The story which yields highest confidence for containing the answer, according to the model (which's probabilistic) is selected and used for following questions in a conversation style.

The entire pipeline (retrieval, model evaluation) is wrapped inside a web server API, for a convenient use. Web server provides the user with a website where he can choose custom context (story) or decide to find the story automatically using this retrieval method, from pre-loaded stories. After this initiation, the user can ask more questions about a given or selected story in a conversational style. An example of one such conversation can be seen in figure 12 in appendix. An access to the API is available by running our code in github, using an available pretrained model.

Conclusion

We proposed a method for inferencing with GPT-3 in the conversational-QA task, and show that the best results are obtained when we encode each challenge by its passage only.

Continuing the work in [2], we again show that using a model that's fine-tuned for the same task as the examined task, as the encoder for queries, is almost always superior to using general models, that weren't fine-tuned for the examined task.
We also show the usefulness of our method, by showing that it outperforms the naive (random) ones.

Furthermore, we have implemented a convenient web API to interface with the model, including an automatic story retrieval system, which you can use through our project repository.

References

[1]. Language Models are Few-Shot Learners.
[2]. What Makes Good In-Context Examples for GPT-3?
[3]. A Simple but Effective Method to Incorporate Multi-turn Context with BERT for Conversational Machine Comprehension.
[4]. CoQA: A Conversational Question Answering Challenge.

Appendix

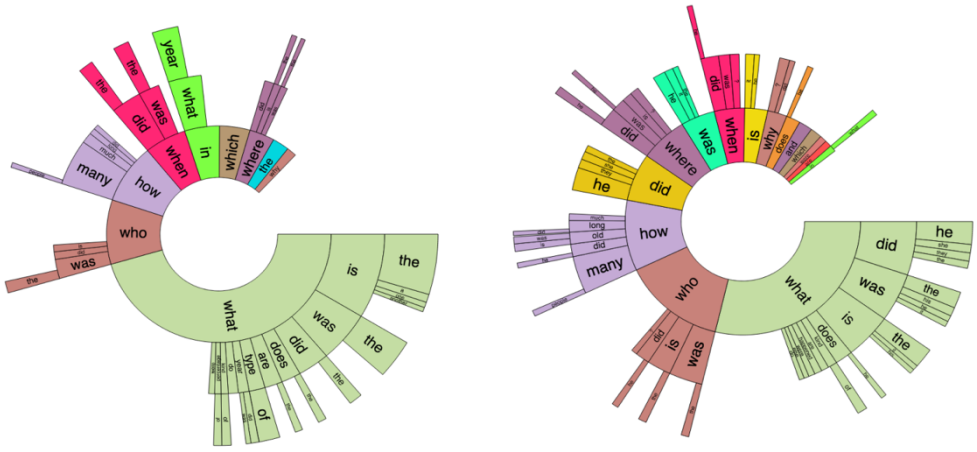
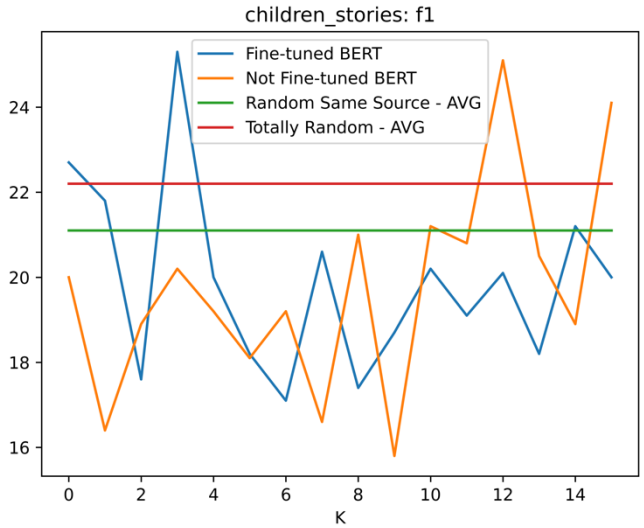
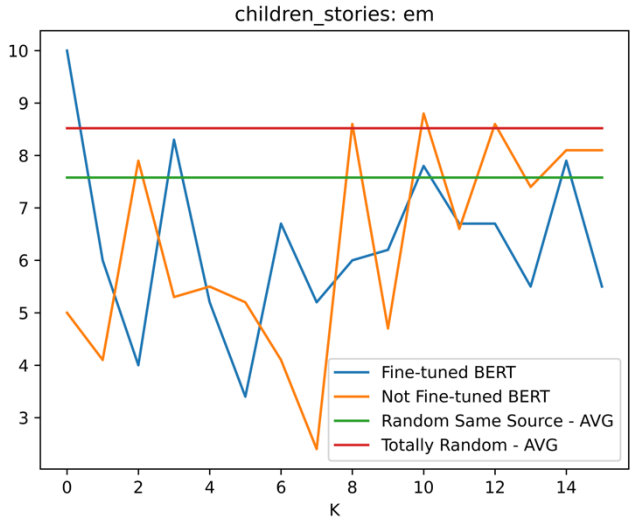


figure 6: Distribution of trigram prefixes of questions in SQUAD and CO-QA datasets

