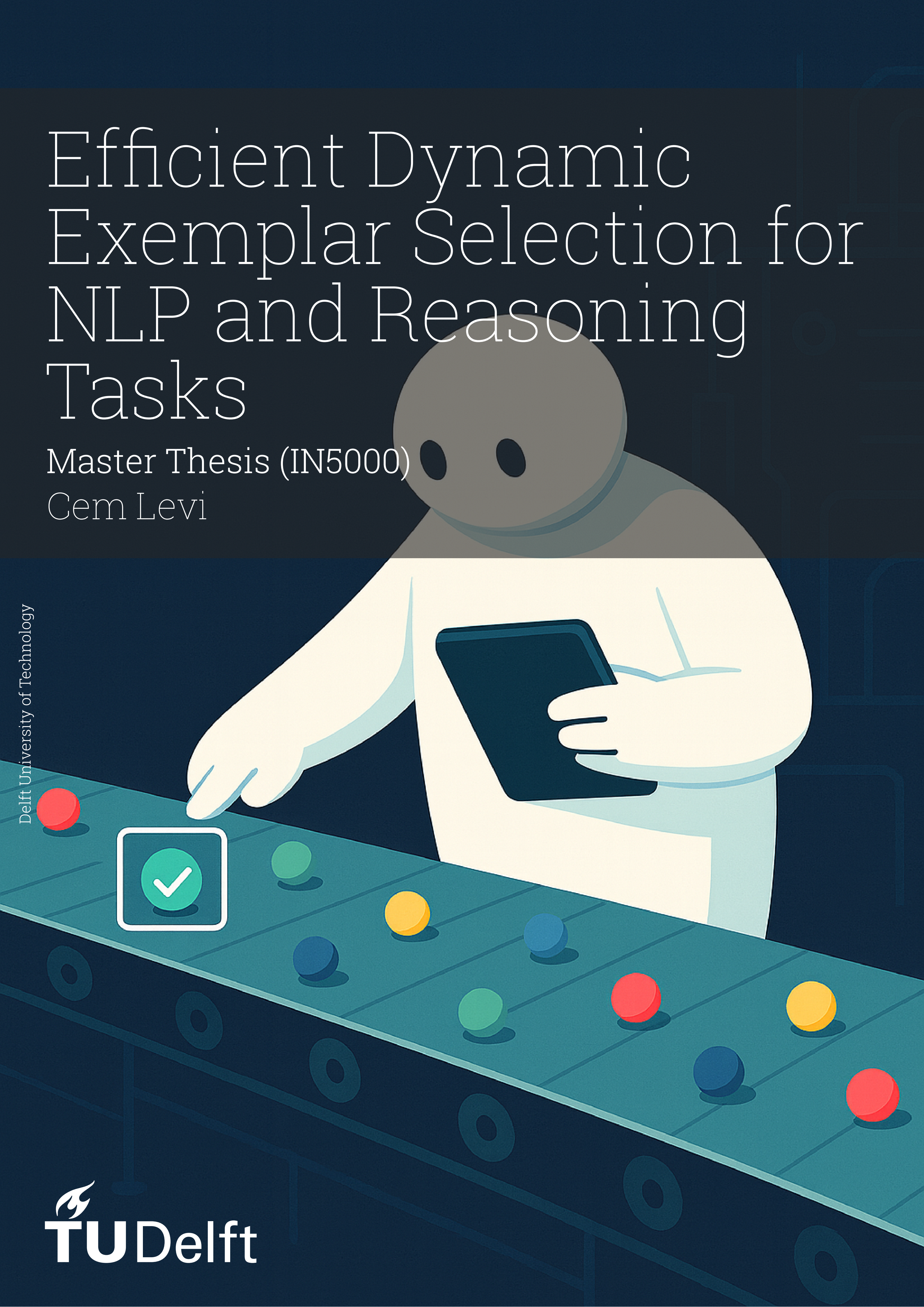
Efficient Dynamic Exemplar Selection for NLP and Reasoning Tasks

Master Thesis (IN5000)

Cem Levi



Efficient Dynamic Exemplar Selection for NLP and Reasoning Tasks

by

Cem Levi

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on 03/09/2025

Information:

Project duration:	December, 2024 - September, 2025
Program:	Master Computer Science
Track:	Artificial Intelligence
Student number:	4683625

Thesis Supervision:

Thesis Chair, Supervisor:	Prof. Dr. Avishek Anand
Daily co-supervisor:	Venktesh Viswanathan
Committee Member:	Dr. Jan van Gemert

Initial credits:

Cover:	Generated on `https://chatgpt.com/`
Style:	TU Delft Report Style, with modifications by Daan Zwaneveld

TU Delft

Abstract

Large Language Models (LLMs) have demonstrated impressive capabilities on wide range of tasks including tasks that entail complex reasoning. They also demonstrate the ability to adapt to new tasks without further training but with the help of exemplars demonstrating how to solve the complex reasoning task. This is due to emergent capabilities such as In-Context Learning (ICL) where the model learns the skills required for a task through the demonstration samples provided. These methods can be categorized as static methods where exemplars are selected offline or dynamic (Instance-level) where exemplars are selected on a per test query basis. Dynamic, instance-level exemplar selection has been shown to be more accurate than static, task-level methods, but it is hard to use in practice because it requires a lot of computing power. In order to mitigate this issue we propose a novel perspective for selecting exemplars by casting it into a ranking problem and use LTR models trained on automatically generated BERTScore-based relevance labels to assign utility to the exemplars. However, randomly selecting and receiving llm feedback for exemplars may not yield the best data to train LTR models. Hence, principled exploration of the exemplar space is critical to learn a selection policy offline that can be easily employed for dynamic exemplar selection during inference. We tackle with these problems in this paper by proposing CASE Rank, a novel non-linear gap-index bandit framework that cuts down on inference-time overhead by learning an exemplar utility estimator offline without hurting performance. CASE Rank solves these problems by combining a gap-index based bandit framework and LTR using PiRank, a lightweight neural ranking model, as a non-linear surrogate loss function within the bandit framework. CASE Rank is a bandit based selection approach to judiciously sample LLM feedback and learn offline policy using a differentiable sorting algorithm. This approach allows for quick and tailored selection of exemplars for each instance during inference. Experiments conducted on datasets such as GSM8K, AQUA-RAT, and WMT19 indicate that CASE Rank enhances reasoning performance compared to previous methods, while also substantially lowering computational requirements. Our results highlight that principled, efficient exemplar selection can be achieved through a combination of exploration strategies and learning-to-rank models tailored to LLM response behavior.

Contents

List of Figures

List of Tables

Nomenclature

Abbreviations

Abbreviation	Definition
NLP	Natural Language Processing
BERT	Bidirectional encoder representations from transformers
CASE	Challenger Arm Sampling for Exemplar selection
LTR	Learning to rank
GSM8K	Grade School Math 8K
AquaRat	Algebra Question Answering with Rationales
NDCG	Normalized Discounted Cumulative Gain
WMT19	Workshop on Statistical Machine Translation 2019
LLM	Large language model
ICL	In-context learning

Synonyms	Explanation
{exemplar, demonstration example, demo}	Task demonstration examples given with the prompt to the LLM for ICL. If examples include the input-rationale-output triplet, they are called exemplars. If they don't include the rationale, the task demonstrations are called demonstration examples. To not periphrase by specifying which of the definitions was meant all the time, the two definitions were used together.

1 Introduction

1.1. Background and Motivation

Large Language Models (LLMs) [42][6][54][48] have demonstrated remarkable capabilities in handling a wide range of problems and tasks through conditioning on a small number of demonstrations in the input prompt, a paradigm referred as in-context learning (ICL) [9][5]. ICL allows these models to learn from a limited number of examples without the need for parameter updates. ICL streamlines the adaptation of a general-purpose LLM to a specific task without the need for feature engineering or additional model training.
Despite [50][49] shows the ICL superior potential for various scenarios, the performance of ICL heavily relies on the careful example selection [28] [44][46][56]. Recent work [13] [38] has also shown the performance of ICL is sensitive to changes in prompts. Thus, the example selection plays a vital role in the behavior of ICL, and an important question in the field of in-context learning is how to improve the selection of in-context exemplars to enhance the performance of LLMs [29]. This has sparked interest in prompt retrieval, where, given a test instance, training examples are chosen for the prompt based on some similarity metric.
Researchers have proposed various heuristic and trial-and-error methods to address these limitations in example selection, including criteria such as entropy [32], influences, and uncertainty. However, only a few studies have taken a more principled approach [51]. Exemplars for In-Context Learning (ICL) can be categorized into two types: task-level, where a fixed set of exemplars representative of the task is chosen for inference, and instance-level, where exemplars are dynamically selected for each test instance. Static and dynamic methods for principled exemplar selection have been developed to address these challenges [53, 32]. Some methods require annotated data and the training of multiple models for exemplar selection. There has been principled approaches researched for task-level exemplar selection. some task-level exemplar selection methods here . Espicially bandit based approaches outrank the others. [40] proposes CASE and [39] proposes EXPLORA as the sampling algorithms for selecting exemplar subsets.

Both [39, 40] model the problem of selecting exemplars as top-m exemplar subset-selection problem. However, both algorithms are used in task-level exemplar selection. [40] uses a linear function based on sentence similarities between the in-context examples and validation examples as the surrogate model for the reward model for a multi-armed bandit-based exemplar selection scheme. This leads to the formulation of selection of top-m exemplar subsets as the problem of identification of top-m arms [43] in the stochastic linear bandit setting [2].

Despite the fact that dynamic exemplar selection often involves additional computational cost and overhead during inference time, task-level exemplar selection may not capture task variations or drift over time, also fixed exemplars may become suboptimal for new learning conditions. This can lead to subpar performance in ICL. Instance-level exemplar selection focuses on specific data points rather than entire tasks. Moreover, the reliability of a model's performance can vary when applied to different datasets. In more specialized domains, most mistakes tend to occur in rare or unusual cases that are often overlooked by methods that focus solely on the overall task. When examples are ranked individually, especially those the model finds confusing or tends to misclassify, the available budget for labeling or memory can be spent on the most important examples. Focusing resources in this way helps the model adjust more quickly as new data becomes available. It also makes it possible to add new or uncommon examples precisely where they are needed. In practice, this approach tends to improve accuracy and reliability on edge cases, at the cost of some extra computation. To the best of our knowledge there is not a principled approach to instance-level exemplar selection. Instance-level approaches often rely on statistical similarity [44, 51] and diversity-based strategies to identify the most appropriate exemplars for each test instance. These approaches do not model interactions between exemplars.

Our goal is to propose a principled and sample-efficient selection scheme for instance-level exemplar selection. Since the explore-exploit nature of bandit algorithms naturally fit the exploration of the large exemplar subsets space, we adapt a gap-index-based top-m arm selection bandit framework to perform instance-level (dynamic) exemplar selection. Moreover, while existing linear stochastic bandits assume the reward structure is locally linear, the structure of real-world rewards, such as LLM feedback, are non-linear. Hence, our goal is to also model this non-linearity through the surrogate design in the gap-index bandit framework, which to the **best of our knowledge** has not been explored. Additionally, following the approach adopted by CASE [40], we also adopt challenger arm sampling.

One direction that can be taken for a sample-efficient instance-level exemplar selection scheme is just to apply bandits online. This can be done by doing the whole bandit optimization online. However, that can lead to huge computational costs. Thus, the resulting scheme would be computationally inefficient. The reason for the huge computational costs is the number of LLM calls. With online optimization, the algorithm could make the same number of LLM calls original CASE made per instance. Offline optimization algorithm makes 1000-500 LLM calls in total. Most of these LLM calls come from arm evaluation. 1000-5000 calls for each test instance brings substantial computational cost. Therefore, it is determined that online inference with online optimization is too expensive an approach to pursue. For this reason, it was decided an exemplar selection approach that does online inference with offline optimization should be explored.

Another alternative scheme could be viewing exemplar selection as a learning to rank problem. Learning to Rank models can be used for offline optimization and online inference for exemplar selection as well. Thus, we propose modeling the exemplar selection problem as learning to rank problem. To the best of our knowledge there has not been any research on this approach.

Moreover, though many learning to rank algorithms exists [47, 36], naively applying these algorithms will not result in the optimal performance because they do not perform exploration of the space. Learning-to-rank systems typically do not explore the search space because they are trained on logged interactions from previously deployed rankers, which biases the data toward items that were already shown. Their standard pointwise, pairwise, or listwise losses optimize accuracy on observed labels and provide no incentive to try unobserved documents or alternative orderings. At serving time, inference is usually deterministic over a fixed candidate set from retrieval, so items outside that set are never considered, and low-scoring items are rarely surfaced.

Hence, we propose **CASERank** (**C**hallenger **A**rm **S**ampling for **E**xemplar **Rank**ing), which integrates a differentiable sorting or learning-to-rank module as a nonlinear surrogate within the bandit algorithm. This surrogate directly ranks exemplars, estimates their utilities, and captures the nonlinear structure of LLM feedback-based rewards. Consequently, we cast dynamic exemplar selection as a *nonlinear* stochastic bandit problem rather than a linear one. Also, we propose this in a learning to rank setting to rank exemplars. So once we collect sufficient data for the nonlinear surrogate using our bandit framework, this can be used train the network, which is trained offline. It can be used online at inference time for selection

1.2. Research Questions

To address the discussed shortcomings in instance-level exemplar selection, the following research questions are proposed:

(1) **RQ1:** Can online dynamic exemplar selection and LTR approaches lead to better performance in reasoning tasks compared to offline exemplar selection approaches?

(2) **RQ2:** How can instance-level exemplar selection be improved through efficient multi-armed bandit selection

(3) **RQ3:** How does offline policy exemplar compare to offline and online optimization in terms of efficiency-performance tradeoff?

1.3. Contribution

The contributions of this work, derived from addressing the proposed research questions, are as follows:

(1) **Novel way to generate relevance scores for A LTR datasets:** Using LTR models for exemplar ranking requires dedicated LTR data. As there are no publicly available ICL datasets designed for this purpose, we introduce our own approach.

(2) **A framework to model LTR problem as an exemplar selection problem:** We propose a method that frames the exemplar-selection problem as a learning-to-rank (LTR) task. This allows LTR models to be trained to rank exemplars and then deployed for online use.

(3) **Introduction of non-linear differentiable sorting surrogate in gap-index based bandit framework:** We introduce a dynamic exemplar-selection algorithm that uses an LTR model as a nonlinear surrogate for the bandit algorithm. This more accurately models the LLMs nonlinear rewards, while a bandit routine explores the search space and selects training data for the LTR model.

1.4. Thesis Outline

The report is laid out in the following way. Chapter 2 discusses current related work relevant to the research objectives of this study. Chapter 3 dives into the methodology used to collect relevance scores, LTR dataset feature generation, and CASE Rank algorithm. Chapter 3 explains the LTR generation pipeline, modelling a LTR model for exemplar selection, and using a non-linear surrogate for a gap-index-based bandit algorithm. Chapter 4 discusses the experiments conducted and the implementation-tion specification used to answer the research questions. Chapter 5 discusses the results obtained and relates them to the research questions and existing literature. Finally, Chapter 6 concludes the thesis while discussing limitations and directions for future improvements.

2 Background and Related Work

2.1. Exemplar Selection

Large Language Models (LLMs) [42][6][54][48] excel on many problems by using a small number of prompt-based demonstrations, a paradigm called in-context learning (ICL) [9][5], which enables learning from limited examples without any parameter updates. ICL makes it much easier for a general-purpose language model to take on new tasks. There is no need for complicated feature engineering or extra rounds of training. Instead, the model learns to handle a variety of tasks by picking up on helpful hints from just a few examples. ICL demonstrates flexibility, allowing LLMs to handle a wide variety of natural language processing (NLP) tasks, including text classification, question answering, and reasoning problems [50].
Despite [50][49] shows the ICL superior potential for various scenarios, the performance of ICL heavily relies on the careful example selection [28] [44][46][56]. Recent work [13] [38] has also shown the performance of ICL is sensitive to changes in prompts. Thus, exemplar selection plays a vital role in ICL behavior, and an important question in the field of in-context learning is how to improve the selection of in-context exemplars to enhance the performance of LLM [29]. This has sparked interest in prompt retrieval, where, given a test instance, training examples are chosen for the prompt based on some similarity metric.
Researchers have proposed various heuristic and trial-and-error methods to address these limitations in exemplar selection, including criteria such as entropy [32], influences, and uncertainty. However, only a few studies have taken a more principled approach [51]. Exemplars for In-Context Learning (ICL) can be categorized into two types: task-level, where a fixed set of exemplars representative of the task is chosen for inference, and instance-level, where exemplars are dynamically selected for each test instance. Static and dynamic methods for automatic exemplar selection have been developed to address these challenges [53, 32].

