Cache & Distil: Optimising API Calls to Large Language Models

Anonymous ACL submission

Abstract

Large-scale deployment of generative AI tools often depends on costly API calls to a Large Language Model (LLM) to fulfil user queries. To curtail the frequency of these calls, one can employ a smaller language model – a *student* – which is continuously trained on the responses of the LLM. This student gradually gains proficiency in independently handling an increasing number of user requests, a process we term *neural caching*. The crucial element in neural caching is a policy that decides which requests should be processed by the student alone and which should be redirected to the LLM, subsequently aiding the student's learning. In this study, we focus on classification tasks, and we consider a range of classic Active Learning-based selection criteria as the policy. Our experiments suggest that Margin Sampling and Query by Committee bring consistent benefits over other policies and baselines across tasks and budgets.

1 Introduction

Large Language Models (LLMs) offer unique capabilities in understanding and generating human-like text. They have gained widespread use in a wide range of applications, such as assistive tools and entertainment bots. However, large models are often very challenging for all but a few companies and institutions to run on their infrastructure (Schwartz et al., 2020). Meanwhile, smaller models typically under-perform in these applications, at least without additional fine-tuning on task-specific labelled data. Consequently, many applications access LLMs via commercial APIs despite the costs involved and the exposure of their entire request stream to the API providers.

To minimise the costs and data exposure associated with calling the API, we propose to train a smaller language model, which we refer to as *student*, on the LLM's predictions and, as the student gets more accurate, it handles an increasing number of requests. The knowledge of the LLM gets continuously distilled into the smaller model. We refer to this scenario as *neural caching* (see Figure 1), as the student can be thought of as a smart cache. Note though that the student not only remembers what the LLM predicted but also generalises beyond these examples. The goal of this paper is to formalise the neural caching problem and investigate simple ways of approaching it.

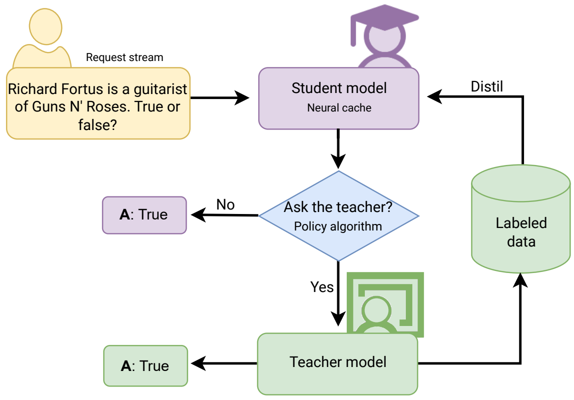

Figure 1: Neural caching (one iteration): A student generates a response to a user request. The policy algorithm determines whether to rely on the student's response or to call an LLM. LLM responses are stored and used to re-train the student as more data becomes available.

The key element in the neural caching scenario is the policy determining which requests the student processes independently. A good policy should weigh the expected immediate user benefit (i.e., if the LLM is substantially more likely to make a correct prediction than the student) and the anticipated benefit for the student (i.e., whether the LLM's prediction will aid in training the student). The latter underscores its relationship with Active Learning (AL, Settles, 2009; Zhan et al., 2022), although AL is typically associated with soliciting human annotations. In particular, there is a similarity to online AL (Cacciarelli and Kulahci, 2023), where new unlabelled data points arrive in a stream and are discarded immediately or sent to an annotator. However, online AL tends to focus on maximising the accuracy of the final model (i.e. student in our terminology). In contrast, what matters in neural caching is the accuracy of the joint system (student, teacher, along with the policy) over its lifetime since this *online accuracy* reflects the average level of service offered to a user.

Despite the aforementioned differences with AL, evaluating the existing AL algorithms – specifically the example selection criteria – remains valuable given the maturity of the AL field and the ease of implementation of some of the AL methods. This study aims to achieve this, as well as to investigate the potential shortcomings of these methods. For instance, will the AL methods end up selecting examples that are too challenging even for the LLM? Would learning from these noisy examples be detrimental to the student? Answering these questions can inform future research on this practically significant scenario.

In this work, our focus is specifically on classification tasks, as opposed to free text generation. Many practical problems, such as routing user requests to relevant departments or answering questions about factual knowledge, can be framed as classification tasks. By confining our focus to classification, we can apply methods developed in AL without modification. This also allows us to circumvent additional challenges tied to the automatic evaluation of text generation (Celikyilmaz et al., 2020).

Our findings reveal the benefits of using AL-based policies such as Margin Sampling (Scheffer et al., 2001) and Query by Committee (Seung et al., 1992). Across datasets and budgets, these methods consistently outperform baselines, such as routing examples randomly or training the student at the very start. Our analysis also reveals that the student appears robust to the noise introduced by an LLM. We also analyse a simplified practical scenario where the student is not retrained and observe even greater improvements in online accuracy from using AL-based policies. We release our code to encourage further work on this problem.[1]

The key contributions of this work are:

- We formulate the *neural caching* problem as a powerful extension of using static caches. In neural caching, LLM calls are optimised, while the student model is periodically retrained on the labels. We believe online Knowledge Distillation could play a key role in saving calls to expensive models.
- We release a benchmark with LLM annotations for classification tasks to facilitate future research in this setup.
- We evaluate and analyse different instance selection criteria for the neural caching setup.
- Our findings reveal that AL-based selection criteria consistently improve performance over baseline methods across various budgets and datasets.

[1]https://anonymous.4open.science/r/neural-caching-780F/README.md

2 Related Work

Active Learning. Active Learning (AL) seeks to reduce the amount of manual data annotation needed. To accomplish this, it selects the most informative examples from unannotated data. These datapoints are then presented to an annotator and the labels are subsequently used to train a model. The most common scenario for AL is pool-based, where a large unlabelled dataset is available from the start and then a subset of examples is selected for labelling. There has been extensive work on applying pool-based techniques to NLP tasks, especially for classification problems (Settles, 2009; Zhan et al., 2022; Zhang et al., 2022).

Online Active Learning. In single-pass online AL (Cacciarelli and Kulahci, 2023), access to a large unlabelled dataset is not available. Instead, we are given one unlabelled instance at a time and need to decide at that time whether to request annotation. Online AL was initially motivated by scenarios in which an instance would not be available for annotation at a later time, such as in defect detection or medical applications, where an item might get shipped or the patient becomes unavailable (Riquelme, 2017). Online AL tends to focus on the final accuracy of the model, rather than the online accuracy of the student and teacher combined, the measure more suitable for our scenario.

