Active Few-Shot Learning for Text Classification

Saeed Ahmadnia[1]   **Arash Yousefi Jordehi**[2]   **Mahsa Hosseini Khasheh Heyran**[2]
Seyed Abolghasem Mirroshandel[2]   **Owen Rambow**[3]   **Cornelia Caragea**[1]

[1]University of Illinois Chicago   [2]University of Guilan   [3]Stony Brook University

sahmad67@uic.edu   arashy76@phd.guilan.ac.ir   mahsahsii@gmail.com
mirroshandel@guilan.ac.ir   owen.rambow@stonybrook.edu   cornelia@uic.edu

Abstract

The rise of Large Language Models (LLMs) has boosted the use of Few-Shot Learning (FSL) methods in natural language processing, achieving acceptable performance even when working with limited training data. The goal of FSL is to effectively utilize a small number of annotated samples in the learning process. However, the performance of FSL suffers when unsuitable support samples are chosen. This problem arises due to the heavy reliance on a limited number of support samples, which hampers consistent performance improvement even when more support samples are added. To address this challenge, we propose an active learning-based instance selection mechanism that identifies effective support instances from the unlabeled pool and can work with different LLMs. Our experiments on five tasks show that our method frequently improves the performance of FSL. We make our implementation available at https://github.com/theSaeed/active-fewshot-learning.

1 Introduction

Deep learning systems have shown great performance when given enough labeled data, yet they struggle to learn from a small amount of labeled data (Sun et al., 2019). However, constructing a large corpus of annotated data is costly and time-consuming, which hinders the building of supervised classifiers for new domains (Zhu et al., 2009). Few-Shot Learning (FSL) addresses this by seeking to grasp new concepts from limited labeled examples for broader applications (Sun et al., 2019). With recent advances in Large Language Models (LLM), the capabilities of FSL can be utilized much better than before and these methods can reach acceptable performance using minimal training data (Gao et al., 2021; Chen et al., 2021; Karimi Mahabadi et al., 2022; Lin et al., 2022).

There are two widely adopted approaches to addressing the FSL problem: In-Context Learning (ICL) and Fine-Tuning (FT). Recent autoregressive decoder-only LLMs have demonstrated strong performance in the realm of FSL (Brown et al., 2020), providing the opportunity to achieve satisfactory results with little to no labeled data without requiring any fine-tuning. On the other hand, FT is a more established strategy for text classification, which leverages a model fine-tuned on labeled support data for prediction, and can show strong results as opposed to ICL (Edwards and Camacho-Collados, 2024), but a large amount of training data is required for model fine-tuning.

Interestingly, most FSL methods typically select the samples randomly. However, the quality of the samples can have a significant impact on the model's performance. In some scenarios, adding un- or less-informative samples can even decrease the accuracy or may cause a high variance in the model's performance (Zhang et al., 2020; Schick and Schütze, 2021b). Few studies in the field of Natural Language Processing (NLP) have explored sample selection strategies. Nevertheless, prior works are limited by their lack of advanced techniques, such as active learning (Chang et al., 2021), and by compromising performance to achieve efficiency (Müller et al., 2022).

Given these challenges, and following the revealed strong performance of FT approaches against ICL (Edwards and Camacho-Collados, 2024), we propose a new FT- and Active Learning (AL)-based Few-Shot (FS) sample selection method to enhance classification performance by choosing the most informative unlabeled samples, thus, minimizing annotation costs. Inspired by successful AL algorithms (Settles, 2009), our algorithm selects instances using entropy and clustering methods to ensure uncertainty, diversity, and representativeness in sample selection.

More specifically, we propose an iterative approach to progressively choose samples for human annotation and label them to be added to the sup-

port set. This process begins with generating an organized depiction of the unlabeled data based on a certain embedding method to capture the requested key features of the data. Following this, a sampling method is applied to strategically select new data points based on the generated embeddings. These data points are then selected and given to a human annotator to be labeled and added to the support set. Afterward, we use this augmented support set for fine-tuning LLMs.

To evaluate our method, we conduct experiments on five classification tasks: the news topic, 5-star rating, and private states' type, polarity, and intensity classification. We use two pre-trained language models, BART and FLAN-T5 (Lewis et al., 2019; Chung et al., 2022), as the backbone and fine-tuning models in our approach. The proposed method is compared against weak and strong baselines to support its effectiveness and efficiency. These baselines include random sampling, in-context learning with Gemma 2 (Team et al., 2024), Llama 3.1 (Dubey et al., 2024), and Mistral v0.3 (Jiang et al., 2023), as well as other related work utilizing more advanced sample selection strategies in FSL. Our approach, especially when combining representative and uncertainty-based sampling techniques, exceeds the baselines considerably on average.

Our contributions can be summarized as follows: 1) We introduce an Active Learning-based sample selection scenario by combining uncertainty and representativeness measures for FS classification problems achieving state-of-the-art performance when paired with recent FSL algorithms. 2) We executed comprehensive experiments on a wide variety of tasks using various LLMs. 3) We present a thorough analysis to assess the performance of the models at different iterations, comparing them to the baselines presented in previous work. 4) We make our implementation publicly available.

2   Related Work

In previous studies, the few-shot scenario has been simulated by randomly sampling a subset from the complete training data (Chen et al., 2020; Schick and Schütze, 2021a; Gao et al., 2021; Chen et al., 2021; Lin et al., 2022; Edwards and Camacho-Collados, 2024). Among different FSL methods in NLP, there are few methods that have paid attention to the sample selection strategies. However, some recent studies in the field of image processing have demonstrated the effectiveness of incorporating AL strategies in the context of FSL (Boney et al., 2019; Pezeshkpour et al., 2020; Li et al., 2021; Shin et al., 2022).

The study conducted by Chang et al. (2021) is one of the few works that specifically addresses sample selection in NLP. Their research focuses on FS training instance selection and using it in three text generation tasks with BART. Their approach is motivated by the idea that few-shot training instances should exhibit diversity and representativeness. To achieve this, the authors utilized K-Means clustering for choosing data points closer to the center of clusters as important (i.e., informative) samples. Their results demonstrate the success of this method even with this simple non-iterative clustering-based approach. In contrast, our research specifically targets classification tasks. Furthermore, our active learning approach incorporates a wider range of tasks and selection strategies (i.e., uncertainty, diversity, and representativeness) compared to their study. We then extend the usage of this idea to iteratively expand the support set.

The only recent study that incorporates FS and AL approaches in NLP is conducted by Müller et al. (2022), using a zero-shot approach for XLM-RoBERTa-based Siamese networks. They use label tuning to fine-tune the label embeddings for faster training. On the contrary, we use multiple language models (i.e., BART and FLAN-T5) and fine-tune the entire model on the support set in every iteration as we prioritize the FS instance selection quality and model performance, which is shown to produce significantly better performance. Also, the related work does not specify how they select samples in the first iteration using an uncertainty approach while the model has never seen any related data before. In contrast, we propose using representative sampling in the first step to boost the initial performance of the model, even in uncertainty sampling methods. Moreover, we introduce four new sampling methods compared to the mentioned work. Importantly, our methods and implementation are open source and publicly available to be freely used by fellow researchers. To the best of our knowledge, the previous work's methods are not freely available to the public.

3   Active Few-Shot Learning

In our definition of the problem, we have a large set of unlabeled training samples to start with. Our
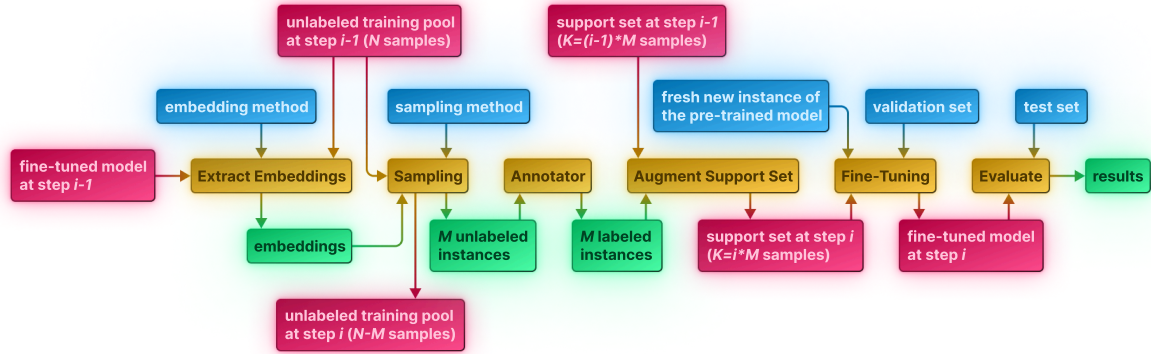
Figure 1: Pipeline of the ith iteration in our approach. Yellow boxes represent different phases of the method. Blue boxes are constant inputs during all iterations. Red boxes are carried over and modified during all consecutive iterations. Green boxes are products of the current iteration that will not be used later in the approach.

goal is to select a small number of samples from this unlabeled pool to be labeled and used as a support set in a way that maximizes a model's performance on the test set. In different FS instance selection methods, we may have one or more iterations of sampling and model fine-tuning. We categorize experiments with a single round of this process as 'non-iterative', and those with multiple rounds as 'iterative'.

Figure 1 illustrates the full pipeline of a single iteration, which can be the one and only iteration in the non-iterative approaches. At each iteration, we first examine the data to determine which additional samples to include in the support set. This process begins by extracting embeddings from a certain source, specified by an **embedding method**, to have a structured representation of the unlabeled data. This source is derived from running inference on the fine-tuned model of the last iteration by feeding it the unlabeled training pool. Subsequently, a **sampling method** is applied to select new samples based on the obtained embeddings, guiding the model's learning toward optimal performance. The embedding and sampling methods are explained in the following subsections in more detail.

The selected samples are then removed from the unlabeled training pool and given to an oracle for labeling. Following that, the newly labeled samples are added to the support set. This augmented support set is next used to fine-tune a new instance of the pre-trained model on the validation set. Finally, we evaluate the latest fine-tuned model on the test set to analyze the performance of our approach at the end of each step. However, the test set is solely used for evaluation purposes and is

not needed for our approach to function in the defined setup. Moreover, in the initial round, the support set is an empty set, all the training data is included in the unlabeled pool, and a pre-trained model without any previous fine-tuning is used to extract embeddings.

3.1 Embedding Methods

We obtain the embeddings by using two distinct methods:

Encoder (En): In this method, we extract the last hidden states from the encoder of the model and apply mean pooling over it. This embedding serves as a dense representation of the input data, while providing a rich feature space that encodes the input sequence.

Scores (Sc): Here, we leverage the output logits of the model and apply a softmax function to calculate the probability distribution over the possible labels. This method focuses on the model's confidence in assigning labels to the input data, which can be interpreted as a measure of how well the model understands the instance.

In both cases, we use a pre-trained model without any fine-tuning during the first iteration and use the fine-tuned model of the previous iteration during the subsequent iterations. Moreover, the embedding methods we use in this study make our approach adaptable to any LLM that provides label probabilities and its encoder's last hidden states.

Since we are working with text generation models, additional processing is required to calculate the scores. Specifically, we need to compute the

probability $P_m^{<t>}[n]$ (Equation 1), which represents the likelihood that the token at position t in sample m's logits corresponds to the nth class out of all N classes.

Each pre-trained model has its own vocabulary that maps distinct numerical indices to the tokens it recognizes. $Logits_m^{<t>}[i]$ indicates the model's logit for the ith word in the vocabulary at position t for sample m. During this procedure, we need our classes to be represented by a single token. For cases where a class name is represented by multiple tokens in the pre-trained model's tokenizer, we handle this by manually replacing such multi-token labels with semantically close single-token labels. We then disregard all the other tokens in the vocabulary that do not correspond to any task-specific class. To manage this, we define and use a dictionary $ClassId(i)$, which maps the ith class to its corresponding index in the vocabulary.

Once the class probabilities are computed, we calculate the score $Score_m[n]$ (Equation 2) by taking the maximum probability of the nth class over all T output tokens for sample m. This is especially important for multi-label tasks such as the MPQA Type task, where the model may generate multiple tokens in the output to indicate multiple labels. Using this method, we will eventually end up with a vector the same size as the label set ($|L|$ in Section 4.1).

$$P_m^{<t>}[n] = \frac{e^{Logits_m^{<t>}[ClassId(n)]}}{\sum_{i=1}^{N} e^{Logits_m^{<t>}[ClassId(i)]}} \quad (1)$$

$$Score_m[n] = \max_{1 \leq i \leq T}(P_m^{<i>}[n]) \quad (2)$$

3.2 Sampling Methods

Within each iteration, M instances need to be sampled from the training set and added to the support set of size K, which is an empty set at the beginning of the initial round. More precisely, these instances are sampled from the (simulated) unlabeled training set by considering the inputs and their corresponding embeddings. Only after choosing the samples, we can look at the labels of the M instances (simulating the human annotation) and use them in the fine-tuning process. M is a small number and is considered to designate **the whole selection size**, unlike typical FSL classification tasks that select M samples for each class (Ren et al., 2018; Chen et al., 2019; Wang et al., 2023), since we do not have access to those classes in our

definition of the problem. The sampling methods that we use in this paper are as follows:

Random: With this method, we simply sample M instances randomly without replacement from the unlabeled pool. This sampling method does not require any embedding data.

