Effective Sample Selection for In-context learning and Analysis of Output Quality for Downstream Fine-tuning

GIJS ADMIRAAL, 4871669@student.tudelft.nl
JOOST JANSEN, 4807170, j.j.jansen-2@student.tudelft.nl
NAZARIY LONYUK, 4596811, n.b.lonyuk@student.tudelft.nl
MARKUS TRASBERG, 5105013, m.traberg@student.tudelft.nl

In-Context Learning (ICL) emerges as a promising approach for leveraging pre-trained LLMs and adapting them to specific unseen tasks without complete retraining. This research paper focuses on efficient sample selection and aiding exploration for In-Context Learning (ICL) of Large Language Models (LLMs) . We introduce a visual interface that extends the OpenICL framework, enabling analysis and exploration of ICL techniques across LLMs, tasks, inferencers and retrievers. Additionally, we propose CDTabuRetriever, a sample selection method combining similarity-based retrieval with model-based confidence scores. Our results show that our heuristic does not improve or decrease accuracies for ICL compared to base methods such as BM25. The publicly available codebase aids reproducibility and development of our proposed contributions.

1 INTRODUCTION

In recent years, remarkable advancements in large language models (LLMs) have revolutionised natural language processing tasks. These models, such as OpenAI's GPT-4 [20] or Google's BARD [7], possess the ability to learn and perform a wide range of tasks with impressive accuracy. However, training these LLMs requires substantial computational resources and is often an expensive endeavour. To alleviate the burden of extensive fine-tuning, In-Context Learning (ICL) emerges as a promising approach for leveraging pre-trained LLMs and adapting them to specific unseen tasks without complete retraining [6]. By sidestepping this resource-intensive process of fine-tuning, ICL manages to yield comparable results to fine-tuned models in specific tasks [10, 17, 32].

ICL, in essence, involves the retrieval and inference of relevant contextual examples from a given dataset to train LLMs for specific tasks. By utilising the knowledge encoded in the pre-trained models and leveraging these in-context examples, it becomes possible to fine-tune LLMs without any parameter updates. This technique significantly reduces the cost associated with training large models from scratch, making it an attractive option for a wide range of applications.

In this research paper, we present our contributions to effective sample selection for In-Context Learning of Large Language Models. We address the challenge of selecting the most relevant and informative samples from a dataset to serve as in-context examples. Our objective is to enhance the performance and efficiency of the fine-tuning process while maximising the utilisation of available data.

Authors' addresses: Gijs Admiraal, 4871669@student.tudelft.nl; Joost Jansen, 4807170, j.j.jansen-2@student.tudelft.nl; Nazariy Lonyuk, 4596811, n.b.lonyuk@student.tudelft.nl; Markus Trasberg, 5105013, m.traberg@student.tudelft.nl.

XXXX-XXXX/2023/2-ART $15.00
https://doi.org/XXXXXXX.XXXXXXX

Our contributions are twofold. First, we introduce a simple yet powerful application that extends the OpenICL [29] framework with a visual interface. Thereby, we further enable the analysis and exploration of ICL techniques across various LLMs, tasks, and datasets. This platform provides researchers and practitioners with a flexible and user-friendly interface to experiment with different configurations and evaluate the effectiveness of different ICL strategies.

Secondly, we propose a sample selection method called CDTabuRetriever that leverages a combination of a similarity-based retrieval method with model-based confidence scores. We utilize examples that demonstrate the highest covariance similarity with all other examples in the embedding space and possess the best overall confidence score from the employed LLM. This heuristic chooses the samples that are expected to have the greatest impact on the In-Context Learning for the whole dataset, allowing for more efficient utilisation of computational resources and reducing the overall training time.

By addressing the critical aspect of sample selection for In-Context Learning of Large Language Models, our contributions aim to optimise the training process, reduce costs, and ultimately facilitate the widespread adoption of LLMs for a wide range of real-world applications.

The structure of this paper is as follows: In section 2, we provide a background on the training of LLMs and the challenges associated with fine-tuning. We then delve into our sample selection method called CDTabuRetriever in section 3, explaining its methodology and the features considered for selecting the optimal in-context examples. Following that, we present the details of our platform application in section 4, including its architecture and functionalities. In section 5 we explain our experimental setup. Resulting in section 6, where we showcase the results obtained through our approach, highlighting the improvements achieved in terms of performance and efficiency. In section 7 we analyze the results in detail, providing insights and interpretations regarding the observed outcomes. Finally, we conclude our research by summarising our findings in section 8, discussing the implications, and suggesting potential avenues for future research in the field of In-Context Learning.

We have made our codebase for this paper publicly available, to aid the reproducibility of our work and to encourage further development, found here: https://github.com/MarkusTrasberg/npl-project.

2 BACKGROUND

To provide a comprehensive explanation of our method, we start by establishing the groundwork and notation of In-Context-Learning. Additionally, we introduce various heuristics that our method employs to enhance its effectiveness.

2.1 In-Context Learning

In-Context Learning is a powerful paradigm that enables a large language model to generate outputs without the need for fine-tuning. By leveraging in-context examples, ICL allows the language model to generate relevant and contextually appropriate responses. The notation used in this field is based on prior work on ICL [31].

ICL involves linearizing each training instance into an input text, represented by the sequence $x = (x_1 ... x_{|x|})$, and an output text, represented by the sequence $y = (y_1 ... y_{|y|})$. Here, each token x_i, y_i belongs to the vocabulary set V, and V represents the LLM's vocabulary.

When faced with a new test input text x_{test}, the generation of the output y_{test} in an in-context learning setting is defined as follows:

$$y_{test} \sim \mathbb{P}_{LM}(y_{test}|x_1, y_1, ..., x_K, y_K, x_{test}) \tag{1}$$

Here, $\sim$ signifies the decoding strategies employed. Each in-context example $e_i = (x_i, y_i)$ is sampled from a training set $D = (xi, yi)_{i=1}^{N}$. The beauty of this generation procedure lies in its ability to generate outputs without the need for parameter updates in the language model when encountering new tasks, which can be both expensive and impractical.

The performance of ICL on downstream tasks can vary significantly, ranging from almost random results to being comparable with state-of-the-art systems. The quality of the retrieved in-context examples play a crucial role in determining the performance [10, 17, 32]. Previous studies have proposed various approaches to selecting in-context examples, modelling the process using a retriever $P(e_i|x_{test})$.

2.2 In-Context Example Selection

In-Context Example Selection refers to the process of the retriever selecting relevant samples e_i from a dataset to serve as in-context examples for fine-tuning LLMs. The selection of high-quality samples plays a crucial role in the success of the fine-tuning process, as it directly impacts the model's ability to effectively respond to queries [16, 24, 30].