Knowledge Distillation of LLMs. Knowledge distillation (KD), i.e., training a smaller model to mimic a larger one, has garnered substantial attention (Bucila et al., 2006; Hinton et al., 2015). The class of methods most closely related to ours is active KD, which effectively applies AL to KD (Liang et al., 2021; Xu et al., 2023; Baykal et al., 2023). Similar to AL, the emphasis is placed on the pool-based setting, as opposed to the online setting, with a particular focus on optimising the final accuracy of the student model, rather than online accuracy as needed for our use case.

Optimisation of Commercial LLM API Calls. Due to the high cost of commercial LLM APIs, several works have explored methods to reduce or otherwise optimise the cost of API calls. GPTCache (Bang, 2023) relies on a vector store of past query embeddings and retrieves their associated labels. It shares similarities with the Coreset version of our approach – which emerged as the weakest method in our experiments. FrugalGPT (Chen et al., 2023) implements a cascade of commercial LLMs, where bigger models are only called if the response from a cheaper model is deemed as too unreliable by a scorer that was trained with in-domain data. In contrast, in this work, we do not assume access to gold data to train a scorer. Zhu et al. (2023) present a method to allocate queries among multiple models, together with traditional caching, in a scenario with highly repetitive queries. Šakota et al. (2023); Shnitzer et al. (2023) optimise routing calls through models by predicting their respective performance. Our work deviates from all these as we propose to use continuous KD in a student model.

Concurrent work of Stogiannidis et al. (2023)[2] also presents a calling strategy that leverages KD to reduce API calls to an LLM. Unlike our paper, which offers a systematic analysis of existing AL criteria, their work concentrates on a specific model design. This design resembles a hybrid of our Coreset and Prediction Entropy, which do not perform well in our experiments. Even more significantly, their method's advantages are only shown in comparison to a scenario where no student model is used. This overlooks trivial baselines of front-loading and random allocation, both of which have shown hard to beat in our experiments. They also use very simple student models (kNN or a non-pretrained MLP versus our smaller pretrained LM), whose relative success may be attributed primarily to the simplicity of the two datasets in their study, which do not necessitate generalisation beyond the teacher model's predictions.

[2]Made public on the same day as ours.

3 The Neural Caching Problem

The objective of neural caching is to optimise the usage of an LLM in a scenario where labels need to be generated for a stream of inputs. As we get more predictions from the LLM, a student model is trained on them. Our goal is to achieve the highest level of service possible within a set budget of LLM calls; hence, calling the LLM serves both to attain high accuracy for the incoming input as well as to train a student model.

To put it formally, our goal is to establish a mapping between elements in the input space $\mathcal{X}$ and the corresponding labels in the space $\mathcal{Y}$. We start with a student model $\mathcal{S}_0$, and we can access a teacher model $\mathcal{T}$ on demand. Our task is to predict labels for a sequence of n examples $(x_1, \ldots, x_n) \overset{\text{iid}}{\sim} \mathcal{X}$.

We retrain the student model on the labels obtained from the LLM every f processed requests. This simulates the situation where the number of requests is uniform in time, and there is a set time to retrain the model, e.g. at night. For simplicity and to follow the convention in AL to retrain the model from scratch (Ren et al., 2022), every time we retrain the student model, we reset it to the original pre-trained model and then use parameter-efficient fine-tuning. Although continual learning methods could be employed (Biesialska et al., 2020; Zhou and Cao, 2021), we believe this is largely orthogonal to our primary focus on policies and resetting enhances the reproducibility of our analysis. Importantly, we do not assume access to ground truth (or human annotation) at any point in learning to simulate a fully automatic scenario.

For every new input x_i, we use the student model $\mathcal{S}_{i/f}$ to obtain the predicted label $\hat{y}_i^{\mathcal{S}}$. Then, we have the option to request the label $\hat{y}_i^{\mathcal{T}}$ from the teacher model (LLM), which incurs a cost of $c(x_i)$. Finally, we return the label $\hat{y}_i$ for x_i: the teacher's label if requested or the student's otherwise.

The processing of the n examples is subject to a budget constraint, where the total cost must not exceed a fixed budget b. We assess the effectiveness of our querying strategy based on the accuracy of our predicted label $\hat{y}_i$ compared to the actual label y_i (*online accuracy*) on the online examples. Additionally, we measure the accuracy of the final student model $\mathcal{S}_{n/f}$ on a test dataset (*final accuracy*). Algorithm 1 describes the process.

Algorithm 1: Pseudo-code for the neural caching algorithm with budget b, retraining frequency f, cost per query c, data from the LLM $\mathcal{D}_{\text{LLM}}$ and an initial student $\mathcal{S}_0$

$\mathcal{D}_{\text{online}} = \emptyset$
for x_i **in** X_{online} **do**
  if $i \bmod f$ == *0* **then**
    $\mathcal{S}_{i/f} = \text{Train}(\mathcal{D}_{\text{LLM}})$
  end
  $\hat{y}_i = \mathcal{S}_{i/f}(x_i)$
  if Call_LLM(b, x_i, $\hat{y}_i$) **and** $b \geq c(x_i)$ **then**
    $\hat{y}_i = \text{LLM}(x_i)$
    $b = b - c(x_i)$
    $\mathcal{D}_{\text{LLM}} = \mathcal{D}_{\text{LLM}} \cup \{\langle x_i, \hat{y}_i \rangle\}$
  end
  $\mathcal{D}_{\text{online}} = \mathcal{D}_{\text{online}} \cup \{\langle x_i, \hat{y}_i \rangle\}$
end
$\mathcal{D}_{\text{test}} = \{\langle x_j, \mathcal{S}_{i/f}(x_j) \rangle \mid x_j \in X_{\text{test}}\}$
$\text{Acc}_{\text{online}} = \text{Evaluate}(\mathcal{D}_{\text{online}})$
$\text{Acc}_{\text{final}} = \text{Evaluate}(\mathcal{D}_{\text{test}})$

3.1 Instance Selection Criteria

We use classical instance selection criteria from AL for the neural caching problem. We use the term *selecting an instance* to denote using the LLM to annotate that example.

Front-loading (FR) This simple approach involves using the entire budget initially by selecting all instances for LLM annotation. Once the budget is used up, subsequent requests are handled by the student model alone. As the examples are i.i.d. in our experiments, this strategy has the same expected *final accuracy* as random selection.