Representative (Rep): This method gets help from the embeddings we retrieved in our desired embedding method to cluster the unlabeled data into M groups using the *K-Means* algorithm. Then, inside each cluster, we sample the data point that its corresponding embedding is the closest (euclidean distance) to the cluster centroid.

Uncertainty (Un): It can only benefit from the *Sc* embeddings to select the M samples about which the model has the most doubts. We will be using *entropy* (Shannon, 1948; Settles, 2009) as our uncertainty measure throughout this paper.

Uncertainty Representative (UnRep): Using this technique, we first choose the $\alpha \times M$ most uncertain samples based on the *Sc* embeddings. Thereafter, we do a representative sampling based on the *En* embeddings only on these selected data points in order to sample the final M unlabeled samples.

Cluster Uncertainty (ClUn): This strategy, at first, splits the data into M clusters considering the given embeddings using the *K-Means* algorithm. It will then pick the data point that the model has the least confidence about inside each cluster by looking at their *Sc* embeddings.

All of these methods can be used during the second iteration onwards, but only the ones that do not involve uncertainty (Random and Rep) can be used within the first iteration and/or non-iterative approaches since there's no previous step for the model to learn enough about the task and decide whether it has doubts about the data.

4 Experimental Setup

4.1 Datasets

We use the **MPQA** Opinion Corpus, which is annotated at the word or phrase level to extract the following features of the expressed attitudes: **type**, **polarity**, and **intensity** (Wiebe et al., 2005; Wilson, 2008). Refer to Appendix A for more details. Additionally, we use **AG News** Corpus (Gulli, 2005)

for news topic classification and the English portion of The Multilingual **Amazon Reviews** Corpus (Keung et al., 2020) for 5-star rating classification tasks.

Dataset	Multi-Label	Train	Val	Test	$|L|$	$U_\%$
MPQA Type	✓	4,248	1,060	1,327	4	85.1
MPQA Polarity	✗	4,505	1,123	1,404	2	8.9
MPQA Intensity	✗	4,505	1,123	1,404	5	34.6
AG News	✗	118,800	1,200	7,600	4	0.0
Amazon Reviews	✗	200,000	1,200	5,000	5	0.0

Table 1: Dataset Statistics. It specifies whether they are multi-label, the size of training, validation, and test set splits, and the number of classes and their uniformness.

Table 1 exhibits the statistics and diversity of the datasets. The MPQA dataset does not offer separate training, validation, and test data, so we use the splits of MPQA that are provided in previous work (Ahmadnia et al., 2024). Although the AG News dataset presents training and test sets, it does not include any validation set. Therefore, we randomly sample the validation data from the training set without replacement. Amazon Reviews, however, offers all three mentioned splits, but the given validation set is relatively large. We generate a new set by randomly down-sampling the original validation set to refrain from overfitting too easily on the validation data as suggested by previous study (Gunel et al., 2020). We maintain the original sets' label distributions for these new validation sets. Following Müller et al. (2022), we define $|L|$ as the cardinality of the label set L, and $U = \sum_{l \in L} |f(l) - \frac{1}{|L|}|$ as a uniformity metric ($U = 0$ is uniform), where $f(l)$ is the relative frequency of label l.

4.2 Fine-Tuning Experiments

We evaluate our method across a wide range of tasks, comparing it to various baselines and related work leveraging several large language models. Specifically, for FT, we use the base versions of BART and FLAN-T5 with 139M and 248M parameters, respectively (Lewis et al., 2019; Chung et al., 2022).

To get better intuition about the tasks, we first calculate the majority baselines, which are the baselines we expect to beat. Additionally, we fine-tune the models using the entire training set as our support set (K = full training set size). These results represent a sort of top-line, which we do not expect to beat in our FS experiments.

Next, we fine-tune the pre-trained models with varying support set sizes, $K \in \{10, 20, 50, 100\}$, using random sampling, representative sampling, and our proposed iterative approaches. In the iterative approaches, within each iteration, we sample $M = 10$ new data points to be progressively added to our support set and present the results when we have fine-tuned the model using support sets of size $K \in \{10, 20, 50, 100\}$. For these experiments, we assign α, in Section 3.2, the value of 10. Appendix D includes additional experiments evaluating the impact of different values of α.

We further repeat FT experiments, this time with $K \in \{5, 10, 25, 50\}$ while having $M = 5$ in iterative approaches on the MPQA tasks to assess the impact of different selection sizes with the same number of iterations. Additionally, we choose the best-performing models from the AG News and Amazon Reviews tasks and fine-tune them with $M = 16$ over 16 iterations, resulting in a total support set size of $K = 256$. We compare our results to the best-performing models provided by Müller et al. (2022), namely LT margin and LT k-means.

4.3 In-Context Learning Experiments

For ICL, we utilize instruction-tuned models and prompt templates similar to related work (Reid et al., 2024; Dubey et al., 2024), and describe each task alongside a list of possible labels. These prompts can be applied directly in 0-shot settings as the system prompt. For FS settings, we append labeled instances to the system prompt to provide additional context. In the end, we query the model for predictions on each test instance in the user prompt and match the predicted output strings with the corresponding labels. The prompt template for each task is included in Appendix C. If a model does not support system prompts, we concatenate and merge the system and user prompts and treat them as a single (user) prompt. We employ two methods to sample training data for ICL:

Random: Similar to random sampling method in FT, we select M instances randomly without replacement, regardless of any information.

Custom: This approach takes advantage of the instance selection used in FT methods by incorporating the support samples constructed by encoder-decoder models. This allows decoder-only models to leverage the encoder of the encoder-decoder models. The goal is to assess the performance gain by using samples interesting for and selected by FT models. However, the decoder-only models

Table 2 (upper block):

Model Name	MPQA Type						MPQA Polarity						MPQA Intensity					
	0	10	20	50	100	Full	0	10	20	50	100	Full	0	10	20	50	100	Full
Majority Baseline	**56.6**						**54.8**						**37.2**					
Random Sampling																		
BART-Random		$56.8_{1.4}$	$56.7_{1.5}$	$59.5_{1.8}$	$63.3_{3.3}$	80.3		$73.2_{5.1}$	$78.9_{4.3}$	$82.9_{1.6}$	$86.8_{1.7}$	92.5		$36.0_{2.4}$	**$37.0_{0.2}$**	$37.1_{0.1}$	$35.2_{2.1}$	47.0
FLAN-T5-Random		$56.2_{4.6}$	$60.0_{2.9}$	$65.3_{2.6}$	$66.7_{2.7}$	**80.7**		$76.5_{2.4}$	$80.6_{2.4}$	$85.3_{1.4}$	$88.4_{0.9}$	**94.2**		$33.0_{3.8}$	$34.0_{3.6}$	$35.5_{1.6}$	$35.5_{1.5}$	**50.0**
Representative Sampling																		
BART-Rep(En)		$56.5_{0.2}$	$57.1_{1.7}$	$59.8_{2.7}$	$64.2_{3.5}$			$71.4_{0.0}$	$76.1_{2.6}$	$81.8_{1.0}$	$87.1_{1.2}$			**$37.0_{0.0}$**	$35.2_{2.1}$	$37.0_{0.4}$	$37.3_{0.3}$	
FLAN-T5-Rep(En)		$53.6_{6.2}$	**$63.7_{2.3}$**	$65.1_{1.5}$	$67.8_{1.9}$			**$77.5_{5.1}$**	$79.3_{1.8}$	$85.3_{2.7}$	$89.2_{1.6}$			$33.9_{2.2}$	$35.4_{0.9}$	$36.3_{1.1}$	$35.6_{1.1}$	
Iterative Approaches																		
FLAN-T5-Rep(En)-Un		$53.6_{6.2}$	$59.8_{1.4}$	$63.7_{2.6}$	$66.9_{2.3}$			**$77.5_{5.1}$**	$81.2_{5.8}$	$88.2_{1.9}$	**$91.4_{0.8}$**			$33.9_{2.2}$	$36.8_{0.3}$	$37.4_{0.7}$	$39.2_{2.1}$	
FLAN-T5-Rep(En)-Rep(Sc)		$53.6_{6.2}$	$61.3_{1.1}$	**$65.8_{1.9}$**	$68.5_{0.8}$			**$77.5_{5.1}$**	$80.8_{3.4}$	$87.4_{0.8}$	$90.6_{1.4}$			$33.9_{2.2}$	$35.3_{2.0}$	$37.4_{2.5}$	$38.0_{1.4}$	
FLAN-T5-Rep(En)-Rep(En)		$53.6_{6.2}$	$61.4_{2.9}$	$64.7_{2.1}$	69.1_{3}			**$77.5_{5.1}$**	$80.4_{2.0}$	$85.6_{0.6}$	$88.1_{1.4}$			$33.9_{2.2}$	$34.5_{2.3}$	$35.4_{2.3}$	$37.8_{1.2}$	
FLAN-T5-Rep(En)-UnRep		$53.6_{6.2}$	$60.6_{4.6}$	$63.2_{2.7}$	$68.8_{2.0}$			**$77.5_{5.1}$**	$82.4_{2.5}$	$87.5_{2.0}$	$90.1_{0.5}$			$33.9_{2.2}$	$36.2_{1.4}$	$37.4_{0.6}$	$38.8_{2.2}$	
FLAN-T5-Rep(En)-ClUn(Sc)		$53.6_{6.2}$	$59.6_{2.8}$	$64.4_{2.9}$	$68.0_{1.9}$			**$77.5_{5.1}$**	**$83.2_{2.4}$**	**$88.3_{0.9}$**	$90.4_{0.7}$			$33.9_{2.2}$	$36.3_{0.8}$	$36.4_{2.0}$	$38.2_{1.4}$	
FLAN-T5-Rep(En)-ClUn(En)		$53.6_{6.2}$	$59.2_{3.6}$	$64.6_{2.0}$	$69.3_{0.8}$			**$77.5_{5.1}$**	$81.5_{1.8}$	$87.5_{1.4}$	$90.8_{0.8}$			$33.9_{2.2}$	$35.1_{2.9}$	**$38.0_{1.6}$**	**$39.7_{1.8}$**	
In-Context Learning																		
Gemma 2-Random	49.3	$50.9_{6.8}$	$55.2_{3.3}$	-	-	-	67.0	$73.0_{2.1}$	$75.0_{3.3}$	-	-	-	32.0	$32.8_{2.6}$	$33.2_{2.6}$	-	-	-
Gemma 2-Custom		$50.5_{5.7}$	$52.9_{5.0}$	-	-			$73.5_{0.7}$	$76.1_{2.3}$	-	-			$33.5_{4.9}$	$33.8_{4.8}$	-	-	
Llama 3.1-Random	43.4	$57.2_{4.6}$	$58.2_{3.9}$	$59.5_{3.0}$	$60.1_{2.9}$	-	64.9	$69.7_{3.7}$	$70.4_{2.9}$	$68.3_{6.4}$	$75.7_{1.9}$	-	23.9	$31.3_{2.2}$	$31.6_{2.5}$	$30.6_{1.7}$	$31.0_{1.9}$	-
Llama 3.1-Custom		**$60.9_{0.9}$**	$61.0_{1.1}$	$61.8_{1.1}$	$61.5_{1.1}$			$63.3_{3.5}$	$67.6_{4.7}$	$72.1_{3.4}$	$67.2_{4.4}$			$32.4_{2.3}$	$33.0_{2.9}$	$31.4_{3.3}$	$31.4_{4.4}$	
Mistral v0.3-Random	39.4	$54.5_{4.3}$	$57.7_{2.6}$	$57.9_{2.6}$	$58.5_{2.1}$	-	**72.4**	$73.8_{2.1}$	$75.2_{1.3}$	$76.5_{2.1}$	$78.1_{3.4}$	-	22.9	$28.9_{2.4}$	$28.5_{2.4}$	$29.3_{0.6}$	$28.7_{2.1}$	-
Mistral v0.3-Custom		$53.9_{4.1}$	$56.9_{2.0}$	$58.6_{2.7}$	$57.8_{4.4}$			$72.1_{1.8}$	$74.1_{1.8}$	$73.6_{1.2}$	$73.9_{4.3}$			$29.4_{7.2}$	$30.7_{7.7}$	$32.6_{5.3}$	$29.7_{5.0}$	

Table 2 (lower block):