Similarity-based Selection. To overcome the limitations of random sample selection, various heuristic methods have been proposed. These methods leverage semantic similarity-based retrieval techniques, such as BM25 [22], TopK [16], and VoteK [26]. These heuristics aim to identify samples that are likely to enhance the fine-tuning process by considering their similarity to the query context.

Model-based Selection. Additionally, model-based methods have been explored in recent research. These methods utilize the models' confidence in their output to select and order examples. For instance, entropy-based retrieval [17] and Minimum Description Length (MDL) retrieval [30] have been investigated as approaches to sample selection.

To the best of our knowledge, a combination of these two techniques have not been proposed yet. By employing a combination of these similarity- and model-based methods, researchers may improve the sample selection process, leading to a higher chance of selecting samples that are relevant and beneficial for fine-tuning LLMs. These advancements in sample selection techniques contribute to the overall success and performance of in-context learning.

3 RETRIEVER IMPLEMENTATION

In this section, we present the methodology used for developing a novel method called CDTabuRetriever (CDT), which aims to improve in-context learning sample selection. The primary objective of CDTabuRetriever is to intelligently retrieve relevant samples from a large dataset, considering the specific context of the learning task.

3.1 Confidence X Diversity metric

To introduce a novel approach for sample selection, we devised a method centred around a newly developed metric called confidence_x_diversity (CD). This metric enables the calculation of the CD score for a given set of in-context examples by considering the confidence score of each input x_i the employed LLM gives, as well as the sum of dissimilarity between that input and all other examples within the set. Where the dissimilarity between

x_i and x_j is seen as $1 - cos\_sim(x_i, x_j)$. The CD score of a set comprising N examples can be computed using the following equation:

$$CD_{set} = \Sigma_i^N \left[confidence(x_i) * \Sigma_j^N \left(1 - cos\_sim\_matrix(x_i, x_j)\right)\right] \quad (2)$$

We hypothesize that by incorporating confidence into our heuristic, we can improve the quality and reliability of selected examples. Confidence scores serve as a measure of the language model's certainty in its predictions, enabling us to identify the examples that the model finds most confident about. This choice is driven by the potential to enhance the learning process and reasoning abilities of the model.

Moreover, by selecting the most diverse set of examples in terms of context, we believe that the model can learn from various perspectives and better infer context when encountering new prompts. Combining diversity and confidence, we aim to create a learning approach that benefits from different contexts while leveraging the model's strongest predictions. This strategy enables us to tap into a wide range of information and improve the model's overall performance.

3.2 Similarities to the Quadratic Knapsack Problem

For our new sampling selection method, we need to find the combination of samples that have the highest total confidence. Additionally, we want the most diverse set we can find. However, we measure diversity of the set by calculating all dissimilarity matrices between any two samples in the set. This might seem straight forward, but in reality this is a hard problem to solve with an algorithm. Essentially the problem of getting the sample set with the highest score is a version of the Quadratic Knapsack Problem (QKP).

The QKP was first introduced by Gallo et al [9] as an extension to the 0-1 knapsack problem. In this problem the value of items placed in the knapsack does not only depend on the individual values of the items but also on a hidden reward that you get when a combination of certain items is present. The QKP is NP-hard and no solution exists for this problem in pseudo-polynomial time. Many works have explored different solutions such as using genetic algorithms for example [25].

We designed a metric, such that the value of an input prompt is calculated by multiplying the model's confidence score in this prompt with the sum of all dissimilarities between the prompt and other selected prompts. This is essentially a transposition of the QKP.

For memory- and time efficiency purposes we decided to find the optimal set of examples using Tabu Search, which is known to produce optimal or highly competitive solutions and shares similar philosophy to genetic algorithms [12].

3.3 Tabu Search

Since the selection of the best set is an NP hard problem, we have to find a way to solve it. One option could be to use a genetic algorithm, but we decided to use a heuristic search algorithm.

Tabu Search (TS) is a well-known heuristic search algorithm that has been used find solutions for other NP-hard problems such as the Travelling Salesman Problem [8]. It was first developed by Glover in [11]. In TS a short-term memory is kept of good solutions, which are used as a basis for creating new solutions in order to find a solution that is as close to optimal as possible. Because the memory is only short-term, it is possible for the algorithm to find its way out of local optima which makes it a viable technique to use to find optimal solutions for NP-hard problems.

In the design of our sample selection algorithm we made two practical decsions. One being that we decided to calculate the confidence by prompting the model used for ICL with unlabeled inputs without examples. The other being that we decided to run our Tabu Search algorithm for $1e4$ iterations, with a short-term memory of best solutions (tabu tenure) of 5.

4 PLATFORM

Another contribution of this paper is a demo platform designed to assess the impact of prompt selection on performance across various models. This section describes the origins of this prototype and outlines its potential value for future research endeavours. The final iteration of the platform can be observed in Figure 4 and Figure 5.

4.1 Prototype

Before starting to build the platform, we developed a medium-fidelity prototype for the design of the platform (see Figure 3) using the Figma software [3]. In the prototype, we decided to split the user flow into three sections, namely training, testing and evalution sections. Even though the final outcome of the platform slightly differed from the proposed prototype solution, this prototype served as a good starting point for building the actual implementation.

4.2 OpenICL

To facilitate the in-context learning process and evaluation on the platform, we utilize the OpenICL framework[29]. It is a recent open-source library designed for efficient retrieval and inference of contextual examples from datasets. It has a highly flexible architecture that allows users to seamlessly combine different ICL components to suit their needs. This thus allows us to easily develop and test our heuristic. To modularity of the system helps us create an environment in which we can test our retrieval heuristic against other already implemented retriever heuristics. Furthermore, modularity creates an adaptable environment in which we can easily do an ablation study to maximize the efficiency of our heuristic.

4.3 Backend

The backend of the demo tool is written in Python 3.10.6 and uses a Flask [14] server to communicate with the frontend. This communication is done through three endpoints, /parameters, /run and /debug, which are documented using Swagger[28] which can be accessed by navigating to the endpoint /apidocs.

GET /parameters returns the allowed parameters and its values it can set for running OpenICL, such as model_name, dataset, retriever, inferencer, etc.

POST /run starts an OpenICL run with the parameters values specified in the body. It returns the accuracy of the OpenICL run, the original prompts, the predictions and the true answers.

POST /debug pretends to start an OpenICL run and returns a dummy object containing an accuracy value, original prompts, predictions and true answers. This endpoint is only used to see if a POST request can be made to the backend and to run the frontend without having OpenICL installed.

