

# ActiveLLM: Large Language Model-Based Active Learning for Textual Few-Shot Scenarios

Markus Bayer<sup>◇</sup> and Justin Lutz and Christian Reuter

PEASEC

Technical University of Darmstadt, Germany

<sup>◇</sup>bayer@peasec.tu-darmstadt.de

## Abstract

Active learning is designed to minimize annotation efforts by prioritizing instances that most enhance learning. However, many active learning strategies struggle with a ‘cold-start’ problem, needing substantial initial data to be effective. This limitation reduces their utility in the increasingly relevant few-shot scenarios, where the instance selection has a substantial impact. To address this, we introduce ActiveLLM, a novel active learning approach that leverages Large Language Models such as GPT-4, o1, Llama 3, or Mistral Large for selecting instances. We demonstrate that ActiveLLM significantly enhances the classification performance of BERT classifiers in few-shot scenarios, outperforming traditional active learning methods as well as improving the few-shot learning methods ADAPET, PERFECT, and SetFit. Additionally, ActiveLLM can be extended to non-few-shot scenarios, allowing for iterative selections. In this way, ActiveLLM can even help other active learning strategies to overcome their cold-start problem. Our results suggest that ActiveLLM offers a promising solution for improving model performance across various learning setups.

## 1 Introduction

The selection of training examples significantly impacts the performance of models. Even Large Language Models (LLMs), such as GPT-3, show high variances based on training example selection (Zhang et al., 2022a). This dependency is even more pronounced in smaller models such as BERT (Zhang et al., 2022a), which are sometimes preferred for their cost and resource efficiency as well as data protection benefits over APIs. In order to combine the advantages of both the light-

ness of BERT-like models and the capabilities of LLMs, we explore the classical concept of Active Learning (AL) for BERT-like models, extended by LLMs such as GPT-4 (OpenAI, 2023).

AL encompasses strategies integrated into the labeling process to select instances with a high learning impact (Zhang et al., 2022b). Typically, a model is trained iteratively during the annotation process to query instances for labeling by the annotators (oracle). While AL can yield significant learning improvements with mid-sized datasets, it often encounters a ‘cold-start’ problem (Chen et al., 2023), rendering it unsuitable for low-data regimes. The cold-start problem occurs in many AL methods because they lack sufficient data at the start of data labeling to accurately measure informativeness and select informative samples. For example, the uncertainty strategy selects the most uncertain instances based on an iteratively learned classifier during annotation. However, the classifier cannot make accurate uncertainty guesses without enough data.

Furthermore, as most AL strategies require a model to be trained iteratively, the use of pre-trained models such as BERT can result in very high delay times during annotation, making the process unusable in real-world annotation scenarios. In so-called model-mismatch scenarios, where the instance selection model (query model) differs from the model used for the final application (successor model), AL yields limited gains (Zhang et al., 2022b).

To address these challenges of few-shot and model-mismatch scenarios, we propose ActiveLLM, an AL method leveraging LLMs (see Figure 1). ActiveLLM can select instances with high learning impact even without initial supervised data and requires no training during the annotation process. This method serves both as a standalone AL approach and as a solution

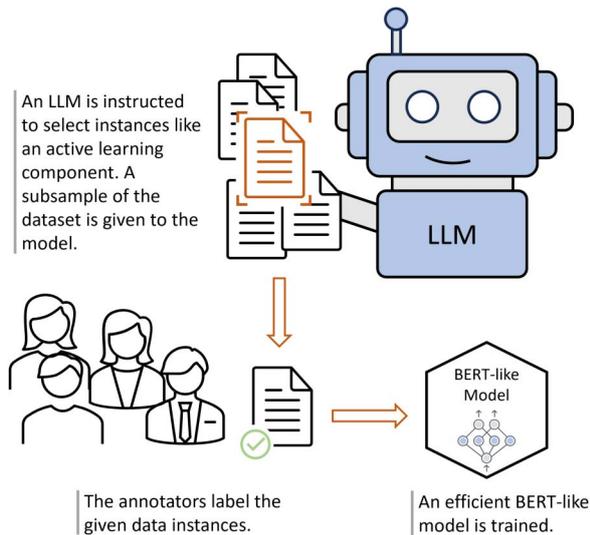


Figure 1: Depiction of ActiveLLM.

to the cold-start problem in other AL strategies. Therefore, our contributions are as follows:

**(C1) Novel AL Method:** We introduce a novel AL strategy using LLMs such as GPT-4, tailored for few-shot learning to overcome the cold-start problem.

**(C2) Efficient and Scalable Methodology:** Our approach decouples the query process from the successor model dependency, enhancing scalability and establishing a high degree of practicality compared to conventional AL strategies.

**(C3) Rigorous Evaluation:** We conduct comprehensive evaluations to demonstrate that ActiveLLM is not constrained to a specific LLM. It exhibits superior performance compared to other AL and few-shot learning methods and can be utilized alongside these methods to achieve beneficial outcomes.

The code for this study is freely available.<sup>1</sup>

## 2 Related Work

### 2.1 Active Learning

The rationale behind AL is that a system selectively chooses instances to be labeled, thereby reducing the labeling effort (Settles, 2012). In this context, annotators serve as oracles, responding to queries from the AL system regarding specific instances. This system utilizes a query strategy that often hinges on measures of uncertainty or diversity. Traditionally, the querying is performed by

the machine learning model itself, which undergoes iterative training during the labeling process. For instance, the model might request labels for instances where its uncertainty is highest, using approaches such as Least Confidence (LC) (Cohn et al., 1994), Margin of Confidence (Tong and Koller, 2001), or Prediction Entropy (PE) (Settles and Craven, 2008). It is then retrained with the newly annotated data.

In addition to uncertainty-based methods, diversity-based approaches also play a critical role in AL. These methods aim to select instances that are diverse to ensure the labeled dataset covers a wide range of feature space. For example, techniques like Embedding KMeans (EKM) (Schröder et al., 2021; Yuan et al., 2020) leverage the model’s ability to embed instances into a latent space. By applying clustering algorithms such as KMeans in this space, the system can identify and select the most diverse examples, thereby enriching the training set with varied and representative data points.

#### 2.1.1 Active Learning with Transformer Models

While AL methods have demonstrated their utility across various traditional machine learning models, their application to transformer models remains challenging. The study by Ein-Dor et al. (2020), among the first to explore AL with pre-trained transformer models such as BERT, shows that AL can enhance the performance of BERT classifiers, particularly in low-data scenarios and when the initial dataset is rich in relevant class instances. However, most of the other works indicate mixed results. For example, Jacobs et al. (2021) combine uncertainty and diversity-based methods, utilizing SentenceBERT embeddings to refine the instance selection process. Their findings indicate that while AL reduces labeling effort, its effectiveness with BERT models is mixed, and the improvements are modest compared to older NLP models. Studies by Yuan et al. (2020), Schröder et al. (2022), Seo et al. (2022), Griebhaber et al. (2020), and Houlshy et al. (2011) contribute to a similar mixed picture of AL methods for transformer models. In Table 1, we present an overview of works utilizing AL for transformer models, categorizing them by query strategies, tested models, and model matching/mismatching scenarios.

<sup>1</sup><https://github.com/PEASEC/ActiveLLM>.

Reference	Query Strategies	Model	Matching	Comparison To Random
(Ein-Dor et al., 2020)	LC, Perceptron Ensemble, EGL, Core-Set, D-AL	BERT	✓	Improvements in imbalanced low-data scenarios
(Jacobs et al., 2021)	VR, PE, BALD, Diversity Heuristics	BERT	✓	Mixed results
(Yuan et al., 2020)	ALPS, PE, BERT-KM BADGE	BERT	✓	Mixed results
(Schröder et al., 2022)	PE, Breaking Ties, LC, C-AL	BERT distilRoBERTa	✓	Small improvements in later iterations
(Seo et al., 2022)	BATL, LC, ALPS, PE, BALD, D-AL, Core-Set, BADGE	DISTRE, BERT SCIBERT	✓	Mostly improvements
(Grießhaber et al., 2020)	BALD	BERT base	✓	Mixed results
(Shelmanov et al., 2021)	MNLP, VR, BALD	BERT base, DistilBERT base, ELECTRA base	✓	Significant improvements
(Shelmanov et al., 2021)	MNLP, VR, BALD	BERT, DistilBERT, ELECTRA	✗	Significant improvements when mismatch is of the form: distilled transformer (query) and normal transformer (successor)
(Tsvigun et al., 2022)	LC, MNLP, Mahalanobis Distance	(Distil-)BERT, (Distil-)RoBERTa, (Distil-)ELECTRA, XL-Net	✗	Significant improvements when successor model (from distilled mismatch setting) is used for pseudo-labeling an independent model
(Nguyen et al., 2022)	MNLP, BERT-KM, BADGE, ALPS	XLM-RoBERTa, miniLM	✗	Significant improvements when successor model (from distilled mismatch setting) is included in query process by giving delayed feedback to the distilled model

Table 1: Overview of different works applying AL to transformer models. ‘‘Model’’ describes the successor model trained from the AL process. ‘‘Matching’’ indicates if the successor model matches the query model.

**Research Gap:** The most significant gap in AL research is the use of LLMs, such as GPT-4, to improve AL with BERT-like transformer models, which, to our knowledge, has not been explored previously.

### 2.1.2 Model-Mismatch Scenarios

As transformer models often require substantial computational resources and time (Treviso et al., 2022), the training runtimes for those can be prohibitively long during labeling, making them impractical for real-world applications of AL. Schröder et al. (2022) noted that the incremental training of these models can be so time-consuming that it can negate the cost savings from reduced

labeling effort. Therefore, the scope of AL has been extended to include scenarios where the querying model may differ from the final (successor) model—a phenomenon known as model mismatch—or situations where no model is used for querying.