2.1.1. Task-level Exemplar Selection

Early techniques such as Chain-of-Thought (CoT) prompting put a strong emphasis on showing clear, step-by-step examples of reasoning. This approach has proven to be much more effective than older prompting methods. Chain-of-Thought (CoT) prompting breaks down difficult reasoning problems into smaller, more manageable steps. This method has led to top results on challenging benchmarks like GSM8K. Still, early versions of CoT depended on examples created by hand, which made it hard to scale and use widely in real-world situations.

To overcome these challenges, task-level exemplar selection methods have become simpler and more general. For example, Zero-Shot-CoT prompting, introduced by [24], uses a single, widely applicable prompt such as "Let's think step by step" that helps the model choose examples as it reasons through a problem, without needing to provide specific examples upfront. This approach enables the model to reason more effectively on various tasks, even when it is not provided with specific examples. It highlights how much can be gained by letting the model choose its own examples based on its own reasoning process.

Further refining exemplar selection, [17] proposed complexity-based prompting, which systematically selects exemplars according to the complexity of their reasoning chains. Their research demonstrated performance improvements over traditional selection methods, including manually-tuned and retrieval-based strategies. Similarly, [14] selects task-specific exemplars using uncertainty-based metrics to improve few-shot chain-of-thought prompting. Unlike prior approaches relying on fixed or randomly chosen exemplars, Active-Prompt introduces a principled selection mechanism that enhances reasoning performance across diverse tasks through targeted annotation of high-uncertainty queries.

2.1.2. Instance-level Exemplar Selection

Instance-level exemplar selection methods in in-context learning (ICL) focus on dynamically selecting the most informative exemplars for individual queries. [44] introduced Efficient Prompt Retrieval (EPR), a method that takes advantage of a pre-trained language model to label training examples as suitable or unsuitable prompts based on their predicted efficacy. Contrastive learning allows EPR to tell which prompts are helpful and which are not, making it easier to find the best examples and improve how well the system works on future tasks.

It is also emphasized that it is important to choose examples that are both different from each other and work well together. [53] highlighted the importance of selecting diverse exemplars, showing that LLMs benefit from varied reasoning processes demonstrated by multiple exemplars. Their maximal marginal relevance-based approach constructs exemplar sets that balance both relevance and diversity, improving performance across multiple reasoning tasks. Similarly, [51] introduced DQ-LoRe, a dual-query and dimensionality-reduction-based re-ranking method, which leverages low-rank approximations to improve the quality of exemplar selection. This method also makes the system more reliable, especially in the cases of distribution shifts.

Moreover, [31]introduced a method based on reinforcement learning (PROMPTPG) to dynamically select in-context examples via policy gradient learning. This method interacts directly with language models, optimizing the selection of prompts to maximize predictive performance. Such adaptive selection strategies not only outperform static or random methods but also reduce performance variability.

2.2. Multi-armed Bandit (MAB) Algorithms

Multi-armed bandit (MAB) algorithms are a fundamental framework for sequential decision-making under uncertainty. They are also widely used in resource allocation and optimization tasks. The classical MAB problem focuses on balancing exploration (learning about arm rewards) and exploitation (selecting the best-known arm) to maximize cumulative rewards [4]. Over time, extensions of the classical framework have been developed to address diverse scenarios, such as contextual bandits [25] and linear bandits [2], which incorporate feature vectors for arms to model their rewards.
In pure exploration settings, MAB algorithms are employed to identify the best or top-m arms instead of maximizing rewards. Algorithms like LUCB [23] and UGapE [19] focus on adaptive sampling to identify optimal arms efficiently.

2.2.1. Linear Bandits

Linear bandit algorithms address sequential decision-making problems where, at each time step, an action is selected from a set of available actions, and a stochastic reward is received, whose expected value is a linear function of the chosen action's features. The equation for the true reward Y_t received when selecting an action (arm) X_t at time t in the linear bandit setting is:

$$Y_t = \langle X_t, \theta^* \rangle + \eta_t \tag{2.1}$$

where X_t is the action vector chosen at time t, θ^* is the unknown parameter vector representing the true relationship between actions and expected rewards, and n_t is stochastic noise with zero mean. Optimism-in-the-face-of-uncertainty ideas were extended to linear models, yielding UCB-style methods such as OFUL with provably sub-linear regret [2]. Bayesian approaches, notably Thompson Sampling for linear payoffs, also achieve near-optimal problem-dependent guarantees [3]. More recently, *gap-index*-based algorithms have become central for pure exploration (fixed-confidence identification), using confidence-enhanced gap indices to concentrate sampling on the most informative comparisons. The linear best-arm identification problem was formulated by [45], and a fully adaptive gap-based algorithm

(LinGapE) was introduced by [52]. Subsequent work refined the sample complexity and moved toward problem-dependent optimality [15, 12]. To unify and extend these ideas, [43] developed a general framework for top-m identification in linear bandits, showing how exploiting feature structure and gap indices can further reduce sample complexity.

2.3. Bandit-Based Exemplar Selection

Bandit-based approaches cast exemplar selection as an exploration-exploitation problem, where the objective is to identify exemplar subsets that maximize model performance with minimal query overhead. Notably, [39] introduces a novel formulation of exemplar subset selection as a top-m arm selection problem in the stochastic linear bandit framework. Each arm corresponds to a candidate subset of exemplars, and rewards are inversely related to the validation loss induced by using that subset in the ICL prompt.
EXPLORA employs a sampling-based bandit algorithm that iteratively estimates a scoring function over subsetsmodeled as a linear function of similarity featuresto identify high-performing exemplar combinations while drastically reducing the number of expensive LLM calls. Complementary to this, [26] adopts a bandit-driven two-stage selection strategy, focusing first on individual exemplar informativeness and then selecting a balanced final set. Unlike EXPLORA, however, LENS depends on output probabilities and does not account for inter-exemplar interactions, limiting its applicability in black-box or subset-sensitive scenarios.

2.3.1. CASE

Challenger Arm Sampling for Exemplar Selection (CASE), proposed by [40], formulates exemplar subset selection as a top-m best-arm identification problem in a stochastic linear bandit setting. CASE manages an exponentially large search space of exemplar subsets by maintaining a shortlist of "challenger" subsets considered potential candidates for the top-performing exemplars. At each iteration, CASE selectively evaluates only one subset from the shortlist or the current top candidates, significantly reducing computational complexity and the number of language model evaluations [40]. Its overall algorithm can be found in Figure 2.1. This approach leverages a linear reward model to evaluate the quality of exemplar subsets by measuring sentence similarity between exemplars and validation examples. Empirical results demonstrate that CASE achieves up to sevenfold reductions in runtime and the number of LLM calls without sacrificing performance compared to state-of-the-art exemplar selection methods like EXPLORA and LENS.

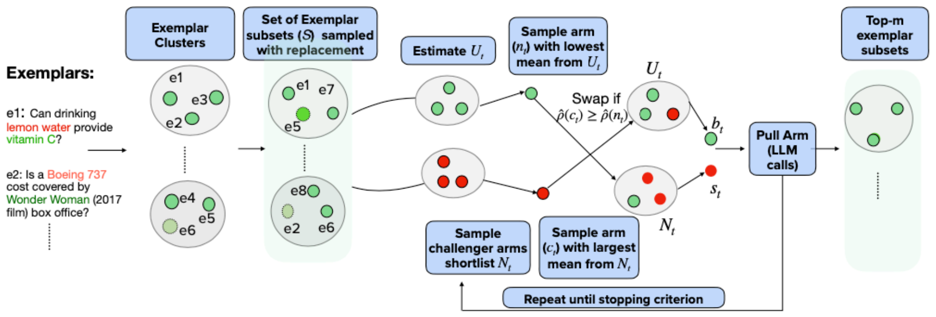

Exemplars:
e1: Can drinking lemon water provide vitamin C?
e2: Is a Boeing 737 cost covered by Wonder Woman (2017 film) box office?
Exemplar Clusters
e1
e3
e2
e4
e5
e6
Set of Exemplar subsets (S) sampled with replacement
e1
e7
e5
e8
e2
e6
Estimate U_t
Sample arm (n_t) with lowest mean from U_t
Swap if $\hat{\rho}(c_t) \geq \hat{\rho}(n_t)$
U_t
b_t
s_t
N_t
Sample challenger arms shortlist N_t
Sample arm (c_t) with largest mean from N_t
Pull Arm (LLM calls)
Top-m exemplar subsets
Repeat until stopping criterion

Figure 2.1: Overview of CASE[40] pipeline

2.4. Learning to Rank

Learning to Rank (LTR) refers to machine learning methods designed to help build ranking models for information retrieval and recommendation systems. These models work by arranging items so that the most relevant results appear at the top, based on what the user is looking for or prefers. Over the past two decades, substantial research has evolved in this domain, leading to significant advancements in both theoretical and practical aspects [30]. Each example in a learning-to-rank dataset is a query-document pair and contains: a query ID, a list of indexed feature values, and a relevance label (usually an integer: 0 = not relevant, higher = more relevant). These labels can be obtained through manual annotation,

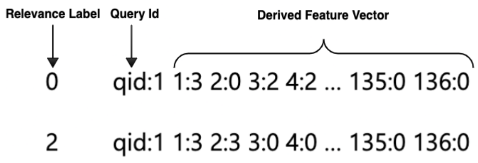

Relevance Label
Query Id
Derived Feature Vector
0 qid:1 1:3 2:0 3:2 4:2 ... 135:0 136:0
2 qid:1 1:3 2:3 3:0 4:0 ... 135:0 136:0

Figure 2.2: Example LTR dataset

implicit user feedback (like clicks or dwell time), or heuristic/synthetic methods. An example LTR dataset can be seen in Figure 2.2.

2.4.1. Surrogate Objectives for LTR

Initially, LTR methods were categorized broadly into pointwise, pairwise, and listwise approaches. The pointwise approaches view ranking as a regression or classification task by individually predicting the relevance of each item without explicitly considering item relationships [18]. Pairwise methods, on the other hand, learn to optimize rankings by minimizing errors associated with pairs of documents, exemplified by algorithms such as RankNet [7]. However, pairwise models often overlook the holistic ranking structure, leading to the development of listwise methods that directly optimize the entire ranked list according to metrics such as NDCG (Normalized Discounted Cumulative Gain) or MAP (Mean Average Precision).
Among notable listwise methods, LambdaRank and LambdaMART emerged as prominent models, leveraging gradient boosting frameworks and approximating direct optimization of ranking measures through lambda gradients [7]. LambdaMART, in particular, has been extensively utilized in web search and recommender systems due to its performance and ability to generalize across diverse datasets [8]. In recent years, advancements in deep neural networks have significantly influenced LTR methods, leading to novel deep learning architectures tailored explicitly for ranking tasks. These neural ranking models, such as DRMM (Deep Relevance Matching Model) and BERT-based rankers, leverage semantic matching and attention mechanisms to enhance the representation of queries and documents, improving ranking quality [21]; [34].

2.4.2. PiRank

PiRank addresses the central challenge in learning-to-rank (LTR) by introducing a scalable and differentiable surrogate loss function designed to closely approximate non-differentiable ranking metrics like NDCG and ARP. Traditional LTR methods rely heavily on surrogate losses loosely connected to true ranking metrics, limiting their effectiveness and scalability. In contrast, PiRank utilizes a temperature-controlled differentiable relaxation of the sorting operator, allowing direct optimization of the exact ranking metrics as the relaxation temperature approaches zero. Formally, the PiRank surrogate loss function for NDCG is defined as follows:

$$\mathcal{L}_{\text{PiRank-NDCG}} = 1 - \frac{\mathrm{D\hat{C}G}(y, \hat{y}, \tau)}{\mathrm{DCG}(y, \pi^*)} \tag{2.2}$$

where $\mathrm{D\hat{C}G}$ is the relaxed Discounted Cumulative Gain calculated via a differentiable sorting operator controlled by temperature parameter τ, y represents the ground-truth relevance scores, denotes predicted scores, and π^* corresponds to the optimal ranking obtained by sorting ground-truth scores in descending order. PiRank uses a divide-and-conquer approach similar to the way merge sort works. This clever strategy makes the training process much more efficient and manageable, even when working with very large datasets.

2.4.3. AllRank

The self-attentive ranker introduced in AllRank is an advancement in neural learning-to-rank by explicitly modeling inter-document dependencies using self-attention. Unlike traditional ranking models that score each document independently, the self-attentive ranker processes the entire list of documents jointly. Its architecture is based on stacked multi-head self-attention layers. Inspired by the Transformer encoder, each documents representation is updated in context with other documents in the same ranking list. This allows the model to capture relative relevance and positional relationships effectively. The output is passed through a feed-forward layer to compute final ranking scores. The model architecture can be seen in Figure 2.3.

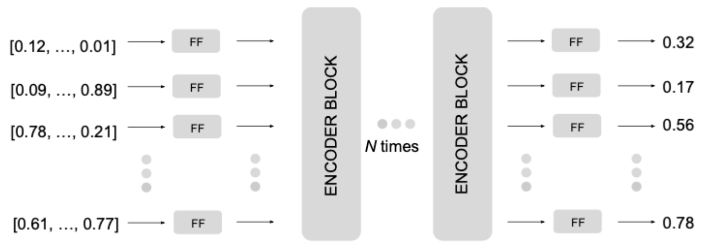


Figure 2.3: Schematic of the proposed model architecture by [37]. Input is a list of real-valued vectors. Output is the list of real-valued scores.

2.4.4. NeuralNDCG

In recent years, optimizing Learning to Rank (LTR) systems using ranking metrics such as Normalized Discounted Cumulative Gain (NDCG) has been hindered by the non-differentiability of these metrics, which depend on sorting operations. The *NeuralNDCG* framework addresses this by introducing a differentiable surrogate for NDCG using a relaxed sorting operator called *NeuralSort.* Instead of relying on discrete permutation matrices, NeuralNDCG uses a continuous approximation of the sorting operator, enabling gradient-based optimization of ranking metrics. The surrogate loss function for NDCG is defined as:

$$\text{NeuralNDCG}_k^{(\tau)}(s, y) = \frac{1}{\text{maxDCG}_k} \sum_{j=1}^{k} \left(\text{scale}(\hat{P}) \cdot g(y)\right)_j \cdot d(j) \tag{2.3}$$

where $\hat{P}$ is the relaxed permutation matrix obtained via NeuralSort with temperature parameter τ,$g(y)$ is the gain function applied to the relevance vector y, $d(j) = \frac{1}{\log_2(j+1)}$ is the discount function for position j,scale($\hat{P}$) denotes a column-wise normalization of $\hat{P}$ via Sinkhorn scaling,maxDCG$_k$ is the maximum achievable DCG at rank k. To solve the problem of distortion introduced by non-column-stochasticity in the relaxed permutation matrix, NeuralNDCG applies Sinkhorn scaling to enforce doubly stochastic constraints. Empirical evaluations demonstrate that NeuralNDCG surpasses existing methods such as ApproxNDCG.

3 Method

3.1. Problem setup

In-context learning (ICL) enables large language models (LLMs) to quickly gain proficiency in a specific task by using only a few examples, without modifying the model's parameters. In this approach, the LLM is given a prompt that includes an *input-rationale-output* triplet, called *exemplars*, which guide the model in performing the task. In the case that the task is not a reasoning task, such as translation, the prompt only includes *input-output* couple and is called a demonstration example instead of an exemplar.

Because processing long contexts is financially and computationally expensive, it is often impractical to include all n training examples. Instead, exemplar selection methods choose a small subset that maximizes overall accuracy.

Formally, let $\mathcal{X} = \{x_i, z_i, y_i\}_{i=1}^{n}$ represent the full set of n training samples (potential exemplars), and let x_{test} denote the test input. The goal is to predict the test output y_{test}.