Margin Sampling (MS) MS (Scheffer et al., 2001; Luo et al., 2004) selects examples with high margin between the top two predictions made by the student model

$$\text{Margin}(x_i) = \log P(y_i = k_1^* \mid x_i) - \log P(y_i = k_2^* \mid x_i) \quad (1)$$

where k_1^* and k_2^* are the first and second most likely labels, respectively, according to the distribution $P(y_i \mid x_i)$ computed by the student model. This is a popular selection criterion for AL (Roth and Small, 2006; Balcan et al., 2007). Schröder et al. (2022) evaluated different uncertainty-based strategies with Transformer models (Devlin et al., 2019) and found MS to be the best-performing one in an offline, pool-based setting. To adapt MS – as well as the other criteria – to an online setting as a selection policy, we define a threshold, and only examples with a margin above this threshold are selected until the budget is exhausted. We refer to Appendix A.1 for more details.

Prediction Entropy (PE) In PE (Schohn and Cohn, 2000; Roy and McCallum, 2001), we select instances with high entropy of the output distribution:

$$\text{Entropy}(x_i) = -\sum_j P(y_i = k_j^* \mid x_i) \log P(y_i = k_j^* \mid x_i) \quad (2)$$

Query by Committee (QBC) In QBC (Seung et al., 1992; Burbidge et al., 2007), we select instances relying on the disagreement among a committee of models. Our committee is the set of $d = 4$ previous student models plus the current – presumably best – student. The disagreement is quantified by computing the proportion of committee members contradicting the current student.

Coreset (CS) CS (Sener and Savarese, 2018) uses an encoder to obtain the embedding representation of the new instance. Then, it calculates the cosine similarity between the embedding of the new input and the embeddings of past examples. If the similarity with respect to the most similar past instance x_i annotated by the LLM is below a certain threshold s, then it requests further annotation from the LLM. To obtain the embeddings, we average the encoder representation across tokens, as this has been proven effective in sentence embedding benchmarks (Ni et al., 2022). Similarity with previous examples has been employed in AL to encourage diversity and coverage (Kim et al., 2006; Zeng et al., 2019). GPTCache (Bang, 2023) also uses the embedding representations to decide whether an incoming instance should be labelled.

4 Experimental Setup

4.1 Datasets

We study the proposed setup on four classification tasks. The first two tasks have been commonly studied in AL for NLP: ISEAR (Shao et al., 2015) and RT-Polarity (Pang and Lee, 2005). The remaining two tasks showcase harder problems where factual knowledge acquired during pre-training of an LLM could be highly beneficial: the fact-checking dataset FEVER (Thorne et al., 2018)

	ISEAR	RT-Polarity	FEVER	Openbook
Accuracy, T5+LoRA (100 gold labels)	0.51	0.85	0.53	0.23
Accuracy, T5+LoRA (5000 gold labels)	0.67	0.90	0.74	0.68
Accuracy, LLM	0.68	0.91	0.78	0.80
Average margin (LLM labels)	10.0	15.4	9.2	10.3
Average margin when wrong (LLM labels)	4.2	10.3	6.9	5.3

Table 1: The accuracy of the LLM is similar to training the simple model with 5000 gold labels.

and the question-answering dataset Openbook (Mihaylov et al., 2018). We split all datasets into online and test portions (80%-20%, except for Openbook, as it has fewer samples). The datasets are balanced.

ISEAR (Shao et al., 2015) annotates personal reports for emotion (classes: *joy*, *fear*, *shame*, *sadness*, *guilt*, *disgust*, *anger*; 7666 examples).

RT-Polarity (Pang and Lee, 2005) provides sentiment polarity labels for movie reviews (classes: *positive*, *negative*; 10662 examples).

FEVER (Thorne et al., 2018) is a fact-checking dataset (classes: *true*, *false*; 6612 examples) with claims that can be checked with 1-3 sentences from Wikipedia.

Openbook (Mihaylov et al., 2018) is a challenging question-answering dataset modelled after open book exams for assessing human understanding of a subject. Each instance consists of a multiple choice question (classes: *A*, *B*, *C*, *D*) and includes one fact that can help answer it. The full dataset consists of 5957 data points; we selected 5457 for the online set and 500 for testing.

4.2 Annotation by LLM

While we are interested in the online caching scenario, to facilitate comparisons between our methods and ensure replicability in future work, we create a dataset in which we obtain LLM predictions for all data points; this dataset is then used to simulate the online setup.

We generate soft labels using OpenAI's `text-davinci-003`, an InstructGPT-based model (Zhan et al., 2022). For each task, we design a prompt that describes the task and the possible classes. Our prompts do not contain any in-context examples (zero-shot), but we use a small part of the dataset (up to 10 examples) for prompt engineering.

On all datasets, we observe that the LLM achieves better accuracy than the smaller model trained on 5000 gold labels, suggesting that KD would be useful in these datasets (Table 1). In our benchmark, we store the log-probabilities of the labels. We note that the average margin for the generated labels is substantially lower when the predicted label is wrong; we observe with additional experiments that the LLM annotations are well calibrated (Figure 4). We release our benchmark with the generated labels to encourage further work on the neural caching problem.

4.3 Experiment Details

We run all our experiments with three random seeds, which also determine the ordering of examples; we present the average scores. For simplicity, we use a retraining frequency $f = 1000$ and a constant cost per query $c(x_i) = 1$. To avoid a cold-start, we train the initial student model $\mathcal{S}_0$ with $N = 100$ (ISEAR, RT-Polarity) or $N = 1000$ (FEVER, Openbook) data points from the LLM; we choose N so that $\mathcal{S}_0$ is better than random choice. For the student model, we use $T5_{base}$ (Raffel et al., 2020) as the backbone model; we freeze the model weights and add LoRA adapter layers for a parameter-efficient fine-tuning (Hu et al., 2022).

We fine-tune the student model with the cross-entropy loss using the log-probabilities assigned by the teacher to each class. Using hard labels seems to work almost as well (Table 9). We split the accumulated data from the LLM into training and validation sets, and train each student from scratch for 30 epochs with early stopping with patience of five epochs. The rest of the hyperparameters can be found in Appendix A.

5 Experiments

We first present our results and then their analysis. To report accuracy across budgets, we use the corresponding Area Under the Curve (AUC) divided

	ISEAR	RT-Polarity	FEVER	Openbook	Average
Random	0.640	0.886	0.704	0.662	0.723
Margin Sampling	**0.666**	**0.896**	**0.725**	**0.703**	**0.748**
Query by Committee	0.656	0.889	**0.725**	0.687	0.739

Table 2: Online accuracy (AUC) for neural caching with no student retraining.

by the budget range, thus obtaining an average accuracy.

5.1 Neural Caching without Student Retraining

We first study a simplified version of neural caching, where the student model is not retrained on new data points. This is a practical scenario, as retraining creates extra overhead for the application provider (e.g., consider a setting where the student is run on a portable device, which is not powerful enough to support retraining).