4.4 Frontend

For developing the front end of the application, we use Next.js by Vercel [1], a React-based framework used in many modern web applications. The design is done through a combination of Tailwindcss [4] and React Bootstrap [2]. Throughout the development phase, the emphasis was placed on the ability to rapidly prototype

and construct the platform. Hence, Next.js was selected over Vanilla JavaScript or standalone React due to its fast project setup capabilities and out-of-the-box working client. Similarly, by utilizing ready-made components like the Form or DataGrid from React-Bootstrap [2] instead of building them from scratch, we ensured that the development process was sped up.

Overall, the frontend is split into two sections:

(1) **Training** - Users select the baseline training parameters such as model, evaluator, retriever, datasets, # of samples and ICE size.
(2) **Results** - Displays the results for the selected parameters. Contains the performance score together with a table of the questions, true answers and predicted answers.

The final frontend solution slightly deviates from the prototype version. For example, it was decided to place the training and testing sections together for better space usage. For similar reasons, it was decided to not display the training/testing rows and only display the final result to the user.

During loadup of the frontend, a API call is sent to /parameters, the result of which is displayed as the available parameter settings under Training. Once the user applies the settings and clicks on 'Get results', POST request to /run is sent. This can run anywhere between 1-10 minutes depending on the selected model and size. Once it is complete, the Results panel will display the respective results sent from the backend.

4.5 Use cases

The tool is developed to allow the user to compare the performance of in-context learning for a variety of large language models and different tasks. It can be run on the browser to simplify the evaluation process and gives control to the user by allowing them to customize the ICL settings. This tool could potentially be an added layer to the existing OpenICL implementation, giving its users access to a more visual and interactive way of evaluating in-context learning and prompt selection for any kind of NLP tasks. In addition, this platform could serve as a good starting point for someone getting into the technicalities of LLM's in-context learning.

5 EXPERIMENTAL SETUP

In order to evaluate our innovative example selection method for ICL, we conducted comprehensive testing of our CDTabuRetriever on diverse factors, including models, tasks, datasets, and other retrievers.

Task specification. To test our new heuristic we decided to test it on the tasks of Multiple-Choice Question Answering (MCQA) and the task of Sentiment Classification (SC). We chose both tasks since a LLM might be better in one than the other while both tasks have a ground truth. The practicality of having a ground truth is that the accuracy of the model can easily be measured.

Model selection. The LLMs we will use for our testing are the model the gpt3/text-davinci-003 [19], google/flan-t5-small[13] and the bigscience/bloom-560m [5] model. The reason for using different models, specifically those trained for text-generation, in our testing was twofold. Firstly, we aimed to assess the performance of both tasks across a variety of models, and text-generation models are known for their versatility and adaptability. Secondly, by incorporating models of varying sizes, we could observe how task performance and example selection were influenced across different model architectures.

MCQA dataset choices. For the MCQA task we decided to use the following datasets, containing common sense and general open book questions respectively: the commonsense_qa [27] and the openbookqa [18] dataset. In order to calculate the confidence and accuracy scores, the zero-shot unlabeled calculation of this requires two input columns: the question and the context. The commonsense_qa dataset already contains a context column, but

for openbookqa we added a context column. The value in this column is "Open book questions". Some additional preprocessing was required for both datasets, where all possible answers are split in their respective columns and passed as input columns.

SC dataset choices. To evaluate the SC task we used the following datasets; gpt3mix/sst2 [15] and rotten_tomatoes[21] dataset, both containing movie reviews. The only preprocessing that needed to be done was the reversing of label values for the rotten_tomatoes dataset as this differed from how this was done in gpt3mix/sst2.

CDTabuRetriever. To test and compare our method of selecting examples, we use the OpenICL framework[29]. We created our own custom $CDTabuRetriever$ class, which extends the $BaseRetriever$ class. Some practical decisions we made were that in order to facilitate the calculation of confidence scores, our $CDTabuRetriever$ also takes a few additional parameters. A model name, to base the confidence scores on and a task name such that the retriever knows which columns of the dataset to base its confidence score on.

Other retrievers. The newly designed retriever, employing our proposed heuristic, will be compared against several retrievers available in the *openicl* Python package. Specifically, we will evaluate its performance against the RandomRetriever, which randomly selects a certain number of examples, the TopKRetriever, which selects the top k nearest embedded neighbors, and the BM25Retriever, which selects examples using the Okapi method as described in Robertson et al.'s work [23].

inferencer selection. For both the MCQA and SC task we decided to use the $GenInferencer$ class. We give this inferencer a generative prompt; which has the form of a string containing the question as well as the options which the model is meant to choose from. Because of this, the model's answer is not restricted to only these options, but a model that is trained for text-generation should correctly understand how to give its answer in the proper format.

Experiment. We conducted a comprehensive experiment to evaluate the impact on the accuracy of different heuristics for in-context learning example (ICE) selection. We tested all models, tasks, and datasets, varying the number of ICE to 1, 2, 4, and 10. Each experiment was averaged over three runs to ensure robust results. For eacht run we evaluated the dataset on a test set of 20 input-output combinations. Due to time and computational limitations, we utilized a limited set of ICE, models, tasks and retrievers.

6 RESULTS

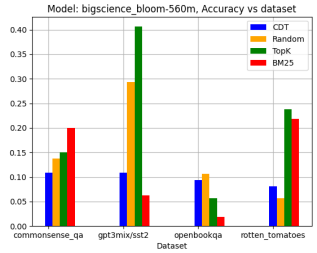
In this section, we present the findings from our experiments on different heuristics for ICL example selection. The detailed results can be found in a table in Appendix Section.

We evaluated the performance of each heuristic by measuring its accuracy. Since for both tasks there is only a small set of options, we define the accuracy to be the amount of correct answers per prompt, scaling from 0 (None correct) to 1 (All correct).

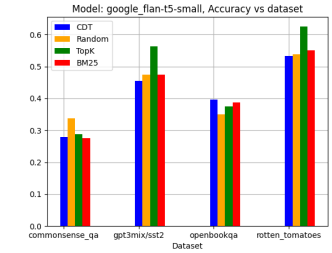
We present two different plots: A bar plot in 1 where we show, for each dataset, the mean of found accuracies per heuristic. The values show the mean of experiments where the number of in-context examples were varied.

We also present a line plot in 2, where we show the accuracies of each heuristic method versus the amount of in-context examples used for learning.

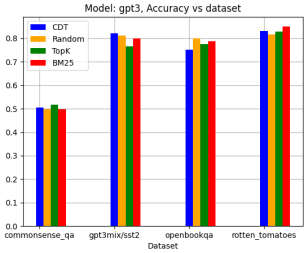
Fig. 1



(a) Bar plot for the 4 analysed retrievers where the mean accuracy for each data set is presented for the "Bloom" model.