But again many studies (Baldrige and Osborne, 2004; Schröder et al., 2022; Shelmanov et al., 2021) report that a model mismatch between the query and successor model often leads to unsatisfactory results (Zhang et al., 2022b). While efforts by Shelmanov et al. (2021) and Nguyen et al. (2022) attempt to address this by using smaller, similar models for querying, there remains a dependency on the model type. Similarly, Tsvigun et al. (2022) offer a promising

approach for decoupling the query and successor models, although a distilled version is still required in the process.

**Research Gap:** In contrast to the use of smaller, resource-efficient models for querying, we propose leveraging much larger language models. This approach may seem counterintuitive at first due to concerns about long runtimes, but as these models are capable of zero-shot learning, no training is required (Shliazhko et al., 2022). Furthermore, these models can be accessed in real-time through several free chat interfaces, requiring no machine learning expertise, making them ideal for practical applications.

### 2.1.3 Cold-Start Problem

While AL can significantly streamline the learning process by reducing labeling efforts, most strategies encounter a cold-start problem. In the absence of sufficient initial labeled data (label seed), the AL system may struggle to make informed predictions about uncertainty or diversity (Yuan et al., 2020). This presents a major challenge for low-data regimes, particularly in few-shot scenarios with very few available data instances. Yuan et al., (2020) propose ALPS, which is one of the only works that directly addresses the cold-start problem in AL with transformer models. ALPS leverages BERT’s pre-existing masked language modeling objective along with clustering to select instances. The authors hypothesize that this strategy provides more reliable confidence scores in the initial stages of model training than a classifier head. However, the results are mixed which is also later confirmed by Nguyen et al. (2022).

**Research Gap:** The cold-start problem is a major challenge in AL research, especially for few-shot scenarios. This work aims to overcome this challenge by proposing an AL strategy that needs no initial labeled data, enabling improvements even at the very beginning of the AL process. This allows the method to also be used to address the cold-start problem of other AL methods by providing them with initial data.

## 2.2 Few-Shot Learning

Few-shot learning refers to methods for scenarios where models need to learn from a limited number of training instances. These scenarios are becoming increasingly relevant due to their importance in real-world problems (Yu and Raschka, 2020) and the increasing capabilities of deep

learning models. While LLMs such as GPT-3 inherently have robust few-shot or zero-shot capabilities through in-context learning, our focus is on enhancing few-shot learning in smaller models such as BERT. With specific few-shot learning methods like ADAPET (Tam et al., 2021), these smaller language models can attain similar or even better few-shot performance than GPT-3.

One approach to this is to formulate tasks as cloze-style tests, where certain words in a text instance are missing and the language model is employed to fill in the blanks. This way, no classifier needs to be trained on top of the language model, leading to a more effective utilization of the language models (Gao et al., 2020). Such an approach is exemplified by the ADAPET method (Tam et al., 2021), as well as in the works of Gao et al. (2020), Zhang et al. (2021), and Schick and Schütze (2020a). PERFECT, introduced by Mahabadi et al. (2022), eliminates the need to manually formulate such cloze tasks by incorporating task-specific adapters and specialized label embeddings. Similarly, in SetFit by Tunstall et al. (2022), no cloze tasks are required, as a sentence transformer (Reimers and Gurevych, 2019) is fine-tuned on the available training data. The instances are then encoded with the resulting model and a regular classification head is trained on them.

While not few-shot methods per se, transfer learning and data augmentation are often employed to address few-shot scenarios. Transfer learning allows models to leverage knowledge from related domains (Ruder et al., 2019), providing a strong foundation in few-shot scenarios. Data augmentation, particularly using LLMs (Ding et al., 2024; Wang et al., 2022; Xu et al., 2024), has been used to generate high-quality synthetic data in low-data regimes (Zheng et al., 2023). Similarly, in this study we investigate whether AL can be useful for few-shot learning, especially by overcoming the cold-start problem.

**Research Gap:** In general, LLMs dominate the machine learning field, in particular, in few-shot learning, but there are numerous reasons why researchers and practitioners still rely on smaller BERT-like models, especially for inference, where concerns such as memory limitations, API costs, and privacy often preclude the use of LLMs (Wang et al., 2024a; Samuel, 2024; Bosley et al., 2023). To the best of our knowledge, our approach is the first to leverage the capabilities

Consider yourself in the position of an active learning component to help a human annotator. You have to choose the instances that the annotator has to label. You are given {the guidelines for the task and} a set of instances of a dataset. You can only choose {32} instances. {You would ideally want to choose those instances that would provide the most informative and diverse data for the model. Here are some strategies to consider: *advices*}

{Label Guidelines for the human annotator: *guidelines*}

{The output format should be a list of the instances that you would label, separated by a comma.}  
 {Please think step by step about what you would do to select the instances to label. After this provide the list of instances that you would label, separated by a comma.}

{Please describe instance by instance why you would select or not select it for labelling. Do not stop before you successfully found {32} that you would suggest to label. After this provide the list of the instances that you would label, separated by a comma.}

For example, if you would label the instances 1, 4, 5 then the output should be: 1, 4, 5. The following instances are given to you (separated with "\n ##### \n"):

Guidelines  
 Instances  
 Advice

Guidelines

No CoT  
 Normal CoT

Explanation  
 Instances

Figure 2: Prompt design of ActiveLLM in few-shot learning mode.

of LLMs to support instance selection for training smaller BERT-like models. In doing so, we combine the flexibility and efficiency of smaller models with the strategic guidance of LLMs.

While Ein-Dor et al. (2020) and Griebhaber et al. (2020) also assess the performance of AL with BERT in low-resource scenarios, their datasets are still considerably larger than those in few-shot scenarios. However, in scenarios with very few instances, example selection has the greatest impact. In this work, we aim to improve state-of-the-art few-shot learning methods with our AL approach.

### 3 Method

ActiveLLM is a pool-based sampling method that operates in batch mode, meaning that it selects a subset from a pool of unlabeled data for querying an oracle. It employs instruction-tuned LLMs as query models, while allowing the choice of a successor model to be independent of these models. A demonstration of the process is given in Figure 1, where the LLM is instructed to select instances, which are then labeled and used to train a BERT-like model, allowing any such model to be trained.

The design of the prompts is critical for achieving optimal results with instruction-tuned LLMs. To this end, we have crafted detailed prompts in various configurations to determine their effectiveness in this context. We differentiate between two modes of ActiveLLM. The first addresses few-shot scenarios, where ActiveLLM is executed only once on the dataset. The second is suitable for general scenarios involving iterative querying,

which also incorporates feedback from previous iterations.

#### 3.1 Few-Shot Learning Mode

The prompts are constructed so that the language model receives a description of AL, including exact details of its tasks, while also keeping the context length small, as this is a limiting factor for current models (Liu et al., 2023). Given a specific task, ActiveLLM creates a prompt consisting of AL role allocation, instructions on the selection process, a description of how to format the output, followed by a batch of unlabeled instances.

An excerpt of the prompt design is illustrated in Figure 2. The parameters displayed on the right indicate certain text phrase constellations which can be included, excluded, or adjusted:

*Guidelines:* This includes a phrase indicating that the model is given not only a set of instances but also guidelines for the task.

*Instances:* This parameter determines how many instances the model should label. We test different values in Section 4.2.3.

*Advice:* Initial experiments showed that the LLMs often respond with explaining general strategies for choosing instances in an AL manner. Hence, we included the parameter ‘advice,’ which incorporates these strategies.

*No CoT/CoT/Explanation:* One of the most common and beneficial prompt engineering strategies is the use of Chain of Thought (CoT) prompting (Wei et al., 2022). We experiment with no CoT, ‘normal’ CoT (“think step by step”), and the task to explain the thoughts on each instance.

Finally, a batch of unlabeled instances is appended to the prompt. We hypothesize that, contrary to common AL strategies, large batches or even all instances do not need to be evaluated by the query model to find instances with high learning impact. We test different values of this parameter in Section 4.2.2.

### 3.2 Iterated Querying Mode

In the second variant, we design ActiveLLM similar to other AL strategies, allowing the LLM to query new instances repeatedly. For this, it is sensible to include data that was selected in previous iterations. As the prompt should not become too large, we consider three additional parameters (see Appendix A and Figure 9 for the full prompt):

*No Recap:* Indicating the baseline, where no instances from previous queries are recalled.

*Recap:* The instances from previous queries are directly included in the prompt.

*Index Recap:* To reduce the context size, the indices of the instances from previous queries are included in the prompt.

For the latter two scenarios, we included instructions in the prompt to indicate that instances from previous iterations are incorporated. In both cases, we did not include the annotated labels, as our experiments showed that they were generally disregarded by the language models. An example of a specific prompt and LLM answer is given in Appendix Figure 10.

## 4 Experiments

### 4.1 Description

Our experimentation perspective with ActiveLLM focuses on addressing the following questions:

1. How should the prompt be designed? (Section 4.2)
2. Can the chosen prompt be applied to other models and datasets? (Section 4.3)
3. How does ActiveLLM compare to other AL strategies? (Section 4.4)
4. Can ActiveLLM improve state-of-the-art few-shot learning methods? (Section 4.5)
5. How does the method perform in non-few-shot-scenarios? (Section 4.6.1)

6. Is ActiveLLM capable of overcoming the cold-start problem in other AL strategies? (Section 4.6.2)

As is common in research, our experiments simulate the AL cycle. To do this, we take a subset of the training set (labels removed) for the respective task and allow the LLM to select a certain number of instances based on the prompt given in Section 3. These instances are then assigned the true labels as if they had been annotated by a perfect annotator.

