Efficient Large Language Model Invocation over Dynamic Data-Streams

Filippos Vlahos

Master of Science
School of Informatics
University of Edinburgh
2024

Abstract

As large language models (LLM) achieve groundbreaking performance in increasingly diverse tasks in natural language processing (NLP), many applications invoke them through expensive API calls. To minimise these costs, as well as the data exposure to, and reliance on the LLM providers, multiple methods to effectively invoke them have been recently proposed. These methods have demonstrated performance comparable to LLMs while significantly reducing the frequency of LLM calls. However, since most of these methods have been developed, fine-tuned, and evaluated with stationary data-streams, their effectiveness on dynamic streams, common in real-world applications, remains unclear. To address this, we conduct experiments on *Neural Caching*, an efficient LLM invocation method, where a smaller locally run model -the 'student'- is continuously trained on the responses of the LLM, gradually gaining proficiency in independently handling user requests. A critical component of Neural Caching is the policy that determines when the student should independently handle the request or when it should redirect it to the LLM. The goal of this project is to evaluate the performance of Neural Caching using the proposed selection policies, identify key factors impacting performance, and explore potential optimisations for improved outcomes, all under dynamic request streams.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Filippos Vlahos*)

Acknowledgements

I would like to sincerely express my strongest gratitude to my supervisor, *Prof Ivan Titov*, as well as to *Guillem Ramírez Santos* for their continuous guidance, advice, and support, for the constructive meetings, and for taking time to review my drafts and provide insightful feedback. I would also like to thank *Shriram*, with whom I underwent this journey, for the collaboration and discussions revolving our projects. Finally, I would like to thank my family, friends, and girlfriend for always supporting me, and motivating me throughout this thesis, and MSc degree.

Table of Contents

Chapter 1

Introduction

The usage of Large Language Models (LLMs) has surged in recent years due to their impressive performance across a multitude of natural language tasks in zero- and few-shot settings, where they excel in understanding and generating human-like text. Given the high costs and significant computational resources required to train and host LLMs, many users are instead turning to readily available LLM services offered through application programming interfaces (APIs) [49, 54]. In addition, many state-of-the-art models such as GPT-4 [2] and Claude are exclusively available as payed API services. Using those services can incur users a significant financial overhead, especially when processing large continual data-streams. Furthermore, as the latest LLMs contain billions of parameters, training them, and invoking them frequently translates into substantial energy consumption and carbon emissions [40, 53]

To minimise these costs, as well as reduce data exposure, and decrease reliance on LLM service providers, while still benefiting from LLMs' capabilities, multiple methods for effective invocation have been proposed. These methods focus on settings in which there is an incoming data stream as the input, and have been found to substantially reduce costs without a significant loss in performance [54]. However, they have been designed and evaluated, almost exclusively, on scenarios where the incoming data stream is stationary. That is, they rely on the assumption that the statistical properties of the incoming data distribution remains constant over time.

This is a significant omission as the assumption that incoming streams are stationary does not hold in many real-world applications. On the contrary, data-streams often are dynamic or drifting, which means that they exhibit changing statistical properties of their data distributions. Real world scenarios exhibiting drifting streams include search engine and social media monitoring, where specific topics become trending and are

highly searched/discussed, and the stock market, where a company's shortcomings or successes lead to higher trading volumes of its shares.

This distinction between stationary and drifting data-streams is of particular importance, as it is known that it can affect performance of online active learning (AL) methods. Strategies designed for stationary data-streams often result in sub-optimal performance when applied to drifting data-streams [12], which raises the question, how do the effective LLM invocation methods proposed perform under dynamic streams?

In this report, we study the effect of drifting data-streams on an effective invocation setup called *Neural Caching*, which has been recently introduced by Ramirez et al. [39]. The setup involves a smaller language model, referred to as the student, being continuously trained on the LLM's predictions and, as the student's accuracy increases, it increasingly handles more requests independently as seen in Figure 1.1. The student therefore, can be thought of as a smart cache, entailing the LLM's past predictions.

A key element in Neural Caching is the policy determining which requests the student should process independently and which should be delegated to the teacher (LLM), the response of which is subsequently used to train the student.

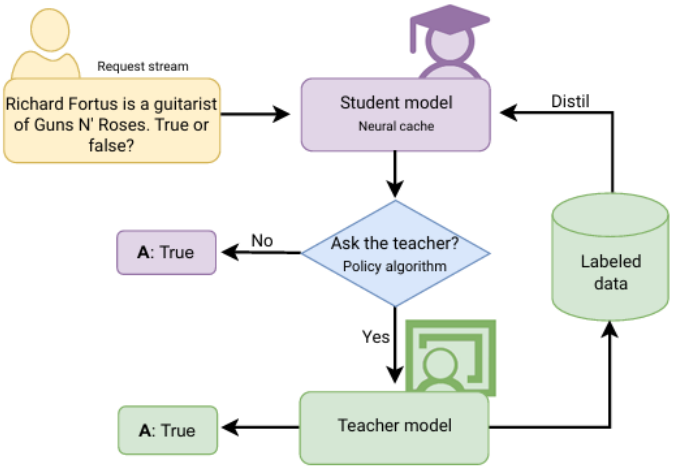


Figure 1.1: One iteration of Neural Caching (Figure from [39]): A student generates a response to a user request. The policy algorithm determines whether to rely on the student's response or to call an LLM. LLM responses are stored and used to re-train the student as more data becomes available.

In this thesis we do not focus on the performance gains of the setup, which have been demonstrated by [39, 49], but rather on the effect of various dynamic data-streams on its performance. This work aims at exploring how the different selection policies and the

Neural Caching setup itself as a whole react to different types of drifting data-streams, find key factors that impact performance in the drifting stream scenario, and identify performance improvement adjustments.

Chapter 2

Background

2.1 Active Learning

As state-of-the-art natural language processing (NLP) models are based on deep learning architectures which are data-hungry and usually require extensive labeled data to deliver good performance [57], Active Learning (AL) reduces costs by selecting a small proportion of samples from unlabeled data for labeling and training. To accomplish this, AL selects the most informative examples from the unlabelled data. These data-points are then presented to an annotator, a system responsible for labeling data-points, and these labels are subsequently used to train a model. In the current study, similarly to AL, we want to select the most informative, but also the hardest examples, for LLM-annotation which are then used to train a smaller model that handles requests independently to reduce costs.

There are three main scenarios in AL: (i) membership query synthesis, (ii) pool-based sampling, and (iii) stream-based selective sampling [27, 45]. In membership query synthesis, the learning algorithm can either select examples for labeling from the input space or generate new examples autonomously. In the pool-based scenario, the learner has access to a fixed set of previously encountered examples and can request labels for any of these instances. This study, however, focuses on the third framework, stream-based AL. Specifically, we investigate AL in the context of streaming data for classification tasks. In this framework, examples continuously emerge from a data stream, and the learning algorithm makes real-time decisions about whether to query the teacher model.

In online AL, there is no large unlabelled dataset available, but instead, we are given one unlabelled instance at a time in a continuous stream and need to decide at that time

whether to request annotation [12]. In general, online AL is crucial for various real-world applications where real-time decision making is required. A well-known example is the secretary problem [12, 16], where a hiring manager makes hiring decisions for each applicant as they are interviewed, without the benefit of seeing all the applicants. An example in NLP is a spam filter that has to decide whether to flag an email as spam in real-time.

2.2 Dynamic Data-Streams

In AL, often the incoming time-varying data-streams are affected by shifts which can occur at different speeds, severity, and distributions. These distribution shifts are often classified into three main categories [12] depending on whether they concern the feature space x or the output dimension y as seen in Figure 2.1.

A shift that only affects the input distribution $p(x)$, and not the conditional distribution $p(y|x)$, is referred to as a *covariate shift*. Under a covariate shift for two time steps t_i and $t_{i+\Delta}$, we have that $p_{t_i}(x) \neq p_{t_{i+\Delta}}(x)$ and $p_{t_i}(y|x) = p_{t_{i+\Delta}}(y|x)$. An example of such shift would be a stream of handwritten digits first appearing on white paper followed by digits written on colored paper, altering the feature distribution with the underlying task remaining the same.

Conversely, a *concept shift* is when the conditional distribution changes, and can be seen as changes in the data distribution and evolution of relationships between attributes and the target feature over time [12, 50]. In this case we have that $p_{t_i}(y|x) \neq p_{t_{i+\Delta}}(y|x)$. When a concept drift occurs, the model's predictive power significantly declines, necessitating a re-fit or replacement [12].

Lastly, a common type of distribution shift is a *label shift*, where the conditional distribution $p(x|y)$ is fixed but the label distribution $p(y)$ changes over time [55] leading to $p_{t_i}(y) \neq p_{t_{i+\Delta}}(y)$. This type of shift is encountered in multiple real-world scenarios such as prediction of diseases like influenza [12], whose distribution can dramatically change depending on the season.

It is worth mentioning that it is possible for a label shift to indirectly cause a covariate shift. This is because when p(y) changes, $p(x)$ may also be affected considering it is derived from $p(x|y)$ and $p(y)$. However, it is important to note that covariate and label shifts usually refer to different phenomena as the former refers specifically to changes in $p(x)$, while the later to changes in $p(y)$.

Data drifts are also classified based on the speed of change in *sudden* drifts which

occur in short periods, *gradual* drifts which occur with a moderate speed, and *incremental* drifts which are characterised by slow speed of change, and differences between data instances in the transition period may not even be statistically significant [50].

In this work, we focus on investigating sudden label and covariate shifts, while excluding concept drifts, as we found them to occur less naturally in the selected NLP tasks, making them less relevant. Another reason for this omission is that when a concept drift occurs, the model needs to be replaced [12], a scenario we do not consider within this work[1]. We also leave incremental and gradual speeds of change in the Neural Caching setup for future research.

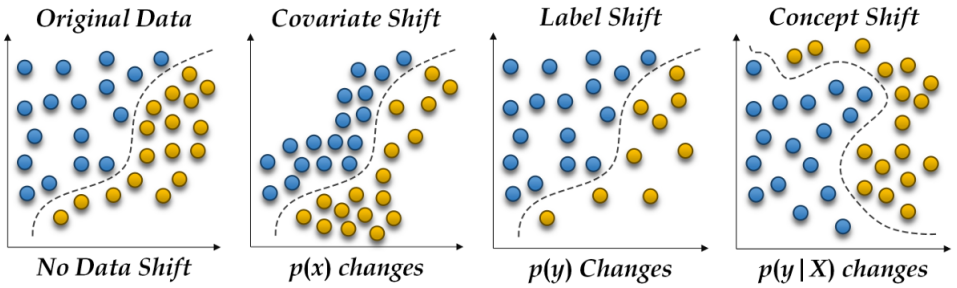


Figure 2.1: The three main categories of distribution shifts on a dataset. Illustration by [6]. The dotted line is the decision boundary between the two classes; i.e., the blue and yellow data points.

2.3 Domain Adaptation

Domain adaptation is a special case of transfer learning which refers to a class of machine learning problems where either the tasks or domains may change between training and testing [15]. In domain adaptation, only domains differ while tasks remain unchanged, and the goal generally is to train a model from a source labeled data that can be generalized to a target domain by minimizing the difference between domain distributions.