Model Name	AG News						Amazon Reviews						Mean					
	0	10	20	50	100	Full	0	10	20	50	100	Full	0	10	20	50	100	Full
Majority Baseline	25.0						20.0						38.7					
Random Sampling																		
BART-Random		$75.1_{7.3}$	$80.9_{4.2}$	$85.1_{1.0}$	$86.8_{1.2}$	94.2		$32.1_{3.0}$	$35.7_{2.7}$	$41.1_{1.8}$	$45.3_{2.7}$	63.2		54.6	57.8	61.1	63.5	75.4
FLAN-T5-Random		$71.8_{6.3}$	$87.3_{1.2}$	$88.4_{1.2}$	$89.3_{0.7}$	**94.4**		$47.2_{4.8}$	$52.7_{4.3}$	$55.9_{0.7}$	$58.8_{1.5}$	**65.7**		56.9	62.9	66.1	67.7	**77.0**
Representative Sampling																		
BART-Rep(En)		$61.4_{0.3}$	$75.9_{11.9}$	$86.3_{0.4}$	$86.6_{1.5}$			$29.9_{0.0}$	$34.9_{1.7}$	$38.7_{2.6}$	$45.9_{2.7}$			51.2	55.8	60.7	64.2	
FLAN-T5-Rep(En)		$86.9_{0.7}$	$86.3_{1.4}$	$88.4_{2.1}$	$89.1_{0.3}$			$51.5_{0.2}$	$51.5_{1.8}$	$55.0_{2.9}$	$59.3_{0.9}$			60.7	63.2	66.0	68.2	
Iterative Approaches																		
FLAN-T5-Rep(En)-Un		$86.9_{0.7}$	$87.7_{1.0}$	$88.5_{0.6}$	$88.7_{1.4}$			$51.5_{0.2}$	$54.7_{3.1}$	$57.0_{1.7}$	$57.5_{2.8}$			60.7	**64.0**	67.0	68.7	
FLAN-T5-Rep(En)-Rep(Sc)		$86.9_{0.7}$	$87.5_{1.2}$	**$88.7_{1.1}$**	**$89.7_{0.2}$**			$51.5_{0.2}$	$52.8_{2.2}$	$57.4_{1.1}$	$59.5_{1.4}$			60.7	63.5	**67.3**	69.3	
FLAN-T5-Rep(En)-Rep(En)		$86.9_{0.7}$	$86.6_{2.4}$	$88.6_{0.4}$	$89.3_{0.2}$			$51.5_{0.2}$	$52.6_{2.9}$	$57.3_{1.0}$	$58.7_{0.9}$			60.7	63.1	66.6	68.6	
FLAN-T5-Rep(En)-UnRep		$86.9_{0.7}$	$87.6_{0.3}$	$88.5_{1.2}$	$89.3_{0.5}$			$51.5_{0.2}$	$45.7_{4.2}$	$52.4_{3.2}$	$55.0_{3.1}$			60.7	62.5	65.8	68.4	
FLAN-T5-Rep(En)-ClUn(Sc)		$86.9_{0.7}$	$86.4_{1.8}$	$87.6_{1.0}$	$88.8_{0.7}$			$51.5_{0.2}$	$54.7_{1.8}$	**$58.4_{0.9}$**	$59.4_{0.4}$			60.7	**64.0**	67.0	69.0	
FLAN-T5-Rep(En)-ClUn(En)		$86.9_{0.7}$	$87.5_{0.7}$	$88.1_{2.0}$	$89.1_{0.8}$			$51.5_{0.2}$	$52.0_{3.5}$	$58.3_{1.6}$	**$59.9_{1.1}$**			60.7	63.1	**67.3**	**69.8**	
In-Context Learning																		
Gemma 2-Random	84.6	$85.2_{1.6}$	$86.8_{1.4}$	-	-	-	**62.2**	$60.6_{1.6}$	$60.1_{1.9}$	-	-	-	**59.0**	60.5	62.1	-	-	-
Gemma 2-Custom		**$87.2_{0.5}$**	**$88.1_{0.6}$**	-	-			**$61.9_{0.6}$**	**$60.5_{0.6}$**	-	-			**61.3**	62.3	-	-	
Llama 3.1-Random	82.5	$85.8_{1.4}$	$85.1_{1.4}$	$86.3_{1.3}$	$86.5_{1.3}$	-	59.4	$53.3_{5.4}$	$54.7_{4.8}$	$57.0_{2.7}$	$55.6_{3.1}$	-	54.8	59.5	60.0	60.3	61.8	-
Llama 3.1-Custom		$84.8_{1.0}$	$85.7_{0.7}$	$86.0_{0.9}$	$85.3_{1.4}$			$50.4_{2.4}$	$52.2_{2.6}$	$56.1_{2.4}$	$54.5_{1.2}$			58.4	59.9	61.5	60.0	
Mistral v0.3-Random	**84.9**	$82.9_{2.6}$	$85.3_{1.4}$	$86.1_{0.8}$	$86.4_{1.4}$	-	54.3	$59.4_{1.2}$	$58.5_{2.2}$	$49.7_{7.9}$	$46.1_{9.0}$	-	54.8	59.9	61.0	60.1	59.6	-
Mistral v0.3-Custom		$80.4_{2.8}$	$86.2_{0.7}$	$83.2_{2.9}$	$81.9_{3.6}$			$60.0_{0.9}$	$56.9_{2.2}$	$45.6_{6.6}$	$47.7_{2.8}$			59.2	61.0	58.7	58.2	

Table 2: The average micro-F1 (%) results for MPQA Type, MPQA Polarity, MPQA Intensity, AG News, and Amazon Reviews when $M = 10$ (i.e., selection size) in iterative approaches, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation. Any experiment that encountered out-of-memory errors is marked with a "-" symbol.

themselves do not participate in identifying or selecting these informative samples in this process, and the encoder-decoder models operate entirely independent of the decoder-only models.

In ICL, we apply both random and custom sampling methods on the instruction-tuned models, i.e., Gemma 2, Llama 3.1, and Mistral v0.3 with 9B, 8B, and 7B parameters, respectively. We report results for support sets of sizes $K \in \{0, 10, 20, 50, 100\}$. Notably, each step's support set is a subset of the support set from the subsequent step. In custom sampling, we only use instances selected by the overall best-performing model in FT experiments.

5 Discussion of Results

To better understand the impact of different approaches, we analyze the experimental results in this section. Table 2 summarizes the main exper-

imental results for FT and ICL approaches. The outcomes for the smaller selection size and comparisons with other related work are also available in Tables 3 and 4, respectively. Additional fine-grained results, including more intermediate support set sizes, are provided in Appendix E as well.

The model names in the tables indicate the employed pre-trained model, followed by the sampling method. If an iterative approach is used, the name reflects the sampling method for the first iteration, followed by the method used in the subsequent iterations. Whenever a referred sampling method can make use of both embedding methods, we specify the used method inside parentheses.

5.1 Active Few-Shot Learning

Table 2 elucidates significant differences in task performance, particularly when fine-tuning on the

Model Name	MPQA Type				MPQA Polarity				MPQA Intensity			
	5	10	25	50	5	10	25	50	5	10	25	50
Random Sampling												
BART-Random	$55.0_{3.8}$	$57.2_{2.4}$	$58.0_{2.1}$	$59.3_{1.3}$	$68.0_{9.0}$	$72.8_{4.2}$	$76.8_{3.3}$	$81.9_{3.4}$	$32.7_{6.2}$	$36.0_{1.9}$	$36.1_{1.5}$	$36.6_{1.2}$
FLAN-T5-Random	$46.8_{8.5}$	$55.6_{4.1}$	$59.7_{3.6}$	$64.5_{2.3}$	$67.2_{8.9}$	$74.4_{4.9}$	$80.5_{1.7}$	$84.3_{2.2}$	$28.0_{5.0}$	$31.0_{4.6}$	$34.6_{4.9}$	$36.0_{1.2}$
Representative Sampling												
BART-Rep(En)	$52.7_{0.0}$	$56.2_{0.7}$	$56.9_{1.6}$	$59.2_{2.3}$	$62.9_{15.3}$	$71.4_{0.0}$	$78.9_{3.4}$	$82.5_{2.7}$	$\mathbf{35.9}_{1.7}$	$\mathbf{37.0}_{0.0}$	$36.4_{1.2}$	$37.0_{0.6}$
FLAN-T5-Rep(En)	$\mathbf{59.3}_{2.4}$	$52.0_{5.6}$	$62.2_{2.7}$	$64.5_{2.0}$	$\mathbf{72.1}_{1.3}$	$78.3_{4.1}$	$80.6_{1.4}$	$85.8_{2.6}$	$29.2_{0.6}$	$34.3_{2.3}$	$35.4_{1.4}$	$36.5_{0.9}$
Iterative Approaches												
FLAN-T5-Rep(En)-Un	$\mathbf{59.3}_{2.4}$	$59.4_{5.2}$	$\mathbf{63.5}_{1.9}$	$\mathbf{65.7}_{1.8}$	$\mathbf{72.1}_{1.3}$	$73.6_{3.1}$	$\mathbf{84.7}_{2.1}$	$\mathbf{88.6}_{1.6}$	$29.2_{0.6}$	$33.3_{2.8}$	$35.7_{2.7}$	$38.0_{2.2}$
FLAN-T5-Rep(En)-Rep(Sc)	$\mathbf{59.3}_{2.4}$	$61.2_{3.2}$	$61.0_{3.5}$	$65.1_{2.1}$	$\mathbf{72.1}_{1.3}$	$\mathbf{81.2}_{1.7}$	$83.5_{2.2}$	$87.7_{2.1}$	$29.2_{0.6}$	$34.0_{2.2}$	$35.7_{1.9}$	$37.4_{1.5}$
FLAN-T5-Rep(En)-Rep(En)	$\mathbf{59.3}_{2.4}$	$\mathbf{62.2}_{2.0}$	$63.2_{3.0}$	$65.4_{2.4}$	$\mathbf{72.1}_{1.3}$	$78.2_{3.2}$	$81.9_{2.2}$	$84.1_{1.7}$	$29.2_{0.6}$	$31.9_{2.8}$	$33.8_{3.1}$	$34.7_{1.8}$
FLAN-T5-Rep(En)-UnRep	$\mathbf{59.3}_{2.4}$	$57.2_{4.7}$	$62.7_{4.3}$	$65.0_{1.3}$	$\mathbf{72.1}_{1.3}$	$79.1_{2.8}$	$84.3_{1.4}$	$87.5_{1.5}$	$29.2_{0.6}$	$32.6_{2.7}$	$35.1_{2.5}$	$\mathbf{38.9}_{1.0}$
FLAN-T5-Rep(En)-ClUn(Sc)	$\mathbf{59.3}_{2.4}$	$61.8_{3.3}$	$\mathbf{63.5}_{2.8}$	$65.0_{2.4}$	$\mathbf{72.1}_{1.3}$	$80.3_{2.3}$	$84.0_{1.8}$	$88.5_{1.8}$	$29.2_{0.6}$	$33.7_{1.5}$	$\mathbf{36.6}_{1.0}$	$37.6_{1.5}$
FLAN-T5-Rep(En)-ClUn(En)	$\mathbf{59.3}_{2.4}$	$60.7_{1.7}$	$63.2_{2.4}$	$65.1_{2.6}$	$\mathbf{72.1}_{1.3}$	$78.2_{3.0}$	$84.5_{1.7}$	$87.8_{1.3}$	$29.2_{0.6}$	$34.1_{1.9}$	$35.2_{3.8}$	$37.4_{2.4}$

Table 3: The average micro-F1 (%) results of MPQA Type, MPQA Polarity, and MPQA Intensity tasks when $M = 5$ (i.e., selection size) in iterative approaches, calculated over ten seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.
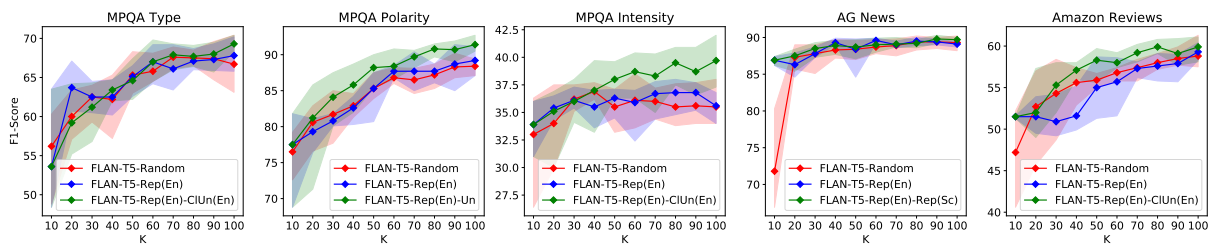

Figure 2: The range (min and max) and average of micro-F1 (%) scores of all tasks with steps of 10 over five runs.

full dataset. These variations stem from the distinct nature of the tasks, which span binary, multi-class, and multi-label classification problems. Notably, the MPQA Type and MPQA Intensity tasks involve imbalanced independent label sets, while the remaining tasks are more balanced. Additionally, the MPQA Intensity and Amazon Reviews tasks have a greater number of labels, which are mostly semantically close, adding more complexity to the classification. Thus, we believe that the chosen tasks offer a diverse and representative spectrum of classification challenges.