Let $S \subseteq \mathcal{X}$ be the subset of k exemplars selected for predicting x_{test}. The prompt P is then constructed as

$$P = [S, x_{test}] = [(x_{i_1}, z_{i_1}, y_{i_1}), \ldots, (x_{i_k}, z_{i_k}, y_{i_k}), x_{test}].$$

The ICL process involves two main steps: (1) a response generator f, and (2) a post-processing step (decoding) δ. The response generator produces multiple responses r from the distribution p_{LLM}, and the post-processing step δ is applied to the output of $f(P)$ to derive the final prediction for y_{test}:

$$\hat{y}_{test} = \delta(f([S, x_{test}])) \quad \text{where} \quad f(P) = \mathcal{G}(p_{LLM}(r|P)).$$

In the instance-based demonstration example subset selection(DESS) for ICL setup, the objective is to dynamically determine, for each test instance u_{test}, an optimal subset of exemplars from a larger candidate pool that maximizes the performance of in-context learning with large language models (LLMs). Given a task with a set $X = \{(u_i, v_i)\}_{i=1}^{n}$ of potential exemplars—each comprising an input u_i, an output v_i, and optionally a rationale—the method aims to select, for every u_{test}, a subset $S_{\text{test}} \subset X$ of fixed size k. The selection is governed by an instance-specific prompt generator π, which uses the similarity, diversity, or other relevance criteria between u_{test} and candidate exemplars to form the prompt $P = [S_{\text{test}}, u_{\text{test}}]$. The LLM f processes this prompt to produce a response, which is then post-processed by δ to yield the predicted output $\hat{v}_{\text{test}}$. The quality of the selection strategy is assessed on a validation set V by computing the average accuracy $A(\pi(\cdot), V)$ across all test instances, where A evaluates how well each dynamically chosen subset S_{test} supports correct prediction for its associated u_{test}. The instance-based DESS problem can thus be expressed as learning a selection policy π^* that maximizes

$$\frac{1}{|V|} \sum_{(u',v') \in V} \mathbf{1}\big(v' = \delta(f(\pi(X, u')))\big).$$

3.2. LTR Dataset for Exemplar Selection

The DESS problem described above can be solved by modeling LTR framework as a demonstration subset selection problem. Let $\mathcal{Q}$ denote a set of queries. For each query $q \in \mathcal{Q}$ with candidate documents $\mathcal{D}_q = \{d_{q,1}, \ldots, d_{q,n_q}\}$, Each candidate $d_{q,j}$ is represented by a feature vector $\mathbf{x}_{q,j} \in \mathbb{R}^m$, and is paired with a relevance label $r_{q,j} \in \{0, 1, \ldots, G\}$ (graded relevance). A learning-to-rank model $f_\theta : \mathbb{R}^m \to \mathbb{R}$ maps features to a scalar score

$$s_{q,j} \; = \; f_\theta(\mathbf{x}_{q,j}), \tag{3.1}$$

and induces a predicted ranking $\hat{\pi}_q$ by sorting $\{s_{q,1}, \ldots, s_{q,n_q}\}$ in descending order. For convenience, define the per-query design matrix $X_q \in \mathbb{R}^{n_q \times m}$ whose j-th row is $\mathbf{x}_{q,j}^\top$, and the label vector $\mathbf{y}_q = (y_{q,1}, \ldots, y_{q,n_q})$. The objective is to learn parameters θ that maximize ranking quality on unseen queries according to standard IR metrics (e.g., NDCG@k, MAP, MRR). In an exemplar selection setting, the query is replaced with a validation sample, the ground truth is replaced with a relevance score, and the document is replaced with an exemplar subset.

$$\begin{aligned} X = \{(u_i, v_i)\}_{i=1}^n \to \mathcal{D}_q = \{d_{q,1}, \ldots, d_{q,n_q}\}, y_{q,j} \in \{0, 1, \ldots, R\} \\ u_{val} \to q \in \mathcal{Q} \end{aligned} \tag{3.2}$$

Collecting relevance scores first is required for generating an LTR dataset for exemplar selection. Since it is not possible to do either human annotations or collect click data in this setting, an alternative approach is required. The following section presents an alternative novel way to collect relevance scores for the generation of LTR datasets.

3.2.1. Relevance Score Collection

In order to collect relevance scores, first k-means clustering is applied on the training data. Then, to form a demonstration example subset, k different demonstration examples are randomly sampled from each cluster. M subsets are sampled to be used for each query. The same M subsets are used as documents for each of the queries. Relevance scores are obtained by inputting each demonstration subset as a k-shot demonstration example to an LLM and then obtaining the LLM's feedback on the question of the validation sample. The process so far can be formularized, following [40]'s notation, as:

$$r \sim f\,(P)), \quad \text{where } P = [S, u_{\text{val}}] \tag{3.3}$$

where

- $f(P)$: Response generator function representing LLM inference. It samples a response r from the LLM distribution:
$$f(P) \sim P_{\text{LLM}}(r \mid P)$$
- P: The full prompt given to the LLM.
- $S = \{(u_1, v_1), (u_2, v_2), \ldots, (u_5, v_5)\}$: One of M randomly sampled subsets of demonstration exemplars.
- u_{val}: The validation input example for which a response is desired.
- $[S, u_{\text{val}}]$: Concatenation of exemplars in S with the validation input u_{val} to form the final prompt.

After the response is post processed, $\hat{v}_{\text{val}}$ is obtained. The final form of the LLM inference equation looks like this:

$$\hat{v}_{\text{val}} = \delta\,(f\,(P)) \tag{3.4}$$

where $\hat{v}_{\text{val}}$ is the predicted output for the validation input u_{val}, and δ is post-processing function applied to the LLM output (e.g., regex extraction).
Relevance scores are collected by evaluating the post-processed answer. F1 BertScore from BertScore[55] is used for evaluation. The F1 score between a post-processed answer $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$ and a ground truth $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$ where each token is represented as a contextual BERT embedding, is computed as:

$$P = \frac{1}{|\mathbf{x}|} \sum_{x_i \in \mathbf{x}} \max_{y_j \in \mathbf{y}} \cos(x_i, y_j) \tag{3.5}$$

$$R = \frac{1}{|\mathbf{y}|} \sum_{y_j \in \mathbf{y}} \max_{x_i \in \mathbf{x}} \cos(y_j, x_i) \tag{3.6}$$

$$\mathrm{F1}(\mathbf{x}, \mathbf{y}) = \frac{2 \cdot P \cdot R}{P + R} \tag{3.7}$$

where

- $\mathbf{x} = \{x_1, x_2, \ldots, x_m\}$: Post-processed answer containing m tokens.
- $\mathbf{y} = \{y_1, y_2, \ldots, y_n\}$: Ground truth containing n tokens.
- $x_i, y_j \in \mathbb{R}^d$: Contextual embedding vectors of tokens x_i and y_j obtained from a pre-trained BERT, where d is the embedding dimension.
- $\cos(x_i, y_j)$: Cosine similarity between the embeddings of token x_i and y_j.
- P: BERTScore Precision for each token x_i in the answer, the maximum similarity to any token y_j in the ground truth, averaged over all x_i.
- R: BERTScore Recall for each token y_j in the ground truth, the maximum similarity to any token x_i in the answer, averaged over all y_j.

After obtaining F1 BertScore for each dataset instance F1 score is multiplied by 10 and then rounded to the nearest integer to attain a dataset with range $y \in \{0, 1, \cdots, R\}$:

$$\text{Relevance score} = \lfloor (F1 \times R) \rceil \tag{3.8}$$

This final calculated value is the relevance score used for an instance of an LTR data set. The overall relevance score collection process can be seen in Figure 3.1.

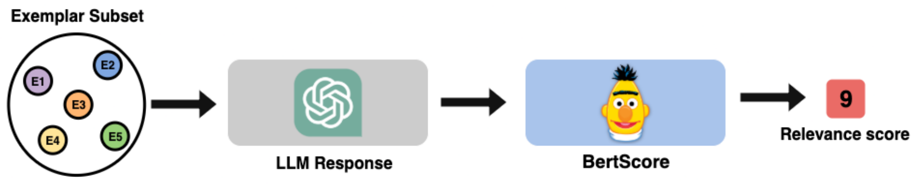


Figure 3.1: Relevance score collection pipeline used in LTR dataset generation for exemplar ranking.

3.2.2. Feature Derivation

Each demonstration example subset needs to have a single set of features for an LTR dataset. This is because each subset formed from k-different demonstration samples corresponds to a single document. Also, as can be seen from Figure 2.2, a dataset instance includes the features of query too. This means that features for a single LTR dataset instance include document features as well. So, a k-dimensional vector along with a single-dimensional one must be projected into a single dimension vector to fit into an LTR dataset structure. Feature derivation is performed to obtain features with a single dimension. A simple feature derivation method was chosen to first determine whether this approach would work: Let $\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \cdots, \mathbf{x}_k \in \mathbb{R}^{1 \times h}$ be k row vectors representing an element of a demonstration subset. h is an array dimension. Row vectors are stacked into a feature matrix:

$$A = \begin{bmatrix} \mathbf{x}_1 \\ \mathbf{x}_2 \\ \mathbf{x}_3 \\ \vdots \\ \mathbf{x}_k \end{bmatrix} \in \mathbb{R}^{k \times h} \tag{3.9}$$

where $h = 768$ and $\mathbf{x_k}$ is demonstration example k of the demonstration subset. Then, the column-wise mean of the matrix is calculated:

$$\bar{\mathbf{x}} = \frac{1}{k}\sum_{i=1}^{k}\mathbf{x}_i \tag{3.10}$$

equivalently,

$$\frac{1}{k}\sum_{i=1}^{k} A_{i,:} = \frac{1}{k}\,\mathbf{1}_k^\top A \;\in\; \mathbb{R}^{1\times h}. \tag{3.11}$$

Finally, the mean of the feature vectors of the demonstration subset is added to the validation sample, used as the query, embedding is obtained from SentenceBert. The sum is the final feature vector in LTR dataset.

$$x_f = \bar{\mathbf{x}} + \mathbf{x}_{val} \tag{3.12}$$

3.3. CASE Rank

There are several shortcomings of the previously mentioned methods. Firstly, even though it has a performance increase, Dynamic CASE is a very expensive method. It's performance is also not high enough to justify its cost. Since CASE models the rewards linearly, it uses a linear loss function. This approach, even though effective enough in a cost-effective static setting, puts a ceiling on performance. This is because, in reality, the rewards are non-linear since they are LLM responses. Since LLMs are non-linear models, their responses are also non-linear. Thus, modeling the rewards non-linearly may increase model performance. To model rewards as non-linear, a non-linear loss function must be used. However, LTR approaches also have a performance limit because of the lack of enough training data and their sampling approach. Data generation is expensive because it takes an LLM call to collect each relevance score. On the other hand, it can be said that randomly sampling training data does not work well especially in similar amounts that used for the experiments in section 5.1. Therefore, a more sophisticated approach needs to be taken to pick training data for an LTR model.
Therefore, we propose a novel bandit-based algorithm for instance-based exemplar selection. Algorithm 1 describes the proposed bandit-based demonstration example selection method CASE Rank. CASE Rank corrects the shortcomings of these two approaches by first using a non-linear loss to model LLM-based reward, which has a non-linear structure. Thus, CASE Rank uses a non-linear surrogate in a gap-index-based bandit algorithmic framework. It specifically uses a differentiable sorting / LTR model as a surrogate, which models the utility of the exemplar subsets through a ranking function. Secondly, CASE Rank adapts the gap-index based bandit algorithm from [40] to pick and collect additional training data for the LTR model. Then, a trained LTR model is used for instance-level inference. Hence, CASE Rank does offline training with online inference.The pipeline can be seen in Figure 3.2.

3.3.1. Updating Means

The DESS problem can be posed as the top-m arm selection problem in stochastic non-linear bandits. In the current setting, the arms correspond to the exemplar subsets S, and actions(pulling an arm) correspond to selecting an exemplar subset. Rewards are LLM response accuracies, which we try to maximize cumulatively. The goal is to select the most rewarding option, which is the best performing exemplar subset
A multi-armed bandit-based approach is taken to solve the DESS problem defined above. Let $a \in \{0,1\}^n$ with $\|a\|_0 = k$ represent the onehot indicator of a size-k exemplar subset drawn from the candidate pool $\mathcal{X}$ (where $|\mathcal{X}| = n$). A multi-armed bandit whose arms are defined with arms which are the subsets $a_i \in \mathcal{S}(\mathcal{X})$ for $i \in \{1,\ldots,|\mathcal{S}(\mathcal{X})|\}$. The number of arms grows exponentially in both k and n. The reward of an arm a is the accuracy $A(\pi(a),\mathcal{V})$, where π denotes the prompt induced by the subset encoded by a, and $\mathcal{V}$ is the validation set.

$$\mathrm{Sim}_{ij} = \frac{\phi(u_i)^\top \phi(u'_j)}{\|\phi(u_i)\|\,\|\phi(u'_j)\|}, \tag{3.13}$$

Algorithm 1 CASE Rank

```
Require: $\mathcal{X}$: training exemplars, $k$: prompt size, $\mathcal{S}$: all $k$-subsets of $\mathcal{X}$, $a \in \mathcal{S}$: an arm, $\mathcal{D}$: training
    dataset, $\mathcal{G}$: $\mathcal{D}$ generator function, $N$: query size, $\mathcal{F}_\theta$: LTR model, $\theta$: model parameters, $\eta$: learning
    rate
Ensure: $U_T$: top-$m$ arms with highest reward
 1: Define: $U_t$: set of currently estimated top-m arms.
 2:         $N_t$: set of currently estimated next best-m arms.
 3:         $b_t$: the most ambiguous arm from $U_t$.
 4:         $s_t$: the most ambiguous sampled arm from $N_t$.
 5: Initialize $U_0 \leftarrow$ set of random $m$ arms from $\mathcal{S}$, $t \leftarrow 1$, $\vec{\alpha}_1 \sim \mathcal{N}(0, 1)$
 6: Initialize $\mathcal{D} \leftarrow$ set of random $M$ subsets from $\mathcal{G}(\mathcal{X})$
 7: while $B_t(s_t, b_t) \leq \epsilon$ do
 8:     $n_t \leftarrow \arg\min_{a \in U_{t-1}} \hat{\rho}_t(a)$
 9:     $c_t \leftarrow \arg\max_{a \in N_{t-1}} \hat{\rho}_t(a)$
10:     if $\hat{\rho}_t(c_t) \geq \hat{\rho}_t(n_t)$ then
11:         Swap $n_t$ and $c_t$ between $U_{t-1}$ and $N_{t-1}$
12:         $U_t, N_t \leftarrow$ updated sets
13:     else
14:         $U_t, N_t \leftarrow U_{t-1}, N_{t-1}$
15:     end if
16:     $M_t \leftarrow s' \sim_{m'} (U_t \cup N_{t-1})^c$                 ▷ Randomly sample
17:     $N_t \leftarrow \text{top}_{m'}(M_t \cup N_{t-1}; \hat{\rho}_t)$                 ▷ Update $N_t$
18:     Compute the revised most ambiguous arms for convergence
19:     $b_{t+1} \leftarrow \arg\max_{b \in U_t} \max_{a \in N_t} B_t(a, b)$
20:     $s_{t+1} \leftarrow \arg\max_{s \in N_t} B_t(s, b_{t+1})$
21:     Pull selected arm, receive reward, and update parameters
22:     $a_{t+1} \leftarrow \text{selection\_rule}(U_t, N_t)$
23:     $r_{t+1} \leftarrow A(\pi(a(t)), \mathcal{V})$                 ▷ LLM inference call
24:     $\mathcal{D} \leftarrow \mathcal{D} \cup \{\mathcal{G}(r_{t+1}, a_{t+1}, \mathcal{X})\}$
25:     for $\forall x \in \mathcal{D}$ do
26:         $\hat{y}_i \leftarrow \mathcal{F}_\theta(x_i)$                 ▷ Forward pass
27:         $\ell_i \leftarrow \mathcal{L}(\hat{y}_i, y_i)$                 ▷ Compute loss
28:         $\theta \leftarrow \theta - \eta \nabla_\theta \ell_i$                 ▷ Gradient step
29:     end for
30:     for $\forall a \in S$ do
31:         for $i = 1, 2, \ldots, N$ do
32:             $\hat{y}_{a_i} = \mathcal{F}_\theta(x_{a_i})$
33:         end for
34:     end for
35:     for $\forall a \in \mathcal{S}$ do
36:         $\hat{\rho}_t(a) \leftarrow \frac{\sum_{i=1}^{N} \hat{y_{a}}_i(t)}{N}$
37:     end for
38:     $t \leftarrow t + 1$
39: end while
40: Output: $\mathcal{F}_{\theta^*}$ Trained PiRank model
```