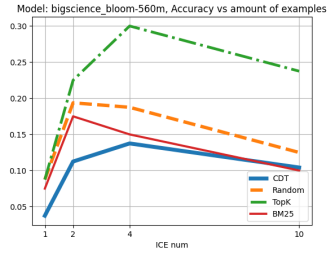


(b) Bar plot for the 4 analysed retrievers where the mean accuracy for each data set is presented for the "Flan" model.

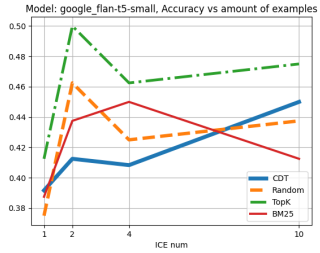


(c) Bar plot for the 4 analysed retrievers where the mean accuracy for each data set is presented for the "GPT3" model.

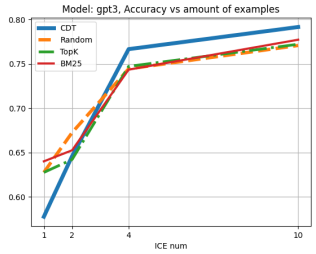
Fig. 2



(a) Line plot for the 4 analysed retrievers where the mean accuracy is plotted versus the number of in-context examples used is presented for the "Bloom" model.



(b) Line plot for the 4 analysed retrievers where the mean accuracy is plotted versus the number of in-context examples used is presented for the "Flan" model.



(c) Line plot for the 4 analysed retrievers where the mean accuracy is plotted versus the number of in-context examples used is presented for the "GPT3" model.

7 DISCUSSION

In this section, we will discuss our results. We will also discuss the limitations of our method, experimental setup, and the platform.

7.1 Performance of Models

When looking at the results per model, we can see that Bloom performs the worst for all datasets. For the commonsense_qa dataset there are 5 multiple choice options, for the openbookqa there are 4 options and for the others there are 2 options. However we can see that Bloom's results are around or below 20%, 25% and 50% respectively. This means that all results obtained by using the Bloom model are inconclusive, since the model can be outperformed by picking at random among the choices.

We can see that GPT-3 performs best on all tasks. Flan performs slightly worse, but definitely not picking random choices.

7.2 Choice of Retriever

We found that the choice of retriever does not impact in-context learning much at all. Therefore in future research, we recommend using a larger test and training set, in order to be able to conclude which sample selection method is better with more certainty.

7.3 CDTabuRetriever performance

Although the results do not show that CDT is necessarily the best or worst performing retriever, there is some information we can learn.

We can see that our model performs better when selecting a larger amount of examples, as we can see from the line plots for the "Flan" and "GPT-3" models.

We can see from both the line and bar plots in 1 and 2 that our model can outperform commonly used retrieval methods. Although it cannot do this consistently, this does indicate that it is a viable option and might be worth exploring further.

7.4 Confidence calculation

In the proposed method, CDTabuReciever, the confidence is given by giving a text-generation model a prompt. This takes a significant amount of time, and makes the sample selection method slower than others it is compared to. To improve the computational speed of this sample selection method, we suggest using a confidence estimation techinque to get confidence scores of training samples.

7.5 Limitations in Computational Power

Given that the nature of LLMs requires high computational power, obtaining results was a cumbersome process. Since we were limited to using our own laptops and Google Colab, we were unable to run extensive experiments. Our proposed retriever heuristic is dependent on the training size, this thus did not allow us to get results from larger training sizes.

7.6 Limitations in Prompt Selection

Another limitation of this paper could be the prompt selection. During our experiments, we noticed that using a wrong prompt can lead to the model not being able to learn the task at hand and giving the wrong type of answer, regardless of the in-context examples given. This means that each dataset requires a different optimal prompt to get the best performance. The obtained results might thus vary if altering the prompts.

7.7 Future Work

The obtained results are only gathered from using a generative inferencer. For the quality assurance of our heuristic it would be beneficial to also test it on other inferencers such as the perplexity-based inferencer or the chain-of-thought inferencer.

Secondly, given the scope and timeframe of this research, the resulting platform is still basic in its evaluation and visualization capabilities. The developed platform would benefit from having more in-depth results section

with added graphs and more evaluation metrics. In addition, the current user flow can be improved by speeding up the running process as current results can take up to a few minutes to generate. In essence, the demo tool could be used as a simple exploratory tool for researchers interested in context learning or even serve as an integral part of the OpenICL framework.

8 CONCLUSION

In conclusion, this research paper focused on the effective sample selection for In-Context Learning (ICL) of Large Language Models. We introduced a visual interface that extends the OpenICL framework, allowing for the analysis and exploration of ICL techniques across LLMs, tasks, inferencers, and retrievers. We also proposed a sample selection method called CDTabuRetriever, which combines similarity-based retrieval with model-based confidence scores. The results present inconclusive evidence that the CDTabuRetriever outperforms other sample selection methods. However, the proposed method is competitive regarding accuracy while slower in computation time against other methods.

REFERENCES

[1] [n. d.]. Next.js by Vercel - The React Framework. https://nextjs.org/

[2] [n. d.]. React Bootstrap | React Bootstrap. https://react-bootstrap.netlify.app/

[3] 2018. Figma: the collaborative interface design tool.

[4] 2020. Tailwind CSS - Rapidly build modern websites without ever leaving your HTML. https://tailwindcss.com/

[5] BigScience. 2021. BigScience Bloom-560M. https://bigscience.huggingface.co/models/bigscience/bloom-560m Accessed on June 17, 2023.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. In *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (Eds.), Vol. 33. Curran Associates, Inc., 1877–1901. https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfcb4967418bfb8ac142f64a-Paper.pdf

[7] Aaron Daniel Cohen, Adam Roberts, Alejandra Molina, Alena Butryna, Alicia Jin, Apoorv Kulshreshtha, Ben Hutchinson, Ben Zevenbergen, Blaise Hilary Aguera-Arcas, Chung-ching Chang, et al. 2022. LaMDA: Language models for dialog applications. (2022).

[8] C-N Fiechter. 1994. A parallel tabu search algorithm for large traveling salesman problems. *Discrete Applied Mathematics* 51, 3 (1994), 243–267.

[9] Giorgio Gallo, Peter L Hammer, and Bruno Simeone. 1980. Quadratic knapsack problems. *Combinatorial optimization* (1980), 132–149.

[10] Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *arXiv preprint arXiv:2012.15723* (2020).

[11] Fred Glover. 1989. Tabu search—part I. *ORSA Journal on computing* 1, 3 (1989), 190–206.