We adapt the AL instance selection criteria in the following way. Given a criterion C, we calculate the respective values from the previous outputs of the student and call this list the history $\hat{C}$. If we have a remaining budget b and n remaining online instances, we use as a threshold for an incoming instance the $\frac{b}{n}$-th percentile of the history $\hat{C}$. The best possible scenario would imply having oracle threshold values for each budget (i.e. as if we had access to the full dataset offline). However, in additional experiments, we found that the above rule yields very similar scores.

To use QBC in this setup, we simulate that we have four previous students trained on subsets of the data. For example, if the student is trained on $N = 1000$ examples, the previous students are trained on 900, 800, 700, and 600 data points, respectively. We find that MS yields results very similar to PE and that Coreset is similar to Random. To ease visualising the results, here we omit PE and Coreset.

Table 2 and Figure 2 contain the results when we train the initial student with $N = 1000$ datapoints annotated by the LLM. Our experiments with different initial budgets N yield similar results (Table 8 in Appendix), and Coreset performs poorly even with different encoders.

We find that MS and PE are the best-performing methods on all datasets and across all the initial student models, followed by QBC, which outperforms the baseline of random selection. Given the simplicity of these methods, these results make a strong case for using AL-based selection methods, especially MS. Unlike QBC, MS does not require storing multiple models and performing inference with each of them.

5.2 Neural Caching with Student Retraining

We now turn to the complete setup proposed in Section 3, in which the selected instances are used to retrain a student model with some periodicity. This creates the incentive to spend the budget early to get a more proficient student model as soon as possible. To observe this effect, we include a random baseline with a uniform sampling rate. This suggests waiting longer for informative examples to arrive counterweights the benefits of getting a strong student as quickly as possible. We select thresholds to encourage spending more of the budget early on (see Appendix A.1).

We show the results averaged across all datasets in Figure 3 and per-dataset in Figure 5. We observe that both MS and QBC substantially outperform the other methods. Coreset (embedding-based) does badly in all the studied setups and encoders (Table 11). Table 3 summarises the results.

5.3 Analysis

Hard examples with noisy labels. We have observed in our experiments that prioritising harder instances for teacher annotation leads to clear gains in online accuracy. However, as discussed in the introduction, LLM accuracy may be significantly affected by the increased 'complexity' of an example, which can inflate the proportion of noisy annotations in the data on which the student is trained (see Figure 4). This problem is known in KD as *confirmation bias* (Arazo et al., 2020; Liu and Tan, 2021). Previous results from offline KD suggest that this type of confirmation bias can be mitigated by avoiding the hardest instances (Baykal et al., 2023), improving the chances that the teacher model makes a correct prediction. However, we observe that the most significant advantage of the LLM with respect to the student in terms of accuracy lies in these samples that are deemed hard by the student (leftmost part of the plot in Figure 4); since we are optimising the online accuracy, the trade-off between providing hard or correct labels may be different in our online case than in the offline scenario. Given the above, we hypothesise that MS and QBC would be more negatively affected by the confirmation bias than front-loading, which does not prioritise hard examples. To test

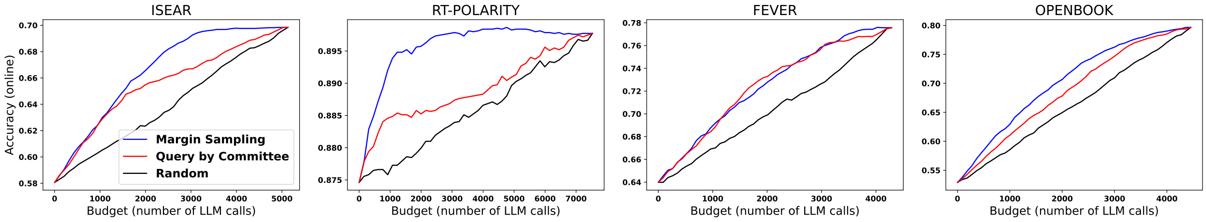

Figure 2: Accuracy curve with respect to budgets for neural caching without student retraining.

	ISEAR	RT-Polarity	FEVER	Openbook	Average
Random	0.614	0.872	0.723	0.703	0.728
Front-loading	0.637	0.879	0.734	0.731	0.745
Coreset	0.637	0.878	0.715	0.726	0.739
Entropy	0.657	0.886	0.728	0.693	0.741
Margin Sampling	**0.658**	**0.889**	**0.753**	0.726	**0.757**
Query by Committee	0.650	0.887	0.748	**0.737**	0.755

Table 3: Online accuracy (AUC) for neural caching with student retraining.

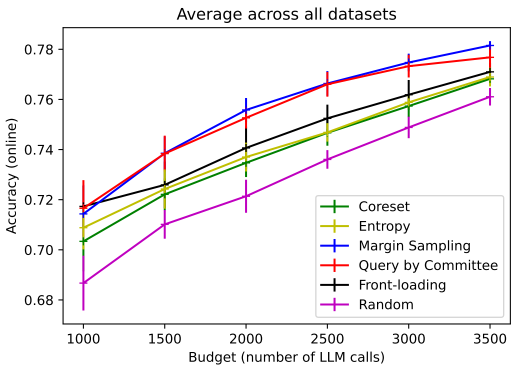

Figure 3: Accuracy curve with respect to budgets, in the neural caching problem with student retraining. Error lines indicate variance.

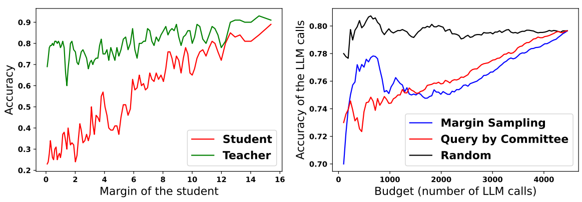

Figure 4: On the left, we order data points by their margin and plot the accuracy of their respective labels generated by the student and teacher. We observe that the greatest advantage of using the labels from the teacher comes from examples with small margins. On the right, the accuracy of the labels generated by the LLM calls in neural caching with no student retraining. We observe that MS and QBC are more likely to generate wrong labels. We focus on Openbook for both plots.

this hypothesis, we designed an experiment to put an upper bound on the effect of wrong LLM annotations. For each strategy, we only retrain the student model on correct labels, simulating an oracle that discards incorrect examples. Table 5 shows the absolute improvements in the online and final accuracy with respect to the values obtained without the oracle (Table 3 and 4). We observe moderate absolute improvements, but surprisingly MS and QBC do not seem to improve more than front-loading, suggesting that the hypothesis is wrong and that the impact of confirmation bias is somewhat limited and - what is surprising - similar across strategies.