The most relevant type of domain adaptation to this work is the so called closed set domain adaptation. In this type of domain adaptation the feature (x) and label (y) spaces are identical between domains, while their probability distributions may differ [15, 26]. The possible types of domain shifts in this scenario are covariate, concept, and label shifts. A difference between our work and typical domain adaptation settings is that we

[1] In the context of Neural Caching, pottentially, both the teacher and student models would need to be replaced upon a concept drift.

do not have any source labeled data nor a target domain to generalise to, but rather a continuous unlabelled stream that exhibits domain shifts throughout time to which we seek to adapt to.

2.4 Knowledge Distillation

Knowledge Distillation (KD) [21] is a technique where knowledge is transferred from a large, complex, parameter-heavy model (teacher) to a smaller, more lightweight, and efficient one (student), with the intention to retain performance while reducing computational costs. The category of methods most closely related to ours is active KD which applies AL to KD in scenarios where a stream of unlabeled data is provided [18]. In our case, we want to distill the knowledge of the LLM into the smaller student model by retraining it on the LLMs outputs.

Notable advancements in active KD include the efficient distillation of LLMs achieving comparable performance with reduced model sizes [19, 28] and improving question answering performance by distilling the capabilities of BERT [13] into smaller models [10, 41].

2.5 Neural Caching

The Neural Caching problem was introduced in 2023 by Ramirez et al. [39] and is formalised as a problem of optimising the usage of an LLM when predictions are required for an input stream. As more predictions are obtained from the LLM, they are used to train a student model with the end goal 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 as well as to train the student model.

Formally, the goal is to establish a mapping between elements in the input space X and the corresponding labels in the output space Y. Initially, we have a student model S in a state S_0 and we have access to a teacher model T (i.e. the LLM). The task therefore, is to predict labels for a sequence of n example $(x_1, ..., x_n)$. Every f processed requests the student model is retrained on the labels obtained from T. When the student model is retrained, we reset it to its original pretrained state S_0 and then use parameter-efficient fine-tuning.

For every input x_i, the student model $S_{i/f}$ predicts the label $\hat{y}_i^S$ and, given that the budget constraint is satisfied, we have the option to request the label $\hat{y}_i^T$ from the teacher

model which incurs a cost of $c(x_i)$. The budget constraint is simply that the total cost must be less than a fixed budget b. Finally, the label $\hat{y}_i$ for input x_i is returned which is equivalent to either the teacher's or the student's label.

The level of service, that is, the effectiveness of the querying strategy, is assessed by the accuracy of the predicted label $\hat{y}_i$ compared to the actual label y_i on the online examples.

2.5.1 Instance Selection Criteria

The instance selection criteria or selection policy, is the strategy that determines which examples will be annotated by the LLM. Similarly to [39], policies inspired from AL are used in addition to two baseline methods. Methods from two commonly used broad families, as identified by [14, 31], are examined in this work, namely two uncertainty or confidence-based methods and one diversity-based method.

Uncertainty methods measure the uncertainty of a trained model on how to label a new example [45]. Uncertainty can reflect either aleatoric uncertainty, due to ambiguity inherent in the example, or epistemic uncertainty, due to limitations of the model [24, 31]. In our setup, we assume epistemic uncertainty to be prevalent, as the student model should not be proficient enough to handle all data-points, especially in the beginning of the stream. Uncertainty methods are some of the most commonly used query frameworks, straight-forward to implement, and computationally efficient [44]. In this work, the methods of this family examined are Margin Sampling and Prediction Entropy.

Methods belonging to the diversity-based family attempt to select diverse data-points that can cover the entire feature space by considering the distance between samples [25]. This family of methods attempts to address a weakness of the uncertainty-based family, that is, the selection of duplicate or very similar samples. In this study, Coreset [44] is the method of this family we examine.

2.5.1.1 Baseline Selection Policies

The two baseline selection policies used are front-loading (FL) and Random (Rand). In FL, all incoming requests are selected for LLM annotation until the entire budget is used. Once the budget is used and the student is retrained, all subsequent requests are handled by the student model alone.

In Rand, we select requests for LLM annotation randomly, with a uniform sampling rate across the entire incoming stream.

2.5.1.2 Margin Sampling (MS)

Margin sampling [42] takes into account the difference between the posterior probabilities of the two most probable labels, referred hereafter as margin. Intuitively, instances with large margins are easy, since the classifier has little doubt in differentiating between the two most likely class labels. Instances with small margins are more ambiguous, thus knowing the true label would help the model discriminate more effectively between them [45].

$$Margin(x_i) = logP(y_i = c_1^*|x_i) - logP(y_i = c_2^*|x_i) \tag{2.1}$$

In Equation 2.1, c_1^* and c_2^* are the first and second most likely classes respectively, based on the posterior probability distribution $P(y_i|x_i)$ computed by the student model.

MS is a popular selection policy in AL [7, 43]. In the original Neural Caching paper [39], it was found to be the most effective method for stationary streams while it has also been successfully used in other effective invocation methods [38].

2.5.1.3 Prediction Entropy (EN)

The entropy [46] of a probability distribution measures the amount of disorder or randomness in the distribution. In AL, the entropy of a model's predicted class probabilities for a data-point can be used as an indicator of the model's uncertainty about the correct class label for that data point [12]. Thus, we select data points for annotation that produce an entropy value above a certain threshold, with the entropy being calculated as follows:

$$Entropy(x_i) = -\sum_j P(y_i = c_j^*|x_i) logP(y_i = c_j^*|x_i) \tag{2.2}$$

where c^* is the most likely class based on the posterior probability distribution $P(y_i|x_i)$ computed by the student model. Entropy is a commonly used uncertainty-based selection policy in AL [31, 39, 49].

2.5.1.4 Coreset (CS)

On every new instance, Coreset [44], uses an encoder to generate its embedding representation. It then computes the cosine similarity between this new embedding and the embeddings of previous examples. If the similarity with the most similar past instance x_i annotated by the teacher (LLM) is below a threshold s, then the instance is selected for annotation by the teacher model. Similarly to [39], the embeddings are obtained by averaging the encoder's representation across tokens, a technique proven effective in sentence embedding benchmarks [33]. GPTCache [8], a widely used effective invocation method also uses embeddings to create a "semantic cache" to decide whether an input request should be labelled by an LLM.

2.6 Other Effective LLM-Invocation Methods

Stogiannidis et al. [49] also explore the Neural Caching problem with a similar setup, where an LLM (GPT-3 or GPT-4) is used as a teacher model, and a simpler cost-effective model (K-NN or Multi-Layer Perceptron) as the student. In their experiments they used Neural Caching to tackle an intent recognition and a sentiment analysis dataset. The selection policy they employ is a combination of the entropy of the probability distribution produced by the student model for the incoming instance, and the distance of the vector representation of the incoming instance from the centroid of the vector representations of the k most similar cached instances. Their results showed promise for Neural Caching as calls to the teacher could be significantly reduced (by $\approx 33\%$ with only a slight performance drop (0.37 %).

A different setup introduced by Nie et al. [34] involves a series of different student models of increasing complexity where smaller models progressively learn from the ongoing outputs of an LLM. In this work the objective is to learn a "cascade" of models (instead of just a single one as in Neural Caching) starting with lower-capacity models such as logistic regression, and ending with a powerful and expensive LLM. Which model is used for which input is determined by a deferral policy which initially allows most examples to reach the LLM and as the setup becomes more stable, and the smaller models can successfully handle most examples, only harder examples reach the LLM. Their experiments showed that this setup can achieve performance comparable to LLMs while significantly reducing inference costs. Additionally, they tested the robustness of their setup by introducing a covariate shift and finding minimal performance decline.

This is the only effective LLM invocation method we are aware of in which the impact of drifting data-streams is analysed.

More recently, Ramirez et al. [38] introduced a setup where instead of retraining a small auxiliary model, they use two LLMs, a smaller and a larger one. Their approach removes the additional complexity of re-training a student model and relies on the information within the LLMs. They use MS, which we also employ, as their selection policy and show that it performs consistently well on a range of short-generation tasks. Similarly to Neural Caching, they use cascading, i.e. all queries are passed through the small model, and depending on its output, the (large) LLM is invoked.

Chapter 3

Methodology

In this chapter we present the utilised datasets, how we simulate online LLM annotation, details of our experimental setup, and our method of evaluation.

3.1 Datasets

In this work, as in [39], we focus on classification tasks using four datasets. We use all four in the first experiment to derive some general, initial results, while one is used in all further experiments allowing to go into more depth and analyse multiple aspects of the effect of dynamic request streams in the Neural Caching problem.

3.1.1 ISEAR

The International Survey on Emotion Antecedents and Reactions (ISEAR) [47] is a psychological dataset that contains responses from individuals across various countries, detailing their emotional experiences. Participants were asked to recall instances where they felt one of seven emotions: joy, fear, anger, sadness, disgust, shame, or guilt. They described the situation, their emotional reactions, and the consequences of those emotions. The dataset contains 6068 data-points with seven possible classes.

ISEAR has been widely used in NLP [39], with recent research continuing to use it extensively in areas such as emotion detection [1, 3], but also in other ML tasks such as in effective LLM innovation research [34, 38, 39]

3.1.2 OpenBook

OpenBook [32] is a dataset modelled after open book exams for assessing human understanding of a subject. The dataset consists of 5957 multiple-choice elementary-level science questions derived from science textbooks, with the 4 possible classes being 'A', 'B', 'C', and 'D'. Each instance consists of a question, along with a common knowledge fact that can be used to answer it. This dataset is particularly challenging as it requires multi-step reasoning, use of additional common and commonsense knowledge, and rich text comprehension to choose the correct answer. OpenBook is commonly used to benchmark state of the art models such as Llama-2 [52] on commonsense reasoning.

3.1.3 RT-Polarity

Polarity [36] is a binary sentiment classification dataset consisting of 10662 movie reviews labeled as either 'positive' or 'negative' with an equal number of examples belonging to each class. Due to its simplicity and effectiveness, Polarity has been extensively used in NLP research in developing and benchmarking models for sentiment analysis tasks [58].

3.1.4 FEVER

The Fact Extraction and VERification (FEVER) dataset [51] is a fact checking dataset that contains claims derived from Wikipedia developed to evaluate the ability of models to verify claims against a corpus of evidence. We use 6610 fact data-points followed by the question *"Was this claim true or false?"*. This is another binary classification task with the two classes being "true" and "false". FEVER tests our setup's ability to perform reasoning and information verification, and has been used to benchmark recent KD setups [34, 39].

3.2 Annotation by LLM

In this study, the online setup is simulated by using the datasets[1] created by [39] which contain LLM predictions for all data-points of each of the four tasks. The soft labels in the dataset have been generated by *OpenAI*'s *text-davinci-003* an InstructGPT-based model [57].

[1] https://huggingface.co/datasets/guillemram97/cache_llm

InstructGPT models are based on GPT-3 [11], which is fine-tuned using supervised learning on a set of prompts along with demonstrations of desired model behaviour. Then, a dataset of rankings of model outputs is collected, which is used to further fine-tune this supervised model using reinforcement learning from human feedback [35].