[12] Fred Glover, James P. Kelly, and Manuel Laguna. 1995. Genetic algorithms and tabu search: Hybrids for optimization. *Computers Operations Research* 22, 1 (1995), 111–134. https://doi.org/10.1016/0305-0548(93)E0023-M Genetic Algorithms.

[13] Google. 2022. Google FLAN-T5 Small. https://github.com/google-research/FLAN Accessed on June 17, 2023.

[14] Miguel Grinberg. 2018. Flask web development: developing web applications with python.

[15] Yuguo Liang, Di He, Luke Zettlemoyer, and Wen-tau Li. 2021. GPT3Mix: Leveraging Large-scale Language Models for Controlled Text Generation. arXiv:2104.08691 [cs.CL] https://arxiv.org/abs/2104.08691

[16] Jiachang Liu, Dinghan Shen, Yizhe Zhang, Bill Dolan, Lawrence Carin, and Weizhu Chen. 2021. What Makes Good In-Context Examples for GPT-3? *arXiv preprint arXiv:2101.06804* (2021).

[17] Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2021. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. *arXiv preprint arXiv:2104.08786* (2021).

[18] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. OpenBookQA: A New Benchmark for Open Book Question Answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. 5038–5048. https://www.aclweb.org/anthology/D18-1513/

[19] OpenAI. 2023. GPT-3: text-davinci-003. https://www.openai.com Accessed on June 17, 2023.

[20] OpenAI. 2023. GPT-4 Technical Report. arXiv:2303.08774 [cs.CL]

[21] Bo Pang and Lillian Lee. 2004. A Sentimental Education: Sentiment Analysis Using Subjectivity Summarization Based on Minimum Cuts. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL)*. 271–278. https://www.aclweb.

org/anthology/P04-1035/

[22] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[23] Stephen E Robertson, Steve Walker, MM Beaulieu, Mike Gatford, and Alison Payne. 1996. Okapi at TREC-4. *Nist Special Publication Sp* (1996), 73–96.

[24] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning To Retrieve Prompts for In-Context Learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Seattle, United States, 2655–2671. https://doi.org/10.18653/v1/2022.naacl-main.191

[25] Tugba Saraç and Aydin Sipahioglu. 2007. A genetic algorithm for the quadratic multiple knapsack problem. In *Advances in Brain, Vision, and Artificial Intelligence: Second International Symposium, BVAI 2007, Naples, Italy, October 10-12, 2007. Proceedings 2*. Springer, 490–498.

[26] Hongjin Su, Jungo Kasai, Chen Henry Wu, Weijia Shi, Tianlu Wang, Jiayi Xin, Rui Zhang, Mari Ostendorf, Luke Zettlemoyer, Noah A Smith, et al. 2022. Selective annotation makes language models better few-shot learners. *arXiv preprint arXiv:2209.01975* (2022).

[27] Alon Talmor, Jonathan Herzig, and Jonathan Berant. 2019. CommonsenseQA: A Question Answering Challenge Targeting Commonsense Knowledge. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 4144–4154. https://www.aclweb.org/anthology/D19-1410/

[28] Tony Tam. 2011. Swagger API.

[29] Zhenyu Wu, YaoXiang Wang, Jiacheng Ye, Jiangtao Feng, Jingjing Xu, Yu Qiao, and Zhiyong Wu. 2023. Openicl: An open-source framework for in-context learning. *arXiv preprint arXiv:2303.02913* (2023).

[30] Zhiyong Wu, Yaoxiang Wang, Jiacheng Ye, and Lingpeng Kong. 2022. Self-adaptive In-context Learning. *arXiv preprint arXiv:2212.10375* (2022).

[31] Jiacheng Ye, Zhiyong Wu, Jiangtao Feng, Tao Yu, and Lingpeng Kong. 2023. Compositional exemplars for in-context learning. *arXiv preprint arXiv:2302.05698* (2023).

[32] Zihao Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*. PMLR, 12697–12706.