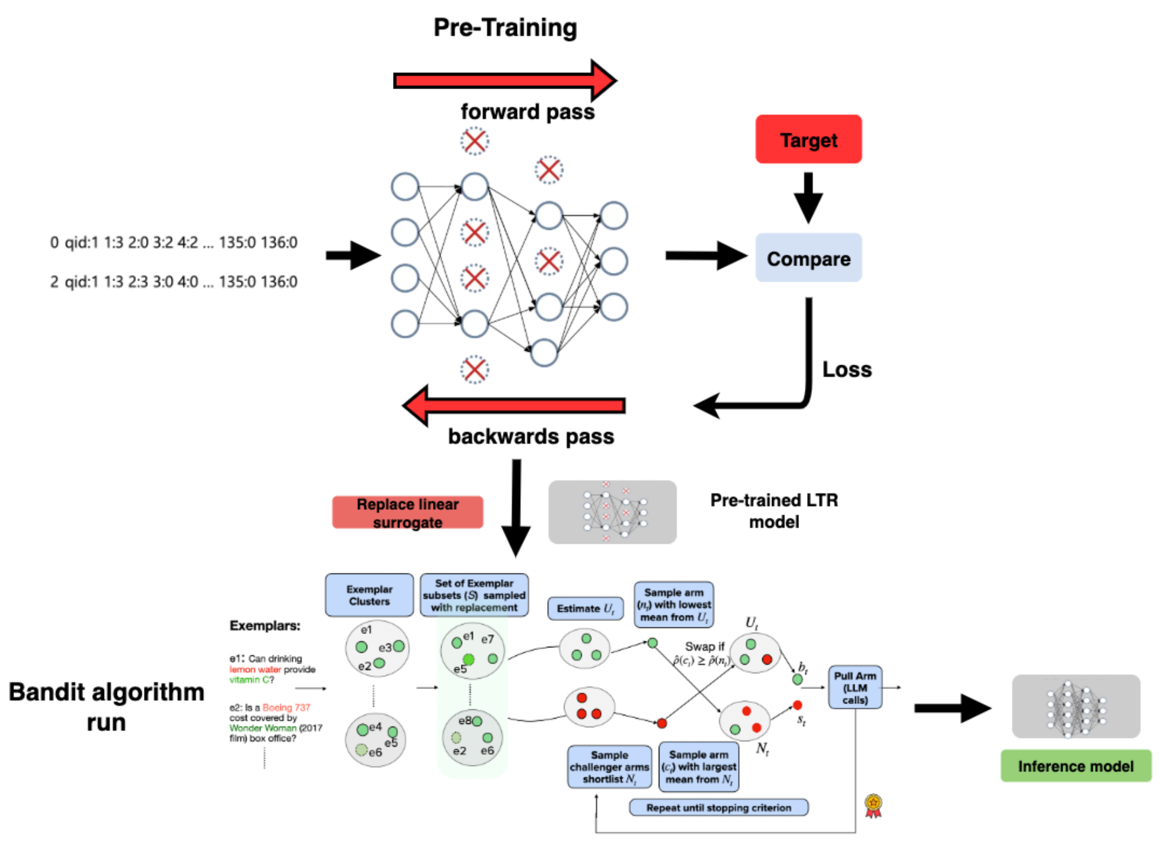

Pre-Training
forward pass
Target
0 qid:1 1:3 2:0 3:2 4:2 ... 135:0 136:0
2 qid:1 1:3 2:3 3:0 4:0 ... 135:0 136:0
Compare
Loss
backwards pass
Replace linear surrogate
Pre-trained LTR model
Bandit algorithm run
Exemplars:
e1: Can drinking lemon water provide vitamin C?
e2: Is a Boeing 737 cost covered by Wonder Woman (2017 film) box office?
Exemplar Clusters
Set of Exemplar subsets (S) sampled with replacement
e1
e2
e3
e4
e5
e6
e7
e8
Estimate U_t
Sample arm (n_t) with lowest mean from U_t
U_t
Swap if $\hat{\rho}(c_t) \geq \hat{\rho}(n_t)$
b_t
s_t
N_t
Pull Arm (LLM calls)
Sample challenger arms shortlist N_t
Sample arm (c_t) with largest mean from N_t
Repeat until stopping criterion
Inference model

Figure 3.2: Complete pipeline of proposed CASE Rank algorithm.

where $\phi(u)$ is the sentence-embedding produced by a pre-trained transformer model (e.g., SENTENCE-BERT). where $\phi(u)$ is the sentence-embedding produced by a pre-trained transformer model (e.g., SENTENCEBERT). CASE Rank also keeps a set $\mathcal{D}$ for training. Whenever an arm is pulled in the CASE Rank algorithm, the data of the arm is put into the set $\mathcal{D}$(line 3 in Algorithm 1). The set $\mathcal{D}$ contains all the data that is going to be used in the LTR model training.

$$\mathcal{D}' = \mathcal{D} \cup \{\mathcal{G}(\mathbf{a})\}, \quad \text{where } \mathcal{D} = \{(u_1, v1), (u_2, v_2), \ldots, (u_n, v_n)\} \tag{3.14}$$

The observed reward when pulling an arm is defined in [40]:

$$\hat{\rho}(a; \alpha) = \rho(a; \alpha) + \eta \tag{3.15}$$

where:

- $\hat{\rho}(a)$ is the observed reward.
- $\rho(a)$ is the expected reward.
- η is zero-mean sub-Gaussian noise.

$\rho(a)$ is defined in [40] as:

$$\rho(a; \alpha) = \alpha^\top x_a \tag{3.16}$$

where

- a: An **arm**, corresponding to a subset of k exemplars selected from a pool X of n total exemplars. It is represented as a binary vector of length n, where $a(i) = 1$ if exemplar i is included, and 0 otherwise.

- $\alpha \in \mathbb{R}^n$: A vector of **learnable coefficients** that represent the importance of each exemplar in contributing to reward.
- $x_a \in \mathbb{R}^n$: The **feature vector** for arm a, constructed using similarity scores between exemplars and validation inputs. Specifically,

$$x_a(i) = a(i) \cdot \left(\frac{1}{n'} \sum_{j=1}^{n'} \mathrm{Sim}_{ij} \right)$$

  where Sim_{ij} is the cosine similarity between the embeddings of exemplar u_i and validation input u'_j.
- η: A **zero-mean sub-Gaussian noise** term, capturing the randomness and uncertainty inherent in LLM outputs.

In CASE Rank, the expected reward is defined as:

$$\rho(a, \mathbf{W}) = f(\mathcal{G}(a)) \tag{3.17}$$

where, $\mathcal{G}(\cdot)$ is the LTR data generator function and $f(\cdot)$ is the LTR model, defined as:

$$f(\mathbf{x}) = \mathbf{W}^{(4)} \,\mathrm{ReLU}\Big(\mathbf{W}^{(3)} \,\mathrm{ReLU}\big(\mathbf{W}^{(2)} \,\mathrm{ReLU}(\mathbf{W}^{(1)}\mathbf{a} + \mathbf{b}^{(1)}) + \mathbf{b}^{(2)}\big) + \mathbf{b}^{(3)}\Big) + \mathbf{b}^{(4)}. \tag{3.18}$$

where:

- $\mathbf{x} \in \mathbb{R}^n$: input vector
- $\mathbf{W}^{(1)} \in \mathbb{R}^{h_1 \times n}$ and $\mathbf{b}^{(1)} \in \mathbb{R}^{h_1}$: weights and bias for layer 1
- $\mathbf{W}^{(2)} \in \mathbb{R}^{h_2 \times h_1}$ and $\mathbf{b}^{(2)} \in \mathbb{R}^{h_2}$: weights and bias for layer 2
- $\mathbf{W}^{(3)} \in \mathbb{R}^{h_3 \times h_2}$ and $\mathbf{b}^{(3)} \in \mathbb{R}^{h_3}$: weights and bias for layer 3
- $\mathbf{W}^{(4)} \in \mathbb{R}^{m \times h_3}$ and $\mathbf{b}^{(4)} \in \mathbb{R}^{m}$: weights and bias for the output layer
- $\mathrm{ReLU}(z) = \max(0, z)$: activation function applied element-wise

Therefore, equation (3.3.1) can be considered the non-linear surrogate function.

The MAB learning algorithm iteratively estimates the unknown coefficients $\mathbf{W}^l$, such that the empirical estimate of reward $\hat{\rho}(a) \equiv \hat{\rho}(a, \mathbf{W}) \approx \rho(a, \mathbf{W})$ for the high scoring arms a and a large time index $t > \tau$. Iterative estimation is done by training and evaluating the LTR model.

Originally in CASE, the reward is used in the model update through the vector b_t in the ridge regression equation. This observed reward from Equation (3.16) is used in the computation of the vector b_t:

$$b_t = \sum_{a \in \mathcal{A}_t} N_a x_a \hat{\rho}(a) \tag{3.19}$$

which is then used in the model update:

$$\hat{\alpha}_t = \Sigma_t^{-1} b_t \tag{3.20}$$

with the design matrix:

$$\Sigma_t = \lambda I + \sum_{a \in \mathcal{A}_t} N_a x_a x_a^\top \tag{3.21}$$

In lines 22-26, CASE Rank updates the means by first doing one epoch of training. All of the data in $\mathcal{D}$ is used for doing a forward and a backward pass through LTR model, which can be modeled for training as:

$$\begin{gathered}\hat{\mathbf{y}} = \mathbf{W}^{(4)} \cdot \Big(\mathbf{m}^{(3)} \odot \mathrm{ReLU}\Big(\mathbf{W}^{(3)} \cdot \big(\mathbf{m}^{(2)} \odot \mathrm{ReLU}\big(\mathbf{W}^{(2)} \cdot \\ (\mathbf{m}^{(1)} \odot \mathrm{ReLU}(\mathbf{W}^{(1)} \cdot \mathbf{x} + \mathbf{b}^{(1)})) + \mathbf{b}^{(2)}\big)\big) + \mathbf{b}^{(3)}\Big)\Big) + \mathbf{b}^{(4)}.\end{gathered} \tag{3.22}$$

where, $\mathbf{m}$ is dropout. Newly seen arm information allows the neural networks weights to be updated with :

$$\begin{aligned}\mathbf{W}^{(l)}_{\text{new}} &= \mathbf{W}^{(l)}_{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{W}^{(l)}} \\ \mathbf{b}^{(l)}_{\text{new}} &= \mathbf{b}^{(l)}_{\text{old}} - \eta \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{(l)}}\end{aligned} \tag{3.23}$$

Like in CASE, in CASE Rank the means of the arms are stored as a numerical value. To calculate the new means using updated weights, arms should be reevaluated. This is done by getting predictions for N different queries for each arm using an updated neural network(lines 27-31 in Algorithm 1). For arm a, the estimated mean is:

$$\hat{\rho}_a(t) \triangleq \frac{\sum_{i=1}^{N} \hat{y}_i(t)}{N} \tag{3.24}$$

where N is query/validation sample number(lines 32-34 in Algorithm 1). In order to align LLM inference feedback and new means, queries used for pulling arms and for making predictions are kept the same. We were inspired by [40] when determining how to calculate the new means. The algorithm we adapted averages N different validation samples to calculate the reward for an arm. Then, the algorithm uses those rewards to linearly update the means. CASE Rank calculates the means by getting predictions to the same N validation samples. So, the intuition behind both approaches is to measure an arms overall performance using these N validation samples. Moreover, CASE Rank uses LTR model as the surrogate, so it can not use validation score accuracy as the reward. This is because LTR-style datasets require a single score per query, unlike a linear surrogate where an average can be used for an accuracy score. Thus, the LTR model observes the rewards as subset performance on individual queries rather than N queries like the algorithm it adapts. Averaging prediction scores to obtain the mean in CASE Rank allows the model to observe the overall performance of an arm.
The other reason for averaging predictions for different queries to calculate mean is because of efficiency concerns. Instead of this approach, if the prediction for a single query was collected multiple times, it would make updating the means very costly. If, for example, ten predictions were to be done per query instead of one, the algorithm would be 10 times slower.
One could argue that obtaining means by averaging predictions for N different queries do not result in calculating true means. This could be because different queries may result in vastly different LTR scores, resulting in an imbalance in scores. Thus, one could say calculating arm mean by averaging the scores for a single could yield better results. However, this approach would result in the algorithm having a bias toward a single arm for each query. That is because that arm would do best for that particular query when means are calculated. Thus, that one arm would always get the highest mean. An additional proof for showing information is not lost when means are calculated this way can be seen in section A.4. It should be noted that for predictions, the dataset from section 3.2 does not contain any relevance scores. Also in order to maintain consistency in predictions, LTR model was set to inference mode, in which there was no dropout.

The problem of identifying top-m arms can be solved using the Gap-index methods, which are unified by the GIFA framework [43]. The GIFA framework defines an iterative family of algorithms that keeps a set U_t of the current estimates of the m-best arms. At iteration t, it selects the most ambiguous arm in U_t,

$$b_t \;=\; \arg\max_{b \in U_t} \; \max_{a \in U_t^c} B_t(a, b),$$

and the most ambiguous arm in the complement U_t^c (the *challenger*),

$$c_t \;=\; \arg\max_{c \in U_t^c} B_t(c, b_t).$$

The algorithm then pulls whichever of b_t or c_t exhibits the larger variance and updates the model parameters. Here $B_t(a,b)$ denotes the gap index between arms a and b.

3.3.2. Gap Calculation

The gap calculation stays the same since CASE Rank also holds a list of arm means like CASE:

$$\hat{\Delta}_{i,j}(t) \triangleq \hat{\rho}_i(t) - \hat{\rho}_j(t) \tag{3.25}$$

where $\hat{\rho}_i(t)$, $\hat{\rho}_j(t)$ are estimated rewards for arms i and j respectively.

3.3.3. Variance Calculation

The gap-index between two arms (i,j) is defined in [40] as:

$$B_t(i,j) = \hat{\rho}_t(i) - \hat{\rho}_t(j) + W_t(i,j) \tag{3.26}$$

where the confidence term is defined as: $W_t(i,j) = C_{t,\delta}\left(\|x_i\|_{\hat{\Sigma}_t^\lambda} + \|x_j\|_{\hat{\Sigma}_t^\lambda}\right)$,where

$$C_{t,\delta} = \sqrt{2\ln\left(\frac{1}{\delta}\right) + N\ln\left(1 + \frac{(t+1)L^2}{\lambda^2 N}\right)} + \frac{\sqrt{\lambda}}{\sigma}S, \tag{3.27}$$

and S and L are constants, N is the total number of arms pulled, and $\hat{\Sigma}_t^\lambda = \sigma^2(V_t)^{-1}$[40].
Equation 3.27 can also be written in UCB-form as:

$$W_t(i,j) := \beta\left(\hat{\sigma}_t(i) + \hat{\sigma}_t(j)\right) \tag{3.28}$$

where $\hat{\sigma}_t(i)$ is $\|x_i\|_{\hat{\Sigma}_t^\lambda} = \sqrt{x_i^\top \hat{\Sigma}_t^\lambda x_i}$, and β is $C_{t,\delta}$. Thus, the two formulations of $(W_t(i,j)$ are **functionally equivalent**, with the first being more precise and the second more computationally efficient. Therefore, Equation 3.28 was referenced when calculating confidence bound for CASE Rank.
To calculate the confidence bound $W(i,j)$ between two arms i and j, first the surrogate loss function PiRank model is put in training mode, where dropout is turned on. The equation for the model in that setting can be seen 3.22. The reason behind using dropout when calculating confidence bounds is so that the variance between predictions can be measured. In order to have variance between predictions, dropout must be utilized during a forward pass.

Dropout randomly sets a fraction of the neurons' outputs to zero in each forward pass. A forward pass through a layer with some layer **h** looks like:

$$\tilde{\mathbf{h}} = \mathbf{m} \odot \mathbf{h} \tag{3.29}$$

where $\odot$ denotes element-wise multiplication, and $\mathbf{m} \in \{0,1\}^{1\times d}$ is a binary mask vector sampled from a Bernoulli distribution.

$$D \sim \mathbf{Bernoulli}(1-p) \tag{3.30}$$

Having enabled dropout, a prediction is made for N different queries for each arm. Using these predictions the confidence term between two arms i and j can be expressed as:

$$W(i,j) = c_{\text{new}} \cdot \sqrt{\frac{1}{N}\sum_{k=1}^{N}\left(y_i^{(k)} - y_j^{(k)} - \bar{d}\right)^2} \tag{3.31}$$

where c_{new} is a scalar confidence multiplier (hyperparameter that scales the standard deviation):

$$\begin{aligned} c_{\text{t}} &= \frac{c_{\text{t-1}}}{\sqrt{t}}, \quad \text{if } t > 0 \\ c_{\text{t}} &= 1.96, \quad \text{otherwise} \end{aligned} \tag{3.32}$$

and,

- $y_i^{(k)}$: The k-th observed score for arm i.
- $y_j^{(k)}$: The k-th observed score for arm j.
- $\bar{d}$: The empirical mean of the pairwise differences, defined as:

$$\bar{d} = \frac{1}{n} \sum_{k=1}^{n} \left(y_i^{(k)} - y_j^{(k)} \right)$$

- N: The number of samples or observations used to estimate the difference.