As observed by [39], the LLM achieves considerably better accuracy than the student model, and slightly better accuracy than when compared to the student trained on 5000 gold labels. This confirms the usefulness of applying the Neural Caching setup on the selected tasks as it is possible to achieve similar accuracy to the LLM while reducing the number of LLM invocations and therefore reducing costs.

3.3 Experiment Details

The experiment setup is similar to [39] as their publicly available repository[2] was our starting point, and we extended it to experiment on drifting data-streams. Upon completion, our code was also made publicly available on *GitHub*[3] for reproducibility and future use.

All experiments have been run with three random seeds, which also determine the ordering of examples where applicable, with the average scores being presented for robustness. For simplicity, a constant cost per query $c(x_i) = 1$ is used.

In our experiments, the student model is a pre-trained $T5_{base}$ transformer, a text-to-text model [37]. For retraining, we freeze the model's original weights and incorporate Low-Rank Adaptation (LoRA) layers for parameter-efficient fine-tuning (PEFT) [22]. PEFT is a practical approach that enables efficient adaptation of a pre-trained model to a specific task with minimal additional parameters. This method selectively adjusts a small subset of the model's parameters while keeping the majority unchanged [20]. Employing PEFT is particularly beneficial in zero- and few-shot settings like ours, where it facilitates faster and more accurate predictions [29]. In this scenario, although the student model is lightweight enough to not require expensive hardware, it is still able to adapt and learn a task using PEFT for efficiency.

Similarly to [39], a LoRA adapter is added to the query, key, value, and output weights in each self-attention layer of T5, the LoRA rank is set to $r = 16$, the scaling to

[2]Neural Caching Repository: `https://github.com/guillemram97/neural-caching`

[3]Neural Caching Drifting Data-Streams Repository: `https://github.com/FilipposVlahos/neural-caching-temporal`

$\alpha = 0.25$, the learning rate to $\eta = 5 \cdot 10^{-4}$, and training batch size $m = 16$.

The student model is fine-tuned using the cross-entropy loss on the log probabilities assigned by the teacher in each class, as [39] found this to be slightly beneficial over using only the most likely class. The accumulated data from the LLM is split into training and validation sets, and each student is trained from scratch for 30 epochs with early stopping with patience of five epochs.

3.4 Evaluation

To report accuracy across budgets, the corresponding Area Under the Curve (AUC) divided by the budget range is used to obtain an average accuracy. The budget range explored is from 1000 to 3500 with a step of 500.

We only consider online accuracy, which is the accuracy of the setup on the incoming stream, in this study, rather than final accuracy, which is the accuracy of the retrained student model on a test dataset upon completing a simulation. We justify this as we are mainly interested in how Neural Caching performs while the data drifts occur, rather than how the student model performs on stationary data at the end of the simulation.

Chapter 4

Experiments and Results

In this chapter, we present the experiments conducted along with their respective results. We show our experiments on label shifts in Section 4.1, and on covariate shifts in Section 4.2. The results presented here are further discussed in Chapter 5.

4.1 Label Shift Experiments

In this section, we first present our results from simulating label shifts in the input data-stream across all four datasets. Next, we focus on the ISEAR dataset aiming to enhance performance under label shifts and, more broadly, any non-stationary input stream. Upon modifying the setup we analyse the behaviour of the AL strategies pre- and post-adaptation, and using the adapted setup, we rerun the label shift experiment across all datasets. Finally, to understand aspects of labels shifts that impact performance, we run two more experiments on ISEAR.

We only use one dataset for the more in-depth experiments due to constraints in time and to make the analysis of the results easier. The ISEAR dataset was chosen for two reasons, (i) it consists of multiple classes which makes it more suitable to study label shifts, and (ii) its classes (emotions) are semantically related to the examples, as opposed to OpenBook which contains multiple choice questions with classes A, B, C, and D.

4.1.1 Label Shift Across Datasets

In this first experiment we simulate sudden label shifts on each dataset. To do so, we arrange the data-points of each dataset based on their label. This results in all

examples belonging to the same label to appear sequentially, followed by examples belonging to the next label and so on. We arrange the labels in random order, and as each experiment is run thrice, with a different seed each time, different sorting combinations are performed in each simulation and an average accuracy is calculated.

Label shifts are expected to deteriorate performance, especially for the baseline selection policies [39], as when a label shift occurs, the student model will have been last retrained on data-points belonging to previously encountered classes. As examples belonging to the previously ”unseen” class appear, the model will be strongly biased to predict previous classes resulting in the accuracy dropping significantly until the student is retrained again.

For this first experiment we use the same hyperparameters used by [39] for stationary request streams. These hyperparameters consist of a retraining frequency $f = 1000$, and of the following thresholds for the AL selection policies: MS=5, CS=0.9, EN=0.5. These hyperparameters have been optimised for stationary streams and are used here to examine the impact on performance upon introducing dynamic streams to a setup tuned for stationary streams.

The results are reported in Table 4.1 and the respective accuracy curves in Figure 4.1. As expected, we observe that consistently, across datasets, applying a label shift reduces performance. Looking closer at each task, starting from ISEAR, surprisingly the Rand selection policy is the one least affected by the shift (7% drop) while all the remaining ones suffer a more significant drop ($\approx 10\%$).

OpenBook is the task with the highest drop in performance, with the only policy without a significant drop being CS, which was already performing poorly with no drift present. FL slightly outperforms CS in this task making it the most successful policy. This is the hardest task, but also the only one in which the labels have no semantic relation with the corresponding example, which could explain the severity of the drop. A multiple choice question being annotated by the LLM with the answer being ‘A’ does not provide information to the model about the answer of subsequent questions. Therefore, annotating an example here serves mostly to attain higher accuracy rather than to train the student which is less beneficial as it results in the student being biased towards the label most present in the training data. This difference could explain why the baseline methods perform the best in this task regardless of the shift.

Similarly to OpenBook, CS also performs comparably well in FEVER as it surpasses its accuracy with stationary streams where again, it was already performing poorly. Finally, in Polarity we observe the least performance decay overall which can be

attributed to it being the easiest task based on accuracy, as well as on it being a binary classification task, which means that there is only single label shift transition.

Overall, we observe an interesting switch in the performance of the baseline policies. While in stationary streams FL was the better method reaching a performance close to the AL-policies, when introducing a label shift FL becomes the worst performing policy, and Rand performs closer, if not better, to the AL-inspired strategies. From the AL-based policies EN seems to have the most consistent performance across tasks when label shifts are introduced.

(a) ISEAR

Shift	Label	None
Margin Sampling	0.564	**0.656**
Coreset	0.516	0.639
Entropy	**0.566**	0.650
Front Loading	0.512	0.637
Random	0.529	0.598

(b) OPENBOOK

Shift	Label	None
Margin Sampling	0.392	0.680
Coreset	0.444	0.490
Entropy	0.394	0.633
Front Loading	0.444	**0.688**
Random	**0.471**	0.598

(c) FEVER

Shift	Label	None
Margin Sampling	0.597	**0.716**
Coreset	**0.681**	0.635
Entropy	0.657	0.714
Front Loading	0.505	0.703
Random	0.645	0.678

(d) POLARITY

Shift	Label	None
Margin Sampling	0.870	**0.888**
Coreset	0.828	0.877
Entropy	**0.873**	0.882
Front Loading	0.824	0.880
Random	0.864	0.866

Table 4.1: Online Accuracy (AUC) for Neural Caching when faced with a Label Shift and with no shift (i.i.d.) in the incoming stream. The same hyperparameters used by Ramirez et al. [39] for a stationary request stream were used for this experiment. Highlighted are the best accuracies achieved for each shift. A significant performance drop is observed across the datasets when a label shift is applied.

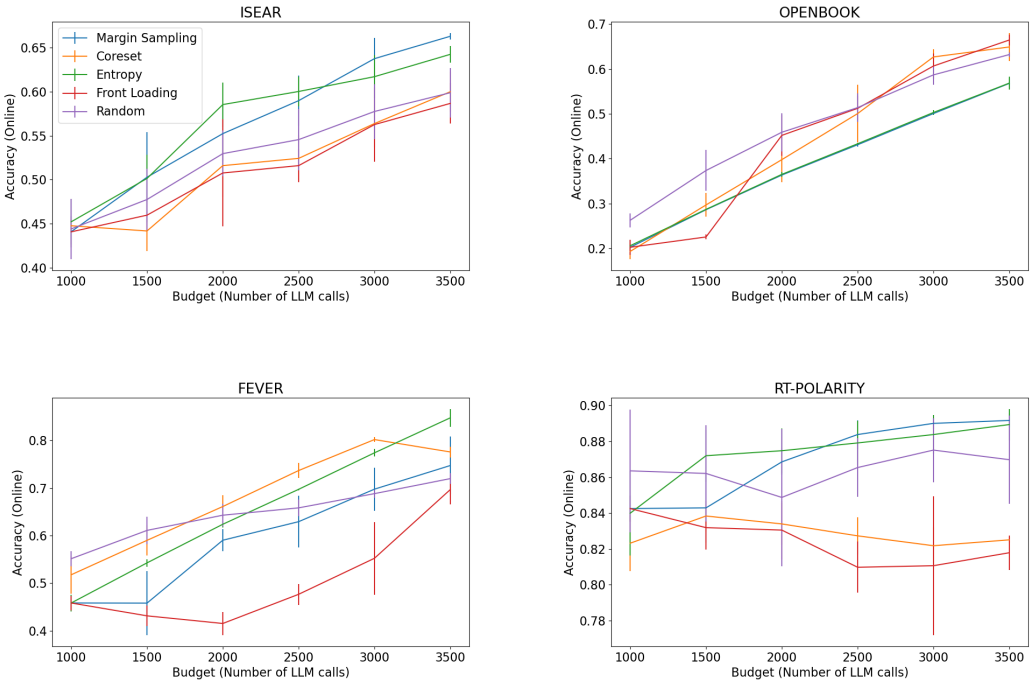


Figure 4.1: Accuracy curves with respect to budgets in the Neural Caching problem with an incoming stream exhibiting a label shift. Error lines indicate variance. The simulations have been run using the hyperparametes used by Ramirez et al. [39].

4.1.2 Adapting Neural Caching for Streams with Label Shifts

In the experiments in the previous section we saw how significantly dynamic label drifts reduce performance. This drop is not surprising as discussed in Section 4.1.1, but it is likely to have been intensified by the experimental set-up being originally developed for, and assessed on, stationary rather than dynamic data-streams. It is known that a setup designed for stationary streams, when applied to dynamic ones, often performs sub-optimally [12]. This naturally raises the question whether adapting parts of the setup and/or the AL-based strategies could contain this performance decline or perhaps even revert it.

4.1.2.1 Increasing Retraining Frequency

The first adaptation that we hypothesise will improve performance for streams with dynamic label shifts is to increase the retraining frequency from $f = 1000$ to $f = 500$. A higher retraining frequency is possible to improve performance regardless of the incoming data distribution, but we hypothesise that when drifts are present, the

improvement will be greater. The reason behind this is that increasing f means that retraining is more likely to occur closer to when label drifts occur, thus enabling the student model to detect the drift faster by being trained on the new data quicker. This change should result into a quicker reaction by the Neural Caching setup to drifts without allowing the performance to drop as much.