As anticipated, the majority baseline generally yields the poorest performance. Both BART-Random and FLAN-T5-Random provide reasonable starting points, with improvements observed as the support set size K increases. A common trend across FT approaches is the progressive enhancement in performance as the support set grows from 10 samples to the full dataset.

Furthermore, the table demonstrates that FLAN-T5-based models work better than BART-based models in most cases. This is why we focus our iterative experiments on FLAN-T5. However, additional experiments with BART can be found in Appendix E. The results also suggest that simple representative sampling is more effective than random sampling when $K = 100$, even when used in a non-iterative setup. Nevertheless, the iterative approaches outperform most non-iterative methods when $K \geq 20$. The MPQA Intensity task, in particular, succeeds in excelling the majority baseline in FSL experiments only when iterative methods are applied.

Although the five tasks produce distinct outcomes, the iterative approach 'FLAN-T5-Rep(En)-ClUn(En)' usually outperforms the random and representative approaches, and it does so consistently when $K \in \{50, 100\}$ in three tasks and when $K = 100$ in four tasks. All of the iterative approaches manage to excel the best non-iterative approaches on average when $K = 100$. Especially, 'FLAN-T5-Rep(En)-ClUn(En)' stands out as it beats the best non-iterative methods on average by 1.2% points at $K = 50$ and 1.6% points at $K = 100$. Hence, we recommend using this method in new use cases. Figure 2 captures the contrast between the non-iterative FLAN-T5-based models and the best-performer model at $K = 100$ for each task in greater detail, including the intermediate steps results.

5.2 Impact of Smaller Selection Size

Table 3 presents additional experimental results for $K \in \{5, 10, 25, 50\}$ on the MPQA dataset across 10 different seeds, using $M = 5$ in the iterative approaches. Even though, in our problem context, K represents the overall size of the support set—distinct from the conventional FSL classification tasks where K refers to the number of samples per class, making it challenging to ensure equitable representation of all labels in tasks like MPQA Intensity—the iterative approaches still surpass the non-iterative methods in most cases. Moreover, the iterative approach 'FLAN-T5-Rep(En)-ClUn(En)' still holds up and beats all the non-iterative approaches at $K = 50$. An interesting insight that can be drawn by comparing the $K = 50$ columns in Tables 2 and 3 shows that having $M = 5$ leads to improved performance across nearly all iterative approaches compared to $M = 10$ at the same support set size ($K = 50$). This suggests using a smaller M can result in a more effective sample selection process.

5.3 In-Context Learning

An examination of the ICL results reveals that these models generally not only underperform iterative approaches but also struggle to beat non-iterative FT methods. They even fail to exceed the majority baseline in the MPQA Intensity task. Although ICL models can usually deliver reasonable performance when K is small and may outperform iterative methods in Amazon Reviews, their performance stagnates or even declines as the number of support samples (K) increases. This inability of ICL approaches to leverage larger support sets has been noted in previous work as well (Pecher et al., 2024). Additionally, ICL methods are inclined to exhibit higher standard deviations compared to FT approaches, indicating greater performance variability.

Comparing the pre-trained models in ICL shows there is no definitive winner between Llama 3.1 and Mistral v0.3. Generally, Mistral v0.3 performs better with smaller K values, while Llama 3.1 surpasses Mistral v0.3 as K increases. However, Gemma 2 stands out as the clear leader, outperforming the other two models in most cases. Interestingly, custom sampling fails to help Llama 3.1 and Mistral v0.3 to gain performance against random sampling, suggesting that these two models do not benefit from instances identified as informative by

FLAN-T5. Gemma 2, however, is the only pretrained model that benefits from custom sampling in most experiments.

5.4 Other Related Work

Model Name	AG News	Amazon Reviews
LT margin	$86.2_{0.7}$	$46.6_{1.4}$
LT k-means	$82.8_{1.2}$	$48.6_{0.9}$
FLAN-T5-Rep(En)-Rep(Sc)	$90.7_{0.2}$	
FLAN-T5-Rep(En)-ClUn(En)		$61.0_{1.0}$

Table 4: The average macro-F1 (%) results for AG News and Amazon Reviews when $M = 16$ and $K = 256$, calculated over five different seeds in the sampling phase. The subscripts denote standard deviation.

Table 4 compares our best-performing models on AG News and Amazon Reviews datasets with the best-performing models reported in previous work on the same datasets, i.e., LT margin for AG News and LT k-means for Amazon Reviews (Müller et al., 2022). Our models outperform these baselines by a significant margin of 4.5% points on AG News and 12.4% points on Amazon Reviews, demonstrating the effectiveness of our methods. Figure 3 illustrates that LT margin and LT k-means at $K = 256$ struggle to beat the best iterative FT approaches even at small K values. The figure also shows how the performance gains of iterative approaches diminish as K increases. For example, there is a performance improvement of 3.3% points in AG News and 8.2% points in Amazon Reviews when K increases from 16 to 96. Meanwhile, there is only a 1.1% and 1.6% points improvement when K increases from 96 to 256 in AG News and Amazon Reviews, respectively. These results suggest the proposed methods are most effective in FS settings.
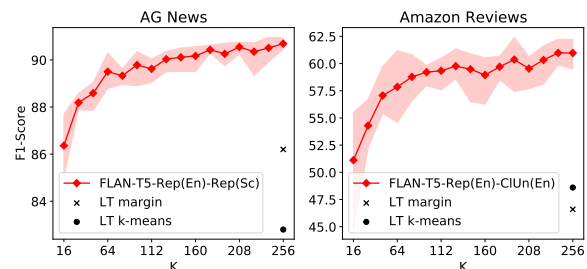

Figure 3: The range (min and max) and average of macro-F1 (%) scores of AG News and Amazon Reviews tasks with steps of 16 over five runs.

5.5 Performance-Efficiency Trade-off

In designing our methodology, we prioritized the quality of few-shot instance selection and model

performance, as these are critical in applications where model accuracy outweighs time efficiency constraints. However, here we analyze run-time performance to enable practical deployment in time-sensitive scenarios. We present a breakdown of execution times for different phases of the process, allowing for informed decisions when tuning key parameters to meet efficiency requirements.

Model Name	Embedding	Sampling	FT	Overall
FLAN-T5-Random	0:00:00	0:00:00	0:10:30	0:10:30
FLAN-T5-Rep(En)	0:00:16	0:00:03	0:08:59	0:09:18
FLAN-T5-Rep(En)-Un	0:01:58	0:00:01	1:48:06	1:50:04
FLAN-T5-Rep(En)-Rep(Sc)	0:02:14	0:00:02	1:55:16	1:57:32
FLAN-T5-Rep(En)-Rep(En)	0:01:38	0:00:07	1:56:44	1:58:28
FLAN-T5-Rep(En)-UnRep	0:02:03	0:00:03	2:08:03	2:10:09
FLAN-T5-Rep(En)-ClUn(Sc)	0:02:17	0:00:02	1:50:20	1:52:39
FLAN-T5-Rep(En)-ClUn(En)	0:01:56	0:00:04	1:49:36	1:51:37

Table 5: Accumulated execution time (h:mm:ss) of embedding extraction, sampling, and fine-tuning across different models.

Table 5 presents execution times for embedding extraction, sampling, and fine-tuning phases across different approaches on the MPQA Polarity dataset over a single seed. Durations are accumulated over ten iterations in iterative approaches. Annotation, support set augmentation, and evaluation times are excluded, as they are either negligible or independent of the approach. As expected, non-iterative methods exhibit shorter execution times compared to iterative ones. The results confirm that increasing the number of iterations directly extends the overall run time in a linear fashion, as it naturally requires more computational steps, though performance gains diminish, as illustrated in Figure 2. Thus, tuning the number of iterations is crucial for balancing performance and efficiency.

The embedding extraction time is also influenced by the embedding method used. Method *En* processes data through the encoder, while *Sc* requires both encoder and decoder passes, increasing computational cost. If both methods are applied, embeddings can still be obtained in a single model pass. Another key factor is the size of the unlabeled training pool; for large datasets (e.g., Amazon Reviews), random downsampling can significantly reduce execution time when necessary.

The sampling phase is relatively insignificant compared to embedding extraction and fine-tuning. Interestingly, iterative sampling does not take ten times the time of representative sampling, since the computations can be more simplified when choosing 10 samples per iteration compared to all 100 samples in a single iteration. Sampling method choice also impacts execution time, with non-K-Means approaches typically being faster. Similar to embedding extraction, downsampling the training pool can further enhance efficiency as well.

Fine-tuning times vary due to early stopping, with no clear correlation between approach type and fine-tuning duration beyond iteration count. However, fine-tuning time is highly dependent on validation set size, and reducing the validation set can significantly shorten run time when needed. Random downsampling of the validation set offers an effective strategy for optimizing efficiency.

Overall, our findings highlight key factors (e.g., iteration count, training pool size, validation set size, etc.) that influence execution time. Proper tuning of these parameters enables a flexible trade-off between model performance and computational efficiency.

6 Conclusion and Future Work

We propose a novel fine-tuning-based method for sampling data to be used in a few-shot setting with active learning, while many others tend to sample data randomly. We show how using different embedding and sampling methods helps us achieve better results in classification tasks by choosing and labeling the most informative unlabeled samples that may represent the variety of data or that the model has the most doubts about. These methods unleash their full potential when used iteratively, using the fine-tuned model from the previous iterations, surpassing in-context learning approaches and other fine-tuning-based sampling strategies in previous studies.

Future work will expand on new embedding and sampling methods in classification tasks as well as other types of NLP tasks, such as text generation. It will also explore the effect of semi-supervised learning methods on top of our approach in a pipeline, making use of the rest of the unlabeled data to improve performance.

Acknowledgment

Limitations

In the current study, we have centered our attention on English. In the future, we plan to focus on other natural languages and alternative datasets. Furthermore, our proposed methods are unable to be directly used in non-classification or non-NLP tasks and they need some modifications to be applied to these types of tasks. These experiments also require a lot of computational resources like the other AL approaches, since we have to iteratively run the same experiment 10 times with an incrementally augmented support set.

Ethics Statement

Our current study is a fundamental research work in the field of NLP and computational linguistics. There are many applications considered for these fields of research. For instance, understanding users' tweets on Twitter, e-commerce applications, and question answering. Although many research projects have been done in these fields, and a large number of them have accomplished remarkable results, we do not explicitly recommend using these systems standalone. The reason is that there are open issues about the robustness and fairness of these systems. Hence, we see a need for human experts in interpreting the results. From our point of view, there are no ethical concerns about the platforms, technologies, tools, and algorithms used or proposed in this study. We should also note that the dataset, language models, tools, and libraries that we have utilized in this work are all publicly available.

References

Saeed Ahmadnia, Arash Yousefi Jordehi, Mahsa Hosseini Khasheh Heyran, SeyedAbolghasem Mirroshandel, and Owen Rambow. 2024. Opinion mining using pre-trained large language models: Identifying the type, polarity, intensity, expression, and source of private states. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 12481–12495, Torino, Italia. ELRA and ICCL.

Rinu Boney, Alexander Ilin, et al. 2019. Active one-shot learning with prototypical networks. In *ESANN*.

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*, volume 33, pages 1877–1901. Curran Associates, Inc.

Ernie Chang, Xiaoyu Shen, Hui-Syuan Yeh, and Vera Demberg. 2021. On training instance selection for few-shot neural text generation. *arXiv preprint arXiv:2107.03176*.

Wei-Yu Chen, Yen-Cheng Liu, Zsolt Kira, Yu-Chiang Frank Wang, and Jia-Bin Huang. 2019. A closer look at few-shot classification. *arXiv preprint arXiv:1904.04232*.

Yiming Chen, Yan Zhang, Chen Zhang, Grandee Lee, Ran Cheng, and Haizhou Li. 2021. Revisiting self-training for few-shot learning of language model. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9125–9135, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhiyu Chen, Harini Eavani, Wenhu Chen, Yinyin Liu, and William Yang Wang. 2020. Few-shot NLG with pre-trained language model. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 183–190, Online. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Aleksandra Edwards and Jose Camacho-Collados. 2024. Language models for text classification: Is in-context learning enough? In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 10058–10072, Torino, Italia. ELRA and ICCL.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. Making pre-trained language models better few-shot learners. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Antonio Gulli. 2005. Ag's corpus of news articles.

Beliz Gunel, Jingfei Du, Alexis Conneau, and Ves Stoyanov. 2020. Supervised contrastive learning for pretrained language model fine-tuning. *arXiv preprint arXiv:2011.01403*.