#### 4.1.1 Datasets

For the initial prompt engineering experiments, we are interested in a less common dataset, as an LLM might be less biased by data leakage. We chose the Specialized CTI dataset (Bayer et al., 2022), where Twitter posts are classified according to their relevance to specific CTI events (further details in Appendix B). In the general applicability tests, we used the best-performing prompt from the prompt engineering phase on the commonly-used GLUE benchmark. For comparison with the few-shot learning method SetFit, we use the AGNews dataset (Zhang et al., 2015), for which an adaptation of the method is available. The SST-2 dataset (Socher et al., 2013), previously used in the general applicability section, is also employed in the non-few-shot experiments. To measure the performance we use the test sets for CTI and AGNews, and the validation sets for GLUE. Unless otherwise specified (see Sections 4.2.3 and 4.6), we select 32 instances per task, either via AL or random sampling for the baseline, which represents a common few-shot learning setting (Tam et al., 2021; Schick and Schütze, 2020b; Mahabadi et al., 2022).

During the prompt engineering experiments, we aim to minimize random factors by generating five different randomizations of the CTI task and applying both the baseline and each prompt to these variations. For each evaluation, we select five different seeds for training the successor model, with the reported results for the baseline and each prompt being the average score from 25 runs (5 dataset randomizations times 5 trained models). For subsequent experiments, we create one randomization of the datasets but continue to train five models and provide the average score.

### 4.1.2 Models and Hyperparameters

In our experiments, we primarily employed GPT-4 (OpenAI, 2023) as the query model for ActiveLLM, as it achieved top results at the time the main experiments were conducted. However, in Section 4.3 we also experiment with other LLMs: o1 (OpenAI, 2024), GPT-4o, GPT-3.5 (Brown et al., 2020), Llama 3 70B, Gemini-Ultra 1.0 (Anil et al., 2023), Mistral Large, and Mixtral 8×7B (Jiang et al., 2024) (ActiveLLM is renamed according to the used LLM; e.g., GPT-4: ActiveGPT4). To underscore the practicality of our method, we utilize chat versions of the LLMs to simulate realistic interactions within chat environments (further details in Appendix C).

As a successor model for the selected instances of ActiveLLM, we chose BERT (Devlin et al., 2018) and RoBERTa (Liu et al., 2019). While foundational works such as Devlin et al. (2018) typically train BERT-like models for only 3-5 epochs, other studies (Gao et al., 2020; Mosbach et al., 2020) suggest that higher epoch counts are necessary for stable and optimal performance in low-data regimes. Consequently, we have opted for 25 epochs for our experiments. Otherwise, we adhere to the default HuggingFace transformer parameters, with a learning rate of 1e-3, a weight decay of 0.01, and a dropout probability of 0.1.

For comparison, we include in each experiment a random sampling baseline where the first instances from a shuffled dataset are selected. In addition, we compare ActiveLLM with the most commonly used AL strategies for transformers identified in our literature overview (see Table 1): LC, BALD, EKM, and PE. The strategies are implemented in a model-matching scenario (i.e., training the same query and successor model), can query the entire dataset of each task, and select 5 instances until the selection of 32 is reached. ActiveLLM, on the other hand, is implemented in a model-mismatch scenario, can only query a subset of the full training set, and selects the 32 instances in one query (unless otherwise specified). To implement the common AL strategies, we use the small-text library (Schröder et al., 2021), which provides a unified framework for pool-based AL with support for transformer-based models.

## 4.2 Prompt Engineering Experiments

*How should the prompt be designed?*

	A	CoT	G	F1 (SD)
Baseline	–	–	–	0.6781 (0.029)
Prompt A1	✓	✗	✗	0.6522 (0.100)
Prompt A2	✓	✓	✗	0.6951 (0.102)
Prompt A3	✓	✓+	✗	0.6876 (0.108)
Prompt B1	✗	✗	✗	0.7135 (0.060)
Prompt B2	✗	✓	✗	<b>0.7214</b> (0.053)
Prompt B3	✗	✓+	✗	0.6271 (0.139)
Prompt C2	✗	✓	✓	<b>0.7329</b> (0.053)

Table 2: Prompt engineering results of the baseline and ActiveGPT4 on the CTI dataset sampling 32 instances. The results are averaged over 25 runs (5 random dataset and 5 random model initializations). A (advice), G (guidelines), C (no advice, but guidelines), 1 (no CoT), 2 (CoT: step-by-step), 3 (CoT+: explain each instance).

To answer the first question, we experiment with different configurations of ActiveLLM, such as whether to include AL strategy advices or guidelines, as shown in Figure 2. We then evaluate the effects of varying batch sizes for the unlabeled data used in the prompts. Finally, we compare different selection sizes, i.e., the number of instances the LLM should select. While we want to find the most appropriate prompt, our goal is not to engineer every detail of it, which is left open for future work and practitioners.

### 4.2.1 Configuration Results

Table 2 presents the results of various prompt configurations. For these, we use a batch of 300 presented unlabeled instances. Prompt configurations A and B differ in whether the advice on how to select the instances is already included in the prompt. We observed that GPT-4 tends to reiterate the advice even when it is provided in the prompt. Furthermore, the results suggest that it is preferable not to include the advice in the prompt, likely due to the increased context size when the advice is repeated in both the prompt and the response.

We also test the three CoT variants: A1 and B1, where there is no CoT; A2 and B2, which use the standard CoT approach (“think step by step”); and A3 and B3, which instruct to include explanations for each instance. The common “think step by step” instruction yielded the best results. This might be because this approach allows GPT more

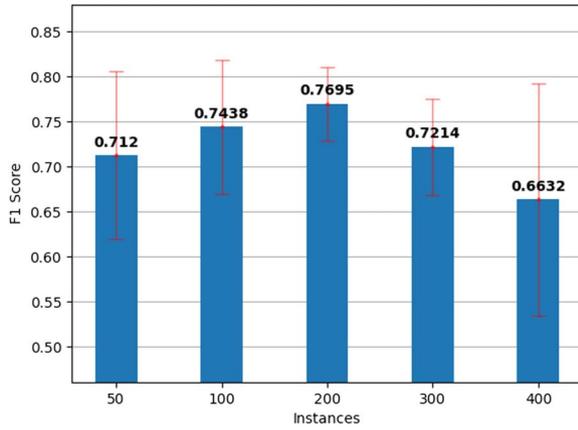


Figure 3: Varying size of the batch of unlabeled instances from which ActiveGPT4 can select 32 instances. All results are averaged over 5 runs on the CTI dataset (F1).

reasoning in structured steps, while not increasing the context size too much.

Finally, we experimented with including labeling guidelines in the prompt. As anticipated, this prompt configuration produced the best results, likely because it enabled the model to better differentiate between instances. Although labeling guidelines are often created prior to the labeling process, they are rarely included in the dataset for common tasks. While the guidelines are available for the CTI task, its inclusion in the prompt should be seen as an exploration of what can be achieved rather than the default case for ActiveLLM. Therefore, in future evaluations, we will utilize ActiveGPT4 with prompt B2, i.e., no guidelines or additional advice on AL, only employing the simple “think step by step” CoT prompting.

#### 4.2.2 Presented Batch Size Results

In the subsequent experiment, we explore varying batch sizes of unlabeled instances presented to ActiveGPT4. Figure 3 displays the results for 50, 100, 200, 300, and 400 examples. There is a noticeable trade-off between the context length and the number of presented examples. While a larger pool of examples generally allows for a more diverse set, an increase in context length tends to degrade the model’s performance. It can be observed that the optimal prompt with 300 examples, as used in the previous section, could be significantly enhanced by reducing the size to 200. Consequently, for the primary experiments in Section 4.3, we limited the number of instances

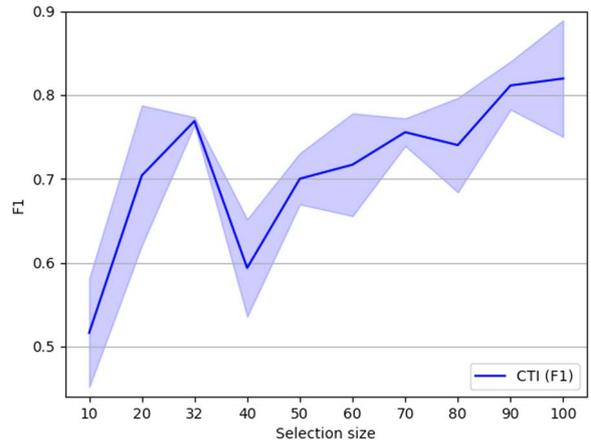


Figure 4: Varying selection size of examples to be selected by ActiveGPT4. All results (F1) are averaged over 5 runs on the CTI dataset.

presented to the model to 200, contingent on the model’s capacity to handle such a context size.

#### 4.2.3 Selection Size Results

In our final experiment on prompt engineering, we investigate the impact of the selection size—how many instances the model should choose—on the outcomes. Many research works (Tam et al., 2021; Schick and Schütze, 2020b; Mahabadi et al., 2022) typically use 32 examples for few-shot scenarios. We evaluate whether smaller or larger sizes might be more effective for ActiveGPT4. Figure 4 illustrates varying selection sizes for the CTI dataset. Selecting 32 examples per query appears to be a sensible choice. Notably, performance declines with up to 90 examples, suggesting that the model may struggle to process such a large number of examples effectively. However, performance increases beyond this point, as expected, because a larger training set is inherently beneficial.