Our experiment results are presented in Table 4.2 where we notice that indeed, increasing the retraining frequency is beneficial. We observe a stronger performance increase on dynamic rather than stationary streams. Specifically, the performance of MS increases only with a label shift present, while EN and Rand have a much stronger performance improvement under label shifts than they do with no shifts. As expected, a higher f seems to enable Neural Caching to respond more swiftly to shifts. In contrast, when no shifts are present, this advantage diminishes, leading to only marginal gains or no improvement at all, a result that agrees with previous studies [39].

It is worth noting that increasing the retraining frequency comes with the increased computational costs of retraining the student model more frequently, and in cases where the retraining does not take place asynchronously, with some downtime necessary each time the student model is retrained. Consequently, and based on our results, we would mostly recommend reducing the retraining frequency in cases where data drifts are expected to be present in the incoming stream.

Subsequent label shift experiments in this section are run with a retraining frequency of 500.

Shift	Label		None	
Retraining Frequency	500	1000	500	1000
Margin Sampling	**0.572**	0.564	0.656	0.656
Coreset	0.516	0.516	0.639	0.639
Entropy	**0.586**	0.566	**0.657**	0.650
Front Loading	0.512	0.512	0.637	0.637
Random	**0.575**	0.529	**0.613**	0.598

Table 4.2: Online Accuracy (AUC) for Neural Caching comparing a higher retrain frequency (500) to the one previously used (1000) on ISEAR exhibiting a label shift and no shift in the incoming stream. We notice that the benefit of the higher frequency is more significant on the dynamic stream.

4.1.2.2 Window-Based Online Active Learning

Often in AL scenarios with drifting streams, as new examples are introduced, older ones become increasingly irrelevant and eventually, when used for retraining, can lead to a drop in performance [17]. For instance, in a spam detection system that continuously learns from new emails, using old examples from years ago, when spam patterns were different, can cause the model to mistakenly flag current emails as spam, thereby reducing its accuracy.

Motivated by this, we hypothesised that as label shifts occur, older data-points biasing the student model to predict classes encountered in the beginning of the stream will not be useful for retraining once examples from that class stop occurring in the stream. To address this, we implement a simple window based online AL approach as seen in Figure 4.2 from [12], where instead of storing all examples previously selected for LLM annotation and using them to retrain the student, we introduce a buffer that only stores the most recent k examples and deletes older ones. This implementation acts as a simple forget mechanism. In our experiment, k equals the retraining frequency f, so at most, the student will be retrained on the last 500 data-points selected for LLM annotation.

The results are presented in Table 4.3 and by comparing them to those in Table 4.2, we observe that the forget mechanism does not benefit all selection policies, but EN which benefits from a small performance gain, and particularly Rand which becomes the most successful policy. On the other hand, MS exhibits a minor accuracy drop, while CS exhibits a more significant decay in performance ($\approx 4\%$).

This poor performance may be attributed to the fixed threshold of the AL-policies being sub-optimal with this configuration due to the changes in the AL-policy values caused by training on much fewer examples which would explain why only Rand's performance significantly increases.

4.1.2.3 Fine-tuning Thresholds

Besides the retraining frequency and forgetting old examples, the AL selection policies themselves could be improved as they were also fine-tuned, and evaluated on stationary streams.

First, we attempted to fine-tune the thresholds for each AL strategy to see if the values optimised for stationary streams translate well when faced with dynamic streams. The results are presented in Table 4.7 where we first observe that fine-tuning the

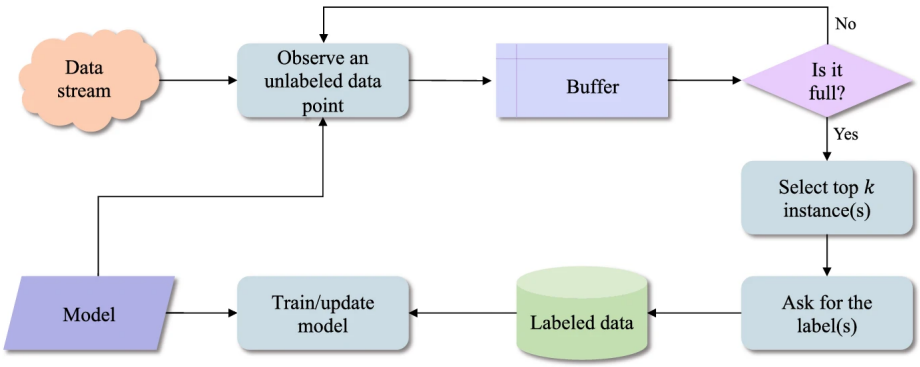


Figure 4.2: Window-based online Active Learning. Figure from Cacciarelli et al. [12].

Strategy	Online Accuracy (AUC)
Margin Sampling	0.568
Coreset	0.473
Entropy	0.588
Front Loading	0.504
Random	**0.591**

Table 4.3: Online Accuracy (AUC) for Neural Caching on ISEAR with window-based online AL under label shifts, using only the previous $k = f = 500$ LLM annotated examples are used to retrain the model.

threshold results in some performance gains for all policies under both stationary and dynamic label shifts. More interestingly, we notice that the optimal thresholds differ between stationary and dynamic streams demonstrating the importance of adapting the setup upon changes in the distribution of the incoming data-stream.

Furthermore, we notice that a smaller threshold performs better for label shifts in all three strategies, contrary to the stationary stream where larger ones perform better. This could be because under label shifts, AL-based selection policies (EN/CS/MS) may display increased confidence after retraining on examples belonging to the same class, until the shift occurs, resulting in fewer examples being delegated to the more accurate LLM.

4.1.2.4 Dynamic Threshold

In the previous experiment we saw that different threshold values work better depending on the incoming stream distribution, but what if instead of having a fixed threshold, we

Thr.	Shift	
	Label	None
4.0	**0.576**	0.660
4.5	0.565	0.656
5.0*	0.565	0.656
5.5	0.570	0.660
6.0	0.557	**0.665**

Table 4.4: MS

Thr.	Shift	
	Label	None
0.8	**0.536**	0.622
0.85	0.513	0.637
0.9*	0.514	0.639
0.95	0.515	0.639
1.0	0.514	**0.641**

Table 4.5: Coreset

Thr.	Shift	
	Label	None
0.4	0.585	0.658
0.45	**0.594**	0.644
0.5*	0.584	0.657
0.55	0.590	**0.665**
0.6	0.589	0.643

Table 4.6: Entropy

Table 4.7: Online Accuracy (AUC) for Neural Caching on ISEAR under label shifts when fine-tuning the threshold of the AL policies. The policies we fine-tune are Margin Sampling, Coreset, and Entropy. We notice that the optimal hyperparameters differ consistently between the two input distributions.

had a mechanism that dynamically adapts the threshold value? We introduce such a mechanism by setting the threshold T of the t-th example as follows:

$$T_t = \mu_t \pm \sigma_t * z \tag{4.1}$$

where μ is the mean and σ is the standard deviation of the strategy-value (MS/CS/EN) of the past x examples up to time t. The scaling factor z determines how many standard deviations away from the mean the threshold should be set and is fine-tuned separately for each AL-policy. In our experiment we've set $x = 50$, and have fine-tuned z finding that the optimal value for MS and EN is $z = 0.8$ and for CS $z = 0.9$. We report the results of our fine-tuning experiments of z in Appendix A. The $\pm$ in Equation 4.1 corresponds to the fact that for MS we select examples for annotation where the strategy value (i.e. the margin) is less than T as we want to select examples with low margin, in which case we use $-$, while for CS and EN we select those that are greater than T and we use $+$.

We use dynamic thresholds as we believe that they are promising for dynamic drifts as, when the drift occurs, we expect the strategy value to diverge from the previous ones, and we would like to select those new data-points for annotation to quickly adapt to the shift, which would not always be the case with a fixed threshold. In addition, this method would be practical in scenarios where the incoming data-stream alternates between different types of distribution shifts as we saw in Section 4.1.2.3 that different threshold values perform best depending on the shift.

For each AL-policy, we run experiments using the dynamic threshold on it's own and then combined with window-based AL with the results being reported in Table 4.8. We observe that by adding the dynamic threshold the performance of CS and EN increases, while that of MS decreases comparing to Table 4.7. When we combine the dynamic threshold with window-based AL though, the performance of all three AL-strategies further increases with EN performing the best. This improvement in performance when using the dynamic threshold with window-based AL is inline with our speculation in Section 4.1.2.2 that using a forget mechanism results in sub-optimal performance with fixed threshold values.

	Dynamic	Dynamic + Window-Based
Margin Sampling	0.548	0.564
Coreset	0.545	0.554
Entropy	0.597	**0.603**

Table 4.8: Online Accuracy (AUC) for Neural Caching using a dynamic threshold standalone and in conjunction with window-based online AL. We observe that by combining the two methods we achieve the highest accuracy.

4.1.3 Optimal Configuration

Having adapted the set-up for dynamic request streams and evaluated this new configuration on ISEAR, we rerun the experiments across the four datasets to see if the new setup generalises well to different tasks. To recap, this optimised configuration consists of a higher retraining frequency of $f = 500$, a window-based AL forget mechanism, and a dynamic threshold for the AL policies.

On Table 4.9 we compare the previous results of applying a label shift on the initial setup with applying the shift on the optimised setup for each of the four datasets. On Table 4.10 the average accuracy (AUC) across the datasets are reported. We observe that although MS's performance had not improved on ISEAR upon modifying the setup, in the other tasks it shows a strong improvement exhibiting an overall average improvement of $\approx 7\%$. EN's performance which had significantly improved on ISEAR also improves on the rest of the datasets showing that for the uncertainty-based selection policies our modifications are highly beneficial and generalise well on different tasks. While CS on the other hand had also showed improvement on ISEAR, it exhibited a

performance decline on the remaining tasks suggesting that either the hyperparameters (e.g. z) need to be fine-tuned separately on each policy for CS to perform well, or that the new setup does not benefit this policy.

Looking at the baseline methods, Rand exhibits a strong improvement in performance with the optimised setup, outperforming the other selection policies in three out of the four tasks.

(a) ISEAR

Parameters	Initial	Optimised
Margin Sampling	0.564	0.564
Coreset	0.516	**0.554**
Entropy	0.566	**0.603**
Front Loading	**0.512**	0.504
Random	0.529	**0.591**

(b) OPENBOOK

Shift	Initial	Optimised
Margin Sampling	0.392	**0.502**
Coreset	**0.444**	0.432
Entropy	0.394	**0.522**
Front Loading	**0.444**	0.418
Random	0.471	**0.552**

(c) FEVER

Shift	Initial	Optimised
Margin Sampling	0.597	**0.748**
Coreset	**0.681**	0.568
Entropy	0.657	**0.747**
Front Loading	0.505	**0.520**
Random	0.645	**0.764**

(d) POLARITY

Shift	Initial	Optimised
Margin Sampling	0.870	**0.897**
Coreset	**0.828**	0.791
Entropy	0.873	**0.893**
Front Loading	**0.824**	0.805
Random	0.864	**0.901**

Table 4.9: Online Accuracy (AUC) for Neural Caching under label shifts across datasets, comparing performance using the initial parameters with that produced using the optimised ones.