As an additional test, we analyse the subset of test examples where the teacher is incorrect. If confirmation bias is a major issue for MS and QBC than for front-loading, we would expect that they are more prone to reproducing the teacher's errors. Again, we do not find any substantial differences between these two strategies vs front-loading (Table 10).

Online accuracy vs. final accuracy. Taking a look at the accuracy of the final student (Table 4), we observe that it is generally consistent with the online accuracy (Table 3). However, MS has a low *final* accuracy on FEVER while having the best *online* accuracy on that dataset, confirming that in some tasks, calling the LLM to obtain labels for hard examples may improve the online accuracy while not necessarily improving the student. This result emphasises the differences between our set-

	ISEAR	RT-Polarity	FEVER	Openbook	Average
Front-loading	0.598	0.879	0.686	**0.647**	0.702
Coreset	0.599	0.879	0.680	0.641	0.700
Entropy	0.608	**0.885**	0.682	0.647	0.705
Margin Sampling	**0.609**	0.884	0.678	0.634	0.701
Query by Committee	**0.609**	0.882	**0.687**	0.646	**0.706**

Table 4: Final accuracy (AUC) of the last student model for neural caching with student retraining.

	ΔOnline	ΔFinal
Front-loading	0.009	0.019
Margin Sampling	0.008	0.022
Query by Committee	0.008	0.018

Table 5: Absolute improvements for the online and final accuracy using an oracle that allows us to discard instances with wrong labels from the LLM, averaged across datasets. The improvements are with respect to values from Table 3 and 4.

	Openbook			**FEVER**		
	N=1000	N=2000	N=3000	N=500	N=1000	N=1500
Front-loading	0.731	0.769	0.751	0.716	0.734	0.734
Margin Sampling	0.726	0.777	0.764	0.718	**0.753**	0.751
Query by Committee	**0.737**	**0.786**	**0.779**	**0.722**	0.748	**0.755**

Table 6: Online accuracy (AUC) of different selection criteria with different initial student models $\mathcal{S}_0$.

up and the setting normally studied in AL.

5.4 Robustness of the Findings

Vary initial training ($\mathcal{S}_0$). We study the effect of the quantity of LLM-annotated data on which the first student model is trained, focusing on the setup with retraining (Table 6). We consider the two more challenging tasks, FEVER and Openbook. We find that QBC performs best overall, and the performance of MS is more sensitive to the initial budget. This observation suggests that better decision criteria for transitioning from a front-loading regime to MS can be beneficial; we leave this for future exploration.

Higher retraining frequency f. We repeat neural caching experiments, setting this time a higher frequency of retraining $f = 100$; this results in much longer runs as the student model has been retrained an order of magnitude more times. Table 7 shows the results. We observe that results are consistent and very similar to those with a lower frequency of retraining (Table 3).

	ISEAR	RT-Polarity	FEVER	Openbook	Average
Front-loading	0.637	0.879	0.734	0.731	0.745
Margin Sampling	**0.661**	**0.892**	0.750	0.728	0.758
Query by Committee	0.657	0.890	**0.751**	**0.740**	**0.759**

Table 7: Online accuracy (AUC) for neural caching with retraining frequency $f = 100$.

6 Conclusions

In this work, we have studied how instance selection criteria from AL behave when they are used to decide in real time whether we should perform an LLM call or use a student model that has been trained on previous LLM predictions. In the scenario where we are not retraining the student model, Margin Sampling performs the best, across different datasets. In the scenario where we retrain the student model with some time periodicity, Query by Committee is the most robust option. In our experiments we observe that, while Margin Sampling outperforms the front-loading baseline on harder tasks, it is more sensitive to the initial budget spent to train the student model $\mathcal{S}_0$.

We find that the embedding-based strategy (Coreset) consistently performs poorly across different encoders; it is the only LLM caching approach which is known to be adopted by practitioners (e.g., GPTCache (Bang, 2023)). We believe these types of strategies could be useful in certain contexts, e.g. multiple near-identical calls to an LLM, a scenario which has not been the focus of this work.[3]

Our results suggest that (i) there is room for smart LLM query allocation in the context of continuously distilling an LLM into a student model and (ii) previous literature in Active Learning can transfer well to this setup. We believe that online Knowledge Distillation could play a key role in caching LLMs and saving unnecessary calls to expensive models.

[3] FEVER does contain paraphrases or statements entailing each other but these constitute only a small fraction of the dataset.

Limitations

Our experiments assume that there is no development set available for each task, which could help improve the results of non-baseline methods by introducing a task-specific hyperparameter for thresholds.

Our experiments refer to a particular configuration with one student model and one LLM. We leave for future work configurations with other models. We also leave for future work experiments on text generation, which would require using text-based AL criteria.

In this work, we focused on a stationary (i.i.d.) stream of requests. In practice, the distribution of requests is likely to change over time (Cacciarelli and Kulahci, 2023). As suggested by the online AL literature (Bifet and Gavaldà, 2007), this should further increase the gap between the AL-based approaches and static strategies, e.g., front-loading. In those cases, we would expect improvements in both online and final accuracy.

Ethics statement We anticipate that our proposed approach will be advantageous for smaller companies and will enhance user privacy by limiting the amount of data shared with API providers. However, we recognise the potential for misuse of this technology. For instance, it might contravene the service policies of API providers and potentially could decrease the revenue of creators if they are compensated for the usage of their content by the API provider. Furthermore, while the original model provided through the API may have been fine-tuned to diminish harmful biases and tailored for fairness across diverse user groups, there is a possibility that the student model derived through our process may not inherit these qualities.

References

Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. 2020. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE.

Maria-Florina Balcan, Andrei Broder, and Tong Zhang. 2007. Margin Based Active Learning. In *Learning Theory*, Lecture Notes in Computer Science, pages 35–50, Berlin, Heidelberg. Springer.

Fu Bang. 2023. GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings. In *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, pages 212–218, Singapore, Singapore. Empirical Methods in Natural Language Processing.

Anson Bastos and Manohar Kaul. 2021. ALLWAS: active learning on language models in wasserstein space. *CoRR*, abs/2109.01691.

Cenk Baykal, Khoa Trinh, Fotis Iliopoulos, Gaurav Menghani, and Erik Vee. 2023. Robust active distillation. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Magdalena Biesialska, Katarzyna Biesialska, and Marta R. Costa-jussà. 2020. Continual lifelong learning in natural language processing: A survey. In *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, pages 6523–6541. International Committee on Computational Linguistics.