The rest of the gap-index equation is the same, as the gap calculation does not change.
Ideally, estimating uncertainty is most accurate when multiple predictions per sample can be made. In our setting, however, an arms mean is computed across multiple validation samples. In our case, we define an arm's mean as the average score across multiple validation samples, so repeated bandit rounds already incorporate dropout-driven variability. Computing a standard deviation from a single validation sample would both overfit to that example and create a mismatch with how we compute the mean. Estimating variance across the validation set instead reduces overfitting, while repeated evaluations on the same arms and samples also capture dropout-based variance.

3.3.4. Uncertainty Calculation

Uncertainty is used to determine in which direction in the input domain to explore. In line 19, uncertainty is used to select which arm to pull. It is calculated per arm. Like confidence bound, also for uncertainty, the loss model is used with dropout turned on. For arm i, N predictions are collected. Using these predictions, uncertainty for arm i is calculated as:

$$\text{uncertainty} = \sqrt{\frac{1}{N} \sum_{k=1}^{N} \left(\hat{y}^{(k)} - \bar{y} \right)^2} \tag{3.33}$$

where:

$$\bar{y} = \frac{1}{N} \sum_{k=1}^{N} \hat{y}^{(k)} \tag{3.34}$$

also,

- $\hat{y}^{(t)}$: The prediction vector from the t-th model (or forward pass).
- T: Total number of predictions (e.g., number of ensemble members or MC dropout samples).
- $\bar{y}$: The mean prediction across all T samples.

As all the changes, including updating means, gap, variance, and uncertainty, were explained, the overall pipeline can be described. First, the LTR model is pre-trained with data from randomly selected M groups. Then, the pre-trained LTR model replaces the linear surrogate of the bandit algorithm, resulting in a bandit algorithm with a non-linear differentiable surrogate. The pre-training data is then used to initialize $\mathcal{D}$. In lines 5-6 in Algorithm 1, the two most ambiguous arms n_t, c_t are taken from U_t and N_t, respectively, and their gap is calculated. If their gap is not lower than ϵ, subsets are swapped between U_t and N_t (lines 7-12 in Algorithm 1). This is done based on which subsets/arms n_t, c_t have the lowest mean and which have the highest mean. The algorithm then uniformly samples m arms from $(U_t \cup N_{t1})^c$, to generate the set M_t. M_t is used to select the top-m arms from $M_t \cup N_{t1}$ to generate the updated N_t(lines 13-17). Then, the selection rule is used to decide which arm to pull from $N_t \cup U_t$, the high-reward selected set. Using the arm, LLM feedback is obtained. Afterwards, the training set is updated as in 3.14. In lines 22-24, the training set is then used for one forward pass through the surrogate, which is then used to reestimate the means as described in subsection 3.3.1. Then again, the convergence criterion is checked. When the algorithm converges, it returns the re-trained LTR model. Finally, the LTR model is used for online inference.

4 Experimental Setup

4.1. Dataset

Experiments were performed on two tasks. For natural language understanding, reasoning experiments were performed. For natural language generation , machine translation experiments were performed.

GSM8K and AQUA-RAT datasets were utilized for reasoning experiments.
GSM8K[10]. GSM8K (Grade School Math 8K) is a dataset containing 8.5K high-quality, linguistically diverse math word problems designed for grade school students. Each of its instances contains a string for the grade-school level math question and a string for the corresponding answer with multiple steps of reasoning and calculator annotations. An example GSM8K prompt can be seen in Figure A.2. GSMK8K dataset consists of 7473 training and 1319 validation samples. Since GSM8K does not have any test samples, its validation samples are used for testing and a 30% split is performed on training samples to obtain 5231 training and 2242 validation samples. Because vanilla dynamic exemplar selection makes several hundred LLM calls per test sample, to perform all experiments in time, only 300 test samples were used for GSM8K instead of all available test samples.

AquaRat[27]. The AQUA-RAT dataset consists of about 100,000 algebraic word problems with natural language rationales. Each AQUA-RAT instance contains a question, option, rationale, and an answer. An example AQUA-RAT prompt can be seen in Figure A.1.AQUA-RAT consists of 97467 training, 254 validation, and 254 test samples.

Machine translation experiments were conducted using the dataset WMT19. Machine translation experiments were done from English to Simplified Chinese.
WMT19[16]. For the task of machine translation, we used the dataset WMT19, and experimented on translation from English to Simplified Chinese. An example WMT19 prompt can be seen in Figure A.3. WMT19 has 98726 training, and 3981 validation samples. Because WMT19 does not have any test samples, its validation samples are used for testing and a 30% split is performed on training samples to obtain validation samples.
All datasets are utilized for the following during this research: training, validation, LTR dataset relevance score collection, CASE algorithm and evaluation.

4.2. Evaluation metrics

Exemplar selection is a thoroughly researched field with standards for evaluation metrics. To ensure comparability of our results with existing studies, we adapt these standard metrics where applicable. Evaluation metrics used for this research can be categorized into three: evaluation metrics for inference, evaluation metrics for arm accuracy, and evaluation metrics for LTR model training.

Exact match (EM) is the first inference metric used in this evaluation. It returns 1 if the models output is identical to the reference string and 0 otherwise, making it appropriate when strict fidelity to a single ground-truth answer is required. EM is applied to evaluate GSM8K and AquaRat, and it is also used to compute arm accuracy for the dynamic and static CASE algorithms on these datasets.

This study adopts the F1 variant of BERTScore as the second evaluation metric when using WMT19 dataset. BERTScore uses contextual representations from advanced language models that have already been trained, known as Transformer-based models, to measure how similar a candidate text is to a reference text. The F1 version of BERTScore focuses on both precision and recall by comparing the similarity of individual words from the candidate to those in the reference, and vice versa, in a way that aims to capture the best possible matches between the two texts. The F1 score is then computed from these precision and recall components to capture a balanced view of semantic alignment. The equation for F1 BertScore can be seen in Equation 3.2.1.

F1 BertScore is also used to compute the arm accuracy for CASE Rank for all datasets. On the other hand, it is only used to compute arm accuracy for Dynamic and Static CASE on WMT19 dataset.

The final metric and only metric used in training of LTR models is normalized Discounted Cumulative Gain (nDCG), described in Equation 4.1. This metric stems from Discounted Cumulative Gain (DCG) [22], which is is specifically intended to capture gradient relevancy labels. It measures user satisfaction by rewarding the most relevant (gradient) results to appear first. nDCG compares the DCG from the actual ranking against an ideal ranking, in which the passages are perfectly sorted by decreasing relevance.

$$\begin{aligned} \mathrm{DCG}(y, \hat{\pi}) &= \sum_{j=1}^{L} \frac{2^{y_{\hat{\pi}_j}} - 1}{\log_2(1+j)} \\ \mathrm{NDCG}(y, \hat{\pi}) &= \frac{\mathrm{DCG}(y, \hat{\pi})}{\mathrm{DCG}(y, \pi^*)} \end{aligned} \tag{4.1}$$

Higher values of DCG and NDCG correspond to better ranking quality. Truncated versions, DCG@k and NDCG@k, are obtained by replacing L with a cutoff value k in Eq. (4.1), restricting evaluation to the top-k ranked items.

4.3. Hardware configuration

The experiments were conducted using both DelftBlue[1] cluster nodes and Kaggle-hosted cloud GPUs. On DelftBlue, the hardware was provisioned via two Slurm partitions: gpua100 and gpua100small. Nodes in the gpua100 partition feature dual Intel Xeon E56448Y processors (totaling 64 CPU cores), 500 GB of system RAM, and 1.5 TB of local SSD storage. Each node is equipped with 4 $\times$ NVIDIA A100 Tensor Core GPUs, each offering 80 GB of HBM2 memory, delivering high throughput for compute-intensive modeling. The gpua100small partition utilizes a single NVIDIA A100 GPU configured with NVIDIAs Multi-Instance GPU (MIG) mode into 28 separate GPU instances, each with 10 GB of dedicated memoryproviding, fine-grained, shortduration compute access.

For embedding generation tasks, additional resources were leveraged via Kaggle notebooks, which provided access to two NVIDIA Tesla T4 GPUs. Each T4 GPU is based on the Turing architecture and includes 2,560 CUDA cores and 320 Tensor cores, with 16 GB of GDDR6 memory and a memory bandwidth of approximately 320 GB/s.

4.4. Baselines

The following section describes the training details, hyperparameters used, and other details concerning each baseline.

Static CASE experiments were conducted using CASE algorithm with the parameterization used in the original work. The number of top arms to be identified was set to $m = 10$, and the total number of available arms was $K = 5$. The confidence parameter was fixed at $\delta = 0.05$, controlling the probability of incorrectly identifying the top-m arms. The minimum gap for switching the arms between U_t and N_t was also kept at $\varepsilon = 0.1$.

Dynamic CASE method is obtained by applying CASE algorithm, used by Static CASE, specifically for each test instance instead of all of the test dataset. Thus, in Dynamic CASE feature matrix is newly formed for each test instance from training and test embeddings instead of forming once from training and validation embeddings. Moreover, the hyperparameters utilized during experiments for CASE algorithm are the same as Static CASE configuration.

PiRank is the first baseline deployed for using LTR models for demonstration example selection. PiRank is also used as the LTR model of CASE experiments. PiRank was chosen because it is both small and has good performance. The ranking temperature parameter $\tau = 5$ controlled how sharp or smooth the differentiable sorting approximation was. Lower values made the ranking distribution more focused, while higher values made it smoother. Each input slate was restricted to a list size of 500, ensuring a fixed-length representation for efficient batch computation. The model architecture consisted of a sequence of fully connected hidden layers with sizes of (256,256,128,64) and ReLU activation function after each layer, processing 768-dimensional feature vectors for each item in the slate. These same parameter values were used also for the PiRank model that was used for non-linear surrogate of CASE Rank.
Optimization was performed using Adam with a learning rate of 1×10^4, balancing stability and convergence speed for this architecture. The training ran for 100 epochs with a batch size of 16.

AllRank is the second baseline utilized in LTR models for demonstration example selection. The model architecture consisted of a shared fully connected pre-projection layer of size 128, followed by a Transformer encoder with 4 blocks (N), 2 attention heads (h), a feed-forward width of 512 (d_{ff}), and dropout probability of 0.3. A final shared linear layer produced a single relevance score per item. Input slates were padded or truncated to a 240 and batch size was 64.
Training was carried out for 100 epochs with no early stopping. The optimizer was Adam with a learning rate of 1×10^{-3}, paired with a StepLR scheduler with decay rate 0.1 every 50 epochs. Gradient clipping was not applied. The primary validation metric was NDCG@5, aligning model selection with the intended emphasis on shallow-ranked items. The main hyperparameters reported here are for the Neural NDCG loss configuration. However, this study also covered experimentation with alternative listwise and differentiable ranking objectives, including ListNet, LambdaRank, Neural NDCG with normalized data, and NDCGLoss 2++. Parameter values for each loss setting can be seen in Table 4.1.

CASE Rank uses PiRank as the LLM for non-linear surrogate loss of the gap-index-based bandit algorithm. The hyperparameters used for the PiRank model in CASE Rank are the same as those of the PiRank baseline. The CASE Rank algorithm performs pre-training and later training in each CASE iteration on PiRank. The training details are the same as the PiRank baseline. As CASE Rank adapts the CASE algorithm to train the PiRank model, the algorithm uses the same CASE configuration as the other bandit-based baselines.
Also, in CASE RANK algorithm, the core procedural components from the original CASE framework were retained to ensure theoretical consistency and comparability of results. Specifically, the computation of the set J_t, estimated m best arms at t, was kept identical to the original CASE definition. Similarly, the computation of b_t, the most ambiguous arm from U_t, follows the same formulation as in CASE, and keeps its important function in directing how the arm selection process works. The stopping rule, which terminates sampling once the confidence intervals separate the top-m arms from the rest by at least ε, was also kept unchanged. Finally, the selection rule, choosing the next arm to sample based on the maximization of the elimination potential as in the original CASE algorithm, was preserved without modification.

4.5. Relevance Score Generation Details

k is selected to be 5 to comply with the choice of using 5-shot demonstration examples, so that a single example is selected from each cluster. $M = 500$ subsets are sampled for each query. Also, the LLM response is be post-processed depending on the dataset for which the relevance scores are collected. For

Table 4.1: Details of hyperparameters used in different LTR model configurations. Categorized by loss function and framework.

Framework	**Loss function**	**Transformer Architecture**	**FC Linear Layer**
PiRank	PiRank surrogate loss	-	(256, 256,128,64)
AllRank	Neural NDCG	N = 2, d_{ff} = 384, h = 1, dropout = 0.1	(768,96)
	ListNet	N = 4, d_{ff} = 512, h = 2, dropout = 0.3	(768,128)
	LambdaRank	N = 2, d_{ff} = 384, h = 1, dropout = 0.1	(768,96)
	Neural NDCG	N = 4, d_{ff} = 512, h = 4, dropout = 0.3	(768,96)
	Neural NDCG With Normalized data	N = 2, d_{ff} = 384, h = 1, dropout = 0.1	(768,96)
	NDCGLoss 2++	-	(256, 512, 1024, 512, 256)

the dataset WMT19, [16], no post-procesing is done since LLM response contains the whole translation sentence that is desired to be compared tothe ground truth. AquaRat and GSM8K are reasoning dataset with numerical answers, but post processing was done only for GSM8K. For responses of the GSM8K dataset, a post processing function was used to extract the numerical answer from the LLM response. AquaRat responses has no post-processing done, and the whole output with rationale was used. In the Table 4.2, part of the output that was scored for each dataset can be seen. The corresponding ground

Scored part of the response	Dataset
Final answer	GSM8K
Complete output/Rationale	AquaRat
Complete output	WMT19

Table 4.2: Part of LLM response that was used for scoring for each dataset

truths are numerical answers, ground truth rationale, and correct simplified Chinese translation for datasets GSM8K, AquaRat, and WMT19, respectively.

Finally, for the collection of the relevance score of the datasets AquaRat and GSM8K, `bert-base-uncased` was used. On the other hand, for WMT19 `bert-base-chinese` was used. The relevance score is multiplied by $R = 10$ to obtain the final relevance score. Thus, the range of relevance scores is 10.

4.6. Implementation Details

`llama3.2:3b` was chosen as the small LLM model for the experiments. It is a 3-billion-parameter, pretrained, and instruction-tuned generative model. `llama3.2:3b`[20] was used for all inference experiments where a demonstration example set is selected using some sort of method, including dynamic CASE, static CASE, LTR model, or CASE Rank. Then this selected exemplar set is used for 5-shot in-context learning. Inference is done by getting a response from `llama3.2:3b` following the 5-shot example and a new query. The temperature parameter of the llama inference model was set to 0.25, with a max new tokens of 256. The tokens are generated by sampling with a top k value of 10 and a top p value of 1.0.

`Mistral` was chosen as the larger LLM model for the experiments. Mistral is a 7-billion-parameter model, available in both instruction and text completion. Its versions `mistral:v0.1` and `mistral:v0.3` were used specifically. As `llama3.2:3b`, they were also used in all inference experiments, generating the 5-shot response to be evaluated. The temperature parameter of both the mistral inference models was set to 0.25, with max new tokens of 256 sampled with a top k value of 10 and a top p value of 1.0. hatali

5 Results

5.1. Reasoning performance of online dynamic exemplar selection and LTR approaches compared to that of offline exemplar selection approaches

To address **RQI**, Dynamic CASE and LTR models are compared with static CASE. The results are presented in Table 5.1. As it can be seen from Table 5.1, Dynamic CASE outperforms Static CASE, and LTR models outperform both Dynamic CASE and Static CASE. Dynamic CASE outperforming Static CASE is expected, as instead of choosing a single set of exemplars for all test instances, an exemplar set is chosen specifically for each test instance with the cost of huge computational overhead. Moreover, in Table 5.1, we observe that LTR models outperform both dynamic and static CASE when the best performing model is taken for each dataset. This higher performance over dynamic and static CASE can be explained by two reasons. Firstly, in both dynamic and static CASE, it is assumed that rewards are stochastically linear, which is empirically not true. Rewards used by the bandit algorithm come from LLM, which is a non-linear model. LTR models are neural networks, which are non-linear. So they do a better job in modeling the rewards from LLM calls. Secondly, LTR models are trained on the ranking metric NDCG, so they improve the actual target of interest. Whereas, CASE tries to decrease the ambiguity of each arm and the gap between arms while also choosing a better set of arms. Thus, the LTR model's goal aligns better with the goal of selecting the best exemplar for each instance than that of CASE.