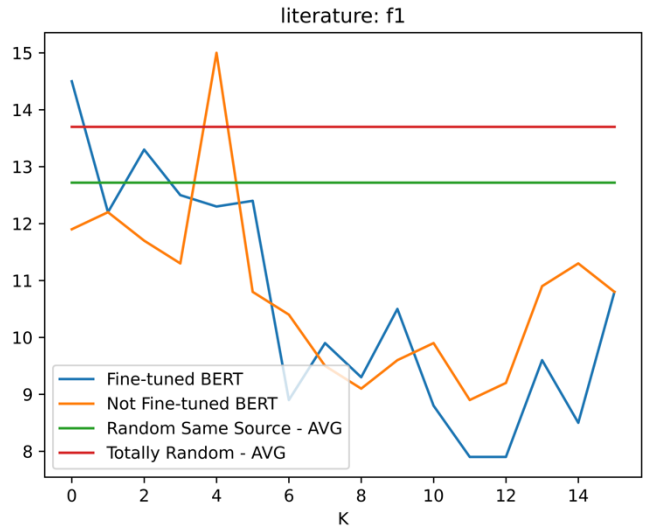


(a). F1 score as a function of k for the children-stories domain

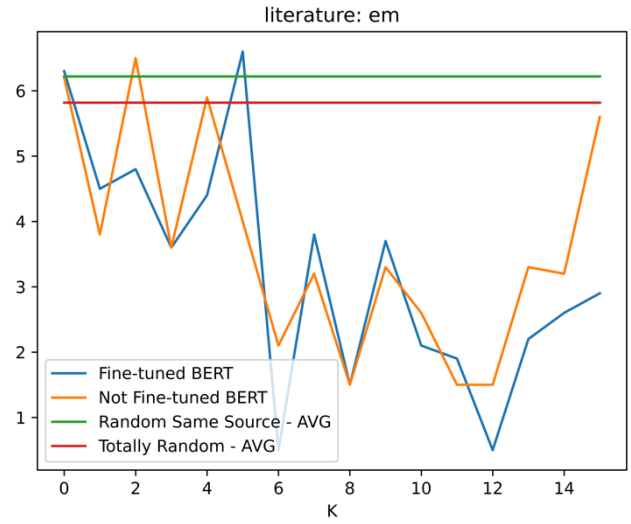


(b). Exact Match score as a function of k for the children-stories domain

figure 7

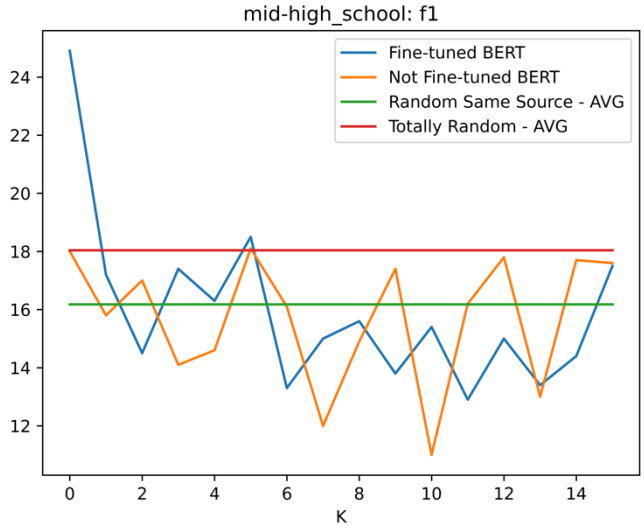


(a). F1 score as a function of k for the literature domain

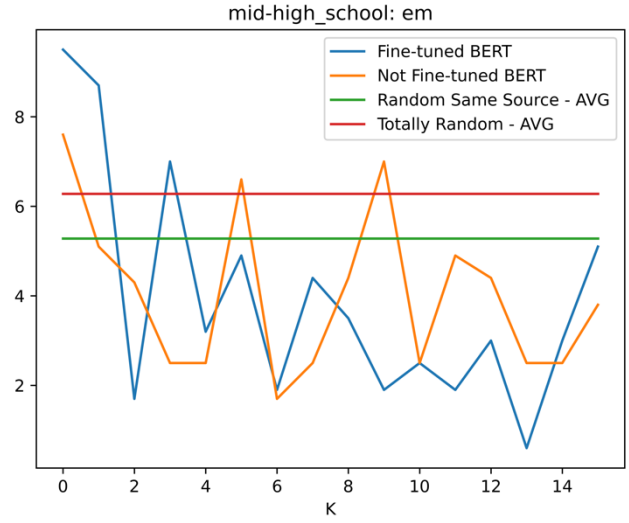