Albert Bifet and Ricard Gavaldà. 2007. Learning from time-changing data with adaptive windowing. In *Proceedings of the Seventh SIAM International Conference on Data Mining, April 26-28, 2007, Minneapolis, Minnesota, USA*, pages 443–448. SIAM.

Cristian Bucila, Rich Caruana, and Alexandru Niculescu-Mizil. 2006. Model compression. In *Proceedings of the Twelfth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Philadelphia, PA, USA, August 20-23, 2006*, pages 535–541. ACM.

Robert Burbidge, Jem J. Rowland, and Ross D. King. 2007. Active Learning for Regression Based on Query by Committee. In *Intelligent Data Engineering and Automated Learning - IDEAL 2007*, Lecture Notes in Computer Science, pages 209–218, Berlin, Heidelberg. Springer.

Davide Cacciarelli and Murat Kulahci. 2023. A survey on online active learning. *CoRR*, abs/2302.08893.

Asli Celikyilmaz, Elizabeth Clark, and Jianfeng Gao. 2020. Evaluation of text generation: A survey. *CoRR*, abs/2006.14799.

Lingjiao Chen, Matei Zaharia, and James Zou. 2023. Frugalgpt: How to use large language models while reducing cost and improving performance. *CoRR*, abs/2305.05176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. 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 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

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.

Tianyu Gao, Xingcheng Yao, and Danqi Chen. 2021. Simcse: Simple contrastive learning of sentence embeddings. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 6894–6910. Association for Computational Linguistics.

Geoffrey Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. In *NIPS Deep Learning and Representation Learning Workshop*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. Lora: Low-rank adaptation of large language models. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net.

Seokhwan Kim, Yu Song, Kyungduk Kim, Jeong-Won Cha, and Gary Geunbae Lee. 2006. MMR-based Active Machine Learning for Bio Named Entity Recognition. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, pages 69–72, New York City, USA. Association for Computational Linguistics.

Kevin J. Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. 2021. Mixkd: Towards efficient distillation of large-scale language models. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. In *Advances in Neural Information Processing Systems*, volume 35, pages 1950–1965. Curran Associates, Inc.

Lu Liu and T. Tan. 2021. Certainty driven consistency loss on multi-teacher networks for semi-supervised learning. *Pattern Recognition*, 120:108140.

Tong Luo, K. Kramer, S. Samson, A. Remsen, D.B. Goldgof, L.O. Hall, and T. Hopkins. 2004. Active learning to recognize multiple types of plankton. In *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.*, volume 3, pages 478–481 Vol.3. ISSN: 1051-4651.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? A new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 2381–2391. Association for Computational Linguistics.

Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B. Hall, Daniel Cer, and Yinfei Yang. 2022. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1864–1874. Association for Computational Linguistics.

Bo Pang and Lillian Lee. 2005. Seeing Stars: Exploiting Class Relationships for Sentiment Categorization with Respect to Rating Scales. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 115–124, Ann Arbor, Michigan. Association for Computational Linguistics.

Edoardo Maria Ponti, Alessandro Sordoni, Yoshua Bengio, and Siva Reddy. 2023. Combining parameter-efficient modules for task-level generalisation. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 687–702, Dubrovnik, Croatia. Association for Computational Linguistics.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(1).

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B. Gupta, Xiaojiang Chen, and Xin Wang. 2022. A survey of deep active learning. *ACM Comput. Surv.*, 54(9):180:1–180:40.

Carlos Riquelme. 2017. *Online Decision Making for Statistical Model Fitting*. Ph.d. thesis, Stanford University. Submitted to the Department of Mathematical and Computational Engineering.

Dan Roth and Kevin Small. 2006. Margin-based active learning for structured output spaces. In *Machine Learning: ECML 2006, 17th European Conference on Machine Learning, Berlin, Germany, September 18-22, 2006, Proceedings*, volume 4212 of *Lecture Notes in Computer Science*, pages 413–424. Springer.

Nicholas Roy and Andrew McCallum. 2001. Toward optimal active learning through sampling estimation of error reduction. In *Proceedings of the Eighteenth International Conference on Machine Learning (ICML 2001), Williams College, Williamstown, MA, USA, June 28 - July 1, 2001*, pages 441–448. Morgan Kaufmann.

Tobias Scheffer, Christian Decomain, and Stefan Wrobel. 2001. Active Hidden Markov Models for Information Extraction. In *Advances in Intelligent Data Analysis*, Lecture Notes in Computer Science, pages 309–318, Berlin, Heidelberg. Springer.

Greg Schohn and David Cohn. 2000. Less is more: Active learning with support vector machines. In *Proceedings of the Seventeenth International Conference on Machine Learning (ICML 2000), Stanford University, Stanford, CA, USA, June 29 - July 2, 2000*, pages 839–846. Morgan Kaufmann.

Christopher Schröder, 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*, pages 2194–2203, Dublin, Ireland. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM*, 63(12):54–63.

Ozan Sener and Silvio Savarese. 2018. Active learning for convolutional neural networks: A core-set approach. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net.

Burr Settles. 2009. Active Learning Literature Survey. Technical Report, University of Wisconsin-Madison Department of Computer Sciences. Accepted: 2012-03-15T17:23:56Z.

H. S. Seung, M. Opper, and H. Sompolinsky. 1992. Query by committee. In *Proceedings of the fifth annual workshop on Computational learning theory*, pages 287–294, Pittsburgh Pennsylvania USA. ACM.

Bo Shao, Lorna Doucet, and David R. Caruso. 2015. Universality Versus Cultural Specificity of Three Emotion Domains: Some Evidence Based on the Cascading Model of Emotional Intelligence. *Journal of Cross-Cultural Psychology*, 46(2):229–251. Publisher: SAGE Publications Inc.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets. *CoRR*, abs/2309.15789.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Ilias Stogiannidis, Stavros Vassos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2023. Cache me if you can: an online cost-aware teacher-student framework to reduce the calls to large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 14999–15008. Association for Computational Linguistics.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. 2018. FEVER: a large-scale dataset for fact extraction and verification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics.

Marija Šakota, Maxime Peyrard, and Robert West. 2023. Fly-swat or cannon? cost-effective language model choice via meta-modeling. *CoRR*, abs/2308.06077.

Guodong Xu, Ziwei Liu, and Chen Change Loy. 2023. Computation-efficient knowledge distillation via uncertainty-aware mixup. *Pattern Recognit.*, 138:109338.

Xiangkai Zeng, Sarthak Garg, Rajen Chatterjee, Udhyakumar Nallasamy, and Matthias Paulik. 2019. Empirical Evaluation of Active Learning Techniques for Neural MT. In *Proceedings of the 2nd Workshop on Deep Learning Approaches for Low-Resource NLP (DeepLo 2019)*, pages 84–93, Hong Kong, China. Association for Computational Linguistics.