4.1.4 Impactful Features in Label Shifts

To gain a better understanding of the behaviour of Neural Caching under label shifts, as well as to identify key aspects of label shifts that impact performance, we conduct two more experiments using the initial parameters, as in Section 4.1.1, on ISEAR.

Hyperparameters	Initial	Optimised
Margin Sampling	0.605	**0.678**
Coreset	**0.617**	0.586
Entropy	0.622	**0.692**
Front Loading	**0.572**	0.562
Random	0.628	**0.702**

Table 4.10: Average Online Accuracy (AUC) across datasets for Neural Caching under label shifts, comparing the optimized setup's performance with the initial one. We observe that the uncertainty-based policies as well as Rand, significantly benefit from the optimisation.

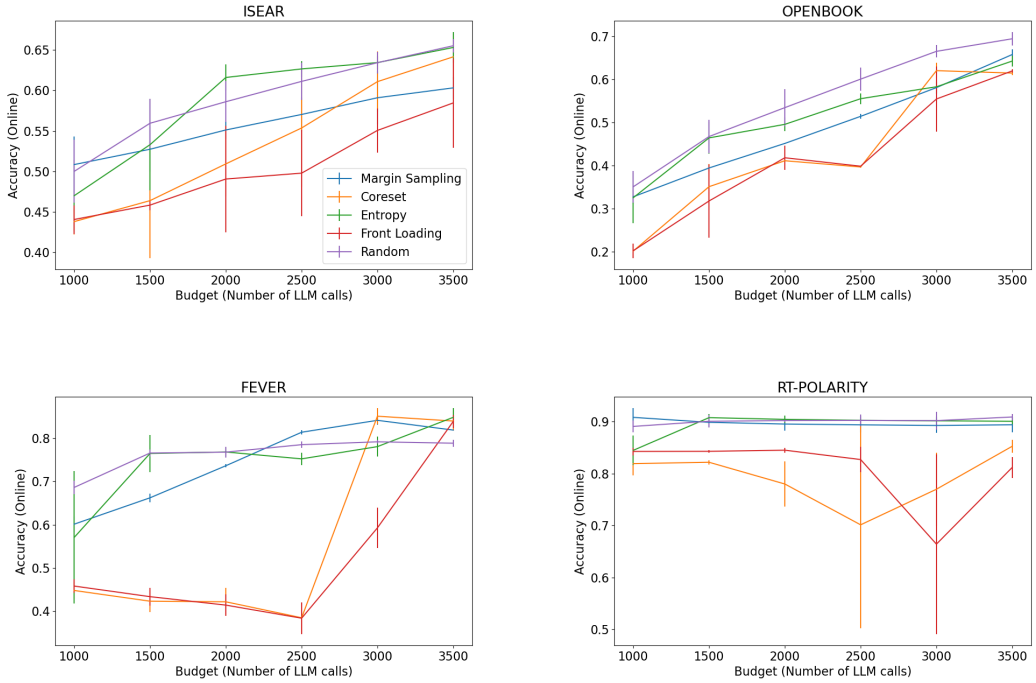


Figure 4.3: Accuracy curves with respect to budgets in the Neural Caching problem with an incoming stream exhibiting label shifts using the optimised setup.

4.1.4.1 Strength of Shift

This next experiment has been designed to analyse how the "strength" of the shift affects performance. By strength here we mean the percentage of examples belonging to the label that is subject to the shift. In real-world scenarios, not all data-points will

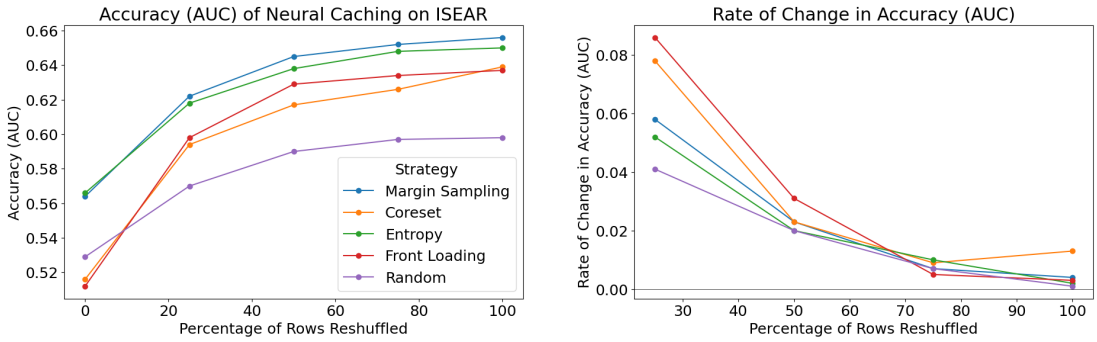
necessarily fall under the label shift, but only some proportion of them. For example, a model that predicts diseases like influenza, whose distribution changes dramatically over the winter months, although the proportion of positive examples will change sharply each month, there will not be a full shift where each disease appears exclusively on certain months of the year. Thus understanding how well each strategy performs under different strengths of label shifts can help determine which approach is most robust. If a strategy shows a smaller rate of decline in accuracy as the shift strength changes, it might be more suitable for environments where the data distribution behaviour is more unpredictable.

To simulate different strengths of label shifts, we apply label shifts as in the experiments in Section 4.1.1 and then shuffle randomly a percentage of examples before initiating the experiment. By incrementally increasing the percentage of shuffled examples, we make the shift weaker allowing us to observe how the accuracy changes for each policy.

We report the experiment results in Table 4.11, in Figure 4.4a we graphically present the same results, while in Figure 4.4b we illustrate the corresponding rate of change in accuracy as the intensity of the label shift varies. We notice that MS and then EN achieve the highest accuracy across the different shift strengths while also maintaining a relatively low rate of change making them very consistent. We also observe that the accuracy decreases as the shift becomes stronger, with the biggest drop being observed between having the 25% mark and the full label shift.

Rows Shuffled	0% (Full Label Shift)	25%	50%	75%	100% (No Shift)
MS	0.564	0.622	0.645	0.652	0.656
CS	0.516	0.594	0.617	0.626	0.639
EN	0.566	0.618	0.638	0.648	0.650
FL	0.512	0.598	0.629	0.634	0.637
RND	0.529	0.570	0.590	0.597	0.598

Table 4.11: Online Accuracy (AUC) of Neural Caching on ISEAR for increasingly weaker label shifts. The percentages indicate the proportion of rows that have been reshuffled, with 0% representing a full label shift and 100% representing no shift i.e. stationary input stream.



(a) (b)

Figure 4.4: Online Accuracy (AUC) for Neural Caching under different label shift strengths. Strengths are simulated by arranging the dataset so that full label shifts are present and then shuffling a proportion of rows. The proportions we simulate are: 0%, 25%, 50%, 75%, 100% where 0% is equivalent to a full label shift and 100% to no shift. On the left (a) we plot the accuracy (AUC) and on the right (b) the rate of change of the accuracy (AUC) with respect to the different label shift strengths.

4.1.4.2 Teacher's Accuracy

This final label shift experiment aims to investigate how the sequence in which labels are presented affects the performance of the Neural Caching setup within the context of a full label shift where the LLM exhibits varying levels of accuracy across different labels.

We observed that the LLM is considerably more accurate in classifying certain emotions than others on ISEAR as seen in Table 4.12. Thus, instead of sorting examples randomly based on their labels, we sort them from the 'easiest' to the 'hardest' label and vice versa, where the easiest class is the one the teacher predicts with the highest accuracy.

This experiment is relevant for real-world scenarios where there is an option to initiate the setup on easier or harder tasks. For example, let's assume that Neural Caching is used to assist students in answering school questions. There are two classes of students; a beginners class and an advanced class. Will performance be impacted if the model is first provided to the advanced class, where the teacher model will be less accurate but the budget will be used on answering more difficult questions, or vice versa, where the LLM will have greater performance, but easier questions will be mostly used for training? To answer this we devised the following experiment.

The results are reported in Table 4.13 where we observe that all selection policies

Class	LLM Accuracy (%)
Joy	98.29
Fear	85.97
Sadness	81.09
Shame	73.58
Anger	58.29
Guilt	55.71
Disgust	42.10

Table 4.12: Accuracy of the LLM -teacher model- for each class (emotion) of ISEAR. Labels sorted from ”easiest” to ”hardest”.

perform significantly better when the harder examples appear first. Impressively, MS, EN, and FL outperform their accuracy even in the absence of any shift (see Table 4.1), highlighting the significant impact that label order has on label shifts in Neural Caching.

The shift simulated in this experiment entails a mixture of a label and a covariate shift as we arrange the examples based on both their label, and their ’difficulty’, a property of the input distribution $p(x)$.

Labels Sorted by LLM Accuracy	Ascending	Descending
Margin Sampling	0.657	0.561
Coreset	0.628	0.573
Entropy	0.651	0.558
Front Loading	0.641	0.522
Ramdom	0.597	0.552

Table 4.13: Online Accuracy (AUC) for Neural Caching on ISEAR under label shifts were labels are sorted based on the predictive accuracy of the LLM as seen in Figure 4.12. We notice that when trained on the toughest class first, where there are many miss-classifications, Neural Caching performs significantly better.

4.2 Covariate Shift Experiments

In this section we examine two different cases of covariate shifts, i.e. shifts that only affect the input distribution $p(x)$, on the ISEAR dataset.

4.2.1 Sentence Length

Longer inputs generally entail more complex semantics. For example, the accuracy of GPT-3.5 is significantly lower on longer movie reviews [34]. Therefore, a covariate shift in the data would be exhibited by arranging the data-points based on sentence length. This way we can assess Neural Caching on a covariate shift over the input's semantic complexity.

The results are reported in Table 4.14 where we see that all the AL-based strategies perform better when the longer sentences are presented first, thus demonstrating that in covariate shifts, it is beneficial for Neural Caching when more complicated examples are present in the beginning of the stream. We also see that MS is the best performing selection policy in this scenario in both types of covariate shift.

Sentence Length	Ascending	Descending
Margin Sampling	0.646	**0.657**
Coreset	0.619	**0.628**
Entropy	0.639	**0.651**
Front Loading	0.634	**0.641**
Random	**0.600**	0.597

Table 4.14: Online Accuracy (AUC) for Neural caching under a covariate shift simulated by sorting the incoming examples by sentence length. We notice that starting with longer sentences in the beginning of the stream consistently results in better performance.

4.2.2 Typos

In this experiment we simulate a different type of covariate shift where we randomly introduce spelling mistakes and typos to half of the instances of the dataset, and test how the ordering of these examples affect performance in Neural Caching. To do so, we first present all the original instances followed by those containing typos and vice versa. We also run simulations and for a version of the dataset where all data-points contain typos to ensure that introducing typos has an effect in accuracy, and where the instances with typos appear randomly in the stream as a baseline that ensures that a covariate shift is meaningful in this scenario.

With this experiment we attempt to understand scenarios in which there is sudden change in the language used in the incoming stream. Such a case, where the input

suddenly contains typos, could be if Neural Caching was first exposed to a data-stream produced by users with learning difficulties that affect spelling, followed by users without. More generally though, with this experiment we attempt to examine the model's behaviour when presented with any kind of sudden variation in the language that does not affect the posterior $p(y|x)$ and $p(y)$. Examples of such variations are any change in the dialect, regional accent, or level of formality used in the input stream.