As part of the following experiments, we also tested the selection size on the SST-2 dataset. The results are equivalent and can be found in Appendix Figure 8.

### 4.3 General Applicability

*Can the chosen prompt be applied to other models and datasets?*

In this section, we aim to assess how ActiveLLM equipped with various LLMs and the prompt configuration B2 from Section 4.2.1 performs on the more common datasets of the GLUE benchmark.

The results of this experiment are given in Table 3. ActiveGPT4, ActiveGPT4o, Activeo1,

	QNLI	QQP	RTE	SST2	WNLI	MNLI-(m/mm)	MRPC	COLA	
Baseline	55.11	63.67	50.83	56.03	35.77	34.92/35.01	67.55	63.89	
200	GPT-4	58.94 ↑	63.99 ↑	53.57 ↑	<b>73.14</b> ↑	40.28 ↑	35.45/35.14 ↑	68.33 ↑	64.70 ↑
	GPT-4o	<b>62.76</b> ↑	63.65 ↓	48.88 ↓	66.31 ↑	41.13 ↑	32.23/32.80 ↓	69.75 ↑	65.41 ↑
	o1	62.10 ↑	64.00 ↑	54.01 ↑	59.08 ↑	39.43 ↑	33.52/33.94 ↓	63.87 ↓	<b>69.91</b> ↑
	Mistral Large	60.23 ↑	65.26 ↑	<b>55.60</b> ↑	60.64 ↑	<b>52.96</b> ↑	36.59/35.92 ↑	67.06 ↓	67.44 ↑
100	Llama 3 70B	56.57 ↑	65.82 ↑	52.27 ↑	65.71 ↑	40.28 ↑	37.19/36.68 ↑	66.86 ↓	68.15 ↑
	Mistral Large	60.68 ↑	64.82 ↑	49.48 ↓	66.86 ↑	46.48 ↑	37.41/36.58 ↑	69.07 ↑	67.56 ↑
	GPT-3.5	55.11 -	<b>65.88</b> ↑	55.09 ↑	61.15 ↑	51.55 ↑	<b>37.44/37.62</b> ↑	67.70 ↑	68.63 ↑
	Gemini-Ultra	57.60 ↑	61.18 ↓	54.58 ↑	49.08 ↓	43.10 ↑	35.74/35.77 ↑	<b>69.95</b> ↑	69.22 ↑
	Mixtral 8x7B	61.13 ↑	65.27 ↑	50.97 ↑	68.00 ↑	41.41 ↑	33.62/33.75 ↓	68.14 ↑	66.96 ↑
AL	LC	56.76 ↑	58.65 ↓	54.44 ↑	56.86 ↑	42.82 ↑	36.48/36.18 ↑	66.96 ↓	69.26 ↑
	BALD	59.91 ↑	n.a.*	52.13 ↑	69.17 ↑	44.79 ↑	36.14/36.34 ↑	68.53 ↑	64.24 ↑
	EKM	52.44 ↓	59.62 ↓	47.65 ↓	69.77 ↑	46.49 ↑	32.87/32.75 ↓	61.08 ↓	66.60 ↑
	PE	53.75 ↓	58.65 ↓	54.44 ↑	56.86 ↑	42.82 ↑	37.11/37.20 ↑	66.96 ↓	69.26 ↑

Table 3: BERT-base few-shot results (32 instances) of ActiveLLM with different LLMs and of other AL strategies (LC, BALD, EKM, and PE) on GLUE tasks. ‘Active 200/100’ describes the size of the batch of unlabeled instances. All results are averaged over 5 runs (accuracy). Standard deviations are given in Appendix Table 6.

\*No result available after 48 hours of execution.

and ActiveMistralLarge are given 200 unlabeled instances as identified in the batch size experiments (Section 4.2.2). Llama 3, GPT-3.5, Gemini-Ultra, and Mixtral cannot process such extensive prompts and are therefore given 100 instances. All LLMs show that they are very good query models for ActiveLLM, outperforming the baseline in most cases. However, we recommend using ActiveLLM with GPT-4 or GPT-3.5, as they consistently improve results across all tasks. An interesting finding is that Mistral Large with 200 examples performed worse than its 100 example counterpart in 5 out of 8 tasks. The tasks where Mistral Large performs better with 200 instances are those with generally shorter text length, suggesting that choosing a batch size based on token length may be better than a static batch size. This would also explain the variation in improvements across tasks and models, as some tasks involve many more tokens per instance, and some models perform better with larger prompts. In addition, the differences between the models may be due to the fact that the prompt design phase was carried out using GPT-4. Other models may work better and give more consistent results with a prompt design specifically for them. The most significant improvements can be observed with ActiveMistralLarge (200) and ActiveGPT4

on WNLI and SST-2, respectively, achieving improvements of 17.19 and 17.11 percentage points over the baseline.

Overall, ActiveLLM, utilizing GPT-4 and GPT-3.5, proves to be a robust approach for enhancing performance in few-shot scenarios across all tasks. While it may even produce better results with other LLMs, the improvements may be inconsistent and should be compared to a baseline.

To verify that our results are not specific to BERT, we repeated the experiments from Table 3 using RoBERTa. The results, shown in Appendix Table 7, confirm that ActiveLLM also improves performance on a different BERT-based architecture. As with BERT, GPT-4 and GPT-3.5 give more consistent results than LLaMA, Mistral/Mixtral, and Gemini-Ultra.

#### 4.4 Comparison

*How does ActiveLLM compare to other AL strategies?*

We are interested to see how ActiveLLM performs against other AL methods. In Table 3 we also included the results for the most used AL strategies with transformers identified in our literature overview (see Table 1): LC, BALD, EKM, and PE. While these methods are implemented with the best preconditions with a model-matching

	QNLI	QQP	RTE	SST2	WNLI	MNLI	MRPC	COLA	
Average	265.26	279.11	11.01	170.34	5.17	627.12	12.31	25.29	
AL	LC	81.58	271.40	6.39	53.11	3.55	297.00	5.50	11.07
	BALD	812.26	n.a*	22.34	520.11	8.37	1610.23	30.54	68.26
	EKM	85.16	289.39	7.23	53.53	4.40	306.43	6.08	10.02
	PE	82.02	276.54	6.47	53.01	4.37	293.24	5.58	10.22

Table 4: Runtime of the AL strategies (LC, BALD, EKM, and PE) on the GLUE tasks from Table 3. The runtime (in minutes) refers to the total time required to select 32 instances over 6 AL iterations. \*No result available after 48 hours of execution.

scenario and access to the whole dataset of each task, they are often not able to improve over the baseline and in every case worse than an LLM-variant of ActiveLLM. This outcome is expected, as these AL strategies face a cold-start problem. Therefore, in Section 4.6.2 we test whether ActiveLLM can be used to overcome the cold-start problem of these techniques. Furthermore, the practical use of these methods is limited, as the selection of examples to be labeled required at least several minutes and, in the case of larger datasets, several hours (see Table 4). For instance, with BALD on the QQP dataset, the approach did not yield any result even after 48 hours of execution. In contrast, ActiveLLM eliminates the need for model training during the AL process and only considers a small subset of the unlabeled data, enabling query times of just a few seconds while still achieving higher improvements.

Standard deviations for the results of Table 3 are given in Appendix Table 6. While the values vary across tasks and models, the overall trend indicates that ActiveLLM does not amplify performance instability, and in many cases, can reduce it substantially.

#### 4.5 ActiveLLM and Few-Shot Learning

*Can ActiveLLM improve state-of-the-art few-shot learning methods?*

While showing significant improvements in few-shot scenarios compared to random sampling and other AL strategies, one might ask why not directly use few-shot learning methods. Therefore, we evaluate the performance of ActiveGPT4 both individually and in combination with the state-of-the-art few-shot learning methods ADAPET, PERFECT, and SetFit. For each method, we have to use a different dataset for which an implementation is available: ADAPET

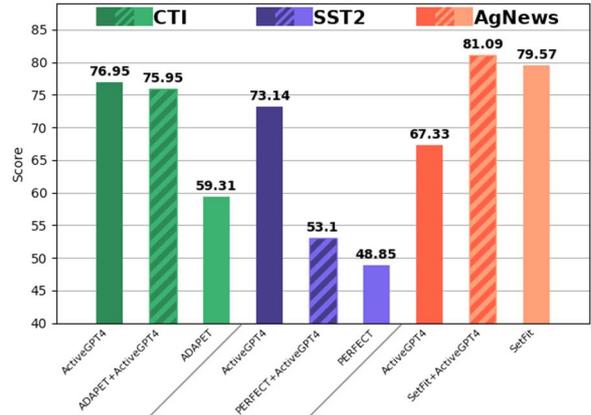


Figure 5: Comparison of ActiveGPT4, both independently and in combination with few-shot methods, across different tasks (32 instances). The methods include ADAPET (evaluated using BERT-base on CTI with F1), PERFECT (using RoBERTa-Large on SST2 with accuracy), and SetFit (using BERT-base on AGNews with accuracy).

with CTI, PERFECT with SST-2<sup>2</sup>, and SetFit with AgNews.

The results of this experiment are shown in Figure 5. ActiveGPT4 significantly outperforms the few-shot learning methods ADAPET and PERFECT (+ 17.64% + 24.29%). While not outperforming SetFit on its own, ActiveLLM, when combined with each few-shot learning method, significantly improves all of these methods. This shows that few-shot learning methods can be improved through the selection of instances, and in some cases, this selection is even more important than the use of few-shot learning methods.