AQ Jiang, A Sablayrolles, A Mensch, C Bamford, DS Chaplot, D de las Casas, F Bressand, G Lengyel, G Lample, L Saulnier, et al. 2023. Mistral 7b (2023). *arXiv preprint arXiv:2310.06825*.

Rabeeh Karimi Mahabadi, Luke Zettlemoyer, James Henderson, Lambert Mathias, Marzieh Saeidi, Veselin Stoyanov, and Majid Yazdani. 2022. Prompt-free and efficient few-shot learning with language models. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3638–3652, Dublin, Ireland. Association for Computational Linguistics.

Phillip Keung, Yichao Lu, György Szarvas, and Noah A. Smith. 2020. The multilingual Amazon reviews corpus. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4563–4568, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Xiaorun Li, Zeyu Cao, Liaoying Zhao, and Jianfeng Jiang. 2021. Alpn: Active-learning-based prototypical network for few-shot hyperspectral imagery classification. *IEEE Geoscience and Remote Sensing Letters*, 19:1–5.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. 2022. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 9019–9052.

Thomas Müller, Guillermo Pérez-Torró, Angelo Basile, and Marc Franco-Salvador. 2022. Active few-shot learning with fasl. In *International Conference on Applications of Natural Language to Information Systems*, pages 98–110. Springer.

Branislav Pecher, Ivan Srba, Maria Bielikova, and Joaquin Vanschoren. 2024. Automatic combination of sample selection strategies for few-shot learning. *arXiv preprint arXiv:2402.03038*.

Pouya Pezeshkpour, Zhengli Zhao, and Sameer Singh. 2020. On the utility of active instance selection for few-shot learning. *NeurIPS HAMLETS*.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Mengye Ren, Eleni Triantafillou, Sachin Ravi, Jake Snell, Kevin Swersky, Joshua B Tenenbaum, Hugo Larochelle, and Richard S Zemel. 2018. Meta-learning for semi-supervised few-shot classification. *arXiv preprint arXiv:1803.00676*.

Timo Schick and Hinrich Schütze. 2021a. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402.

Timo Schick and Hinrich Schütze. 2021b. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352, Online. Association for Computational Linguistics.

Burr Settles. 2009. Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Junsup Shin, Youngwook Kang, Seungjin Jung, and Jongwon Choi. 2022. Active instance selection for few-shot classification. *IEEE Access*.

Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. 2019. Meta-transfer learning for few-shot learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. 2024. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*.

Xixi Wang, Xiao Wang, Bo Jiang, and Bin Luo. 2023. Few-shot learning meets transformer: Unified query-support transformers for few-shot classification. *IEEE Transactions on Circuits and Systems for Video Technology*.

Janyce Wiebe, Theresa Wilson, and Claire Cardie. 2005. Annotating expressions of opinions and emotions in language. *Language resources and evaluation*, 39(2):165–210.

Theresa Ann Wilson. 2008. *Fine-grained subjectivity and sentiment analysis: recognizing the intensity, polarity, and attitudes of private states*. University of Pittsburgh.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Jingbo Zhu, Huizhen Wang, Benjamin K Tsou, and Matthew Ma. 2009. Active learning with sampling by uncertainty and density for data annotations. *IEEE Transactions on audio, speech, and language processing*, 18(6):1323–1331.

A   MPQA Opinion Corpus

A sentence may contain expressions that reflect different private states with various attitudes. These attitudes can belong to different types, and each type can express negative or positive opinions (polarity) toward targets with varying degrees of strength (intensity) (Wiebe et al., 2005; Wilson, 2008).

Task	Input		Output
	Attitude Type	**Sentence**	
T	-	The new US policy **deserves to be closely analyzed and monitored**.	arguing sentiment
P	intention	Canada is among the countries that have **pledged to** ratify the accord.	positive
I	sentiment	There is **a deep faith** here, however, in the power of democracy.	high

Table 6: The examples for Type (T), Polarity (P), and Intensity (I) tasks. The expressions within the sentences are in bold.

The original MPQA annotation scheme comprises 6 types of attitudes. We remove the *other* and *speculation* types in our experiments as these types of attitudes do not hold a polarity. That leaves us with a 4-class classification task for the type. Furthermore, an expression in a sentence may have zero to four labels as attitude types based on the expression itself and the sentence that contains the expression. This leads the type identifier task to be a multi-label classification task. Subsequently, we identify polarity and intensity using the attitude type, the expression that holds the attitude, and the expression's container sentence as the input. This input can only have one specific polarity and one in-

tensity, which makes these tasks binary and 5-class multi-class classification tasks, respectively.

An example for each task is available in Table 6, and all labels and their distribution are as follows: **type:** *agreement* (×284), *arguing* (×2,466), *intention* (×420), and *sentiment* (×3,862) **polarity:** *negative* (×3,200) and *positive* (×3,832); and **intensity:** *low* (×658), *low-medium* (×1,262), *medium* (×2,615), *medium-high* (×1,258), and *high* (×1,239).

B   Implementation Details

Our models were implemented on PyTorch[1] neural network framework. Furthermore, we utilized the scikit-learn library[2], NumPy[3], and Matplotlib[4] packages. We used `facebook/bart-base` and `google/flan-t5-base` models in FT and `google/gemma-2-9b-it`, `meta-llama/Llama-3.1-8B-Instruct`, and `mistralai/Mistral-7B-Instruct-v0.3` in ICL and their tokenizers from the *Hugging Face* Transformers library[5] (Wolf et al., 2020). For the ICL tasks, non-MPQA tasks, performance-efficiency tradeoff experiments, BART-based MPQA Polarity iterative experiments, and the analysis of the impact of parameter α, the models were executed on a single NVIDIA RTX A5000 GPU 24 GB GPU and AMD EPYC 7662 3.28 GHz 64-Core CPU. The rest of the tasks (many of the FT MPQA tasks) were executed on a single NVIDIA A100 40 GB GPU and dual AMD Rome 7742 CPUs (each with 2.25 GHz 64-Cores). The maximum amount of GPU memory we used for our approaches was 18 GB. They also required a maximum of 16 GB of RAM.

Parameter	BART	FLAN-T5
Batch Size	10	10
Learning Rate	5e-5	1e-4
Dropout Rate	0.1	0.1
Optimizer	AdamW	AdamW
Early Stopping	20 epochs	20 epochs

Table 7: The hyperparameters for BART- and FLAN-T5-based models.

All results in this paper are reproducible by setting the random seeds to fixed numbers. The hyperparameters used in our experiments are listed

---

[1]https://pytorch.org/
[2]https://scikit-learn.org/stable/
[3]https://numpy.org/
[4]https://matplotlib.org/
[5]https://github.com/huggingface/transformers

in Table 7. In the present study, we utilized publicly available datasets. Hence, we did not use any human annotators.

C   ICL Templates

This section contains prompt templates that we used in ICL experiments for MPQA Type in Figure 4, MPQA Polarity in Figure 5, MPQA Intensity in Figure 6, AG News in Figure 7, and Amazon Reviews in Figure 8.

D   Impact of Parameter α

Table 8 presents the results of the "FLAN-T5-Rep(En)-UnRep" approach for $\alpha \in \{1, 2, 5, 10, 20, 50\}$ with $M = 10$, using the MPQA Polarity dataset.

E   Fine-Grained Experiments

Tables 9 (MPQA Type), 10 (MPQA Polarity), 11 (MPQA Intensity), 12 (AG News), and 13 (Amazon Reviews) present the main experiments from Table 2, along with intermediate steps for better comparison. Table 10 also includes additional experiments for BART-based iterative approaches using the MPQA Polarity dataset. The results are reported for support set sizes of $K \in \{0, 5, 10, 20, 30, 40, 50, 60, 70, 80, 90, 100, Full\}$. Some of the experiments encountered out-of-memory errors.

```
System:
You are a sentiment classifier. For each message, predict the type of an expressed private state as (sentiment, intention,
arguing, or agreement) given the whole sentence for context, followed by the expression itself. Each entry may belong to
one or more classes. Return the predicted classes with exactly one space between each if there is more than one class. Do
not add any additional information, comment, or formatting.

User:
Text: [Test Sample]
Type:
```

```
System:
You are a sentiment classifier. For each message, predict the type of an expressed private state as (sentiment, intention,
arguing, or agreement) given the whole sentence for context, followed by the expression itself. Each entry may belong to
one or more classes. Return the predicted classes with exactly one space between each if there is more than one class. Do
not add any additional information, comment, or formatting. Here are some samples:

Text: [Support Sample #1]
Type: [Label of Support Sample #1]

...

Text: [Support Sample #K]
Type: [Label of Support Sample #K]

User:
Text: [Test Sample]
Type:
```
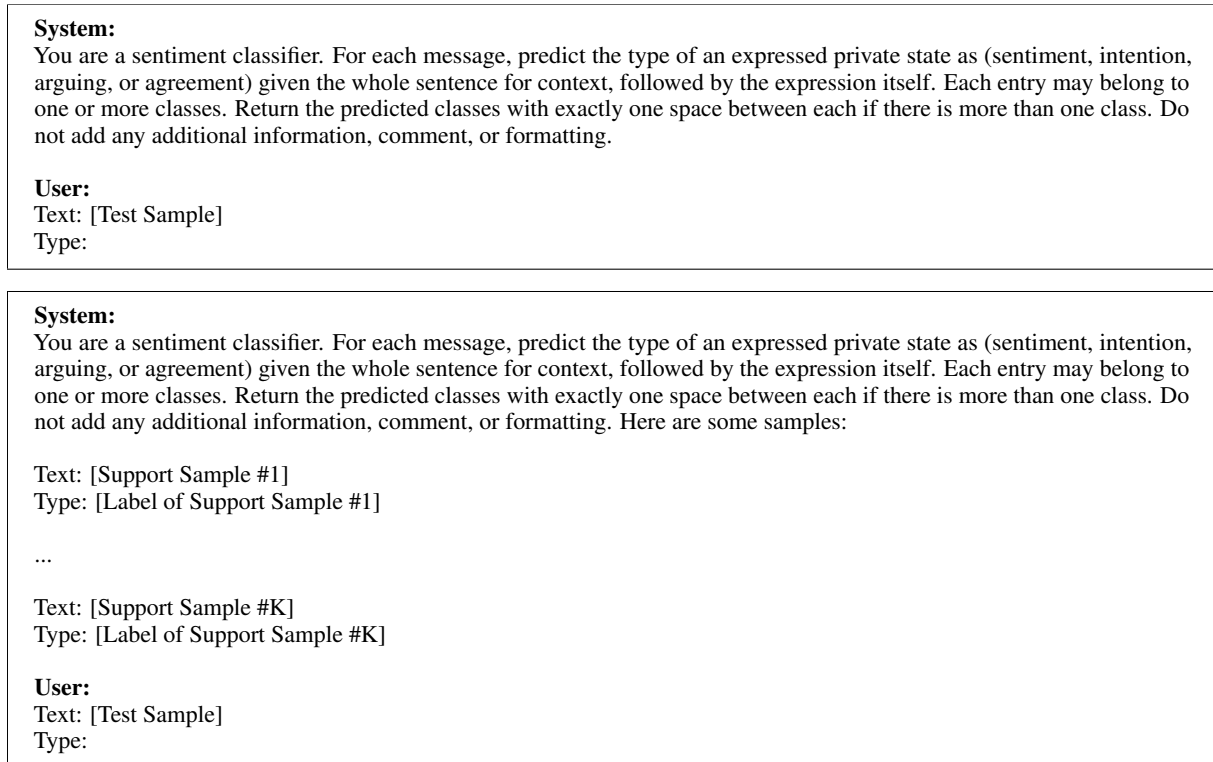
Figure 4: Prompt templates for **MPQA Type** task. The upper box shows a sample for zero-shot learning, and the
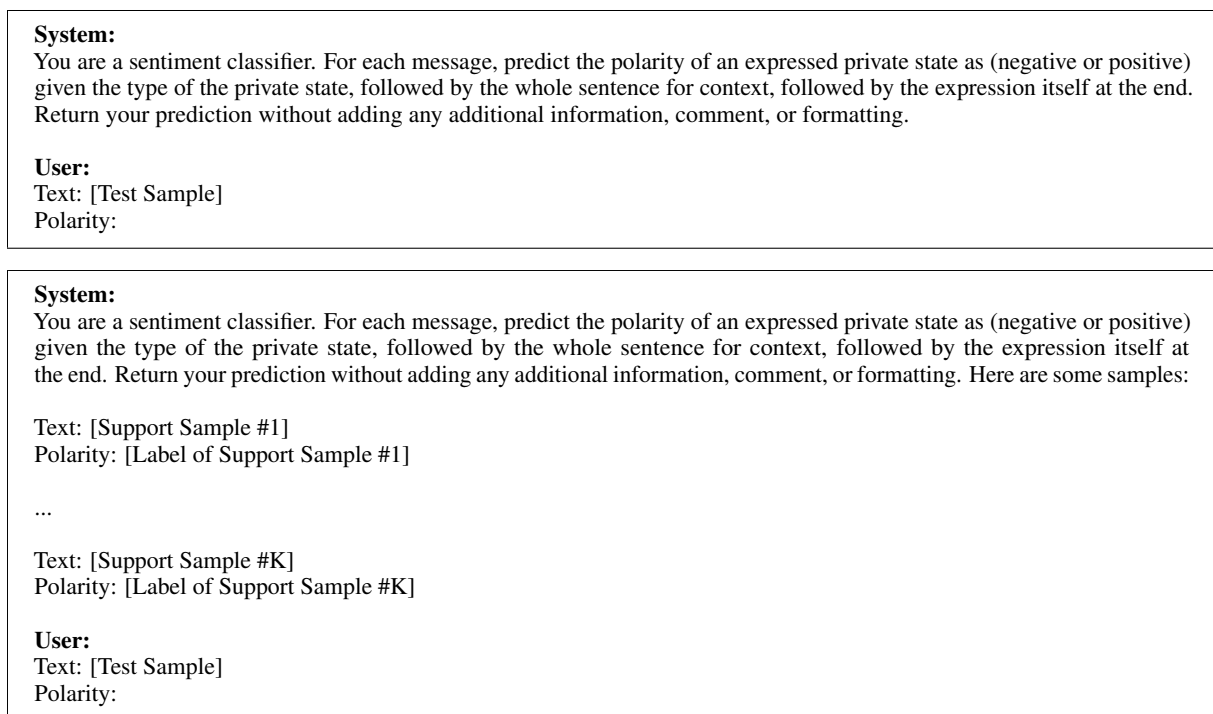lower one shows a sample for K-shot learning.

```
System:
You are a sentiment classifier. For each message, predict the polarity of an expressed private state as (negative or positive)
given the type of the private state, followed by the whole sentence for context, followed by the expression itself at the end.
Return your prediction without adding any additional information, comment, or formatting.

User:
Text: [Test Sample]
Polarity:
```

```
System:
You are a sentiment classifier. For each message, predict the polarity of an expressed private state as (negative or positive)
given the type of the private state, followed by the whole sentence for context, followed by the expression itself at
the end. Return your prediction without adding any additional information, comment, or formatting. Here are some samples:

Text: [Support Sample #1]
Polarity: [Label of Support Sample #1]

...

Text: [Support Sample #K]
Polarity: [Label of Support Sample #K]

User:
Text: [Test Sample]
Polarity:
```