Insight: *CASE Rank outperforms static and dynamic CASE because it focuses on optimizing the goal itself.*

On the other hand, the results in Table 5.1 also demonstrate that the ceiling for using a learning to rank model to select exemplars dynamically is limited in this setting. Firstly, a typical benchmark learning to rank dataset like [41] has 30,000 search queries, whereas all datasets used for this experiment, as explained in 4, were trained with datasets with only 35 queries but 500 groups. Therefore, the amount of data used for training may not be enough for any of the LTR models to converge with proper generalization, resulting in diminished results. Yet, one could argue that because PiRank is a relatively small-sized model with a hidden size of 3, the data used should be enough for it to converge with generalization. Therefore, the data at hand should be enough to train the PiRank model properly. This implies that the method of random sampling data is not helpful for the training process. It can be deduced that randomly sampling data to generate training data may not result in getting the best training data in this setting. Although novel and not as expensive, data generation for exemplar selection with learning to rank models is costly. Because of this, it can be said that random sampling data for data generation for this task is not a good option if only a small portion of data can be generated. Therefore, if LTR models are to be utilized to their full potential for the task of exemplar selection, a more advanced method, other than random sampling, should be employed.

Table 5.1: Demonstration example selection results across 3 datasets for different Learning-to-Rank models and Dynamic CASE using `llama3.2:3b`

Method	**GSM8K**	**AquaRat**	**WTM19**
Task level			
Static CASE(Llama2)	67.00	25.90	65.40
Instance level			
Dynamic CASE	70.00	36.40	71.70
Learning To Rank			
PiRank	69.30	37.00	71.30
NeuralNDCG	72.23	44.09	71.13
ListNet	70.30	43.30	71.26
LambdaRank	67.30	44.09	71.22
NDCGLoss 2++	70.00	43.70	72.56
NeuralNDCG with Normalized Data	71.33	45.66	71.80

5.2. Multi-armed bandit selection using non-linear surrogate loss function improves instance-level exemplar selection

To address **RQII**, the study compares the proposed CASE Rank algorithm with Dynamic CASE and learning-to-rank (LTR) baselines. As shown in Table 5.2, CASE Rank outperforms both Dynamic CASE and the LTR models. This improvement arises from addressing limitations in each baseline. Static CASE implicitly assumes that rewards from the LLM are linear, which is misaligned with the inherently non-linear behavior of LLMs. LTR models, when trained on randomly sampled data, exhibit a ceiling in exemplar-selection performance irrespective of dataset size. CASE Rank replaces linear matrix scoring with the *PiRank* loss and trains PiRank on data sampled using the CASE procedure. In effect, CASE Rank combines the strengths of both approaches while mitigating their shortcomings, yielding superior performance. Post-training, PiRank can also be viewed as a decoder that estimates the quality of the LLMs response for each exemplar set.

Insight: *PiRank learns to estimate the qualities of each exemplar post-training so that it can rank them to select which is best.*

To further support the claim that CASE Rank selects more representative instance-specific exemplars, qualitative analysis is done on exemplars selected by both the AllRank NeuralNDCG model and CASE Rank. Table 5.3 presents the exemplar subsets selected by both methods for a particular AquaRat test instance where CASE Rank had a correct answer and AllRank NeuralNDCG model did not.
For an LLM to learn from context for a specific test instance, it needs demonstration examples that have a similar context. Context can be categorized in this setting by the type of question asked, since the question type determines the rationale. Different types of mathematical questions require different types of thinking. For an LLM to learn from context, the number of demonstration examples that are similar to the test instance should be as many as possible. However, it should also be kept in mind that the exemplar subsets are formed by randomly sampling an element from each cluster. Thus, the chances of having exemplars that are all similar in an exemplar set are very low. That is because each exemplar needs to have similar rationales while having different contextual embeddings that capture semantic and syntactic information. Therefore, it can be said that in this setting, theoretically, the more a subset has questions and rationales that are similar to those of a test instance, the greater the chance it has to get a correct answer. This can also be seen in Table 5.3.
The test instance from the example at Table 5.3 is a Profit, Loss & Interest problem that requires algebra, in particular linear equations. Thus, exemplars with similar questions and similar skills are

Table 5.2: Demonstration example selection results across 3 datasets (we use 5-shot for all methods). Percentage improvements are reported over Dynamic CASE

Method	**GSM8K**	**AquaRat**	**WTM19**
	LLama3.2:3b		
Instance level			
Dynamic CASE	70.00	36.40	71.70
Task level			
Static CASE(Llama2)	67.00	25.90	65.40
Learning To Rank			
NeuralNDCG	72.23	44.09	71.13
Our Approach			
CASE Rank	**73.60**(▲3.60%)	**51.18**(▲14.78%)	**78.49**(▲6.79%)

required from each subset. The subset chosen by the AllRank NeuralNDCG model has only one such exemplar, which is the last exemplar in the list. It is a Profit, Loss & Interest problem that requires solving linear equations. On the other hand, CASE Rank chose a subset with two similar exemplars. Hence, it adds additional information to solve a particular question from the test instance that the LLM encounters during inference. Second exemplar of the subset of CASE Rank, even though not a Profit, Loss & Interest problem, requires solving linear equations. Also, the fourth exemplar of the subset is a Profit, Loss & Interest problem that requires solving linear equations. The same pattern can be seen in Table A.2, where the test instance is a proportionality problem. CASE Rank manages to select a subset with 2 proportionality questions (1.,2.), which is more than a single one(3.) AllRank chose. Therefore, it can be concluded that having more similar exemplars helps a subset in getting correct LLM responses by giving additional information to LLM. This pattern is further seen in Tables A.3, A.4, A.5.

Insight: *CASE Rank outperforms its competition by selecting subsets with more skills similar to that of test instance.*

To corroborate the results in Table 5.2 and address **RQII**, AllRank NeuralNDCG and CASE Rank are compared against Static CASE for exemplar selection using `mistral:v0.3`. Due to the computational cost and the larger model size of `mistral:v0.3`, no Dynamic CASE experiments were conducted. The results, shown in Table 5.4, indicate that CASE Rank consistently outperforms Zero-Shot CoT [24] and Few-Shot CoT [6], which rely on no or hand-picked exemplars and do not account for interactions among exemplars. This suggests that CASE Rank effectively selects instance-level exemplars that yield additional in-context gains for the LLM. CASE Rank also surpasses both AllRank NeuralNDCG and Static CASE, although the relative improvements are smaller than those observed with `llama3.2:3b`. The reduced gains across datasets with `mistral:v0.3` likely reflect cross-model instability of instance-level selection: what constitutes a good demonstration is model-specific. Differences in preferred reasoning style, full chain-of-thought vs. brief rationales vs. answer-only and prompt formatting and exemplar order (headers, role tags, separators, whitespace, numbering, and recency/primacy effects), which change how models parse the prompt, can all shift effectiveness across models. Also, sampling parameters (temperature, top-p/top-k, repetition penalties), stop sequences, and max tokens interact with exemplars. So, parameters that work for one model arent optimal for another. Moreover, `llama3.2:3b` is not specifically a reasoning model, but multiple sources [35, 11, 33] state that it has superior reasoning, which explains its superior performance in reasoning compared to `mistral` models.

To further evaluate the performance of CASE Rank and the LTR baselines on a larger, different language model, exemplarselection experiments were conducted with `mistral:v0.1`. Results for static and dynamic demonstration selection across methods are reported in Table A.1. CASE Rank outperforms

Table 5.3: Exemplar subsets selected by AllRank NeuralNDCG and CASE Rank using LLama3.2:3b for a particular AquaRat test instance where CASE Rank had a correct answer and the AllRank NeuralNDCG model did not.

Question	A travel company wants to charter a plane to the Bahamas. Chartering the plane costs $5,000. So far, 12 people have signed up for the trip. If the company charges $200 per ticket, how many more passengers must sign up for the trip before the company can make any profit on the charter?
Method	**Exemplars**
AllRank	**Question:** A group consists of 4 couples in which each of the 4 boys has one girlfriend. In how many ways can they be arranged in a straight line such that boys and girls occupy alternate positions? **Question:** A man cycling along the road noticed that every 12 minutes a bus overtakes him and every 4 minutes he meets an oncoming bus. If all buses and the cyclist move at a constant speed, what is the time interval between consecutive buses? **Question:** Find large number from below question. The difference of two numbers is 1365. On dividing the larger number by the smaller, we get 6 as quotient and the 15 as remainder? **Question:** A certain sum is invested at simple interest at 18% p.a. for two years instead of investing at 12% p.a. for the same time period. Therefore the interest received is more by Rs. 840. Find the sum?
CASE Rank	**Question:** 64, 81, 100, 144, 196, ?, Find the missing number(?). **Question:** In a school 10% of the boys are same in number as 1/2th of the girls. What is the ratio of boys to the girls in the school? **Question:** Anna left for city A from city B at 5.20 a.m. She traveled at the speed of 80 km/hr for 2 hrs 15 min. After that the speed was reduced to 60 km/hr. If the distance between two cities is 350 kms, at what time did Anna reach city? **Question:** If a number when divided by 44, gives 432 as quotient and 0 as remainder. What will be the remainder when dividing the same number by 34? **Question:** On a sum of money, the S.I. for 2 years is Rs. 660, while the C.I. is Rs. 696.30, the rate of interest being the same in both the cases. The rate of interest is? **Question:** How many options are there for license plate numbers if each license plate can include 2 digits and 3 letters (in that order) or 3 digits and 2 letters (in that order)? (Note: there are 26 letters in the alphabet?)

Rationale is not shown to conserve space. However, in our experiments, all exemplars for AquaRat include rationales. The test instance is a Profit, Loss & Interest problem that requires algebra. Subset selected by CASE Rank contains two similar questions(2.,4.), whereas AllRank NeuralNDCG one(5.).

both LTR methods as well as zero-shot CoT [24] and manual few-shot CoT [6]. However, both CASE Rank and the LTR methods achieve lower exemplarselection performance with `mistral:v0.1` than with `llama3.2:3b`. Moreover, while CASE Rank still improves over LTR, the gains are smaller than with `llama3.2:3b`, indicating that providing higher-quality demonstrations yields limited additional benefit for `mistral:v0.1`.

Insight: *`llama3.2:3b` is the stronger model overall and leverages in-context examples more effectively than `mistral`.*

5.3. Relevance score approach effects CASE Rank performance

Table 5.2 highlights the relative gains CASE Rank provides in performance compared to Dynamic CASE. It can be seen from Table 5.2 that each dataset has different relative gains. The relative gains for datasets AquaRat and WMT19 are much higher than those of GSM8K. Since two of three datasets have relative gains much higher than 5%, one could expect that the third dataset would have at least close to 5%. This difference between relative gains can be explained by the difference between relevance score generation approaches for each dataset. As it was previously presented in Table 4.2 in Chapter 3 when generating the relevance score for each dataset, a different part of the LLM output was scored by BERTScore. Table 4.2 shows that the extracted final answer is the scored part to obtain the final relevance score for the dataset GSM8K. The extracted final answer contains much less information compared to the rationale of an LLM few-shot output of AquaRat or a whole text translation of WMT19. Since it contains much less information, there is also less information change between answers

Table 5.4: Demonstration example selection results across 3 datasets (we use 5-shot for all methods) in different settings. Percentage improvements are reported over next best performing model.

Method	GSM8K	AquaRat	WTM19
	Mistral-7B-v0.3		
Task level			
Zero-shot-COT	41.47	25.00	64.50
Manual Few-shot COT	36.66	30.31	71.81
Learning To Rank			
NeuralNDCG	45.00	35.80	75.90
Our Approach			
CASE Rank	**50.30**(▲5.30%)	**36.43**(▲0.63%)	**77.00**(▲1.10%)

that are going to be scored. For example, if the extracted final answer is 169 and the ground truth is 163 this affects the relevance score by very little. In the case of a very different answer relevance score becomes zero. Thus, the whole dataset ends up having a relative score range of 3. Not having a lot of variance in relevance score makes it difficult for PiRank model to differentiate between good and better. This is because bad exemplar subsets will be eliminated by getting zero; however, some of the two different exemplar subsets that have non-zero relevance scores will have very similar subset scores for GSM8K. This causes PiRank model to struggle to identify the best exemplar subsets, which causes less performance gain in selection. On the other hand, both AquaRat and WMT19 have better quality relevance scores since their relevance scores are more informative. For obtaining relevance scores for AquaRat and WMT1,9, whole outputs are used for comparison, which allows much more diversity in relevance scores. This diversity allows better performance when using CASE Rank for exemplar selection, resulting in higher relative gains.

5.4. LTR model as a non-linear surrogate loss makes offline policy exemplar selection more efficient than online optimization and LTR approaches

To address **RQIII**, the study compares the efficiency of CASE Rank with Dynamic CASE, Static CASE, and an LTR baseline. Efficiency is measured as inference time (seconds). As shown in Table 5.5, CASE Rank substantially reduces inference time relative to Dynamic CASE approximately 100 × faster. The speedup arises because CASE Rank executes the CASE procedure once, whereas Dynamic CASE reruns CASE for every test instance, incurring considerable overhead. In conclusion, CASE Rank not only accelerates inference but also improves performance, making the overall improvement substantial.

Inference Time[s]	**GSM8K**	**AquaRat**	**WMT19**
Dynamic CASE	1554.19	2378.04	106.54
CASE Rank	11.74	10.59	1.68

Table 5.5: Inference Time comparison between CASE Rank and Dynamic CASE using Llama3.2:3b for each of the datasets. Inference time is measured in seconds and is calculated per instance.

To measure CASE Rank's efficiency better, it is compared with the fastest overall exemplar selection algorithm, Static CASE. Inference time speed comparison between CASE and Static CASE is presented in Figure 5.1. With the exception of the AquaRat dataset, Static CASE attains the lowest inference time. The AquaRat outlier may be explained by latency spikes in several LLM responses that inflate the average. Static CASE's efficiency dominance is expected, as it is a task-level method: At inference, it simply queries the LLM once with a pre-selected exemplar. By contrast, CASE Rank selects an exemplar subset at inference using PiRank predictions, which introduces additional computational overhead. Even so, the difference in inference time between the two methods is modest. Figure 5.1 also displays the fact that CASE Rank is fastest when performing machine translation using WMT19 dataset. This is expected as large language models are already very good at machine translation without any fine-tuning or in-context learning. This is because LLMs are pre-trained on massive bilingual or parallel corpora LLMs, especially multilingual models trained with translation tasks in their pretraining objective, have seen large quantities of translation pairs.

Insight: *The incremental additional cost of CASE Rank is justified by its performance gains over Static CASE.*

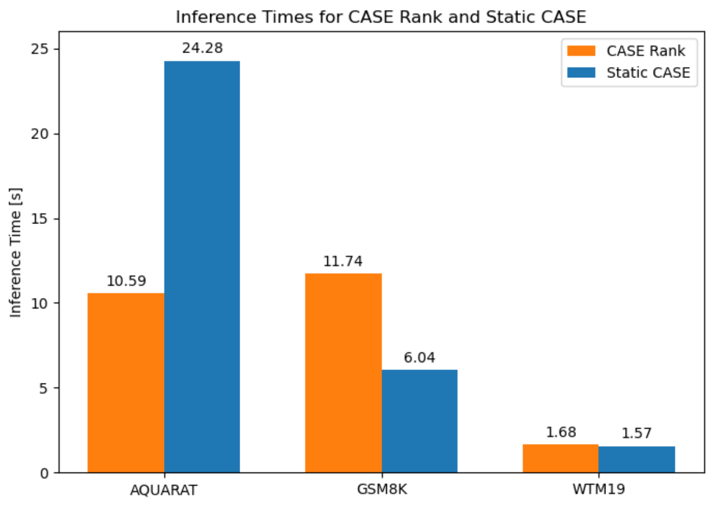


Figure 5.1: Inference Time comparison between CASE Rank and Static CASE using Llama3.2:3b for each of the datasets. Inference time is measured in seconds and is calculated per instance.