We introduced typos to ISEAR by iterating through its data-points and prompting an LLM (Llama-3-8B [4]). We designed a prompt that describes the task without containing any in-context examples i.e. zero-shot, but we used a small part of the dataset (up to 10 examples) for prompt engineering.

Looking at Table 4.15, where we report the results, first we confirm that the experiment is valid, as introducing typos indeed reduces performance (we compare the 'All Typos' to the 'No Typos' column). We can also confirm that arranging examples based on whether they contain typos and exhibit a covariate shift is also impactful as there is change in accuracy between mixing randomly the examples and sorting them. We observe that for all strategies but CS, presenting typos first results in Neural Caching performing better.

CS is the policy that is overall worse affected by typos which intuitively makes sense as it selects examples for annotation based on their semantic similarity, which will be harder to compute when words are misspelled. This could explain why it reacts differently to the other selection policies when the covariate shifts are present.

As the remaining policies perform better when typos appear first, we see again that harder examples in the beginning of the stream result in a better performance. We also see that MS is the best performing policy in all the different simulations.

Shift	Typos First	Typos Last	Mixed	All Typos	No Typos
Margin Sampling	0.638	0.617	0.632	0.614	0.656
Coreset	0.568	0.577	0.573	0.547	0.639
Entropy	0.631	0.615	0.630	0.608	0.650
Front Loading	0.626	0.593	0.613	0.594	0.637
Random	0.578	0.574	0.574	0.552	0.598

Table 4.15: Online Accuracy (AUC) for Neural Caching uncer a covariate shift simulated by introducing typos to (part of) the incoming examples.

Chapter 5

Discussion

5.1 Selection Policies under Dynamic Streams

5.1.1 General Observations

Evaluating how the selection policies react to dynamic streams, we start by observing that across experiments, the performance of the original Neural Caching setup exhibits a drop when a dynamic (with label or covariate shifts) data-stream is present regardless of the selection policy used. Such performance fluctuations based on the incoming shift underscore the need to carefully consider which selection policy to employ.

We observe, that across our experiments, in both label and covariate shifts, the uncertainty-based methods seem to consistently perform relatively well. In the covariate shifts, in both experiments MS performs the best, closely followed by EN. In the label shift experiments, although not always the best performing methods, they retain a comparatively high performance across experiments, regardless of the shifts or the methods used. Furthermore, they were also the methods consistently performing the best as the strength of the shift changed (see Figure 4.4).

On the other hand, we have found CS to be the most unreliable AL-based method, performing poorly in most of our experiments. This poor performance of CS is in agreement with previous results on Neural Caching [39]. However, we have seen a couple of instances where CS performs better than its competition, namely, in the label shift experiments with use of the initial setup on OpenBook and FEVER.

Furthermore, looking at Figure 4.3 we observe that consistently across the datasets CS performs very poorly for lower budgets, but for budgets greater than 2500, its performance is similar to that of other policies. This might suggest that in scenarios

where the budget constraint is generous, using CS might not be an unreasonable choice. Besides, a similar diversity-based strategy is the main LLM caching technique used by practitioners (GPTCache), but is used mostly for cases in which multiple near-identical calls are present in the incoming stream [8, 39], a scenario we do not examine in this study.

5.1.2 Baseline VS. AL-based Selection Policies

A key factor in the decision of which selection policy to employ is the difference in performance between the baseline policies and the AL-inspired ones. We saw in Table 4.1 that when no shifts are present, FL has a performance comparable, if not better, to the AL-policies, while Rand performs significantly worse. It was believed that if the incoming data exhibited dynamic shifts, this difference in performance would automatically increase [39], but in this work we have seen this not to be the case necessarily. Upon applying label shifts, all policies suffered a significant drop in performance with Rand emerging as one of the most accurate policies as it its performance suffered the least, performing comparably to the AL-based ones. When adapting the setup to improve performance under dynamic streams in Section 4.1.2, the accuracy of FL further decreased, but Rand was still performing comparably to the uncertainty policies whose performance improved.

This unexpected performance gain from Rand under label shifts may make a case for using simpler selection policies, but much caution is suggested. We noticed that the baseline methods are significantly less consistent across the different experiments. FL performs well when there is no drift, but performs poorly for label shifts and is average under covariate shifts. Conversely, Rand performs impressively well under label shifts, but not so much under no shift and covariate shifts. This means that if an incoming stream exhibits shift changes, then using baseline methods makes for a risky choice and is likely to lead to sub-optimal performance.

In addition, we believe that uncertainty-based policies have significant room for improvement in their performance, while the baseline methods are less flexible. As mentioned, although the uncertainty-based strategies saw an impressive improvement in performance after optimising, they do not consistently outperform Rand in the label shift scenario. A potential reason behind this is that with the dynamic threshold we use, they under-utilise their budget leading to them mostly spending it at the end of the stream, as opposed to Rand which evenly distributes the budget across the stream. This

is evident in Figure 5.1 where we see that EN, and especially MS, suffer from not using their budget enough early on the stream. This would be less of a concern in streams with more examples, in which case, the budget would be spread more evenly, and would possibly outperform Rand.

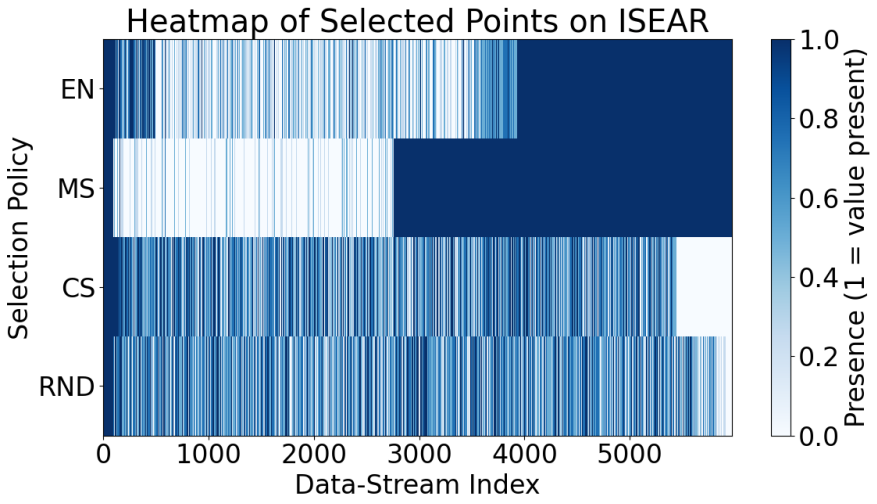


Figure 5.1: Heatmap of selected data-points across strategies on the ISEAR Dataset. The y-axis represents the different selection strategies (EN, MS, CS, RND), while the x-axis corresponds to the value indices in the data stream. The color intensity reflects the frequency of selection, where a value of 1 indicates that the point was consistently selected across runs. This heatmap was generated by recording the data points chosen by each strategy for annotation by the LLM in a binary array, with 1 indicating selection and 0 indicating exclusion. The arrays were averaged across multiple runs with different seeds to produce the final visualization.

To address this problem in scenarios where the number of data-points in the stream is known in advance, we suggest adding pressure to the selection policy to use the budget more evenly across the incoming data-stream. One way of achieving this would be dynamically reducing the threshold when the budget is under-used. In case the stream length is unknown, a scaling threshold could be used that selects all data-points above one standard deviation σ from the mean value, 90% of data-points above 0.9 σ, and so on.

5.1.3 In-Depth Analysis of AL-based Selection Policies

To closer observe the behaviour of the AL-based metrics, in Figure 5.2 we plot how the semantic difference value of CS, the margin of MS, and the prediction entropy of

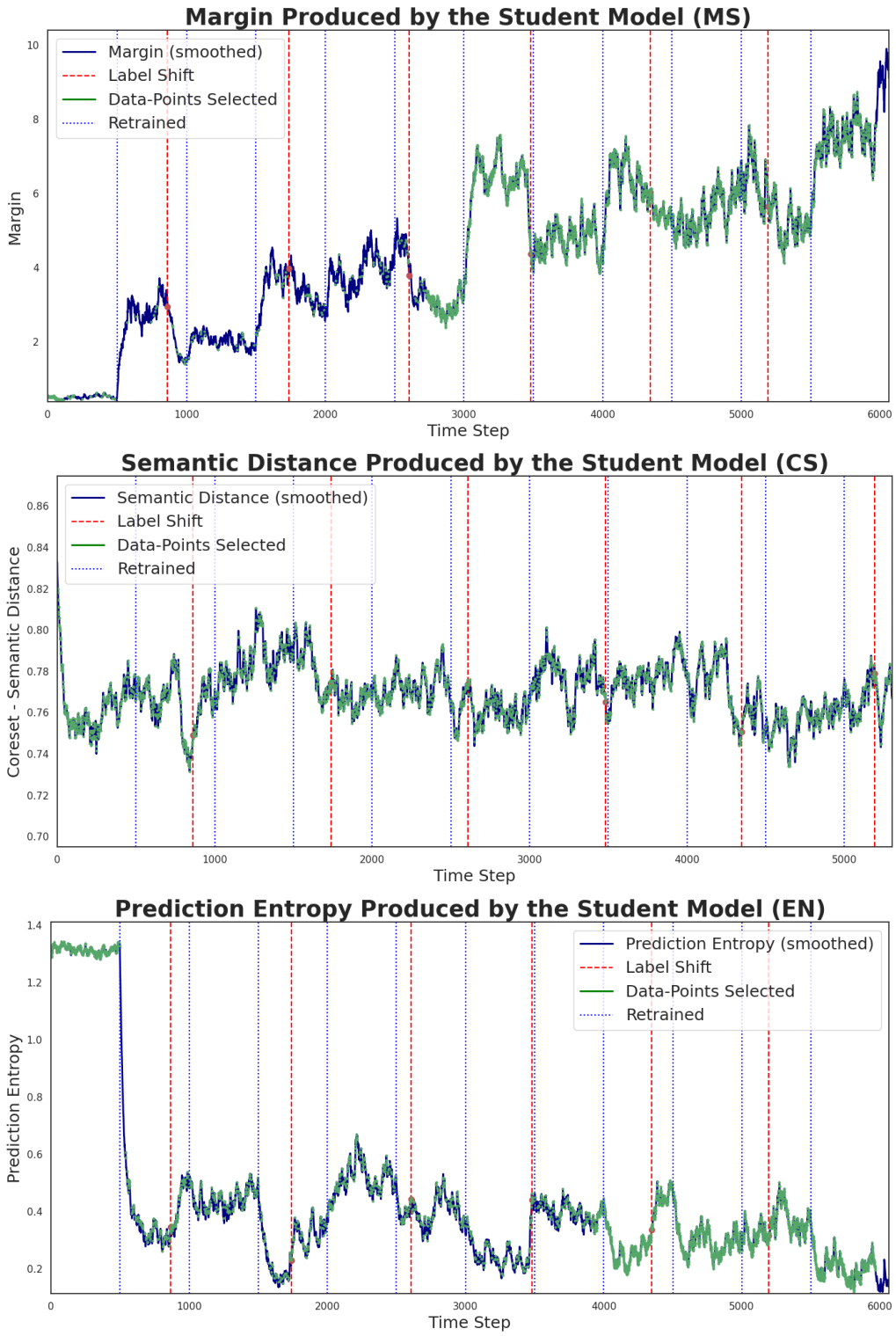


Figure 5.2: Values of each metric (y-axis) for each time step (x-axis) as the experiment on ISEAR with label shifts progresses. The red vertical lines show where a label shift takes place, the blue vertical lines where retraining occurs ($f = 500$), and the green dots signify the examples selected for LLM-annotation. The values have been smoothed using exponential smoothing with $\alpha = 0.3$ to reduce noise while the original values are displayed in Figure C.1.