Figure 5: Prompt templates for **MPQA Polarity** task. The upper box shows a sample for zero-shot learning, and the
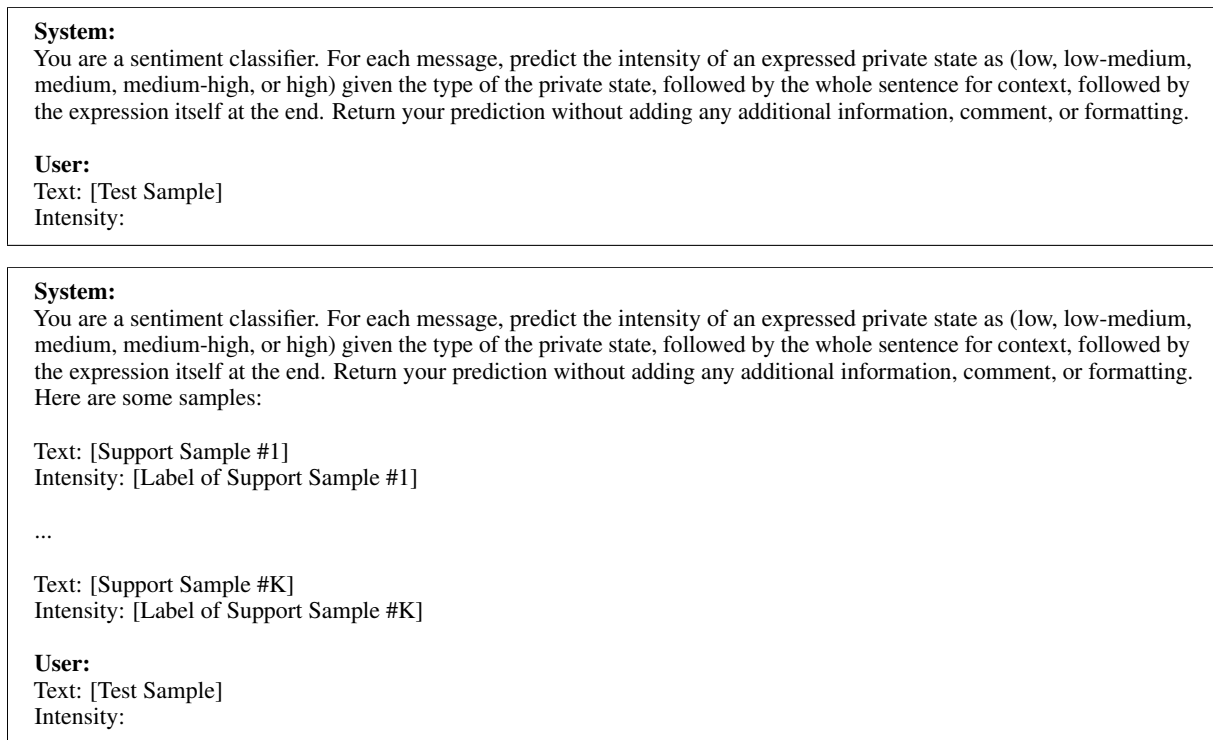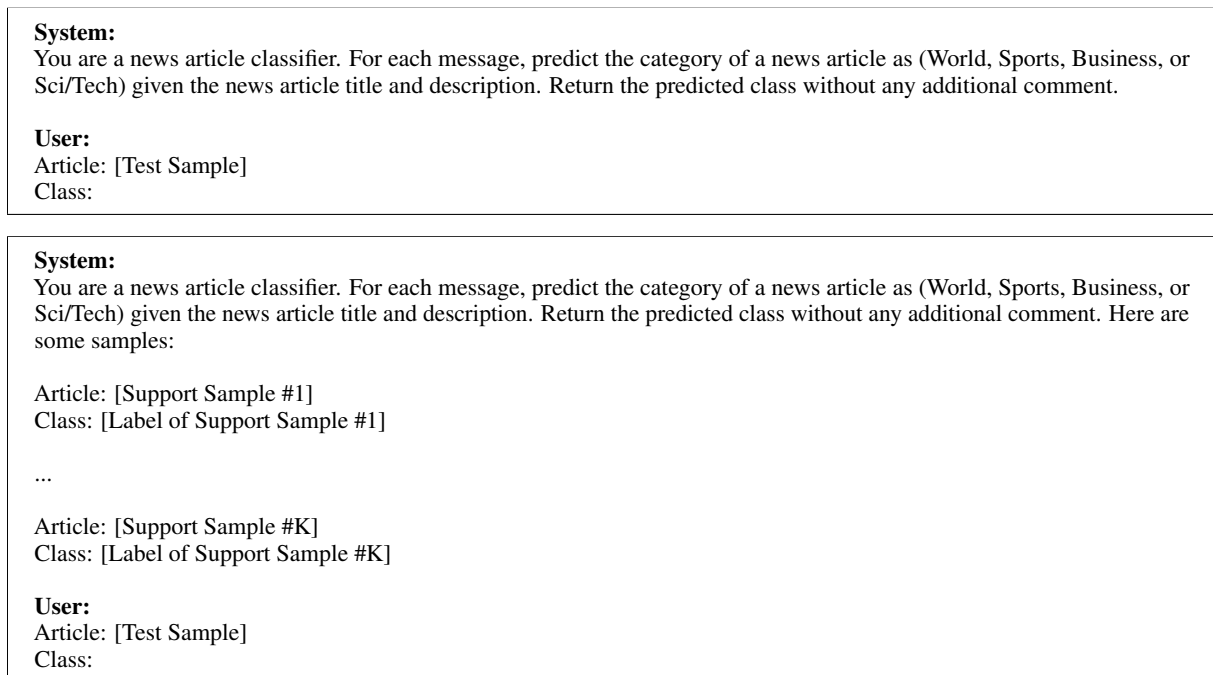lower one shows a sample for K-shot learning.

System:
You are a sentiment classifier. For each message, predict the intensity of an expressed private state as (low, low-medium, medium, medium-high, or high) given the type of the private state, followed by the whole sentence for context, followed by the expression itself at the end. Return your prediction without adding any additional information, comment, or formatting.

User:
Text: [Test Sample]
Intensity:

System:
You are a sentiment classifier. For each message, predict the intensity of an expressed private state as (low, low-medium, medium, medium-high, or high) given the type of the private state, followed by the whole sentence for context, followed by the expression itself at the end. Return your prediction without adding any additional information, comment, or formatting. Here are some samples:

Text: [Support Sample #1]
Intensity: [Label of Support Sample #1]

...

Text: [Support Sample #K]
Intensity: [Label of Support Sample #K]

User:
Text: [Test Sample]
Intensity:

Figure 6: Prompt templates for **MPQA Intensity** task. The upper box shows a sample for zero-shot learning, and the lower one shows a sample for K-shot learning.

System:
You are a news article classifier. For each message, predict the category of a news article as (World, Sports, Business, or Sci/Tech) given the news article title and description. Return the predicted class without any additional comment.

User:
Article: [Test Sample]
Class:

System:
You are a news article classifier. For each message, predict the category of a news article as (World, Sports, Business, or Sci/Tech) given the news article title and description. Return the predicted class without any additional comment. Here are some samples:

Article: [Support Sample #1]
Class: [Label of Support Sample #1]

...

Article: [Support Sample #K]
Class: [Label of Support Sample #K]

User:
Article: [Test Sample]
Class:

Figure 7: Prompt templates for **AG News** task. The upper box shows a sample for zero-shot learning, and the lower one shows a sample for K-shot learning.

Figure 8: Prompt templates for **Amazon Reviews** task. The upper box shows a sample for zero-shot learning, and the lower one shows a sample for K-shot learning.

α	10	20	30	40	50	60	70	80	90	100
1	$\mathbf{76.4}_{3.7}$	$80.3_{1.0}$	$82.9_{3.1}$	$84.3_{2.8}$	$86.3_{1.6}$	$88.6_{1.2}$	$88.1_{1.3}$	$90.3_{1.5}$	$90.2_{0.5}$	$90.6_{1.2}$
2	$\mathbf{76.4}_{3.7}$	$\mathbf{81.6}_{1.1}$	$\mathbf{83.5}_{1.4}$	$85.2_{1.9}$	$86.5_{0.6}$	$87.8_{2.5}$	$88.9_{1.6}$	$\mathbf{90.5}_{1.4}$	$90.7_{1.0}$	$91.0_{0.6}$
5	$\mathbf{76.4}_{3.7}$	$81.3_{3.3}$	$83.4_{2.2}$	$\mathbf{86.7}_{1.9}$	$\mathbf{87.4}_{2.6}$	$87.5_{2.1}$	$88.9_{1.7}$	$90.2_{0.9}$	$\mathbf{91.0}_{1.1}$	$91.1_{0.9}$
10	$\mathbf{76.4}_{3.7}$	$80.7_{2.6}$	$83.1_{0.5}$	$84.9_{1.6}$	$86.7_{1.3}$	$\mathbf{89.5}_{1.7}$	$\mathbf{89.8}_{1.0}$	$90.0_{0.4}$	$\mathbf{91.0}_{0.6}$	$91.1_{0.5}$
20	$\mathbf{76.4}_{3.7}$	$80.4_{1.4}$	$80.4_{5.6}$	$83.3_{2.7}$	$84.5_{1.3}$	$86.5_{1.8}$	$88.2_{0.6}$	$89.4_{1.1}$	$90.3_{1.0}$	$91.2_{0.7}$
50	$\mathbf{76.4}_{3.7}$	$79.8_{3.0}$	$83.2_{1.8}$	$84.8_{1.9}$	$85.9_{1.8}$	$88.3_{2.1}$	$89.1_{2.5}$	$90.0_{1.8}$	$90.7_{1.3}$	$\mathbf{91.3}_{1.4}$

Table 8: The average micro-F1 (%) results for **MPQA Polarity** when $M = 10$ (i.e., selection size) in the "FLAN-T5-Rep(En)-UnRep" approach, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.