Figure A.4 reports inference times across datasets for Static CASE as well. It should be noted that as a result of an outlier due to high latency, average inference times are larger than the median. From Figure A.4, it can be established that CASE Rank is faster in inference time, thus more efficient than LTR model. Therefore, it is concluded that offline policy exemplar selection can be more efficient than online optimization and learning-to-rank approaches when using CASE Rank. The inference time speed difference can be explained by the fact that LTR models are larger than PiRank. PiRank is a 3-layer neural network, whereas LTR models are transformers. So, PiRank has many fewer parameters than a transformer. Because of this, a forward pass takes longer for a transformer than for PiRank. Therefore, it can be said that model size is decisive in determining efficiency in this setting.
Moreover, it can also be said that both the AllRank LTR model and CASE Rank's inference times when performing arithmetic reasoning are lower than those of GSM8K. This is unexpected since AquaRat is more diverse and has challenging math problems that often include algebra, logical reasoning, and distraction choices. These results can only be explained by the structure of the datasets. AquaRat offers rationale and multiple choices per question, whereas each GSM8K problem has only a single numerical answer. Thus, the more informative structure of AquaRat results in inference time speedup when performing in-context learning.

6 Conclusion

6.1. Conclusion

In this study, three different approaches to demonstration example selection for in-context learning were investigated in the context of instance-level exemplar selection. The first approach examined is Dynamic CASE, an application of the CASE algorithm from [40] to an instance-level setting. Dynamic CASE outperforms static CASE by using the CASE algorithm to select a demonstration example group for each test instance. However, experiments also showed that Dynamic CASE is much more expensive than its static counterpart because it makes the same number of LLM calls that static CASE does per test instance. Even though Dynamic CASE outperforms the original CASE, its performance increase is not high enough for it to be worth using as a tool over the original CASE algorithm.

To address the limitations associated with the inefficiency of Dynamic CASE, a novel framework for using LTR models for demonstration example selection is introduced. By using LLM feedback and BertScore F1 evaluation metric, relevance scores are collected, allowing LTR data for exemplar selection generation. Using an LTR model overcomes the problem of efficiency since a trained LTR model can be used for prediction for each test instance. Because the prediction process is very fast, efficiency is not a problem in this setting.

Empirical findings indicate that LTR models trained for exemplar selection outperform Dynamic CASE while also being more efficient. It can be emphasized that LTR models trained on optimizing the actual goal rather than decreasing uncertainty while also choosing the highest reward is the reason Dynamic CASE is outperformed. On the other hand, this approach has a performance ceiling in this setting because of the relevance score generation cost. Since no more data can be generated, training can not be improved by randomly sampling more data. Therefore, this approach is more effective in settings where a small amount of data can be afforded to generated. Nonetheless, LTR models proves to be an alternative and efficient method to select exemplars for in-context learning.

Third and final approach introduced, CASE Rank, addresses the limitations of both CASE algorithm and LTR models. CASE Rank introduces a non-linear differentiable surrogate in gap-index framework. It uses an LTR model as a non-linear surrogate function to eliminate the problem of not modeling loss non-linearly. Also, the CASE algorithm is used to select training data to overcome the random sampling obstacle.
Experimental results show that CASE Rank is better in terms of performance than both approaches. Using a LTR model as non-linear surrogate for the adapted-CASE algorithm while using the algorithm to select training data for LTR model proves to be beneficial. However, it is observed that the relevance score collection approach determines the performance increase magnitude. Depending on which part of the LLM response is used for evaluation, the spread of relevance scores changes. More spread allows higher-quality training data.

Finally, the inference time speed of each of the three approaches and Static CASE was investigated and compared by measuring the inference time speed per instance in seconds of each approach. CASE Rank is shown to be the fastest approach out of the three. Dynamic CASE is expected to be slowest because CASE Rank uses LTR model for prediction. Using a smaller neural network makes CASE Rank faster than the vanilla LTR model approach. These results highlight the efficiency of the CASE Rank approach. Even though CASE Rank is slower than Static CASE at inference time, the difference in inference time between the two methods is modest. This suggests that the incremental cost of CASE Rank may be justified by its performance gains over Static CASE.

Even though its relative gains can be unpredictable across models and generating a lot of training data is costly, CASE Rank proves to be efficient and beneficial to exemplar selection for domain-specific in-context learning in scenarios where the computational resources are scarce.

6.2. Limitations and Assumptions

Choice of hyperparameter values
This section details the studys limitations and the assumptions embedded in its experimental design. Owing to resource constraints, experiments were conducted using a single hyperparameter configuration. Hyperparameters for the inference LLMs, the CASE algorithm, and the relevance-score collectionalong with CASE-specific parameterswere predetermined. These selections constitute design choices rather than values substantiated by prior literature. No ablation studies were performed to assess the impact of these hyperparameters.

LTR data generation
As it was stated before, to collect each relevance score, an LLM call needs to be made. Because of resource constraints, generating 50 queries for 500 groups takes several days. This amount is below standard, where benchmark datasets have over 300,000 queries. Thus, a small amount of data was used for training LTR models. This limited LTR model training. The cost of LTR data generation prevented from generating more data to increase performance and try various different experiments. On the other hand, instead of 500 groups, for fewer groups relevance score could be collected, allowing data generation involving more queries. However, it was reasoned that using a large number of groups would allow generating more varied data.

Below average LLM size
Since LLM response time depends on the model size, to efficiently conduct experiments, smaller LLMs were chosen. Due to resource constraints, using larger LLMs was not an option since not all of the experiments would be done in time. Smaller LLMs tend to have worse performance than larger LLMs. Therefore, the performance increase when using methods like CASE Rank is limited in experimental results when using smaller LLMs.

6.3. Future Works

Dynamic Exemplar Selection Across Steps
While this thesis casts exemplar (prompt) selection as a one-shot bandit with a nonlinear LLM surrogate, a natural next step is to model reasoning as a sequential decision process over multiple possible response paths. In complex queries, the system may need to decompose the task into sub-questions, each requiring its own exemplar set; moreover, for any chosen prompt, the LLM can produce diverse continuations, yielding a branching space of trajectories rather than a single outcome. To capture this, we propose extending the framework beyond gap-index linear bandits to more general bandit formulations, in particular, the restless bandit setting, where the value of each arm evolves over time even when not selected due to changes in the intermediate reasoning state and the distribution of potential LLM continuations. This model supports careful planning across steps. It lets the system allocate pulls across subquestions. It can reselect exemplars as new evidence appears. In practice, we learn state representations from intermediate LLM outputs. We specify how each arm transitions between states.

We then optimize activation policies that act over trajectories rather than single pulls. This aligns exemplar selection with multistep LLM reasoning in large, branching search spaces.

Using CASE Rank in other domains
Beyond in-context learning, the proposed bandit-based exemplar selector is a generic subset selection primitive. A natural extension is to study it as a data curation tool for pretraining, dataset distillation, and active learning, where the algorithm adaptively chooses informative subsets under budget constraints. The LLM-based surrogate used here can be replaced with domain reward models, simulators, or human preference models, enabling deployment in settings that do not rely on LLMs. Promising applications include drug discovery and repurposing (selecting candidate compounds for costly assays) and protein design or folding (prioritizing sequences or structures), where the search space is large and evaluations are expensive and noisy. Also, comprehensive benchmarks across domains would demonstrate the algorithms value as a drop-in selection module, independent of ICL, and clarify how to amortize evaluation costs at scale.

Dataset size vs performance
Since the number of training data for LTR models was limited, the performance of LTR models in selecting exemplars for in-context learning was limited. To better measure their performance, future research should aim to generate more data for training. With more training data the performance of LTR models when using different amounts of training data can be investigated as well.

6.4. Disclosure

In line with TU Delft's publishing policies[1], I acknowledge the use of AI tools powered by ChatGPT to assist in rephrasing certain sections of this thesis.

[1]TU Delft publishing policies: `https://www.tudelft.nl/library/actuele-themas/open-publishing/about/policies`

References

[1] Delft High Performance Computing Centre (DHPC). *DelftBlue Supercomputer (Phase 2)*. https://www.tudelft.nl/dhpc/ark:/44463/DelftBluePhase2. 2024.

[2] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. "Improved algorithms for linear stochastic bandits". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems*. NIPS'11. Granada, Spain: Curran Associates Inc., 2011, pp. 2312–2320. ISBN: 9781618395993.

[3] Shipra Agrawal and Navin Goyal. "Thompson Sampling for Contextual Bandits with Linear Payoffs". In: *Proceedings of the 30th International Conference on Machine Learning*. 2013, pp. 127–135.

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. "Finite-time Analysis of the Multiarmed Bandit Problem". In: *Mach. Learn.* 47.23 (May 2002), pp. 235–256. ISSN: 0885-6125. DOI: 10.1023/A:1013689704352. URL: https://doi-org.tudelft.idm.oclc.org/10.1023/A:1013689704352.

[5] Iz Beltagy et al. "Zero- and Few-Shot NLP with Pretrained Language Models". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*. Ed. by Luciana Benotti et al. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 32–37. DOI: 10.18653/v1/2022.acl-tutorials.6. URL: https://aclanthology.org/2022.acl-tutorials.6/.

[6] Tom B. Brown et al. *Language Models are Few-Shot Learners*. 2020. arXiv: 2005.14165 [cs.CL]. URL: https://arxiv.org/abs/2005.14165.

[7] Christopher JC Burges. "From ranknet to lambdarank to lambdamart: An overview". In: *Learning* 11.23-581 (2010), p. 81.

[8] Olivier Chapelle and Yi Chang. "Yahoo! Learning to Rank Challenge Overview". In: *Proceedings of the Learning to Rank Challenge*. Ed. by Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Vol. 14. Proceedings of Machine Learning Research. Haifa, Israel: PMLR, June 2011, pp. 1–24. URL: https://proceedings.mlr.press/v14/chapelle11a.html.

[9] Aakanksha Chowdhery et al. *PaLM: Scaling Language Modeling with Pathways*. 2022. arXiv: 2204.02311 [cs.CL]. URL: https://arxiv.org/abs/2204.02311.

[10] Karl Cobbe et al. *Training Verifiers to Solve Math Word Problems*. 2021. arXiv: 2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

[11] CrewAI. URL: https://docs.crewai.com/en/concepts/llms#meta-llama.

[12] Rémy Degenne et al. "Gamification of Pure Exploration for Linear Bandits". In: *Proceedings of the 37th International Conference on Machine Learning*. Vol. 119. Proceedings of Machine Learning Research. 2020, pp. 2432–2442.

[13] Mingkai Deng et al. *RLPrompt: Optimizing Discrete Text Prompts with Reinforcement Learning*. 2022. arXiv: 2205.12548 [cs.CL]. URL: https://arxiv.org/abs/2205.12548.

[14] Shizhe Diao et al. *Active Prompting with Chain-of-Thought for Large Language Models*. 2024. arXiv: 2302.12246 [cs.CL]. URL: https://arxiv.org/abs/2302.12246.

[15] Tanner Fiez et al. "Sequential Experimental Design for Transductive Linear Bandits". In: *Advances in Neural Information Processing Systems*. Vol. 32. 2019.

[16] Wikimedia Foundation. *ACL 2019 Fourth Conference on Machine Translation (WMT19), Shared Task: Machine Translation of News*. URL: http://www.statmt.org/wmt19/translation-task.html.

[17] Yao Fu et al. *Complexity-Based Prompting for Multi-Step Reasoning.* 2023. arXiv: 2210.00720 [cs.CL]. URL: https://arxiv.org/abs/2210.00720.

[18] Norbert Fuhr. “Optimum polynomial retrieval functions based on the probability ranking principle”. In: *ACM Trans. Inf. Syst.* 7.3 (July 1989), pp. 183–204. ISSN: 1046-8188. DOI: 10.1145/65943.65944. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/65943.65944.

[19] Victor Gabillon, Mohammad Ghavamzadeh, and Alessandro Lazaric. “Best arm identification: a unified approach to fixed budget and fixed confidence”. In: *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2.* NIPS’12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 3212–3220.

[20] Aaron Grattafiori et al. *The Llama 3 Herd of Models.* 2024. arXiv: 2407.21783 [cs.AI]. URL: https://arxiv.org/abs/2407.21783.

[21] Jiafeng Guo et al. “A Deep Relevance Matching Model for Ad-hoc Retrieval”. In: *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management.* CIKM16. ACM, Oct. 2016, pp. 55–64. DOI: 10.1145/2983323.2983769. URL: http://dx.doi.org/10.1145/2983323.2983769.

[22] Kalervo Järvelin and Jaana Kekäläinen. “Cumulated gain-based evaluation of IR techniques”. In: *ACM Trans. Inf. Syst.* 20.4 (Oct. 2002), pp. 422–446. ISSN: 1046-8188. DOI: 10.1145/582415.582418. URL: https://doi-org.tudelft.idm.oclc.org/10.1145/582415.582418.

[23] Shivaram Kalyanakrishnan and Peter Stone. “Efficient selection of multiple bandit arms: theory and practice”. In: *Proceedings of the 27th International Conference on International Conference on Machine Learning.* ICML’10. Haifa, Israel: Omnipress, 2010, pp. 511–518. ISBN: 9781605589077.

[24] Takeshi Kojima et al. *Large Language Models are Zero-Shot Reasoners.* 2023. arXiv: 2205.11916 [cs.CL]. URL: https://arxiv.org/abs/2205.11916.

[25] John Langford and Tong Zhang. “The Epoch-Greedy algorithm for contextual multi-armed bandits”. In: *Proceedings of the 21st International Conference on Neural Information Processing Systems.* NIPS’07. Vancouver, British Columbia, Canada: Curran Associates Inc., 2007, pp. 817–824. ISBN: 9781605603520.

[26] Xiaonan Li and Xipeng Qiu. “Finding Support Examples for In-Context Learning”. In: *Findings of the Association for Computational Linguistics: EMNLP 2023.* Ed. by Houda Bouamor, Juan Pino, and Kalika Bali. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 6219–6235. DOI: 10.18653/v1/2023.findings-emnlp.411. URL: https://aclanthology.org/2023.findings-emnlp.411/.

[27] Wang Ling et al. *Program Induction by Rationale Generation : Learning to Solve and Explain Algebraic Word Problems.* 2017. arXiv: 1705.04146 [cs.AI]. URL: https://arxiv.org/abs/1705.04146.

[28] Jiachang Liu et al. “What Makes Good In-Context Examples for GPT-3?” In: *Proceedings of Deep Learning Inside Out (DeeLIO 2022): The 3rd Workshop on Knowledge Extraction and Integration for Deep Learning Architectures.* Ed. by Eneko Agirre, Marianna Apidianaki, and Ivan Vuli. Dublin, Ireland and Online: Association for Computational Linguistics, May 2022, pp. 100–114. DOI: 10.18653/v1/2022.deelio-1.10. URL: https://aclanthology.org/2022.deelio-1.10/.

[29] Pengfei Liu et al. *Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing.* 2021. arXiv: 2107.13586 [cs.CL]. URL: https://arxiv.org/abs/2107.13586.

[30] Tie-Yan Liu. “Learning to Rank for Information Retrieval”. In: *Found. Trends Inf. Retr.* 3.3 (Mar. 2009), pp. 225–331. ISSN: 1554-0669. DOI: 10.1561/1500000016. URL: https://doi-org.tudelft.idm.oclc.org/10.1561/1500000016.

[31] Pan Lu et al. *Dynamic Prompt Learning via Policy Gradient for Semi-structured Mathematical Reasoning.* 2023. arXiv: 2209.14610 [cs.LG]. URL: https://arxiv.org/abs/2209.14610.

[32] Yao Lu et al. *Fantastically Ordered Prompts and Where to Find Them: Overcoming Few-Shot Prompt Order Sensitivity.* 2022. arXiv: 2104.08786 [cs.CL]. URL: https://arxiv.org/abs/2104.08786.

[33] Meta-Llama. *Llama 3.2 3b instruct (free) - API, providers, stats.* URL: https://openrouter.ai/meta-llama/llama-3.2-3b-instruct%3Afree?utm_source=chatgpt.com.

[34] Rodrigo Nogueira and Kyunghyun Cho. *Passage Re-ranking with BERT.* 2020. arXiv: 1901.04085 [cs.IR]. URL: https://arxiv.org/abs/1901.04085.

[35] NVIDIA. URL: https://build.nvidia.com/meta/llama-3.2-3b-instruct/modelcard.