EN change over time, as well as the data-points selected under a dynamic stream of ISEAR data-points exhibiting label shifts. We observe quite consistently that when a label shift occurs, there is a fluctuation of the value (increase for EN and CS, decrease for MS) while conversely, when the model is retrained we see the opposite effect. This reconfirms the impact of these shifts as well as the potential of detecting these shifts using the selected metrics.

We notice that CS is the metric with the least amount of fluctuations in its value, making this method potentially less suitable for detecting shifts and selecting the most useful examples, potentially further explaining its poor performance. On the other hand, both MS and EN exhibit stronger variation and both seem to progressively improve with increasingly higher margin, and smaller entropy values being observed as the simulation progresses.

As discussed earlier, in MS and EN most of the budget is used towards the end of the stream, but in Figure 5.2 we also see that contrary to our expectations, it is not consistently clear that more data-points are selected for annotation after a shift takes place. We attribute this to the large fluctuation and noise in the policy-values as observed in Figure C.1, resulting in data-points being selected for annotation regardless of whether a shift has occurred as they will often diverge from the standard deviation. When smoothing the data we see that there is an overall trend, so it is more likely that a data-point will be selected after a shift, but this effect is not strong enough. We believe that smoothing the selection-policy values as the new data-points appear would result in more points closer to the shift being selected. This also reconfirms that with more fine-tuning, these selection policies could be further improved.

5.2 Calibration Required upon Data-Stream Distribution Changes

Throughout our experiments we have seen differences in the performance of Neural Caching depending on the characteristics of the incoming data-stream. First, we observe a strong performance decline upon introducing rapid label shifts, then, when increasing the retraining frequency, we notice that Neural Caching reacts differently based on the incoming data-stream distribution. In addition, when fine-tuning the thresholds for the AL-based selection policies, the optimal values differ between no shifts and label shifts consistently.

These differences lead us to the conclusion that to reach an optimal performance in Neural Caching, and more generally in KD-based effective LLM invocation methods, it is important to adapt the hyperparameters and the selection policy to changes in the distribution of the incoming data-stream. Failing to do so, a significant performance decline is likely to occur. If the incoming data-stream distribution is expected not to be constant exhibiting different types of drifts over time, then techniques that automatically adapt to change like the dynamic threshold we implemented could be beneficial.

5.3 Prioritising Hard Examples may be Beneficial

Consistently across our experiments we saw that when harder examples were present in the beginning of the stream, the accuracy of Neural Caching increased. We observed this phenomenon with both label and covariate shifts. In the former, when data-points belonging to labels harder for the LLM to classify appear early in the stream, the resulting accuracy increases (Table 4.13), and in the latter, both when longer, more complicated sentences (Table 4.14), or when examples with typos (Table 4.15) appeared early in the stream, the performance of Neural Caching improved.

As seen in the experiments mentioned, accuracy is significantly affected by the order in which data-points are presented in Neural Caching with better performance when 'difficult' examples appear first. This is contradictory to commonly used methods based on Curriculum Learning, where gradually increasing the complexity of the data-points during training is used as a technique to improve performance [48].

Additionally, this reverse type of curriculum learning benefiting performance is observed despite the fact that harder examples appearing first means that more 'misleading' annotations by the LLM will be used to retrain the student. The phenomenon of training the student model on misclassified labels by the teacher is called 'confirmation bias' in KD [5], and is usually harmful in these settings. However, in Neural Caching we consistently observe clear gains in accuracy when these 'hard', noisy examples are used for training. This conclusion has been found to be true in Neural Caching for stationary streams before [39], and in this work we show that the same effect is strong in dynamic request streams too.

5.4 Limitations and Future Work

One of the main limitations of our study is that we only consider sudden data shifts, and did not examine gradual or incremental drifts. Sudden shifts, as well as no shifts in the data-stream, are the two extreme ends of the spectrum and thus we see Rand and FL performing the best in each case respectively. In the real-world, it is likely that drifts will not be sudden, which could result in different behaviour of Neural Caching, and possibly an increased benefit in using the AL-based policies. We leave for future work to explore such cases.

Another limitation is that we conduct experiments on datasets that do not naturally exhibit shifts, but rather we synthetically simulate them. In addition, we only consider classification datasets, which are a subset of NLP tasks that may exhibit drifts. We can only hypothesise that our findings generalise in more realistic settings, but we have not conducted any such experiments to demonstrate this. For future work, datasets that naturally entail data drifts like StockNet [56], a stock movement prediction dataset from tweets and historical stock prices, could be analysed to evaluate how the results derived from synthetically produced drifts generalise on real tasks that exhibit data drifts.

A third limitation is that the selection policies we deploy come from only two families. Exploring a wider range of selection policies could potentially result in greater performance and deeper insight into the effect of dynamic shifts on Neural Caching. The uncertainty selection policies we use benefit from computational efficiency due to their simplicity and are relatively straightforward to analyse as they contain few hyperparameters and can be concisely described mathematically. However, more advanced methods both in the uncertainty family and beyond have been proposed. In the former, a lot of work is been made on improving uncertainty estimation with LLMs [23, 30].

Beyond the uncertainty family, a method that has been found to outperform simple uncertainty-based methods in detecting shifts in multi-domain question answering and sentiment analysis [31] is H-divergence [9, 31]. This method is based on the idea of quantifying the difficulty for a discriminator to differentiate between two domains. We believe that such methods or more recent uncertainty methods could further increase accuracy in effective LLM-invocation techniques under dynamic streams. We leave this investigation for future work.

Chapter 6

Conclusions

In this thesis we explored the effect of dynamic request streams on Neural Caching, an effective LLM invocation method. We conducted experiments under two types of distribution shifts in the incoming data-stream. Under label shifts, we demonstrated the severe impact in performance that distribution shifts can have when not optimising the setup accordingly. We then identified adjustments to the setup and showed that a combination of increased retraining frequency, a mechanism for forgetting old examples and a dynamic threshold for the AL-based policy values can significantly improve performance under these shifts.

Throughout our experiments, we evaluated the performance of three AL-based policies, and compared them with two baseline methods showing that the uncertainty-based policies, MS and EN, were the most consistent across experiments. They exhibited strong performance under both regular label and covariate shifts, as well as under different strengths of label shifts. CS, the diversity-based policy we examined, was found to not be as suitable in most cases, with a few exceptions, as, for example, when the budget is on the higher end.

The experimental results demonstrated the importance of adjusting elements of Neural Caching upon drift changes to maintain an optimal performance. In addition, through experiments under both label and covariate shifts we have shown that prioritising hard examples may result in significantly improved performance in Neural Caching.

Some limitations of our work include that the optimal performance of MS and EN has probably not been achieved as too many examples were found to have been selected for LLM annotation at the end of the stream. Also, we did not notice a significant increase in examples selected after a shift as we had expected. Therefore, there are further improvements to be made in order to optimally configure the setup's

hyperparameters, which suggests that the potential of the proposed methods is even better than the results in this thesis indicate. We leave for future work the further optimization of the setup, as well as the exploration of more complicated selection policies, of gradual data drifts, and datasets that naturally exhibit these drifts we synthetically produce.

Finally, we believe that the cost reduction associated with Neural Caching while maintaining a strong performance, as shown by previous studies, is already a compelling reason to utilise the proposed effective LLM-invocation methods. With this thesis we hope to have further strengthened the case for these methods by demonstrating their behaviour under various dynamic data drifts and by showing the potential of using uncertainty-based selection policies with the proposed setup optimisations.

Bibliography

[1] Ahmed R Abas, Ibrahim Elhenawy, Mahinda Zidan, and Mahmoud Othman. Bert-cnn: A deep learning model for detecting emotions from text. *Computers, Materials & Continua*, 71(2), 2022.

[2] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

[3] Acheampong Francisca Adoma, Nunoo-Mensah Henry, and Wenyu Chen. Comparative analyses of bert, roberta, distilbert, and xlnet for text-based emotion recognition. In *2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 117–121, 2020.

[4] Meta AI. Llama 3 model card. *https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md*, 2024.

[5] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International joint conference on neural networks (IJCNN)*, pages 1–8. IEEE, 2020.

[6] Mehdi Ataei, Murat Erdogdu, Sedef Akinli, Sedef Ben-David, Shems Saleh, Ali Pesaranghader, Andrew Alberts-Scherer, George Sanchez, Saeed Pouryazdian, Ahmad Ghazi, Jennifer Nguyen, Karim Khayrat, Ali Pesaranghader, Andrew Alberts-Scherer, George Sanchez, Saeed Pouryazdian, and Bo Zhao. Understanding dataset shift and potential remedies. Technical report, Vector Institute, 2021.

[7] Maria-Florina Balcan, Andrei Broder, and Tong Zhang. Margin based active learning. In *International Conference on Computational Learning Theory*, pages 35–50. Springer, 2007.

[8] Fu Bang. 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, 2023.

[9] Shai Ben-David, John Blitzer, Koby Crammer, and Fernando Pereira. Analysis of representations for domain adaptation. *Advances in neural information processing systems*, 19, 2006.

[10] Yasaman Boreshban, Seyed Morteza Mirbostani, Gholamreza Ghassem-Sani, Seyed Abolghasem Mirroshandel, and Shahin Amiriparian. Improving question answering performance using knowledge distillation and active learning. *Engineering Applications of Artificial Intelligence*, 123:106137, 2023.

[11] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[12] Davide Cacciarelli and Murat Kulahci. Active learning for data streams: a survey. *Machine Learning*, 113(1):185–239, 2024.

[13] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[14] Hady Elsahar and Matthias Gallé. To annotate or not? predicting performance drop under domain shift. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2163–2173, 2019.

[15] Abolfazl Farahani, Sahar Voghoei, Khaled Rasheed, and Hamid R Arabnia. A brief review of domain adaptation. *Advances in data science and information engineering: proceedings from ICDATA 2020 and IKE 2020*, pages 877–894, 2021.

[16] PR Freeman. The secretary problem and its extensions: A review. *International Statistical Review/Revue Internationale de Statistique*, pages 189–206, 1983.

[17] João Gama, Indrė Žliobaitė, Albert Bifet, Mykola Pechenizkiy, and Abdelhamid Bouchachia. A survey on concept drift adaptation. *ACM computing surveys (CSUR)*, 46(4):1–37, 2014.

[18] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[19] Yuxian Gu, Li Dong, Furu Wei, and Minlie Huang. Knowledge distillation of large language models. *arXiv preprint arXiv:2306.08543*, 2023.

[20] Zeyu Han, Chao Gao, Jinyang Liu, Sai Qian Zhang, et al. Parameter-efficient fine-tuning for large models: A comprehensive survey. *arXiv preprint arXiv:2403.14608*, 2024.