A MEDIUM-FIDELITY PROTOTYPE

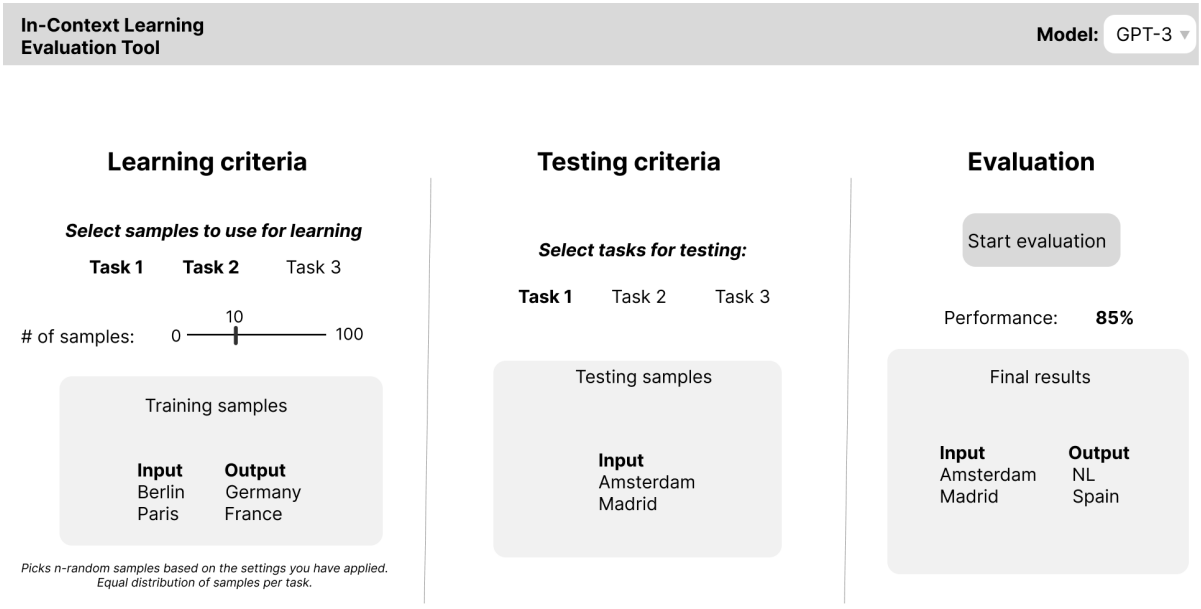


Fig. 3. Prototype

B PLATFORM

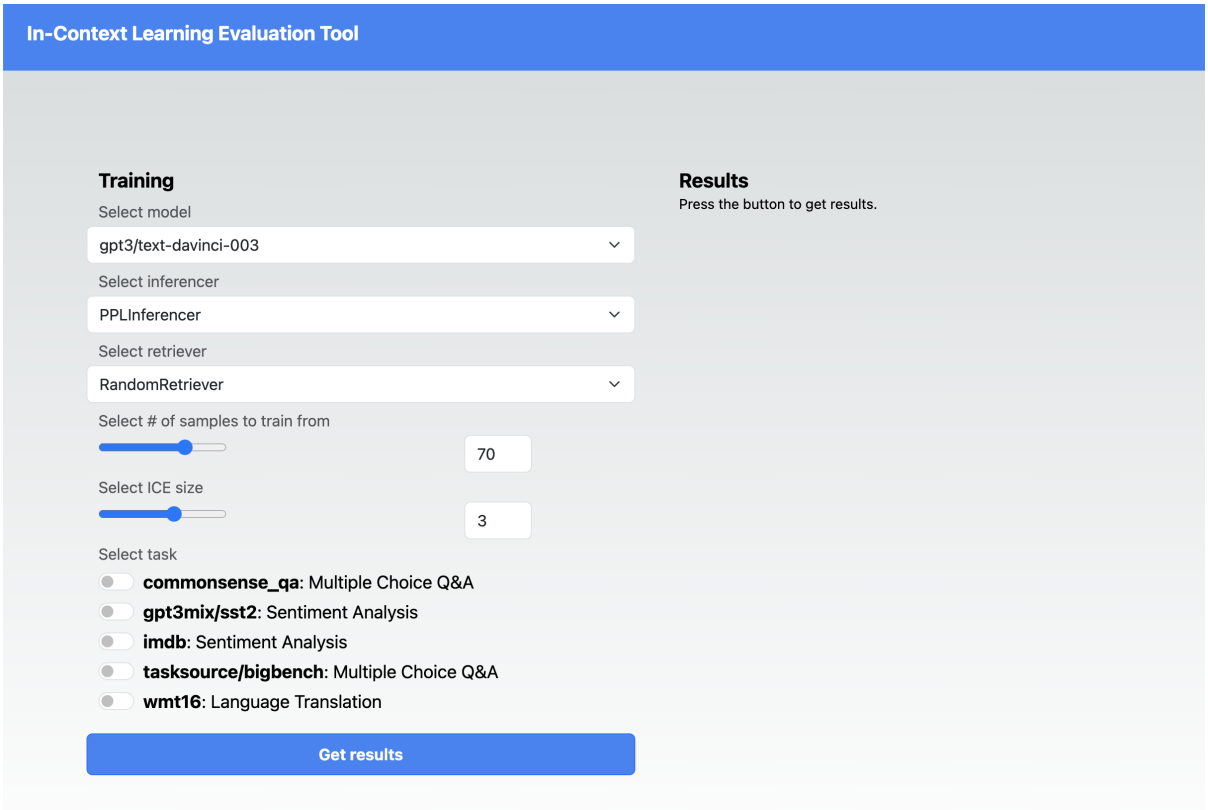


Fig. 4. Final platform look in the browser upon landing

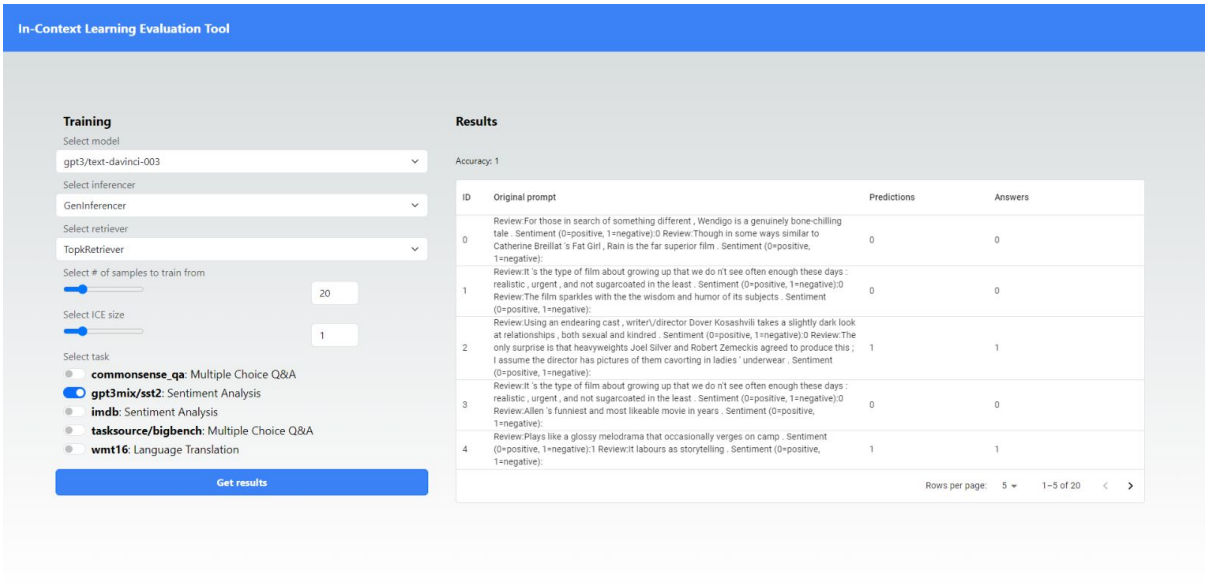
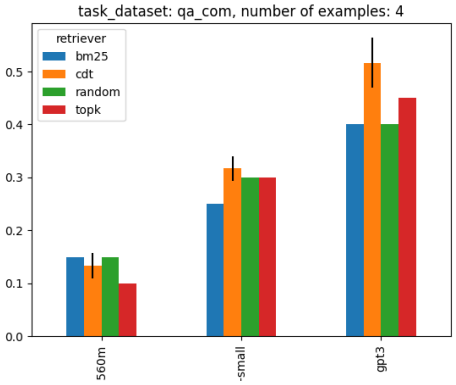


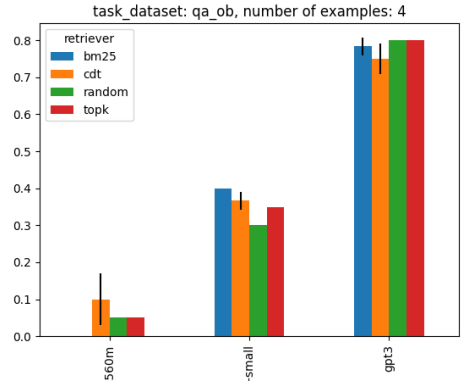
Fig. 5. Final platform look after generating results