#### 4.6 Non-Few-Shot Scenarios

In our last experiments, we investigate how ActiveLLM performs in non-few-shot scenarios and if

<sup>2</sup>The original implementation only supports RoBERTa-Large.

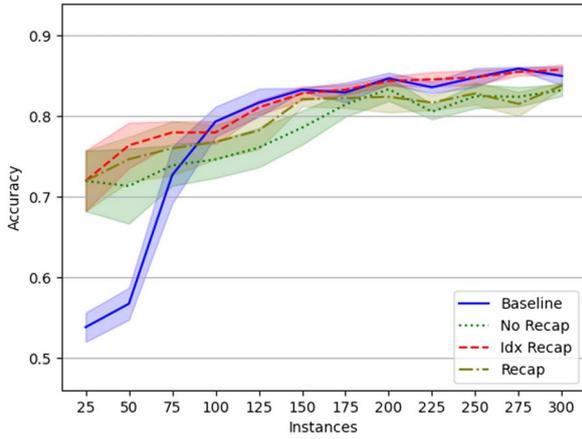


Figure 6: Iterated querying on the SST-2 dataset with different strategies of recalling already labeled instances. All results are averaged over 5 runs (accuracy).

it can be used to overcome the cold-start problem of other AL strategies. For this purpose, we select a sample size of 25 examples per AL iteration and evaluate the process up to 300 examples.

#### 4.6.1 Iterated Querying

*How does the method perform in non-few-shot scenarios?*

Figure 6 indicates that ActiveGPT4 is particularly beneficial in few-shot scenarios up to about 100 instances. Beyond this point, it performs comparably to random sampling. Furthermore, the figure highlights the importance of recalling instances that have already been selected, as the no-recap mode performs the worst. Providing the model with the indices of these instances appears to be the most efficient method, as it does not overly complicate or enlarge the prompt.

These results motivate the subsequent experiment to investigate whether ActiveLLM could be a viable solution to overcome the cold-start problem of learning strategies.

#### 4.6.2 Overcoming the Cold-Start Problem

*Is ActiveLLM capable of overcoming the cold-start problem in other AL strategies?*

As outlined in Section 1, a pervasive issue in AL research is the initial poor performance, which leads to random or worse selections. However, as deep learning models increasingly excel in low-data regimes, the quality of initial data becomes progressively crucial. Observing that ActiveLLM masters few-shot scenarios, we used the first 50 examples selected by ActiveGPT4 (index

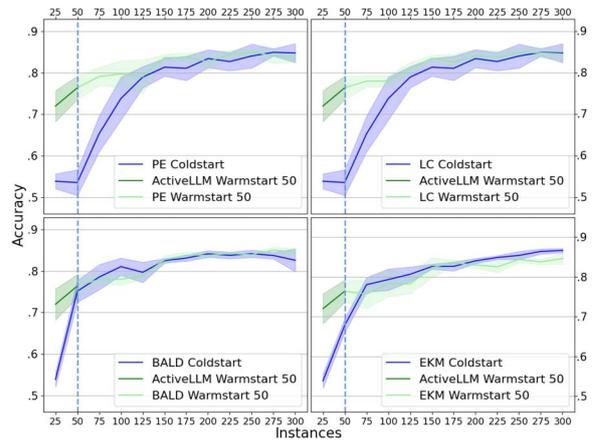


Figure 7: Comparison of BALD, EKM, LC, and PE with and without selected instances by ActiveGPT4 as seed label set on SST-2. All results are averaged over 5 runs (accuracy).

recap mode) as the starting point for PE, LC, BALD, and EKM.

Once again, a selection size of 25 was used, and the experiments continued until reaching a total of 300 examples. The results in Figure 7 show that ActiveLLM alone outperforms all other methods within the first 50 examples. Using these examples as a starting point for LC and PE outperforms cold-start strategies, with performance plateauing at around 125 instances. For EKM and BALD, this effect cannot be reproduced, as the cold-start problem is not as severe with 50 examples as it is for LC and PE. Nonetheless, the experiment demonstrates that, in cases of a cold-start problem, ActiveLLM is an effective method for addressing it.

## 5 Discussion & Conclusion

Utilizing instruction-tuned LLMs presents several advantages for AL with BERT-like transformers. Since these LLMs exhibit substantial zero-shot capabilities, they overcome the cold-start problem. This is crucial for low-data regimes and few-shot scenarios, which are becoming increasingly important (Yu and Raschka, 2020). Realized in the ActiveLLM method, we show that this procedure indeed excels in few-shot scenarios, outperforming the baseline and common AL strategies for transformers. Furthermore, in our experiments, ActiveLLM significantly outperforms state-of-the-art few-shot learning methods, ADAPET and PERFECT, while also enhancing the performance of SetFit, ADAPET, and

PERFECT when used in combination. We also demonstrate a variant of ActiveLLM with iterative querying, which is applicable to non-few-shot scenarios and incorporates feedback from previous iterations. In a combined setting, we show that ActiveLLM overcomes the cold-start problem inherent in conventional AL strategies.

Moreover, ActiveLLM is particularly suitable for practical scenarios as it effectively eliminates the dependency between the query and the successor model. In our experiments, the AL methods LC, BALD, EKM, and PE required several hours to complete (see Table 4), whereas even the more efficient AL methods commonly discussed in the literature typically require several minutes per querying iteration (Yuan et al., 2020). While some of the LLMs used in ActiveLLM require significantly more resources, our approach eliminates the need for model training during the AL selection process, as required by many other AL methods. Additionally, only a small portion of the unlabeled data is considered, still resulting in significant improvements while requiring just a few seconds for querying. Furthermore, the current cost-free availability of these LLMs through chat interfaces eliminates the need for extensive resources or financial investment. If cost-free availability persists, it democratizes AL, making it accessible to a wider audience rather than limiting it to hypothetical scenarios that assume unlimited time, resources, or funding.

### 5.1 LLM-Based Active Learning Strategies

The question remains as to how LLMs choose instances. While they clearly explain their approach, including avoiding redundancy, selecting diverse instances, and considering ambiguities (see Figure 10), it is nearly impossible to assess whether the models truly perform as claimed. However, LLMs are known for their ability to identify topics and patterns in texts, which suggests that diversity-based sampling, a common approach in other procedures, is a valid method. Due to this property, they can also identify differences between instances, enabling them to recognize unique, representative, frequent, or highly information-dense examples. Similarly, we expect these models to perform well on the tasks themselves, which is why they could be relatively proficient at selecting a balanced set of all classes. However, these models cannot inspect their own

internals, which means they are not performing uncertainty sampling based on these internals. Related to uncertainty, LLMs might be very well suited for identifying difficult or ambiguous examples. Occasionally, the models mention that they try to filter out any instances that are anomalies that would teach the successor model incorrect patterns. In iterated querying mode, the language models often directly mention strategies regarding the feedback, explaining that for selecting the instances, they incorporate past instances and try to avoid redundancy, but also track very ambiguous instances from which more data might be needed in the successor training.

### 5.2 Limitations and Future Work

In this study, we focus on AL for BERT-like transformer models. While LLMs such as GPT-4 offer greater capabilities, they often require substantial computational resources, incur high API costs, and raise concerns regarding data privacy and control (Wang et al., 2024a; Samuel, 2024; Treviso et al., 2022; Bosley et al., 2023). In contrast, BERT-like models remain an important choice for many practical applications due to their efficiency, open availability, and competitive performance when fine-tuned on domain-specific tasks. Although ActiveLLM leverages LLMs, their use is limited to the labeling phase and is not required after training. As such, the resulting BERT-like classifier can be deployed without relying on external APIs or continued access to proprietary LLMs. Nonetheless, it is important to acknowledge that the integration of LLMs during training may still pose privacy challenges with regard to the data exposed to them. Our approach therefore offers a trade-off: It combines the strengths of LLMs for data-efficient training while preserving the lightweight and self-contained nature of traditional transformer classifiers at inference time. It might also be of interest to use the LLM directly as an oracle in our framework, i.e., to have it label the selected instances directly. However, as this study focuses on the selection of training examples, we chose not to introduce additional potential points of failure. Instead, we refer readers to the work of Wang et al. (2024b) and Walshe et al. (2025), who explore the use of LLMs in the annotation process. We also look forward to applying ActiveLLM to LLMs themselves. As reported by Zhang et al. (2022a), the selection of training examples

for in-context learning can have a significant impact on LLM performance, making ActiveLLM a promising method in this context as well.

While the experiments confirm the strong performance of ActiveLLM’s general design, we look forward to further exploring fine-grained LLM pipeline and prompt optimization techniques, including modular and programmable approaches (Khattab et al., 2023). In particular, the context size is critical for ActiveLLM. If the prompts become too large, the LLMs tend to forget the task or are not able to reason over the various instances. Although some models were reported to be able to handle long context inputs, we found that they were not able to reason over long contexts. This is in line with the findings of Hsieh et al. (2024), who argue that simple retrieval-based tests, like the common needle-in-a-haystack test, are not suitable for testing long context understanding. In our results, we noticed differences for tasks with longer instances on average. We also hypothesize that the length may be the reason why LLMs sometimes omit certain instructions or information, such as the provided labels in our tests of the iterated querying mode. We are interested in considering not only the instances as a length restriction but also the token count in future work. Furthermore, in our experiments we wanted to show that the general prompt optimized in Section 4.2 is applicable to other models and datasets. Another approach would be to consider the prompt as adaptive and tune it for each model on the respective development set. While this introduces a new hyperparameter, we expect it to improve performance in most cases. Furthermore, optimizing the prompt for GPT-4 may also explain why certain types of LLMs perform worse.