[21] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[22] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

[23] Yuheng Huang, Jiayang Song, Zhijie Wang, Shengming Zhao, Huaming Chen, Felix Juefei-Xu, and Lei Ma. Look before you leap: An exploratory study of uncertainty measurement for large language models. *arXiv preprint arXiv:2307.10236*, 2023.

[24] Alex Kendall and Yarin Gal. What uncertainties do we need in bayesian deep learning for computer vision? *Advances in neural information processing systems*, 30, 2017.

[25] Yeachan Kim and Bonggun Shin. In defense of core-set: A density-aware core-set selection for active learning. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 804–812, 2022.

[26] Wouter M Kouw and Marco Loog. An introduction to domain adaptation and transfer learning. *arXiv preprint arXiv:1812.11806*, 2018.

[27] Janez Kranjc, Jasmina Smailović, Vid Podpečan, Miha Grčar, Martin Žnidaršič, and Nada Lavrač. Active learning for sentiment analysis on data streams: Methodology and workflow implementation in the clowdflows platform. *Information Processing & Management*, 51(2):187–203, 2015.

[28] Kevin J Liang, Weituo Hao, Dinghan Shen, Yufan Zhou, Weizhu Chen, Changyou Chen, and Lawrence Carin. Mixkd: Towards efficient distillation of large-scale language models. *arXiv preprint arXiv:2011.00593*, 2020.

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

[30] Linyu Liu, Yu Pan, Xiaocheng Li, and Guanting Chen. Uncertainty estimation and quantification for llms: A simple supervised approach. *arXiv preprint arXiv:2404.15993*, 2024.

[31] Shayne Longpre, Julia Reisler, Edward Greg Huang, Yi Lu, Andrew Frank, Nikhil Ramesh, and Chris DuBois. Active learning over multiple domains in natural language tasks. *arXiv preprint arXiv:2202.00254*, 2022.

[32] Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. Can a suit of armor conduct electricity? a new dataset for open book question answering. *arXiv preprint arXiv:1809.02789*, 2018.

[33] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877*, 2021.

[34] Lunyiu Nie, Zhimin Ding, Erdong Hu, Christopher Jermaine, and Swarat Chaudhuri. Online cascade learning for efficient inference over streams. *arXiv preprint arXiv:2402.04513*, 2024.

[35] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[36] Bo Pang and Lillian Lee. Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales. *arXiv preprint cs/0506075*, 2005.

[37] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.

[38] Guillem Ramírez, Alexandra Birch, and Ivan Titov. Optimising calls to large language models with uncertainty-based two-tier selection. *arXiv preprint arXiv:2405.02134*, 2024.

[39] Guillem Ramírez, Matthias Lindemann, Alexandra Birch, and Ivan Titov. Cache & distil: Optimising api calls to large language models. *arXiv preprint arXiv:2310.13561*, 2023.

[40] Santiago Rojas. Evaluating the environmental impact of large language models: Sustainable approaches and practices. *Innovative Computer Sciences Journal*, 10(1):1–6, 2024.

[41] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

[42] Tobias Scheffer, Christian Decomain, and Stefan Wrobel. Active hidden markov models for information extraction. In *International symposium on intelligent data analysis*, pages 309–318. Springer, 2001.

[43] Christopher Schröder, Andreas Niekler, and Martin Potthast. Revisiting uncertainty-based query strategies for active learning with transformers. *arXiv preprint arXiv:2107.05687*, 2021.

[44] Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

[45] Burr Settles. Active learning literature survey. *University of Wisconsin, Madison*, 52, 07 2010.

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

[47] Bo Shao, Lorna Doucet, and David R Caruso. 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, 2015.

[48] Petru Soviany, Radu Tudor Ionescu, Paolo Rota, and Nicu Sebe. Curriculum learning: A survey. *International Journal of Computer Vision*, 130(6):1526–1565, 2022.

[49] Ilias Stogiannidis, Stavros Vassos, Prodromos Malakasiotis, and Ion Androutsopoulos. Cache me if you can: an online cost-aware teacher-student framework to reduce the calls to large language models. *arXiv preprint arXiv:2310.13395*, 2023.

[50] Andrés L Suárez-Cetrulo, David Quintana, and Alejandro Cervantes. A survey on machine learning for recurring concept drifting data streams. *Expert Systems with Applications*, 213:118934, 2023.

[51] James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. Fever: a large-scale dataset for fact extraction and verification. *arXiv preprint arXiv:1803.05355*, 2018.

[52] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[53] Aida Usmanova, Junbo Huang, Debayan Banerjee, and Ricardo Usbeck. Reporting and analysing the environmental impact of language models on the example of commonsense question answering with external knowledge. *arXiv preprint arXiv:2408.01453*, 2024.

[54] Can Wang, Bolin Zhang, Dianbo Sui, Zhiying Tum, Xiaoyu Liu, and Jiabao Kang. A survey on effective invocation methods of massive llm services. *arXiv preprint arXiv:2402.03408*, 2024.

[55] Ruihan Wu, Chuan Guo, Yi Su, and Kilian Q Weinberger. Online adaptation to label distribution shift. *Advances in Neural Information Processing Systems*, 34:11340–11351, 2021.

[56] Yumo Xu and Shay B Cohen. Stock movement prediction from tweets and historical prices. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1970–1979, 2018.

[57] Xueying Zhan, Qingzhong Wang, Kuan-hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B Chan. A comparative survey of deep active learning. *arXiv preprint arXiv:2203.13450*, 2022.

[58] Ce Zhou, Qian Li, Chen Li, Jun Yu, Yixin Liu, Guangjing Wang, Kai Zhang, Cheng Ji, Qiben Yan, Lifang He, et al. A comprehensive survey on pretrained foundation models: A history from bert to chatgpt. *arXiv preprint arXiv:2302.09419*, 2023.

Appendix A

Fine-Tuning Dynamic Thresholds

In Tables A.1, A.2, A.3, and A.4 we present the results form fine-tuning the z value from Equation 4.1 for each selection policy. We experimented with four values (0.7, 0.8, 0.9, 1).

	Dynamic	Dynamic + Window-Based
Margin Sampling	0.551	0.553
Coreset	0.535	0.541
Entropy	0.578	0.580

Table A.1: $z = 1$

	Dynamic	Dynamic + Window-Based
Margin Sampling	0.547	0.548
Coreset	0.545	0.554
Entropy	0.593	0.601

Table A.2: $z = 0.9$

	Dynamic	Dynamic + Window-Based
Margin Sampling	0.548	**0.564**
Coreset	0.546	0.522
Entropy	0.597	**0.603**

Table A.3: $z = 0.8$

	Dynamic	Dynamic + Window-Based
Margin Sampling	0.538	0.539
Coreset	0.525	0.495
Entropy	0.599	0.603

Table A.4: $z = 0.7$

Appendix B

LLM-Prompt to Insert Typos

The prompt we used to insert typos for our experiment presented in Section 4.2.2: *"You are an automatic typo generator. Given an input phrase, you generate the exact same phrase, but insert typos that appear natural. It is very important that you only provide the final output without any additional comments or remarks. Input Prompt: {sentence"}*.

Appendix C

Raw Selection Policy Values

In Figrue C.1 we present the un-smoothed raw selection policy values. We observe that they are noisy, making it hard to observe an overall trend.

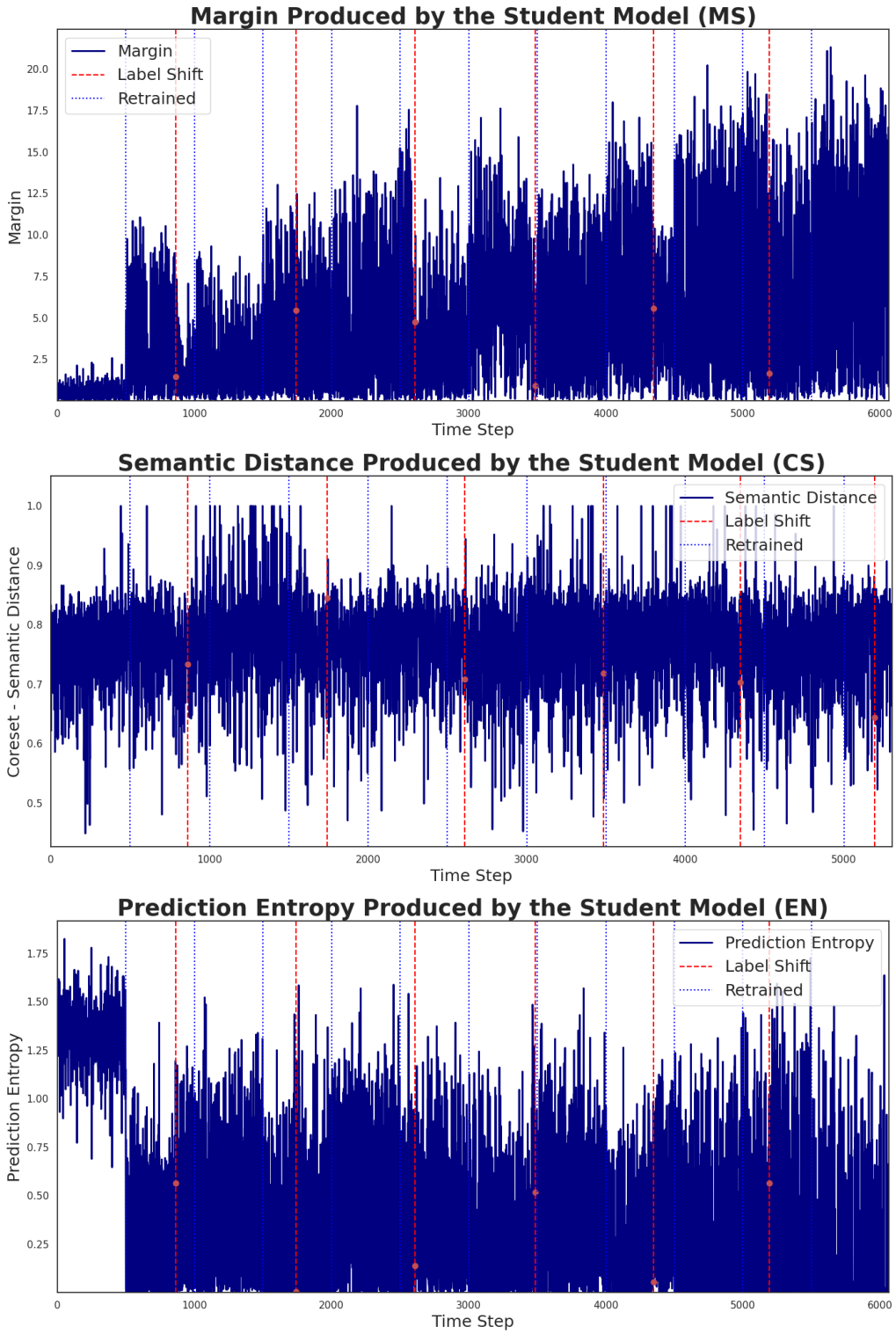


Figure C.1: Raw values of each metric (y-axis) for each time step (x-axis) as the experiment on ISEAR with label shifts progresses. The red vertical lines show where a label shift takes place, while the blue vertical lines where retraining occurs ($f = 500$).