(b). Exact Match score as a function of k for the literature domain

figure 8

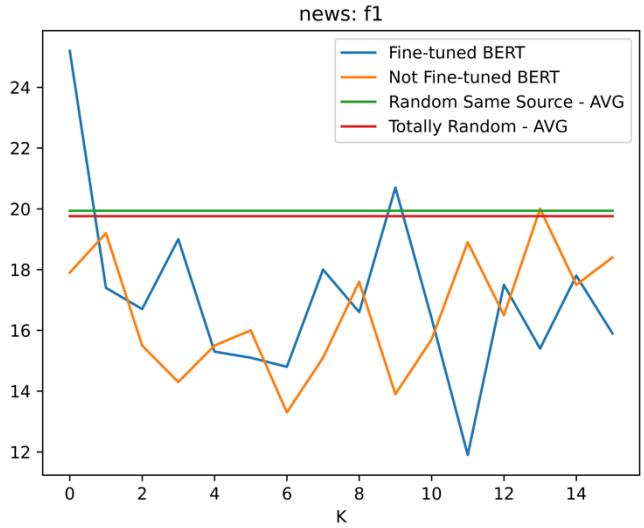


(a). F1 score as a function of k for the mid-high-school exams domain

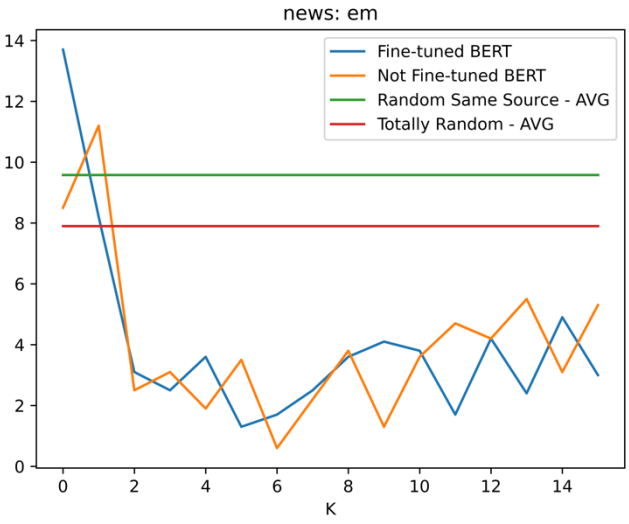


(b). Exact Match score as a function of k for the mid-high-school exams domain

figure 9

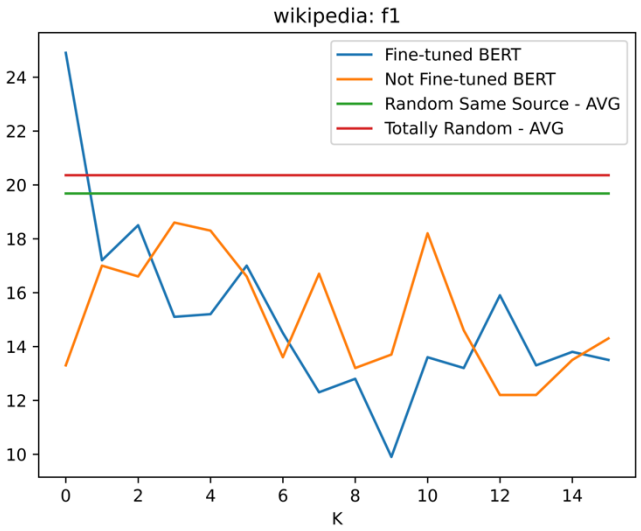


(a). F1 score as a function of k for the news domain