Model Name	0	5	10	20	30	40	50	60	70	80	90	100	Full
Majority Baseline	**56.6**												
Random Sampling													
BART-Random			$56.8_{1.4}$	$56.7_{1.5}$			$59.5_{1.8}$					$63.3_{3.3}$	80.3
FLAN-T5-Random			$56.2_{4.6}$	$60.0_{2.9}$	$62.5_{2.1}$	$62.2_{3.2}$	$65.3_{2.6}$	$65.8_{1.6}$	$67.6_{1.3}$	$67.5_{0.9}$	$67.4_{1.8}$	$66.7_{2.7}$	**80.7**
Representative Sampling													
BART-Rep(En)			$56.5_{0.2}$	$57.1_{1.7}$			$59.8_{2.7}$					$64.2_{3.5}$	
FLAN-T5-Rep(En)			$53.6_{6.2}$	$\mathbf{63.7}_{2.3}$	$62.5_{1.8}$	$62.5_{1.9}$	$65.1_{1.5}$	$67.0_{2.0}$	$66.1_{2.6}$	$67.1_{1.1}$	$67.3_{1.4}$	$67.8_{1.9}$	
Iterative Approaches													
FLAN-T5-Rep(En)-Un			$53.6_{6.2}$	$59.8_{1.4}$	$59.9_{2.2}$	$61.4_{1.9}$	$63.7_{2.6}$	$64.0_{1.9}$	$66.3_{2.8}$	$66.5_{2.5}$	$67.4_{2.4}$	$66.9_{2.3}$	
FLAN-T5-Rep(En)-Rep(Sc)			$53.6_{6.2}$	$61.3_{1.1}$	$\mathbf{63.4}_{2.1}$	$64.7_{1.4}$	$\mathbf{65.8}_{1.9}$	$\mathbf{68.2}_{1.3}$	$67.1_{1.7}$	$67.8_{0.9}$	$68.2_{1.8}$	$68.5_{0.8}$	
FLAN-T5-Rep(En)-Rep(En)			$53.6_{6.2}$	$61.4_{2.9}$	$63.1_{2.2}$	$63.4_{2.0}$	$64.7_{2.1}$	$66.4_{1.4}$	$67.1_{2.2}$	$\mathbf{69.0}_{1.5}$	$\mathbf{68.7}_{1.7}$	$68.9_{1.3}$	
FLAN-T5-Rep(En)-UnRep			$53.6_{6.2}$	$60.6_{4.6}$	$62.5_{2.8}$	$62.0_{3.1}$	$63.2_{2.7}$	$65.8_{2.2}$	$66.8_{1.3}$	$66.7_{1.9}$	$67.4_{0.5}$	$68.8_{2.0}$	
FLAN-T5-Rep(En)-ClUn(Sc)			$53.6_{6.2}$	$59.6_{2.8}$	$63.3_{2.5}$	$\mathbf{64.8}_{1.0}$	$64.4_{2.9}$	$65.6_{1.8}$	$65.4_{1.3}$	$66.9_{1.7}$	$68.0_{2.2}$	$68.0_{1.9}$	
FLAN-T5-Rep(En)-ClUn(En)			$53.6_{6.2}$	$59.2_{3.6}$	$61.2_{3.2}$	$63.4_{1.3}$	$64.6_{2.0}$	$67.0_{2.3}$	$\mathbf{67.9}_{1.3}$	$67.7_{1.2}$	$68.0_{1.4}$	$\mathbf{69.3}_{0.8}$	
In-Context Learning													
Gemma 2-Random	49.3	$48.4_{6.2}$	$50.9_{6.8}$	$55.2_{3.3}$	$63.0_{1.5}$								
Gemma 2-Custom			$50.5_{5.7}$	$52.9_{5.0}$									
Llama 3.1-Random	43.4	$\mathbf{57.0}_{4.7}$	$57.2_{4.6}$	$58.2_{3.9}$	$59.4_{2.9}$	$58.9_{3.6}$	$59.5_{3.0}$					$60.1_{2.9}$	
Llama 3.1-Custom			$60.9_{0.9}$	$61.0_{1.1}$			$61.8_{1.1}$					$61.5_{1.1}$	
Mistral v0.3-Random	39.4	$49.4_{4.5}$	$54.5_{4.3}$	$57.7_{2.6}$	$58.0_{0.8}$	$57.6_{1.8}$	$58.7_{0.9}$					$58.5_{2.1}$	
Mistral v0.3-Custom			$53.9_{4.1}$	$56.9_{2.0}$			$58.6_{2.7}$					$57.8_{4.4}$	

Table 9: The average micro-F1 (%) results for **MPQA Type** when $M = 10$ (i.e., selection size) in iterative approaches, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.

Model Name	0	5	10	20	30	40	50	60	70	80	90	100	Full
Majority Baseline	54.8												
Random Sampling													
BART-Random			$73.2_{5.1}$	$78.9_{4.3}$			$82.9_{1.6}$					$86.8_{1.7}$	92.5
FLAN-T5-Random			$76.5_{2.4}$	$80.6_{2.4}$	$81.7_{2.7}$	$82.9_{1.2}$	$85.3_{1.4}$	$86.8_{0.9}$	$86.5_{1.3}$	$87.2_{1.3}$	$88.3_{1.4}$	$88.4_{0.9}$	**94.2**
Representative Sampling													
BART-Rep(En)			$71.4_{0.0}$	$76.1_{2.6}$			$81.8_{1.0}$					$87.1_{1.2}$	
FLAN-T5-Rep(En)			$77.5_{5.1}$	$79.3_{1.8}$	$80.8_{0.6}$	$82.6_{1.7}$	$85.3_{2.7}$	$87.7_{0.7}$	$87.7_{2.0}$	$87.7_{1.1}$	$88.7_{2.6}$	$89.2_{1.6}$	
Iterative Approaches													
BART-Rep(En)-Un			$\mathbf{77.8}_{5.5}$	$80.7_{2.0}$	$81.7_{3.0}$	$80.5_{4.2}$	$84.6_{2.4}$	$83.5_{2.6}$	$84.7_{4.8}$	$87.0_{1.3}$	$87.8_{2.9}$	$87.5_{2.5}$	
BART-Rep(En)-Rep(Sc)			$\mathbf{77.8}_{5.5}$	$77.5_{4.9}$	$77.4_{3.4}$	$82.1_{4.4}$	$81.7_{5.4}$	$82.9_{3.9}$	$84.9_{2.0}$	$84.3_{2.5}$	$86.0_{2.0}$	$86.6_{1.9}$	
BART-Rep(En)-Rep(En)			$\mathbf{77.8}_{5.5}$	$79.8_{3.9}$	$81.2_{3.1}$	$80.4_{3.9}$	$84.1_{3.3}$	$84.6_{2.7}$	$87.4_{1.2}$	$86.4_{1.4}$	$85.7_{2.5}$	$88.2_{0.8}$	
BART-Rep(En)-UnRep			$\mathbf{77.8}_{5.5}$	$78.3_{4.3}$	$80.1_{2.5}$	$80.7_{4.5}$	$83.5_{4.2}$	$82.9_{4.5}$	$85.1_{1.9}$	$83.9_{4.9}$	$82.5_{5.5}$	$86.2_{1.5}$	
BART-Rep(En)-ClUn(Sc)			$\mathbf{77.8}_{5.5}$	$80.1_{4.5}$	$81.1_{3.4}$	$83.6_{2.5}$	$84.6_{2.2}$	$83.5_{2.6}$	$85.3_{1.6}$	$83.2_{3.4}$	$86.3_{2.1}$	$87.2_{2.5}$	
BART-Rep(En)-ClUn(En)			$\mathbf{77.8}_{5.5}$	$79.2_{4.4}$	$83.0_{2.8}$	$83.4_{2.4}$	$84.0_{1.6}$	$84.1_{2.2}$	$82.2_{2.3}$	$84.8_{4.6}$	$86.6_{2.1}$	$87.2_{1.6}$	
FLAN-T5-Rep(En)-Un			$77.5_{5.1}$	$81.2_{5.8}$	$84.1_{3.0}$	$\mathbf{85.8}_{2.7}$	$88.2_{1.9}$	$88.4_{1.6}$	$89.7_{1.0}$	$\mathbf{90.8}_{0.8}$	$90.7_{1.1}$	$\mathbf{91.4}_{0.8}$	
FLAN-T5-Rep(En)-Rep(Sc)			$77.5_{5.1}$	$80.8_{3.4}$	$82.6_{1.2}$	$84.9_{2.3}$	$87.4_{0.8}$	$86.5_{2.6}$	$87.4_{2.4}$	$89.6_{1.1}$	$89.9_{1.5}$	$90.6_{1.4}$	
FLAN-T5-Rep(En)-Rep(En)			$77.5_{5.1}$	$80.4_{2.0}$	$82.2_{1.7}$	$84.0_{2.3}$	$85.6_{0.6}$	$85.5_{1.2}$	$87.0_{1.5}$	$86.4_{2.0}$	$87.4_{0.7}$	$88.1_{1.4}$	
FLAN-T5-Rep(En)-UnRep			$77.5_{5.1}$	$82.4_{2.5}$	$84.1_{2.4}$	$85.3_{1.0}$	$87.5_{2.0}$	$88.4_{1.7}$	$89.0_{1.2}$	$89.2_{1.5}$	$89.7_{0.8}$	$90.1_{0.5}$	
FLAN-T5-Rep(En)-ClUn(Sc)			$77.5_{5.1}$	$\mathbf{83.2}_{2.4}$	$\mathbf{85.4}_{2.6}$	$85.5_{2.0}$	$\mathbf{88.3}_{0.9}$	$\mathbf{89.1}_{0.5}$	$89.6_{0.8}$	$89.5_{0.8}$	$90.4_{0.7}$	$90.4_{0.7}$	
FLAN-T5-Rep(En)-ClUn(En)			$77.5_{5.1}$	$81.5_{1.8}$	$85.0_{2.2}$	$85.5_{1.4}$	$87.5_{1.4}$	$88.8_{1.5}$	$\mathbf{89.8}_{1.0}$	$89.7_{1.2}$	$\mathbf{91.0}_{0.5}$	$90.8_{0.8}$	
In-Context Learning													
Gemma 2-Random	67.0	$\mathbf{71.9}_{1.9}$	$73.0_{2.1}$	$75.0_{3.3}$	$76.1_{3.7}$								
Gemma 2-Custom			$73.5_{0.7}$	$76.1_{2.3}$									
Llama 3.1-Random	64.9	$67.0_{3.9}$	$69.7_{3.7}$	$70.4_{2.9}$	$72.9_{3.1}$	$75.4_{2.8}$	$68.3_{6.4}$					$75.7_{1.9}$	
Llama 3.1-Custom			$63.3_{3.5}$	$67.6_{4.7}$			$72.1_{3.4}$					$67.2_{4.4}$	
Mistral v0.3-Random	**72.4**	$71.2_{3.0}$	$73.8_{2.1}$	$75.2_{1.3}$	$75.8_{1.1}$	$75.8_{1.7}$	$76.5_{2.1}$					$78.1_{3.4}$	
Mistral v0.3-Custom			$72.1_{1.8}$	$74.1_{1.8}$			$73.6_{1.2}$					$73.9_{4.3}$	

Table 10: The average micro-F1 (%) results for **MPQA Polarity** when $M = 10$ (i.e., selection size) in iterative approaches, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.