C ALL RESULTS

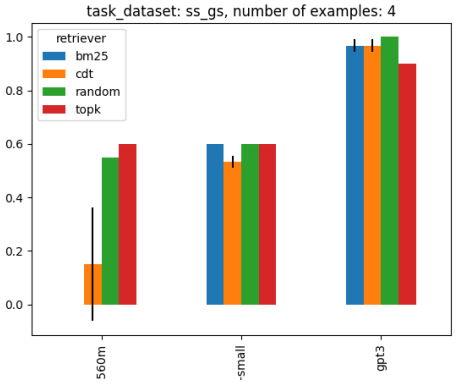
model	test_size	task	dataset	retriever	accuracy_mean	accuracy_std
google/flan-t5-small	20	qa	cs	qkp	0.37	0.02
google/flan-t5-small	20	qa	cs	random	0.35	0.0
google/flan-t5-small	20	qa	cs	topk	0.25	0.0
google/flan-t5-small	20	qa	cs	bm25	0.35	0.0
google/flan-t5-small	20	qa	ob	qkp	0.43	0.05
google/flan-t5-small	20	qa	ob	random	0.4	0.0
google/flan-t5-small	20	qa	ob	topk	0.4	0.0
google/flan-t5-small	20	qa	ob	bm25	0.4	0.0
google/flan-t5-small	20	qa	cs	qkp	0.18	0.02
google/flan-t5-small	20	qa	cs	random	0.3	0.0
google/flan-t5-small	20	qa	cs	topk	0.35	0.0
google/flan-t5-small	20	qa	cs	bm25	0.25	0.0
google/flan-t5-small	20	qa	ob	qkp	0.35	0.04



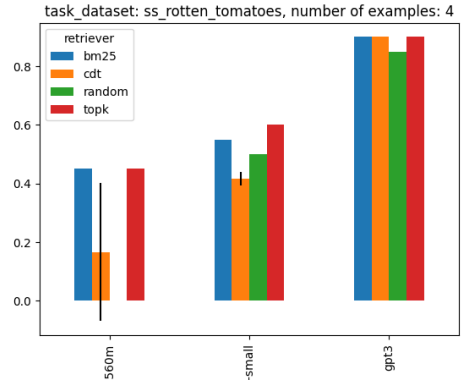
(a) Question and answering task with the Commonsense dataset



(b) Question and answering task with the Openbookqa dataset



(c) Sentiment analysis task with the Gpt3mix/sst2 dataset



(d) Sentiment analysis task with the Rotten Tomatoes dataset

Fig. 6. Four Subfigures of different tasks and dataset runned on 4 In-Context Examples. The used models are google/flan-t5-small, gpt3/text-davinci-003 and the bigscience/bloom-560m with the corresponding retrievers. The used retrievers are Random, Bm25, TopK and QKP which is our CDTabuRetriever.

google/flan-t5-small	20	qa	ob	random	0.35	0.0
google/flan-t5-small	20	qa	ob	topk	0.35	0.0
google/flan-t5-small	20	qa	ob	bm25	0.35	0.0
google/flan-t5-small	20	qa	cs	qkp	0.32	0.02
google/flan-t5-small	20	qa	cs	random	0.3	0.0

google/flan-t5-small	20	qa	cs	topk	0.3	0.0
google/flan-t5-small	20	qa	cs	bm25	0.25	0.0
google/flan-t5-small	20	qa	ob	qkp	0.37	0.02
google/flan-t5-small	20	qa	ob	random	0.3	0.0
google/flan-t5-small	20	qa	ob	topk	0.35	0.0
google/flan-t5-small	20	qa	ob	bm25	0.4	0.0
google/flan-t5-small	20	qa	cs	qkp	0.25	0.0
google/flan-t5-small	20	qa	cs	random	0.4	0.0
google/flan-t5-small	20	qa	cs	topk	0.25	0.0
google/flan-t5-small	20	qa	cs	bm25	0.25	0.0
google/flan-t5-small	20	qa	ob	qkp	0.43	0.02
google/flan-t5-small	20	qa	ob	random	0.35	0.0
google/flan-t5-small	20	qa	ob	topk	0.4	0.0
google/flan-t5-small	20	qa	ob	bm25	0.4	0.0
google/flan-t5-small	20	sa	rt	qkp	0.47	0.02
google/flan-t5-small	20	sa	rt	random	0.45	0.0
google/flan-t5-small	20	sa	rt	topk	0.6	0.0
google/flan-t5-small	20	sa	rt	bm25	0.5	0.0
google/flan-t5-small	20	sa	sst2	qkp	0.3	0.07
google/flan-t5-small	20	sa	sst2	random	0.3	0.0
google/flan-t5-small	20	sa	sst2	topk	0.4	0.0
google/flan-t5-small	20	sa	sst2	bm25	0.3	0.0
google/flan-t5-small	20	sa	rt	qkp	0.75	0.14
google/flan-t5-small	20	sa	rt	random	0.75	0.0
google/flan-t5-small	20	sa	rt	topk	0.8	0.0
google/flan-t5-small	20	sa	rt	bm25	0.7	0.0
google/flan-t5-small	20	sa	sst2	qkp	0.37	0.08
google/flan-t5-small	20	sa	sst2	random	0.45	0.0

google/flan-t5-small	20	sa	sst2	topk	0.5	0.0
google/flan-t5-small	20	sa	sst2	bm25	0.45	0.0
google/flan-t5-small	20	sa	rt	qkp	0.42	0.02
google/flan-t5-small	20	sa	rt	random	0.5	0.0
google/flan-t5-small	20	sa	rt	topk	0.6	0.0
google/flan-t5-small	20	sa	rt	bm25	0.55	0.0
google/flan-t5-small	20	sa	sst2	qkp	0.53	0.02
google/flan-t5-small	20	sa	sst2	random	0.6	0.0
google/flan-t5-small	20	sa	sst2	topk	0.6	0.0
google/flan-t5-small	20	sa	sst2	bm25	0.6	0.0
google/flan-t5-small	20	sa	rt	qkp	0.5	0.04
google/flan-t5-small	20	sa	rt	random	0.45	0.0
google/flan-t5-small	20	sa	rt	topk	0.5	0.0
google/flan-t5-small	20	sa	rt	bm25	0.45	0.0
google/flan-t5-small	20	sa	sst2	qkp	0.62	0.06
google/flan-t5-small	20	sa	sst2	random	0.55	0.0
google/flan-t5-small	20	sa	sst2	topk	0.75	0.0
google/flan-t5-small	20	sa	sst2	bm25	0.55	0.0
gpt3	20	qa	cs	qkp	0.43	0.02
gpt3	20	qa	cs	qkp	0.52	0.08
gpt3	20	qa	cs	random	0.55	0.0
gpt3	20	qa	cs	random	0.55	0.0
gpt3	20	qa	cs	topk	0.58	0.02
gpt3	20	qa	cs	topk	0.5	0.09
gpt3	20	qa	cs	bm25	0.55	0.0
gpt3	20	qa	cs	bm25	0.5	0.05
gpt3	20	qa	ob	qkp	0.75	0.04
gpt3	20	qa	ob	qkp	0.74	0.03