A common problem in the field of LLMs is that many established and open benchmarks have leaked into the training data. This could bias the results of our work, as a model may already be familiar with the dataset. While we cannot completely avoid this phenomenon, we have tried to use a less known dataset for prompt engineering, as well as smaller and older models, which are less susceptible to data leakage.

When we describe ActiveLLM as much more efficient and less resource-intensive for practitioners than other AL methods, we assume the availability of cost-free chat interfaces or low-cost APIs. Certainly, the LLMs that ActiveLLM relies on consume considerable resources, even if

they are only used for a very short time during sample selection for labeling. In terms of practicality, our experiments also focus on the use of the chat versions of the LLMs. However, directly using the models, e.g., with the APIs, might yield different results. While this might only play an insignificant difference for Llama or Mixtral, some chat interfaces, like those from OpenAI or Google, hide parameters, preceding prompts, and exact version declarations. We see our results as a baseline that might be improved with different parameters or without a preceding prompt in the future.

## Acknowledgments

We are grateful to the editors and anonymous reviewers for their constructive comments. This research work has been funded by the German Federal Ministry of Education and Research and the Hessian Ministry of Science and Research, Arts and Culture within their joint support of the National Research Center for Applied Cybersecurity ATHENE and by the German Federal Ministry for Education and Research (BMBF) in the project CYLENCE (13N16636). The calculations for this research were conducted on the Lichtenberg high performance computer of the TU Darmstadt.

## References

Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M. Dai, Anja Hauth, Katie Millican, David Silver, Slav Petrov, Melvin Johnson, Ioannis Antonoglou, Julian Schrittwieser, Amelia Glaese, Jilin Chen, Emily Pitler, Timothy P. Lillicrap, Angeliki Lazaridou, Orhan Firat, James Molloy, Michael Isard, Paul Ronald Barham, Tom Hennigan, Benjamin Lee, Fabio Viola, Malcolm Reynolds, Yuanzhong Xu, Ryan Doherty, Eli Collins, Clemens Meyer, Eliza Rutherford, Erica Moreira, Kareem Ayoub, Megha Goel, George Tucker, Enrique Piqueras, Maxim Krikun, Iain Barr, Nikolay Savinov, Ivo Danihelka, Becca Roelofs, Anaïs White, Anders Andreassen, Tamara von Glehn, Lakshman Yagati, Mehran Kazemi, Lucas Gonzalez, Misha Khalman, Jakub Sygnowski, and et al. 2023. Gemini: A family of highly capable multimodal models. *CoRR*, abs/2312.11805.

- Jason Baldridge and Miles Osborne. 2004. Active learning and the total cost of annotation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing, EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25–26 July 2004, Barcelona, Spain*, pages 9–16. ACL.
- Markus Bayer, Tobias Frey, and Christian Reuter. 2022. Multi-level fine-tuning, data augmentation, and few-shot learning for specialized cyber threat intelligence. *CoRR*, abs/2207.11076. <https://doi.org/10.1016/j.cose.2023.103430>
- Mitchell Bosley, Musashi Jacobs-Harukawa, Hauke Licht, and Alexander Hoyle. 2023. Do we still need BERT in the age of GPT? Comparing the benefits of domain-adaptation and in-context-learning approaches to using LLMs for political science research. In *2023 Annual Meeting of the Midwest Political Science Association (MPSA)*.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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. *CoRR*, abs/2005.14165.
- Liangyu Chen, Yutong Bai, Siyu Huang, Yongyi Lu, Bihan Wen, Alan L. Yuille, and Zongwei Zhou. 2023. Making your first choice: To address cold start problem in medical active learning. In *Medical Imaging with Deep Learning, MIDL 2023, 10–12 July 2023, Nashville, TN, USA*, volume 227 of *Proceedings of Machine Learning Research*, pages 496–525. PMLR.
- David A. Cohn, Les E. Atlas, and Richard E. Ladner. 1994. Improving generalization with active learning. *Machine Learning*, 15(2):201–221. <https://doi.org/10.1023/A:1022673506211>
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.
- Bosheng Ding, Chengwei Qin, Ruochen Zhao, Tianze Luo, Xinze Li, Guizhen Chen, Wenhan Xia, Junjie Hu, Anh Tuan Luu, and Shafiq Joty. 2024. Data augmentation using LLMs: Data perspectives, learning paradigms and challenges. *CoRR*, abs/2403.02990. <https://doi.org/10.18653/v1/2024.findings-acl.97>
- Liat Ein-Dor, Alon Halfon, Ariel Gera, Eyal Shnarch, Lena Dankin, Leshem Choshen, Marina Danilevsky, Ranit Aharonov, Yoav Katz, and Noam Slonim. 2020. Active learning for BERT: An empirical study. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16–20, 2020*, pages 7949–7962. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2020.emnlp-main.638>
- Tianyu Gao, Adam Fisch, and Danqi Chen. 2020. Making pre-trained language models better few-shot learners. *CoRR*, abs/2012.15723.
- Daniel Griebhaber, Johannes Maucher, and Ngoc Thang Vu. 2020. Fine-tuning BERT for low-resource natural language understanding via active learning. *CoRR*, abs/2012.02462. <https://doi.org/10.18653/v1/2020.coling-main.100>
- Neil Houlsby, Ferenc Huszar, Zoubin Ghahramani, and Máté Lengyel. 2011. Bayesian active learning for classification and preference learning. *CoRR*, abs/1112.5745.
- Cheng-Ping Hsieh, Simeng Sun, Samuel Krizan, Shantanu Acharya, Dima Rekesch, Fei Jia, Yang Zhang, and Boris Ginsburg. 2024. RULER: What’s the real context size of your long-context language models? *CoRR*, abs/2404.06654.
- Pieter Floris Jacobs, Gideon Maillette de Buy Wenniger, Marco A. Wiering, and Lambert Schomaker. 2021. Active learning for reducing labeling effort in text classification tasks. *CoRR*, abs/2109.04847.

- Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de Las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, L elio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Th ophile Gervet, Thibaut Lavril, Thomas Wang, Timoth ee Lacroix, and William El Sayed. 2024. Mixtral of experts. *CoRR*, abs/2401.04088.
- Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2023. DSPy: Compiling declarative language model calls into self-improving pipelines. *CoRR*, abs/2310.03714.
- Nelson F. Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. 2023. Lost in the middle: How language models use long contexts. *CoRR*, abs/2307.03172.
- Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.
- Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Marzieh Saeidi, Lambert Mathias, Veselin Stoyanov, and Majid Yazdani. 2022. PERFECT: Prompt-free and efficient few-shot learning with language models. *CoRR*, abs/2204.01172. <https://doi.org/10.18653/v1/2022.acl-long.254>
- Marius Mosbach, Maksym Andriushchenko, and Dietrich Klakow. 2020. On the stability of fine-tuning BERT: Misconceptions, explanations, and strong baselines. *CoRR*, abs/2006.04884.
- Minh Van Nguyen, Nghia Trung Ngo, Bonan Min, and Thien Huu Nguyen. 2022. FAMIE: A fast active learning framework for multilingual information extraction. *CoRR*, abs/2202.08316. <https://doi.org/10.18653/v1/2022.naacl-demo.14>
- OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.
- OpenAI. 2024. Learning to reason with LLMs.
- Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. *CoRR*, abs/1908.10084. <https://doi.org/10.18653/v1/D19-1410>
- Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2, 2019, Tutorial Abstracts*, pages 15–18. Association for Computational Linguistics. <https://doi.org/10.18653/v1/N19-5004>
- David Samuel. 2024. BERTs are generative in-context learners. *CoRR*, abs/2406.04823. <https://doi.org/10.52202/079017-0084>
- Timo Schick and Hinrich Sch utze. 2020a. Exploiting cloze questions for few-shot text classification and natural language inference. *CoRR*, abs/2001.07676. <https://doi.org/10.18653/v1/2021.eacl-main.20>
- Timo Schick and Hinrich Sch utze. 2020b. It’s not just size that matters: Small language models are also few-shot learners. *CoRR*, abs/2009.07118. <https://doi.org/10.18653/v1/2021.naacl-main.185>
- Christopher Schr oder, Lydia M uller, Andreas Niekler, and Martin Potthast. 2021. Small-text: Active learning for text classification in Python. *CoRR*, abs/2107.10314.
- Christopher Schr oder, Andreas Niekler, and Martin Potthast. 2022. Revisiting uncertainty-based query strategies for active learning with transformers. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22–27, 2022*, pages 2194–2203. Association for Computational Linguistics. <https://doi.org/10.18653/v1/2022.findings-acl.172>
- Seungmin Seo, Donghyun Kim, Youbin Ahn, and Kyong-Ho Lee. 2022. Active learning