Xueying Zhan, Qingzhong Wang, Kuan-Hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B. Chan. 2022. A comparative survey of deep active learning. *CoRR*, abs/2203.13450.

Zhisong Zhang, Emma Strubell, and Eduard H. Hovy. 2022. A survey of active learning for natural language processing. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 6166–6190. Association for Computational Linguistics.

Fan Zhou and Chengtai Cao. 2021. Overcoming catastrophic forgetting in graph neural networks with experience replay. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 4714–4722. AAAI Press.

Banghua Zhu, Ying Sheng, Lianmin Zheng, Clark W. Barrett, Michael I. Jordan, and Jiantao Jiao. 2023. On optimal caching and model multiplexing for large model inference. *CoRR*, abs/2306.02003.

A Experimental details and hyperparameters

Student model We use the T5 implementation from Huggingface's `transformers` library. We use LoRA adapters (Hu et al., 2022), as they have been considered one of the most parameter-efficient architectures in few-shot settings (Liu et al., 2022). Following Ponti et al. (2023), we add a LoRA adapter to the query, key, value and output weights in each self-attention layer of T5. We set the LoRA rank to $r = 16$, and the scaling to $\alpha = 0.25$.
We use learning rate $\eta = 5 \cdot 10^{-4}$, training batch size $m = 16$ and weight decay $\lambda = 0.01$. We validate this hyperparameter choice based on experiments using the soft labels from the teacher.

Adaptation of strategies For Entropy, we normalise before computing it by applying a softmax over the classes.

Reporting of results In order to report accuracy across budgets, we use the corresponding Area Under the Curve (AUC) divided by the budget range. By budget range, we refer to the biggest budget minus the smallest one for that task. Intuitively, this gives us an average accuracy across budgets.

Normalisation, pre-processing and evaluation We do not apply normalisation or pre-processing before using the T5 tokeniser. This can be consulted in our code.

A.1 Threshold values

To encourage an early expense of the budgets in the setting with student retraining, we have selected threshold values to ensure initially a higher proportion of calls for LLM annotation (PE=0.5, MS=5, QBC=4, CS=0.9); we have selected these values so that the first student model selects at least 50% of instances for LLM annotation on RT-Polarity. However, we observe very similar results when we use the empirical threshold from Section 5.1.

A.2 Labels from the LLM

We use a budget for LLM annotation of \$200. All the labels are obtained during May 2023. Since the OpenAI API can only return up to the five most likely tokens, we add a bias $b = 100$ to the tokens that represent each class:

- ISEAR: ' *joy*', ' *fear*', ' *anger*', ' *sadness*', ' *disgust*', ' *shame*', ' *guilt*'
- RT-POLARITY: ' *positive*', ' *negative*'
- FEVER: ' *true*', ' *false*'
- OPENBOOK: ' *A*', ' *B*', ' *C*', ' *D*'

If a class is not among the five most likely tokens, it gets assigned in our experiments a log probability of -100.

A.3 Computational resources

$T5_{base}$ has 220 million parameters. We additionally added LoRA modules, which comprise 3.5 million parameters. Only LoRA modules are trained, making it a lightweight student overall. For our experiments, we used clusters with NVIDIA Tesla V100-SXM2-16GB and NVIDIA A100-SXM4-40GB. Runs with frequency f=100 take between 1 and 4 hours to run.

B Additional results

B.1 Neural caching with no student retraining

We observe that Margin Sampling is the best-performing method on all datasets and across all the initial student models, followed by Query by Committee and outperforming the baseline of random selection (Table 8). The gap between Margin Sampling and the baseline widens as we have a better initial student.

B.2 Neural caching with retraining

Figure 5 shows the online accuracy per-dataset in the setup with retraining of the student.

B.3 Soft labels

We conduct experiments to study the effect of using soft labels (using the logprobabilities for each class from the LLM) or hard labels (only using the first class from the LLM). To do this, we train a student model on multiple budgets and obtain the final accuracy. We observe this has some gains in FEVER (Table 9).

B.4 Effect of confirmation bias in neural caching with retraining

To study the confirmation bias, we select the samples from the test dataset where the LLM produces a wrong answer. If the model performance is affected by the noise of the labels it was trained on, it is expected it will reproduce the mistakes of the LLM; therefore, we would expect that it will have a lower score in this subset of the test dataset. We do not find that Margin Sampling and Query by Committee have lower performance

N		**ISEAR**	**RT-Polarity**	**FEVER**	**Openbook**	**Average**
	Random	0.629	0.882	0.679	0.567	0.689
	Margin Sampling	0.656	**0.895**	**0.698**	**0.587**	**0.709**
	Query by Committee	0.644	0.887	0.693	0.568	0.698
500	Entropy	**0.657**	**0.895**	**0.698**	0.586	**0.709**
	Coreset (T5)	0.633	0.886	0.669	0.570	0.689
	Coreset (SimCSE)	0.636	0.887	0.682	0.569	0.694
	Coreset (MPNet)	0.632	0.886	0.675	0.566	0.690
	Random	0.640	0.886	0.704	0.662	0.723
	Margin Sampling	**0.666**	**0.896**	0.725	**0.703**	**0.748**
	Query by Committee	0.656	0.889	0.725	0.687	0.739
1000	Entropy	0.665	0.895	**0.726**	0.700	0.747
	Coreset (T5)	0.643	0.887	0.699	0.665	0.724
	Coreset (SimCSE)	0.646	0.888	0.704	0.661	0.725
	Coreset (MPNet)	0.641	0.888	0.704	0.661	0.724
	Random	0.652	0.884	0.724	0.729	0.747
	Margin Sampling	**0.673**	**0.896**	**0.751**	**0.764**	**0.771**
	Query by Committee	0.667	0.891	0.745	0.760	0.766
2000	Entropy	0.672	0.893	0.747	0.756	0.767
	Coreset (T5)	0.656	0.886	0.719	0.733	0.749
	Coreset (SimCSE)	0.657	0.888	0.725	0.728	0.750
	Coreset (MPNet)	0.655	0.888	0.724	0.727	0.749
	Random	0.648	0.885	0.738	0.734	0.752
	Margin Sampling	**0.669**	**0.895**	0.757	0.767	**0.772**
	Query by Committee	0.665	0.890	**0.758**	**0.773**	0.771
3000	Entropy	0.664	0.893	0.752	0.760	0.767
	Coreset (T5)	0.651	0.885	0.733	0.740	0.752
	Coreset (SimCSE)	0.652	0.885	0.735	0.736	0.752
	Coreset (MPNet)	0.653	0.885	0.735	0.732	0.751

Table 8: Online accuracy (AUC) for neural caching without retraining.

than front-loading in this subset of the dataset (Table 10).