Model Name	0	5	10	20	30	40	50	60	70	80	90	100	Full
Majority Baseline	**37.2**												
Random Sampling													
BART-Random			$36.0_{2.4}$	$\mathbf{37.0}_{0.2}$			$37.1_{0.1}$					$35.2_{2.1}$	47.0
FLAN-T5-Random			$33.0_{3.8}$	$34.0_{3.6}$	$36.2_{0.9}$	$36.9_{0.7}$	$35.5_{1.6}$	$36.1_{1.7}$	$36.0_{1.4}$	$35.5_{1.5}$	$35.6_{1.4}$	$35.5_{1.5}$	**50.0**
Representative Sampling													
BART-Rep(En)			$\mathbf{37.0}_{0.0}$	$35.2_{2.1}$			$37.0_{0.4}$					$37.3_{0.3}$	
FLAN-T5-Rep(En)			$33.9_{2.2}$	$35.4_{0.9}$	$36.1_{0.8}$	$35.5_{1.6}$	$36.3_{1.1}$	$35.9_{2.0}$	$36.7_{1.4}$	$36.8_{1.3}$	$36.8_{1.0}$	$35.6_{1.1}$	
Iterative Approaches													
FLAN-T5-Rep(En)-Un			$33.9_{2.2}$	$36.8_{0.3}$	$\mathbf{36.8}_{1.2}$	$37.2_{1.8}$	$37.4_{0.7}$	$37.0_{2.4}$	$37.2_{2.6}$	$38.4_{2.7}$	$38.5_{1.9}$	$39.2_{2.1}$	
FLAN-T5-Rep(En)-Rep(Sc)			$33.9_{2.2}$	$35.3_{2.0}$	$35.5_{2.5}$	$36.9_{2.5}$	$37.4_{2.5}$	$37.5_{3.1}$	$37.6_{1.2}$	$37.5_{0.7}$	$38.4_{1.3}$	$38.0_{1.4}$	
FLAN-T5-Rep(En)-Rep(En)			$33.9_{2.2}$	$34.5_{2.3}$	$35.1_{3.1}$	$36.2_{1.5}$	$36.9_{2.2}$	$37.1_{0.7}$	$37.8_{1.6}$	$37.2_{1.3}$	$37.0_{0.7}$	$37.8_{1.2}$	
FLAN-T5-Rep(En)-UnRep			$33.9_{2.2}$	$36.2_{1.4}$	$36.7_{0.7}$	$36.8_{0.5}$	$37.4_{0.6}$	$37.3_{2.7}$	$37.2_{2.8}$	$36.5_{1.0}$	$38.4_{2.4}$	$38.8_{2.2}$	
FLAN-T5-Rep(En)-ClUn(Sc)			$33.9_{2.2}$	$36.3_{0.8}$	$35.9_{1.7}$	$36.2_{1.0}$	$36.4_{2.0}$	$37.2_{1.1}$	$37.5_{2.9}$	$37.6_{1.3}$	$36.8_{1.3}$	$38.2_{1.4}$	
FLAN-T5-Rep(En)-ClUn(En)			$33.9_{2.2}$	$35.1_{2.9}$	$36.0_{1.7}$	$37.0_{2.9}$	$\mathbf{38.0}_{1.6}$	$\mathbf{38.7}_{1.7}$	$\mathbf{38.3}_{2.0}$	$\mathbf{39.5}_{1.5}$	$\mathbf{38.7}_{1.9}$	$\mathbf{39.7}_{1.8}$	
In-Context Learning													
Gemma 2-Random	32.0	$\mathbf{32.3}_{4.1}$	$32.8_{2.6}$	$33.2_{2.6}$	$34.6_{1.9}$								
Gemma 2-Custom			$33.5_{4.9}$	$33.8_{4.8}$									
Llama 3.1-Random	23.9	$26.9_{1.0}$	$31.3_{2.2}$	$31.6_{2.5}$	$31.8_{2.2}$	$31.2_{0.7}$	$30.6_{1.7}$					$31.0_{1.9}$	
Llama 3.1-Custom			$32.4_{2.3}$	$33.0_{2.9}$			$31.4_{3.3}$					$31.4_{4.4}$	
Mistral v0.3-Random	22.9	$30.8_{4.8}$	$28.9_{2.4}$	$28.5_{2.4}$	$29.6_{2.1}$	$29.7_{0.6}$	$29.3_{0.6}$					$28.7_{2.1}$	
Mistral v0.3-Custom			$29.4_{7.2}$	$30.7_{7.7}$			$32.6_{5.3}$					$29.7_{5.0}$	

Table 11: The average micro-F1 (%) results for **MPQA Intensity** when $M = 10$ (i.e., selection size) in iterative approaches, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.

Model Name	0	5	10	20	30	40	50	60	70	80	90	100	Full
Majority Baseline	25.0												
Random Sampling													
BART-Random			$75.1_{7.3}$	$80.9_{4.2}$	$84.0_{3.0}$	$84.3_{2.5}$	$85.1_{1.0}$	$85.3_{1.5}$	$86.7_{1.6}$	$86.5_{1.0}$	$86.5_{0.9}$	$86.8_{1.2}$	94.2
FLAN-T5-Random			$71.8_{6.3}$	$87.3_{1.2}$	$87.8_{1.5}$	$88.3_{1.0}$	$88.4_{1.2}$	$88.7_{0.9}$	$88.9_{1.0}$	$89.4_{0.7}$	$89.4_{0.5}$	$89.3_{0.7}$	**94.4**
Representative Sampling													
BART-Rep(En)			$61.4_{0.3}$	$75.9_{11.9}$	$79.2_{7.3}$	$86.5_{0.8}$	$86.3_{0.4}$	$87.1_{0.5}$	$86.4_{1.8}$	$86.5_{0.4}$	$86.8_{1.0}$	$86.6_{1.5}$	
FLAN-T5-Rep(En)			$86.9_{0.7}$	$86.3_{1.4}$	$87.8_{0.6}$	$\mathbf{89.3}_{0.4}$	$88.4_{2.1}$	$\mathbf{89.6}_{0.3}$	$89.0_{0.4}$	$\mathbf{89.5}_{0.4}$	$89.4_{0.4}$	$89.1_{0.3}$	
Iterative Approaches													
FLAN-T5-Rep(En)-Un			$86.9_{0.7}$	$87.7_{1.0}$	$88.1_{0.9}$	$88.0_{0.9}$	$88.5_{0.6}$	$88.2_{0.9}$	$88.9_{0.4}$	$88.7_{1.1}$	$88.6_{1.0}$	$88.7_{1.4}$	
FLAN-T5-Rep(En)-Rep(Sc)			$86.9_{0.7}$	$87.5_{1.2}$	$\mathbf{88.5}_{0.8}$	$88.9_{0.7}$	$\mathbf{88.7}_{1.1}$	$89.0_{0.8}$	$89.1_{0.5}$	$89.2_{0.4}$	$\mathbf{89.8}_{0.5}$	$\mathbf{89.7}_{0.2}$	
FLAN-T5-Rep(En)-Rep(En)			$86.9_{0.7}$	$86.6_{2.4}$	$88.1_{1.1}$	$87.4_{2.1}$	$88.6_{0.4}$	$89.1_{0.5}$	$\mathbf{89.2}_{0.3}$	$89.1_{0.4}$	$89.3_{0.4}$	$89.3_{0.2}$	
FLAN-T5-Rep(En)-UnRep			$86.9_{0.7}$	$87.6_{0.3}$	$88.2_{1.0}$	$89.0_{1.1}$	$88.5_{1.2}$	$89.0_{1.0}$	$89.1_{1.4}$	$89.2_{0.5}$	$89.3_{0.8}$	$89.3_{0.5}$	
FLAN-T5-Rep(En)-ClUn(Sc)			$86.9_{0.7}$	$86.4_{1.8}$	$86.2_{1.8}$	$87.6_{1.3}$	$87.6_{1.0}$	$87.3_{1.6}$	$88.4_{1.1}$	$88.3_{1.1}$	$88.5_{0.7}$	$88.8_{0.7}$	
FLAN-T5-Rep(En)-ClUn(En)			$86.9_{0.7}$	$87.5_{0.7}$	$86.9_{2.1}$	$87.3_{1.8}$	$88.1_{2.0}$	$88.4_{1.3}$	$88.8_{0.9}$	$88.8_{1.3}$	$88.9_{0.7}$	$89.1_{0.8}$	
In-Context Learning													
Gemma 2-Random	84.6	$\mathbf{85.7}_{1.3}$	$85.2_{1.6}$	$86.8_{1.4}$	$87.7_{0.9}$								
Gemma 2-Custom			$87.2_{0.5}$	$88.1_{0.6}$									
Llama 3.1-Random	82.5	$84.4_{1.5}$	$85.8_{1.4}$	$85.1_{1.4}$	$86.0_{0.6}$	$86.1_{0.8}$	$86.3_{1.3}$					$86.5_{1.3}$	
Llama 3.1-Custom			$84.8_{1.0}$	$85.7_{0.7}$			$86.0_{0.9}$					$85.3_{1.4}$	
Mistral v0.3-Random	84.9	$82.3_{3.9}$	$82.9_{2.6}$	$85.3_{1.4}$	$85.5_{1.7}$	$84.9_{2.3}$	$86.1_{0.8}$					$86.4_{1.4}$	
Mistral v0.3-Custom			$80.4_{2.8}$	$86.2_{0.7}$			$83.2_{2.9}$					$81.9_{3.6}$	

Table 12: The average micro-F1 (%) results for **AG News** when $M = 10$ (i.e., selection size) in iterative approaches, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.

Model Name	0	5	10	20	30	40	50	60	70	80	90	100	Full
Majority Baseline	20.0												
Random Sampling													
BART-Random			$32.1_{3.0}$	$35.7_{2.7}$	$37.1_{2.7}$	$37.8_{2.7}$	$41.1_{1.8}$	$42.3_{3.0}$	$44.0_{2.2}$	$46.0_{2.3}$	$47.0_{1.8}$	$45.3_{2.7}$	63.2
FLAN-T5-Random			$47.2_{4.8}$	$52.7_{4.3}$	$54.3_{3.5}$	$55.6_{2.1}$	$55.9_{0.7}$	$56.8_{0.8}$	$57.4_{1.0}$	$58.0_{0.8}$	$58.5_{0.9}$	$58.8_{1.5}$	**65.7**
Representative Sampling													
BART-Rep(En)			$29.9_{0.0}$	$34.9_{1.7}$	$35.3_{1.5}$	$35.5_{2.6}$	$38.7_{2.6}$	$42.1_{2.5}$	$43.8_{2.4}$	$43.0_{3.3}$	$43.8_{2.1}$	$45.9_{2.7}$	
FLAN-T5-Rep(En)			$51.5_{0.2}$	$51.5_{1.8}$	$50.9_{1.9}$	$51.6_{2.4}$	$55.0_{2.9}$	$55.7_{2.7}$	$57.3_{0.7}$	$57.6_{1.2}$	$57.9_{1.3}$	$59.3_{0.9}$	
Iterative Approaches													
FLAN-T5-Rep(En)-Un			$51.5_{0.2}$	$54.7_{3.1}$	$53.9_{3.3}$	$56.2_{3.7}$	$57.0_{1.7}$	$56.1_{3.6}$	$57.6_{1.7}$	$57.5_{2.1}$	$57.4_{1.9}$	$57.5_{2.8}$	
FLAN-T5-Rep(En)-Rep(Sc)			$51.5_{0.2}$	$52.8_{2.2}$	$55.7_{1.4}$	$56.6_{1.1}$	$57.4_{1.1}$	$58.0_{0.8}$	$58.7_{0.6}$	$59.4_{1.6}$	$\mathbf{59.7}_{0.6}$	$59.5_{1.4}$	
FLAN-T5-Rep(En)-Rep(En)			$51.5_{0.2}$	$52.6_{2.9}$	$55.4_{1.0}$	$56.7_{0.9}$	$57.3_{1.0}$	$57.2_{1.1}$	$57.3_{0.9}$	$57.7_{0.6}$	$57.7_{0.6}$	$58.7_{0.9}$	
FLAN-T5-Rep(En)-UnRep			$51.5_{0.2}$	$45.7_{4.2}$	$47.8_{4.9}$	$50.0_{4.6}$	$52.4_{3.2}$	$52.8_{4.4}$	$52.3_{4.2}$	$53.0_{3.1}$	$54.3_{3.2}$	$55.0_{3.1}$	
FLAN-T5-Rep(En)-ClUn(Sc)			$51.5_{0.2}$	$54.7_{1.8}$	$55.5_{1.7}$	$\mathbf{58.1}_{1.4}$	$\mathbf{58.4}_{0.9}$	$58.4_{1.6}$	$58.9_{1.2}$	$58.4_{2.0}$	$59.5_{0.8}$	$59.4_{0.4}$	
FLAN-T5-Rep(En)-ClUn(En)			$51.5_{0.2}$	$52.0_{3.5}$	$55.3_{2.3}$	$57.1_{0.7}$	$58.3_{1.6}$	$58.0_{0.7}$	$\mathbf{59.2}_{0.8}$	$\mathbf{59.9}_{0.6}$	$59.1_{1.3}$	$\mathbf{59.9}_{1.1}$	
In-Context Learning													
Gemma 2-Random	$\mathbf{62.2}$	$60.3_{1.5}$	$60.6_{1.6}$	$60.1_{1.9}$	$\mathbf{60.5}_{2.6}$								
Gemma 2-Custom			$61.9_{0.6}$	$60.5_{0.6}$									
Llama 3.1-Random	59.4	$52.2_{3.7}$	$53.3_{5.4}$	$54.7_{4.8}$	$55.5_{4.5}$	$55.3_{4.4}$	$57.0_{2.7}$					$55.6_{3.1}$	
Llama 3.1-Custom			$50.4_{2.4}$	$52.2_{2.6}$			$56.1_{2.4}$					$54.5_{1.2}$	
Mistral v0.3-Random	54.3	$59.8_{1.1}$	$59.4_{1.2}$	$58.5_{2.2}$	$54.0_{4.9}$	$49.6_{5.0}$	$49.7_{7.9}$					$46.1_{9.0}$	
Mistral v0.3-Custom			$60.0_{0.9}$	$56.9_{2.2}$			$45.6_{6.6}$					$47.7_{2.8}$	

Table 13: The average micro-F1 (%) results for **Amazon Reviews** when $M = 10$ (i.e., selection size) in iterative approaches, calculated over five different seeds for the sampling phase. The sub-columns denote K (i.e., total support set size), and the subscripts indicate the standard deviation.