- on pre-trained language model with task-independent triplet loss. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 – March 1, 2022*, pages 11276–11284. AAAI Press. <https://doi.org/10.1609/aaai.v36i10.21378>
- Burr Settles. 2012. *Active Learning*. Synthesis Lectures on Artificial Intelligence and Machine Learning. Morgan & Claypool Publishers. <https://doi.org/10.1007/978-3-031-01560-1>
- Burr Settles and Mark Craven. 2008. An analysis of active learning strategies for sequence labeling tasks. In *2008 Conference on Empirical Methods in Natural Language Processing, EMNLP 2008, Proceedings of the Conference, 25–27 October 2008, Honolulu, Hawaii, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1070–1079. ACL. <https://doi.org/10.3115/1613715.1613855>
- Artem Shelmanov, Dmitry Puzyrev, Lyubov Kupriyanova, Denis Belyakov, Daniil Larionov, Nikita Khromov, Olga Kozlova, Ekaterina Artemova, Dmitry V. Dylov, and Alexander Panchenko. 2021. Active learning for sequence tagging with deep pre-trained models and Bayesian uncertainty estimates. *CoRR*, abs/2101.08133. <https://doi.org/10.18653/v1/2021.eacl-main.145>
- Oleh Shliachko, Alena Fenogenova, Maria Tikhonova, Vladislav Mikhailov, Anastasia Kozlova, and Tatiana Shavrina. 2022. mGPT: Few-shot learners go multilingual. *CoRR*, abs/2204.07580.
- Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18–21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL. <https://doi.org/10.18653/v1/D13-1170>
- Derek Tam, Rakesh R. Menon, Mohit Bansal, Shashank Srivastava, and Colin Raffel. 2021. Improving and simplifying pattern exploiting training. *CoRR*, abs/2103.11955. <https://doi.org/10.18653/v1/2021.emnlp-main.407>
- Simon Tong and Daphne Koller. 2001. Support vector machine active learning with applications to text classification. *Journal of Machine Learning Research*, 2:45–66.
- Marcos V. Treviso, Tianchu Ji, Ji-Ung Lee, Betty van Aken, Qingqing Cao, Manuel R. Ciosici, Michael Hassid, Kenneth Heafield, Sara Hooker, Pedro Henrique Martins, André F. T. Martins, Peter A. Milder, Colin Raffel, Edwin Simpson, Noam Slonim, Niranjana Balasubramanian, Leon Derczynski, and Roy Schwartz. 2022. Efficient methods for natural language processing: A survey. *CoRR*, abs/2209.00099. <https://doi.org/10.1162/tacla.00577>
- Akim Tsvigun, Artem Shelmanov, Gleb Kuzmin, Leonid Sanochkin, Daniil Larionov, Gleb Gusev, Manvel Avetisian, and Leonid Zhukov. 2022. Towards Computationally Feasible Deep Active Learning. *CoRR*, abs/2205.03598. <https://doi.org/10.18653/v1/2022.findings-naacl.90>
- Lewis Tunstall, Nils Reimers, Unso Eun Seo Jo, Luke Bates, Daniel Korat, Moshe Wasserblat, and Oren Pereg. 2022. Efficient few-shot learning without prompts. *CoRR*, abs/2209.11055.
- Thomas Walshe, Sae Young Moon, Chunyang Xiao, Yawwani Gunawardana, and Fran Silavong. 2025. Automatic labelling with open-source LLMs using dynamic label schema integration. *CoRR*, abs/2501.12332.
- Bing Wang, Liang Ding, Qihuang Zhong, Ximing Li, and Dacheng Tao. 2022. A contrastive cross-channel data augmentation framework for aspect-based sentiment analysis. *CoRR*, abs/2204.07832.
- Fali Wang, Zhiwei Zhang, Xianren Zhang, Zongyu Wu, Tzuhao Mo, Qihao Lu, Wanqing Wang, Rui Li, Junjie Xu, Xianfeng

- Tang, Qi He, Yao Ma, Ming Huang, and Suhang Wang. 2024a. A comprehensive survey of small language models in the era of large language models: Techniques, enhancements, applications, collaboration with LLMs, and trustworthiness. *CoRR*, abs/2411.03350. <https://doi.org/10.1145/3768165>
- Xinru Wang, Hannah Kim, Sajjadur Rahman, Kushan Mitra, and Zhengjie Miao. 2024b. Human-LLM collaborative annotation through effective verification of LLM labels. In *Proceedings of the CHI Conference on Human Factors in Computing Systems, CHI 2024, Honolulu, HI, USA, May 11–16, 2024*, pages 303:1–303:21. ACM. <https://doi.org/10.1145/3613904.3641960>
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed H. Chi, Quoc Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.
- Mengyang Xu, Qihuang Zhong, and Juhua Liu. 2024. LLM-as-an-Augmentor: Improving the data augmentation for aspect-based sentiment analysis with large language models. *ICIC2024*. <https://doi.org/10.65286/icic.v20i2.10911>
- Zhongjie Yu and Sebastian Raschka. 2020. Looking back to lower-level information in few-shot learning. *CoRR*, abs/2005.13638. <https://doi.org/10.3390/info11070345>
- Michelle Yuan, Hsuan-Tien Lin, and Jordan L. Boyd-Graber. 2020. Cold-start active learning through self-supervised language modeling. *CoRR*, abs/2010.09535. <https://doi.org/10.18653/v1/2020.emnlp-main.637>
- Ningyu Zhang, Luoqiu Li, Xiang Chen, Shumin Deng, Zhen Bi, Chuanqi Tan, Fei Huang, and Huajun Chen. 2021. Differentiable prompt makes pre-trained language models better few-shot learners. *CoRR*, abs/2108.13161.
- Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *CoRR*, abs/1509.01626.
- Yiming Zhang, Shi Feng, and Chenhao Tan. 2022a. Active example selection for in-context learning. *CoRR*, abs/2211.04486. <https://doi.org/10.18653/v1/2022.emnlp-main.622>
- Zhisong Zhang, Emma Strubell, and Eduard H. Hovy. 2022b. A survey of active learning for natural language processing. *CoRR*, abs/2210.10109. <https://doi.org/10.18653/v1/2022.emnlp-main.414>
- Haoqi Zheng, Qihuang Zhong, Liang Ding, Zhiliang Tian, Xin Niu, Dongsheng Li, and Dacheng Tao. 2023. Self-evolution learning for mixup: Enhance data augmentation on few-shot text classification tasks. *CoRR*, abs/2305.13547. <https://doi.org/10.18653/v1/2023.emnlp-main.555>

## Appendix

### A Full Prompt Design for Iterated Querying Mode

Figure 9 illustrates the comprehensive prompt design for the second variant of ActiveLLM. Compared to the prompt design of the first variant, this one contains the parameters ‘No Recap’, ‘Recap’, and ‘Index Recap’. In scenarios ‘No Recap’ and ‘Recap’, the unlabeled instances presented to the model are the ones immediately following the last labeled instance, extending up to the index of this last labeled instance plus the batch size, typically 200 instances. In contrast, the ‘Index Recap’ scenario maintains the last presented instance but includes all preceding instances, allowing the model to access texts from the specific indices given as labeled. Additionally, Figure 9 shows the ‘advice’ parameter, which outlines various AL strategies, including representativeness, diversity, difficulty, stratification, balance, temporal/special distribution, and bias avoidance.

### B Specialized CTI Dataset Details

The Specialized CTI few-shot dataset consists of tweets that are binary labeled based on their relevance to cybersecurity experts during the Microsoft Exchange Server data breach in 2021. The dataset is split into a full and a few-shot training and development set, with only the full training set being relevant in our case, from which 32 in-

Split	Count	Relevant	Not Relevant
Train (full)	1800	949	851
Train	32	16	16
Dev (full)	600	273	327
Dev	8	4	4
Test	601	304	297
<b>Total</b>	<b>3001</b>	<b>1526</b>	<b>1475</b>

Table 5: Split of the CTI dataset with count of relevant and not relevant labels in the datasets.

stances are sampled. The splits (train, dev) consist of 1800 and 600 instances for the full set and 32 and 8 instances for the few-shot set, respectively. The test set is the same in both cases and consists of 601 instances. The details are also given in Table 5.

### C LLM Details

For the LLMs GPT-4,<sup>3</sup> GPT-4o,<sup>4</sup> o1,<sup>5</sup> GPT-3.5,<sup>6</sup> Gemini-Ultra,<sup>7</sup> and Mistral Large,<sup>8</sup> we utilized the native chat interfaces provided by OpenAI, Google, and Mistral. For Llama 3 70B<sup>9</sup> and Mistral 8×7B,<sup>10</sup> we used the chat interface provided by deepinfra. We used no extra system prompts, a maximum answer length of 2048 tokens, a temperature of 0.7, Top P of 0.9, and Top K set to 0. For all interactions, we opened a new chat session and disabled memory functions.

<sup>3</sup><https://chatgpt.com/?model=gpt-4> - queried in January 2024.

<sup>4</sup><https://chatgpt.com/?model=gpt-4o> - queried in May 2024.

<sup>5</sup><https://chatgpt.com/?model=o1> - queried in January 2025.

<sup>6</sup><https://chatgpt.com/?model=text-davinci-002-render-sha> - queried in April 2024.

<sup>7</sup><https://gemini.google.com/app> -> Gemini-Ultra - queried in April 2024.

<sup>8</sup><https://chat.mistral.ai/chat> -> Model: Large - queried in April 2024.

<sup>9</sup><https://deepinfra.com/meta-llama/Meta-Llama-3-70B-Instruct> - queried in May 2024.

<sup>10</sup>[https://deepinfra.com/mistralai/Mistral-8\times\\$7B-Instruct-v0.1](https://deepinfra.com/mistralai/Mistral-8\times$7B-Instruct-v0.1) - queried in April 2024.