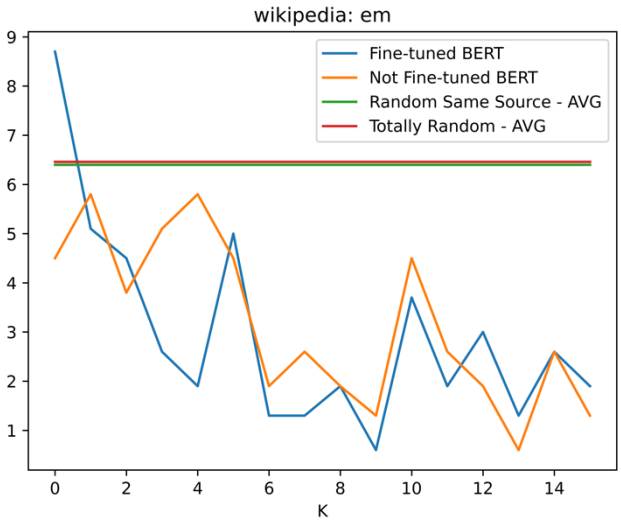


(b). Exact Match score as a function of k for the news domain

figure 10



(a). F1 score as a function of k for the Wikipedia domain



(b). Exact Match score as a function of k for the Wikipedia domain

figure 11

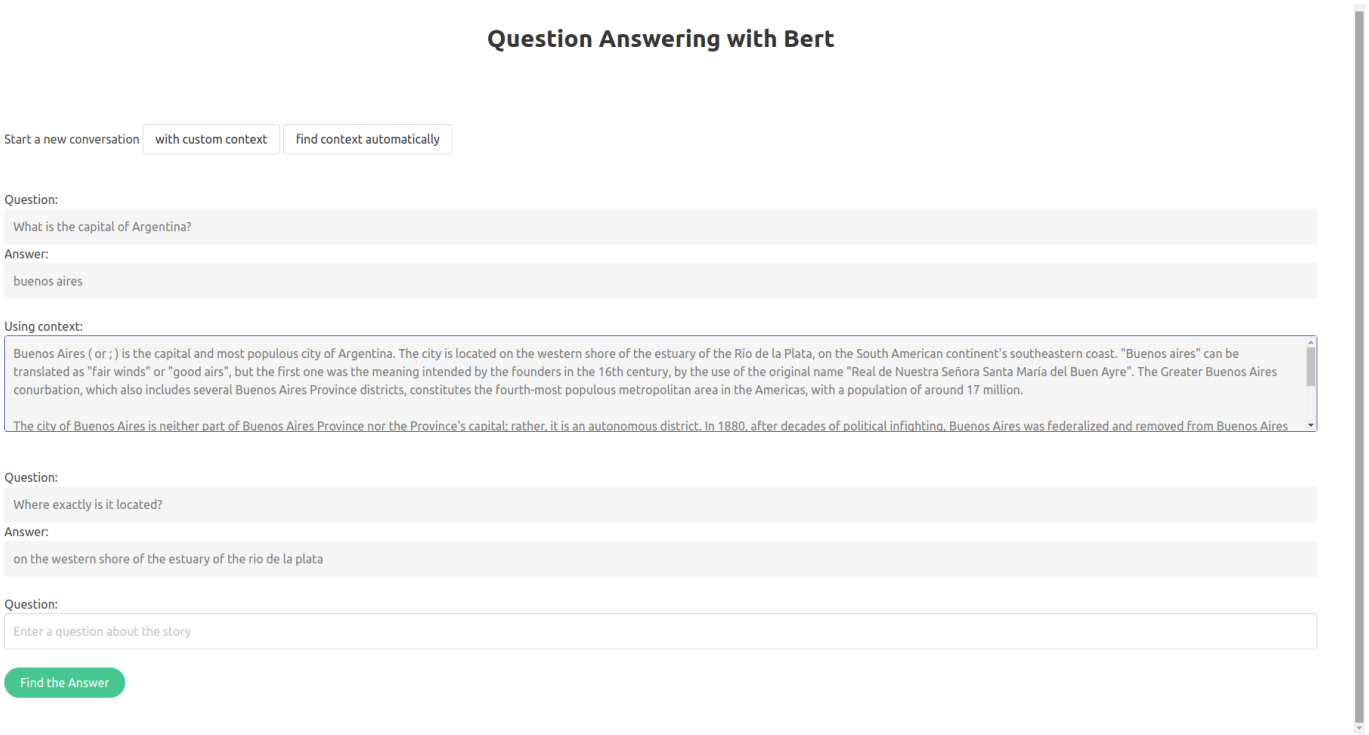


figure 12: A screenshot of the conversational retriever-reader API we've built using our model and the TD-IDF algorithm