gpt3	20	qa	ob	random	0.8	0.0
gpt3	20	qa	ob	random	0.8	0.0
gpt3	20	qa	ob	topk	0.77	0.02
gpt3	20	qa	ob	topk	0.76	0.02
gpt3	20	qa	ob	bm25	0.8	0.0
gpt3	20	qa	ob	bm25	0.78	0.03
gpt3	20	sa	rt	qkp	0.62	0.02
gpt3	20	sa	rt	qkp	0.78	0.16
gpt3	20	sa	rt	random	0.65	0.0
gpt3	20	sa	rt	random	0.8	0.15
gpt3	20	sa	rt	topk	0.6	0.0
gpt3	20	sa	rt	topk	0.77	0.17
gpt3	20	sa	rt	bm25	0.7	0.0
gpt3	20	sa	rt	bm25	0.8	0.1
gpt3	20	sa	sst2	qkp	0.38	0.06
gpt3	20	sa	sst2	qkp	0.67	0.29
gpt3	20	sa	sst2	random	0.3	0.0
gpt3	20	sa	sst2	random	0.58	0.27
gpt3	20	sa	sst2	topk	0.3	0.0
gpt3	20	sa	sst2	topk	0.6	0.3
gpt3	20	sa	sst2	bm25	0.3	0.0
gpt3	20	sa	sst2	bm25	0.6	0.3
bigscience/bloom-560m	20	qa	cs	qkp	0.0	0.0
bigscience/bloom-560m	20	qa	cs	qkp	0.15	0.17
bigscience/bloom-560m	20	qa	cs	random	0.1	0.0
bigscience/bloom-560m	20	qa	cs	random	0.15	0.05
bigscience/bloom-560m	20	qa	cs	topk	0.15	0.0
bigscience/bloom-560m	20	qa	cs	topk	0.23	0.07

bigscience/bloom-560m	20	qa	cs	bm25	0.25	0.0
bigscience/bloom-560m	20	qa	cs	bm25	0.25	0.0
bigscience/bloom-560m	20	qa	ob	qkp	0.08	0.02
bigscience/bloom-560m	20	qa	ob	qkp	0.08	0.05
bigscience/bloom-560m	20	qa	ob	random	0.15	0.0
bigscience/bloom-560m	20	qa	ob	random	0.15	0.0
bigscience/bloom-560m	20	qa	ob	topk	0.05	0.0
bigscience/bloom-560m	20	qa	ob	topk	0.1	0.05
bigscience/bloom-560m	20	qa	ob	bm25	0.0	0.0
bigscience/bloom-560m	20	qa	ob	bm25	0.05	0.05
bigscience/bloom-560m	20	qa	cs	qkp	0.13	0.02
bigscience/bloom-560m	20	qa	cs	qkp	0.15	0.03
bigscience/bloom-560m	20	qa	cs	random	0.15	0.0
bigscience/bloom-560m	20	qa	cs	random	0.15	0.0
bigscience/bloom-560m	20	qa	cs	topk	0.1	0.0
bigscience/bloom-560m	20	qa	cs	topk	0.12	0.02
bigscience/bloom-560m	20	qa	cs	bm25	0.15	0.0
bigscience/bloom-560m	20	qa	cs	bm25	0.15	0.0
bigscience/bloom-560m	20	qa	ob	qkp	0.1	0.07
bigscience/bloom-560m	20	qa	ob	qkp	0.11	0.05
bigscience/bloom-560m	20	qa	ob	random	0.05	0.0
bigscience/bloom-560m	20	qa	ob	random	0.07	0.03
bigscience/bloom-560m	20	qa	ob	topk	0.05	0.0
bigscience/bloom-560m	20	qa	ob	topk	0.03	0.03
bigscience/bloom-560m	20	qa	ob	bm25	0.0	0.0
bigscience/bloom-560m	20	qa	ob	bm25	0.03	0.03
bigscience/bloom-560m	20	sa	rt	qkp	0.05	0.07
bigscience/bloom-560m	20	sa	rt	qkp	0.02	0.06

bigscience/bloom-560m	20	sa	rt	random	0.05	0.0
bigscience/bloom-560m	20	sa	rt	random	0.17	0.12
bigscience/bloom-560m	20	sa	rt	topk	0.05	0.0
bigscience/bloom-560m	20	sa	rt	topk	0.23	0.17
bigscience/bloom-560m	20	sa	rt	bm25	0.05	0.0
bigscience/bloom-560m	20	sa	rt	bm25	0.15	0.1
bigscience/bloom-560m	20	sa	sst2	qkp	0.02	0.02
bigscience/bloom-560m	20	sa	sst2	qkp	0.19	0.25
bigscience/bloom-560m	20	sa	sst2	random	0.05	0.0
bigscience/bloom-560m	20	sa	sst2	random	0.3	0.25
bigscience/bloom-560m	20	sa	sst2	topk	0.1	0.0
bigscience/bloom-560m	20	sa	sst2	topk	0.35	0.25
bigscience/bloom-560m	20	sa	sst2	bm25	0.0	0.0
bigscience/bloom-560m	20	sa	sst2	bm25	0.25	0.25
bigscience/bloom-560m	20	sa	rt	qkp	0.17	0.24
bigscience/bloom-560m	20	sa	rt	qkp	0.08	0.19
bigscience/bloom-560m	20	sa	rt	random	0.0	0.0
bigscience/bloom-560m	20	sa	rt	random	0.0	0.0
bigscience/bloom-560m	20	sa	rt	topk	0.45	0.0
bigscience/bloom-560m	20	sa	rt	topk	0.23	0.23
bigscience/bloom-560m	20	sa	rt	bm25	0.45	0.0
bigscience/bloom-560m	20	sa	rt	bm25	0.23	0.23
bigscience/bloom-560m	20	sa	sst2	qkp	0.15	0.21
bigscience/bloom-560m	20	sa	sst2	qkp	0.07	0.17
bigscience/bloom-560m	20	sa	sst2	random	0.55	0.0
bigscience/bloom-560m	20	sa	sst2	random	0.28	0.28
bigscience/bloom-560m	20	sa	sst2	topk	0.6	0.0
bigscience/bloom-560m	20	sa	sst2	topk	0.57	0.02

bigscience/bloom-560m 20 sa sst2 bm25 0.0 0.0

bigscience/bloom-560m 20 sa sst2 bm25 0.0 0.0