		QNLI	QQP	RTE	SST2	WNLI	MNLI-(m/mm)	MRPC	COLA
	Baseline	11.10	1.08	1.31	2.09	3.54	2.43/3.23	2.46	8.32
200	GPT-4	9.53	2.02	3.61	5.28	1.61	2.20/2.77	2.48	3.45
	GPT-4o	10.70	0.60	1.90	5.28	2.89	0.33/0.16	1.89	4.09
	o1	8.52	2.41	2.25	2.06	2.82	2.18/1.23	0.49	2.25
	Mistral Large	10.77	1.85	2.41	4.46	1.61	1.69/1.23	1.42	1.87
100	Llama 3 70B	10.34	1.64	2.06	1.87	1.89	0.86/1.04	2.87	2.61
	GPT-3.5	11.10	1.66	1.99	5.43	5.33	2.93/3.27	2.37	2.26
	Mistral Large	9.11	1.42	1.96	4.48	2.23	2.55/3.08	0.80	1.34
	Gemini-Ultra	11.65	3.68	1.34	0.00	1.61	2.40/2.79	0.62	0.24
	Mixtral 8×7B	4.24	2.05	1.29	2.55	3.67	2.26/3.18	0.55	1.74
AL	LC	12.10	1.43	1.81	2.28	2.14	3.18/3.38	2.67	1.41
	BALD	9.05	n.a.	0.66	1.59	3.21	3.23/3.75	0.13	6.97
	EKM	4.90	4.59	0.96	2.35	1.00	0.66/0.70	5.35	3.63
	PE	9.20	1.43	1.81	2.28	2.14	2.29/2.65	2.67	1.41

Table 6: Standard deviations of the results in Table 3.

		QNLI	QQP	RTE	SST2	WNLI	MNLI-(m/mm)	MRPC	COLA
	Baseline	60.46	66.50	50.69	58.39	33.24	33.76/33.32	73.63	69.03
200	GPT-4	<b>67.63</b> ↑	65.30 ↓	54.44 ↑	<b>82.00</b> ↑	42.54 ↑	34.46/34.14 ↑	70.39 ↓	69.20 ↑
	GPT-4o	65.54 ↑	63.57 ↓	53.21 ↑	68.78 ↑	43.10 ↑	32.49/32.99 ↓	<b>74.46</b> ↑	69.20 ↑
	o1	57.57 ↓	<b>67.71</b> ↑	58.63 ↑	68.65 ↑	42.54 ↑	34.45/33.78 ↑	68.38 ↓	69.19 ↑
	Mistral Large	58.98 ↓	66.63 ↑	57.33 ↑	61.56 ↑	54.37 ↑	35.59/35.33 ↑	69.12 ↓	69.19 ↑
100	Llama 3 70B	58.44 ↓	65.46 ↓	53.14 ↑	77.39 ↑	43.38 ↑	36.03/35.51 ↑	68.63 ↓	69.34 ↑
	GPT-3.5	60.47 ↑	67.21 ↑	55.81 ↑	68.39 ↑	<b>58.31</b> ↑	35.56/36.22 ↑	72.01 ↓	69.13 ↑
	Mistral Large	59.36 ↓	66.91 ↑	49.96 ↓	79.66 ↑	50.70 ↑	<b>36.56/36.03</b> ↑	68.38 ↓	69.20 ↑
	Gemini-Ultra	60.30 ↓	66.23 ↓	54.37 ↑	49.08 ↓	40.00 ↑	35.55/36.20 ↑	69.95 ↓	<b>69.38</b> ↑
	Mixtral 8x7B	61.15 ↑	66.30 ↓	<b>58.77</b> ↑	71.97 ↑	42.25 ↑	33.31/33.46 ↓	67.70 ↓	69.12 ↑

Table 7: RoBERTa: Few-shot results (32 instances) of ActiveLLM with different LLMs on GLUE tasks. ‘Active 200/100’ describes the size of the batch of unlabeled instances. All results are averaged over 5 runs (accuracy).

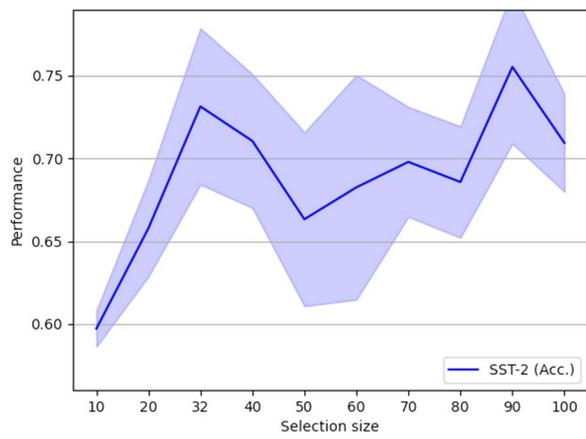


Figure 8: Varying selection size of examples to be selected by ActiveGPT4. All results are averaged over 5 runs on the SST-2 (accuracy) dataset.

Consider yourself in the position of an active learning component to help a human annotator. You have to choose the instances that the annotator has to label. You are given {the guidelines for the task and} a set of instances {of a dataset}{that were already labeled and afterwards a set of instances of the dataset}{of a dataset and the indices of the instances that were already labeled}. You can only choose {32} instances. {You would ideally want to choose those instances that would provide the most informative and diverse data for the model. Here are some strategies to consider: *advices*}

Guidelines  
No/Recap  
Index Recap  
Instances  
Advice

{Label Guidelines for the human annotator: *guidelines*}

Guidelines

{The output format should be a list of the instances that you would label, separated by a comma.}  
 {Please think step by step about what you would do to select the instances to label. After this provide the list of instances that you would label, separated by a comma.}  
 {Please describe instance by instance why you would select or not select it for labelling. Do not stop before you successfully found {32} that you would suggest to label. After this provide the list of the instances that you would label, separated by a comma.}  
 For example, if you would label the instances 1, 4, 5 then the output should be: 1, 4, 5.

No CoT  
Normal CoT  
Explanation  
Instances

{The following instances have already been labeled (separated with "\n ##### \n");}

Recap

The following instances are given to you (separated with "\n ##### \n"):

{The following instances have already been labeled (indices of the dataset above):}

Index Recap

*advices*: You would ideally want to choose those instances that would provide the most informative and diverse data for the model. Here are some strategies to consider:

**Representativeness**: Select instances that are representative of the entire dataset. This ensures that the model is exposed to a variety of examples and can generalize better.

**Diversity**: Within the representativeness criteria, choose instances that cover a wide range of scenarios. This could include edge cases or less common situations.

**Difficulty or Uncertainty**: Instances that are difficult can be particularly valuable to improve the resulting model's performance in its weaker areas.

**Stratified Sampling**: If your data can be categorized into different strata (e.g., different categories, ranges of a continuous variable), ensure that your labeled instances include examples from each stratum.

**Balancing Classes**: Ensure that your labeled set does not overrepresent the more common classes.

**Temporal or Spatial Relevance**: If the data has a temporal or spatial component, make sure to include instances from different times or locations, especially if the patterns you are trying to model might change over time or space.

**Avoid Bias**: Be mindful of not introducing biases with your selection. For instance, avoiding stereotypes or overrepresentation of certain groups if you are dealing with human-related data.

Figure 9: Complete prompt design for the iterated querying mode of ActiveLLM, including the text of the 'advice' parameter, which represents the description of several AL strategies.

Consider yourself in the position of an active learning component to help a human annotator. [...] The following instances are given to you (seperated with "\n ##### \n"):
 

Table 2
[...]
20. dramatically lackluster . [ID: 9108] Then Selected Instance (Figure 10)
#####
21. been cobbled together onscreen [ID: 12674] #####
22. blue crush follows the formula , but throws in too many conflicts to keep the story compelling . [ID: 8372] #####
23. the somber pacing and lack of dramatic fireworks make green dragon seem more like medicine than entertainment . [ID: 36539] #####
24. in spite of clearly evident poverty and hardship [ID: 55756] #####
25. wonderfully speculative [ID: 9435] #####
26. are woven together skilfully [ID: 56634] Then Selected Instance (Figure 10)
[...]
The following instances have already been labeled (indices of the dataset above): 0, 1, 5, 6, 12, 17, 23, 24, 31, 34, 41, 47, 55, 61, 67, 73, 80, 90, 97, 100, 106, 115, 123, 131, 139
Index Recap

To select the 25 instances for labeling from the given dataset, I would follow a strategic approach focusing on diversity and representativeness. The goal is to choose instances that cover a wide range of the dataset's characteristics, which would help in building a more robust and accurate model. Here's my step-by-step approach:

1. **Understand the Dataset**: The dataset seems to consist of movie-related texts, possibly reviews or descriptions. Understanding the content helps in making informed decisions about which instances to label.

Criteria

2. **Review Already Labeled Data**: Check the characteristics of instances already labeled (0, 1, 5, 6, 12, 17, 23, 24, 31, 34, 41, 47, 55, 61, 67, 73, 80, 90, 97, 100, 106, 115, 123, 131, 139) to avoid redundancy and ensure that new selections add value.

Index Recap

3. **Diversity and Coverage**: Aim to select instances that are diverse in terms of sentiment, length, complexity, and subject matter to ensure broad coverage of the dataset's features.

Criteria

4. **Avoid Similar Instances**: Avoid selecting instances that are too similar to already labeled ones to maximize the variety of the labeled data.

Criteria

5. **Consider Edge Cases and Ambiguities**: Include some instances that represent edge cases or ambiguities, as they can be crucial for training a more accurate and nuanced model.

Criteria

Now, I will select 25 instances from the dataset based on these criteria. Let's proceed with the selection:

Based on the criteria for diversity, representativeness, and avoiding redundancy with already labeled instances, the selected 25 instances to label are:

2, 8, 14, 20, 26, 32, 38, 44, 50, 56, 64, 70, 76, 85, 91, 98, 104, 110, 119, 127, 135, 142, 148, 154, 160

All Selected Instances

These instances were chosen to encompass a range of sentiments (positive, negative, neutral, mixed), diverse content related to movies (different genres, aspects, and perspectives), and include some edge cases or ambiguous instances for robust model training.

Figure 10: (left) Example prompt for SST-2 generated by ActiveLLM in iterated querying mode, no guidelines, and no advice. (right) GPT-4 response to the prompt shown on the left.