B.5 Experiments with different encoders

To test the validity of results for Coreset, we repeat experiments from Section 5.2 with encoders SimCSE (Gao et al., 2021) and MP-Net (Song et al., 2020). For SimCSE, we use `sup-simcse-bert-base-uncased` from the project repository.[4] For MPNet, we use `sentence-transformers/all-mpnet-base-v2` from Huggingface. We use threshold = 0.9 for all the methods and retraining frequency $f = 100$.

We show our results in Table 11. We observe that front-loading outperforms Coreset with the three encoders.

[4] https://github.com/princeton-nlp/SimCSE

C Prompts used

The following are the prompts we used when calling the LLM. We have marked in blue one example, and in red the expected answer.

- **ISEAR**: This is an emotion classification task. Only answer one of: 'joy', 'fear', 'anger', 'sadness', 'disgust', 'shame', 'guilt'.
  INPUT: During the period of falling in love, each time that we met and especially when we had not met for a long time.
  OUTPUT: joy

- **RT-Polarity**: This is a sentiment classification task for movie reviews. Only answer either 'positive' or 'negative'.

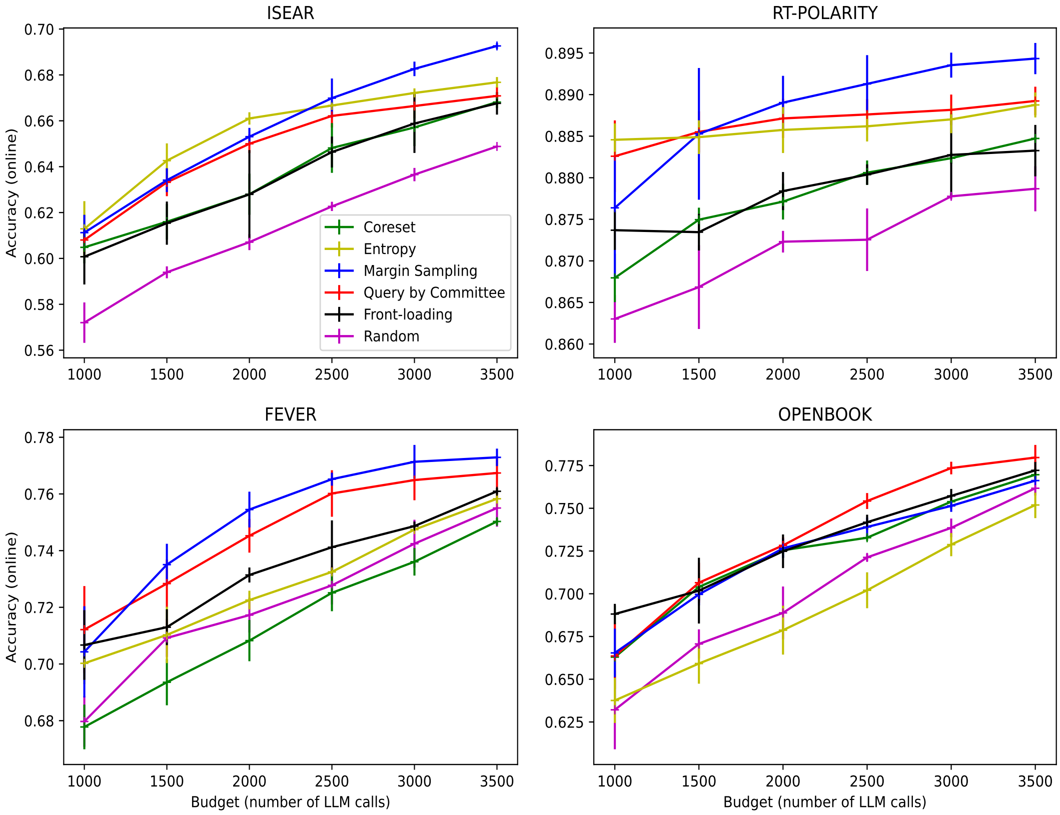

Figure 5: Accuracy curve with respect to budgets, in the neural caching problem with student retraining. Error lines indicate variance.

	ISEAR	RT-Polarity	Openbook	FEVER	Average
Soft labels	**0.598**	**0.880**	**0.617**	**0.670**	**0.691**
Hard labels	**0.598**	0.879	0.616	0.659	0.688

Table 9: Final accuracy (AUC) of the last student model, taking either soft or hard labels from the LLM.

	ISEAR	FEVER	Openbook
Front-loading	0.171	0.513	0.339
Margin Sampling	0.180	0.497	0.340
Query by Committee	0.178	0.512	0.344

Table 10: Accuracy (AUC) over the subset of the test dataset where the LLM produces wrong labels for the last student model for neural caching with student retraining.

	FEVER	Openbook
Front-loading	**0.734**	**0.731**
Coreset (SimCSE)	0.707	0.726
Coreset (MPNet)	0.716	0.724
Coreset (T5)	0.715	0.726

Table 11: Online accuracy (AUC) for neural caching with student retraining.

INPUT: if you sometimes like to go to the movies to have fun , wasabi is a good place to start .
OUTPUT: positive

- **FEVER**: This is a fact-checking task. Only answer either 'true' or 'false'.
  INPUT: On June 2017, the following claim was made: Jeb Bush is former President George H. W. Bush's daughter. Q: Was this claim true or false?
  OUTPUT: false

- **Openbook**: This is a multiple-choice test. You are presented a fact and a question. Only answer one letter, producing no more output.
  FACT: the sun is the source of energy for physical cycles on Earth
  QUESTION: The sun is responsible for
  A: puppies learning new tricks
  B: children growing up and getting old
  C: flowers wilting in a vase
  D: plants sprouting, blooming and wilting
  OUTPUT: D

D Additional information about datasets

Datasets used RT-Polarity, FEVER and Openbook were relased as NLP benchmarks for classification. While ISEAR was originally released as part of a psychological study on emotion across cultures, it has been used as an Active Learning benchmark in the past (Ein-Dor et al., 2020; Bastos and Kaul, 2021).

We did not check if these datasets contain any information that names or uniquely identifies individual people or offensive content because the data we use comes from established classification/multiple choice benchmarks. A custom filtering or modification of the data would hamper comparability with other works using these benchmarks.

Dataset generated We release the soft labels from the LLM under the CC BY 4.0 DEED license. We refer to the original data sources for documentation such as coverage of domains, languages or linguistic phenomena.