[36] Przemyslaw Pobrotyn and Radoslaw Bialobrzeski. "NeuralNDCG: Direct Optimisation of a Ranking Metric via Differentiable Relaxation of Sorting". In: *ArXiv* abs/2102.07831 (2021).

[37] Przemyslaw Pobrotyn et al. "Context-Aware Learning to Rank with Self-Attention". In: *CoRR* abs/2005.10084 (2020). arXiv: 2005.10084. URL: https://arxiv.org/abs/2005.10084.

[38] Reid Pryzant et al. *Automatic Prompt Optimization with "Gradient Descent" and Beam Search.* 2023. arXiv: 2305.03495 [cs.CL]. URL: https://arxiv.org/abs/2305.03495.

[39] Kiran Purohit et al. "EXPLORA: Efficient Exemplar Subset Selection for Complex Reasoning". In: *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing.* Ed. by Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen. Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 5367–5388. DOI: 10.18653/v1/2024.emnlp-main.307. URL: https://aclanthology.org/2024.emnlp-main.307/.

[40] Kiran Purohit et al. *Sample Efficient Demonstration Selection for In-Context Learning.* 2025. arXiv: 2506.08607 [cs.LG]. URL: https://arxiv.org/abs/2506.08607.

[41] Tao Qin and Tie-Yan Liu. "Introducing LETOR 4.0 Datasets". In: *CoRR* abs/1306.2597 (2013). URL: http://arxiv.org/abs/1306.2597.

[42] Alec Radford et al. "Language Models are Unsupervised Multitask Learners". In: *OpenAI* (2019). Accessed: 2024-11-15. URL: https://cdn.openai.com/better-language-models/language_models_are_unsupervised_multitask_learners.pdf.

[43] Clémence Réda, Emilie Kaufmann, and Andrée Delahaye-Duriez. "Top-m identification for linear bandits". In: *Proceedings of The 24th International Conference on Artificial Intelligence and Statistics.* Ed. by Arindam Banerjee and Kenji Fukumizu. Vol. 130. Proceedings of Machine Learning Research. PMLR, Apr. 2021, pp. 1108–1116. URL: https://proceedings.mlr.press/v130/reda21a.html.

[44] Ohad Rubin, Jonathan Herzig, and Jonathan Berant. "Learning To Retrieve Prompts for In-Context Learning". In: *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies.* Ed. by Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz. Seattle, United States: Association for Computational Linguistics, July 2022, pp. 2655–2671. DOI: 10.18653/v1/2022.naacl-main.191. URL: https://aclanthology.org/2022.naacl-main.191/.

[45] Maria-Florina Balcan Soare, Alessandro Lazaric, and Rémi Munos. "Best-Arm Identification in Linear Bandits". In: *Advances in Neural Information Processing Systems.* Vol. 27. 2014, pp. 828–836.

[46] Hongjin Su et al. *Selective Annotation Makes Language Models Better Few-Shot Learners.* 2022. arXiv: 2209.01975 [cs.CL]. URL: https://arxiv.org/abs/2209.01975.

[47] Robin Swezey et al. *PiRank: Scalable Learning To Rank via Differentiable Sorting.* 2021. arXiv: 2012.06731 [cs.LG]. URL: https://arxiv.org/abs/2012.06731.

[48] Hugo Touvron et al. *Llama 2: Open Foundation and Fine-Tuned Chat Models.* 2023. arXiv: 2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

[49] Xuezhi Wang et al. *Self-Consistency Improves Chain of Thought Reasoning in Language Models.* 2023. arXiv: 2203.11171 [cs.CL]. URL: https://arxiv.org/abs/2203.11171.

[50] Jason Wei et al. *Emergent Abilities of Large Language Models.* 2022. arXiv: 2206.07682 [cs.CL]. URL: https://arxiv.org/abs/2206.07682.

[51] Jing Xiong et al. *DQ-LoRe: Dual Queries with Low Rank Approximation Re-ranking for In-Context Learning.* 2024. arXiv: 2310.02954 [cs.CL]. URL: https://arxiv.org/abs/2310.02954.

[52] Longbo Xu, Junya Honda, and Masashi Sugiyama. “A Fully Adaptive Algorithm for Pure Exploration in Linear Bandits”. In: *Proceedings of the 21st International Conference on Artificial Intelligence and Statistics*. Vol. 84. Proceedings of Machine Learning Research. 2018, pp. 843–851.

[53] Xi Ye et al. “Complementary Explanations for Effective In-Context Learning”. In: *Findings of the Association for Computational Linguistics: ACL 2023*. Ed. by Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki. Toronto, Canada: Association for Computational Linguistics, July 2023, pp. 4469–4484. DOI: `10.18653/v1/2023.findings-acl.273`. URL: `https://aclanthology.org/2023.findings-acl.273/`.

[54] Susan Zhang et al. *OPT: Open Pre-trained Transformer Language Models*. 2022. arXiv: `2205.01068 [cs.CL]`. URL: `https://arxiv.org/abs/2205.01068`.

[55] Tianyi Zhang et al. *BERTScore: Evaluating Text Generation with BERT*. 2020. arXiv: `1904.09675 [cs.CL]`. URL: `https://arxiv.org/abs/1904.09675`.

[56] Yiming Zhang, Shi Feng, and Chenhao Tan. “Active Example Selection for In-Context Learning”. In: *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*. Ed. by Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang. Abu Dhabi, United Arab Emirates: Association for Computational Linguistics, Dec. 2022, pp. 9134–9148. DOI: `10.18653/v1/2022.emnlp-main.622`. URL: `https://aclanthology.org/2022.emnlp-main.622/`.

A Appendix

A.1. Dataset prompts

AQUA Prompt

Instruction:`You are a helpful, respectful, and honest assistant helping to solve math word problems or tasks requiring reasoning or math. Follow given examples and solve the problems in step by step manner.`

Exemplars :
[Question]: *The average age of three boys is 45 years and their ages are in proportion 3:5:7. What is the age in years of the youngest boy?*
[Options]: A) 9, B) 10, C) 11, D) 12, E) 13
[Explanation]: $3x + 5x + 7x = 45$,
$x = 3$,
$3x = 9$
[Answer]: The option is A
...
...

Test Input : Question: Options:
Explanation: [INS] Answer: [INS]

Figure A.1: Prompt for Aqua

A.2. Mistral:v0.1 Results

A.3. Qualitative analysis

A.4. Mean calculation proof

Even though only a single prediction for a query is taken into account, since each time means are updated a new prediction for that same query will be obtained for each arm. By the time the number of iterations is equal the number of validation samples/queries, the information obtained by calculating means this way will be same as calculating the mean of an arm for each query individually. Thus, this approach does not result in information loss. Take matrix Y_a, predictions for arm a over time and

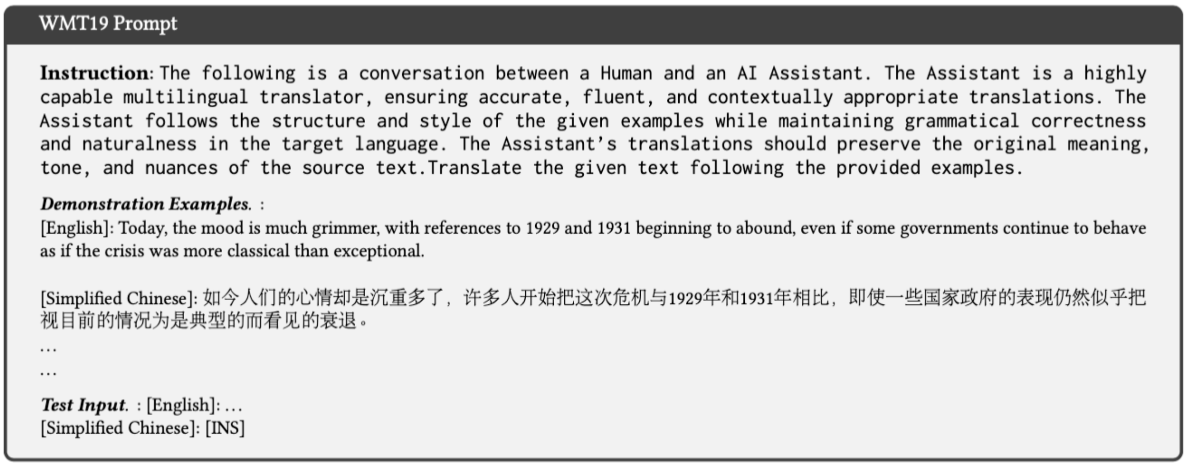
GSM8K Prompt

Instruction:`You are a helpful, respectful and honest assistant helping to solve math word problems or tasks requiring reasoning or math. Follow given examples and solve the problems in step by step manner.`

Exemplars :
[Question]: *Samir just turned half the age Hania was 10 years ago. If in five years Hania will be 45 years old, what will Samirs age be five years from now?*
[Explanation]: If in five years, Hania will be 45 years old, currently she is $45 - 5 = 40$ years old. Samir just turned half the age Hania was 10 years ago, which means she is $30/2 = 15$ years old. In five years, Samir will be $15 + 5 = 20$ years old.
[Answer]: 20 years old
...
...

Test Input : Question:
Explanation: [INS] Answer: [INS]

Figure A.2: Prompt for GSM8K



WMT19 Prompt

Instruction: `The following is a conversation between a Human and an AI Assistant. The Assistant is a highly capable multilingual translator, ensuring accurate, fluent, and contextually appropriate translations. The Assistant follows the structure and style of the given examples while maintaining grammatical correctness and naturalness in the target language. The Assistant's translations should preserve the original meaning, tone, and nuances of the source text.Translate the given text following the provided examples.`

Demonstration Examples. :
[English]: Today, the mood is much grimmer, with references to 1929 and 1931 beginning to abound, even if some governments continue to behave as if the crisis was more classical than exceptional.

[Simplified Chinese]: 如今人们的心情却是沉重多了，许多人开始把这次危机与1929年和1931年相比，即使一些国家政府的表现仍然似乎把视目前的情况为是典型的而看见的衰退。
...
...

Test Input. : [English]: ...
[Simplified Chinese]: [INS]

Figure A.3: Prompt for WMT19

different queries:

$$Y_a = \begin{array}{c|ccc} & & \xrightarrow{\text{iteration}} & \\ \hline \downarrow \text{query} & y_{11} & y_{12} & y_{13} \\ & y_{21} & y_{22} & y_{23} \\ & y_{31} & y_{32} & y_{33} \end{array} \tag{A.1}$$

Rows are predictions over time for single query and columns are predictions for different queries for a single iteration. When iteration number and query number is equal, the mean of arm a when first averaging per query is:

$$\begin{aligned} \hat{\mu}_{t=n_q} &= \frac{\frac{y_{11}+y_{12}+y_{13}}{3} + \frac{y_{21}+y_{22}+y_{23}}{3} + \frac{y_{31}+y_{32}+y_{33}}{3}}{3} \\ \hat{\mu}_{t=n_q} &= \frac{y_{11}+y_{12}+y_{13}+y_{21}+y_{22}+y_{23}+y_{31}+y_{32}+y_{33}}{9} \end{aligned} \tag{A.2}$$

And, the mean of arm a when first averaging per iteration is:

Table A.1: `mistral:v0.1` results across 3 datasets (we use 5-shot for all methods).

Method	**GSM8K**	**AquaRat**	**WTM19**
	Mistral-7B-v0.1		
Task level			
Static CASE	32.60	21.20	74.63
Zero-shot-COT	7.42	18.89	61.28
Manual Few-shot COT	22.36	14.90	-
LENS	26.08	14.17	-
Learning To Rank			
NeuralNDCG	29.33	18.90	74.80
Our Approach			
CASE Rank	34.33	23.62	76.01

Table A.2: Exemplar subsets selected by AllRank NeuralNDCG and CASE Rank using LLama3.2:3b for a particular AquaRat test instance where CASE Rank had a correct answer and AllRank NeuralNDCG model did not

Question	Three birds are flying at a fast rate of 900 kilometers per hour. What is their speed in miles per minute? [1km = 0.6 miles]
Method	**Exemplars**
AllRank	**Question:** In certain year in Country C, x sets of twins and y sets of triplets were born. If there were z total babies born in Country C in this year, and x and y were both greater than 0, which of the following represents the fraction Q of all babies born who were NOT part of a set of twins or triplets? **Question:** The perimeter of a rhombus is 68 cm and one of its diagonals is 16 cm. Find its area? **Question:** A rectangular lawn of dimensions 90 m × 60 m has two roads each 10 m wide running in the middle of the lawn,one parallel to the length and the other parallel to the breadth. What is the cost of traveling the two roads at Rs.3 per sq/m? **Question:** a is the hundreds digit of the three digit integer x, b is the tens digit of x, and c is the units digit of x. 3a = b = 3c, and a > 0. What is the difference between the two greatest possible values of x? **Question:** he population of a town is 10000. It increases annually at the rate of 20% p.a. What will be its population after 3 years?
CASE Rank	**Question:** Machine A can make 350 widgets in 1 hour, and machine B can make 250 widgets in 1 hour. If both machines work together, how much time will it take them to make a total of 800 widgets? **Question:** If two trains are 120 miles apart and are traveling toward each other at constant rate of 30 miles per hour and 40 miles per hour, respectively, how far apart will they be 1 hour before they meet? **Question:** If 2a = 3b and ab0, what is the ratio of a/3 to b/2? **Question:** An businessman earns an income of Re 2 on the first day of his business. On every subsequent day, he earns an income which is just double of that made on the previous day. On the 20th day of business, he earns an income of? **Question:** Four of the five parts numbered (a), (b), (c), (d) and (e) are exactly equal. Which of the parts is not equal to the other four? The number of that part is the answer.

Rationale is not completely shown for some questions to conserve space. However, in our experiments all exemplars include rationales.

$$\begin{aligned}\hat{\mu}_{t=n_q} &= \frac{\frac{y_{11}+y_{21}+y_{31}}{3}+\frac{y_{12}+y_{22}+y_{32}}{3}+\frac{y_{13}+y_{23}+y_{33}}{3}}{3}\\ \hat{\mu}_{t=n_q} &= \frac{y_{11}+y_{21}+y_{31}+y_{12}+y_{22}+y_{32}+y_{13}+y_{23}+y_{33}}{9}\end{aligned} \tag{A.3}$$

Table A.3: Exemplar subsets selected by AllRank NeuralNDCG and CASE Rank using LLama3.2:3b for a particular AquaRat test instance where CASE Rank had a correct answer and AllRank NeuralNDCG model did not

Question	Let A, B and C denote the vertices of a triangle with area 10. Let point D be on side AB, point E be on side BC and point F be on side CA with AD = 2 and DB = 3. The area of $\triangle ABE$ and the area of quadrilateral DBEF are the same. What is the value of this area?
Method	**Exemplars**
AllRank	**Question:** In a certain tournament certain rules are followed: any player is eliminated the moment he losses for the third time (irrespective of how many wins he has), any player can play another player any number of times. If 512 contestants enter the tournament what is the largest number of games that could be played? **Question:** Car A runs at the speed of 33 km/h & reaches its destination in 7 h. Car B runs at the speed of 44 km/h & reaches its destination in 5 h. What is the respective ratio of distances covered by Car A & Car B? **Question:** What is the sum of all the 4 digit numbers that can be formed using all of the digits 2, 3, 5 and 7? **Question:** A certain companys profit in 1996 was 20 percent greater than its profit in 1995, and its profit in 1997 was 20 percent greater than its profit in 1996. The companys profit in 1997 was what percent greater than its profit in 1995? **Question:** Evaluate: $13 + \sqrt{-4 + 5 \times 3 \div 3} = ?$
CASE Rank	**Question:** The ratio of number of boys and girls in a class is 3 : 2. In the 1st semester exam 20% of boys and 25% of girls get more than or equal to 90% marks. What percentage of students get less than 90% marks ? **Question:** Two trains of length 120 m and 280 m are running towards each other on parallel lines at 42 km/h and 30 km/h respectively. In what time will they be clear of each other from the moment they meet? If $5^{25} \cdot 4^{13} = 2 \times 10^n$, what is the value of n? **Question:** If $5^{25} \cdot 4^{13} = 2 \times 10^n$, what is the value of n? **Question:** Rs. 1200 divided among P, Q and R. P gets half of the total amount received by Q and R. Q gets one-third of the total amount received by P and R. Find the amount received by R? **Question:** What is the greatest 6-digit number when divided by 6, 7, 8, 9, and 10 leaves a remainder of 4, 5, 6, 7, and 8 respectively?

Rationale is not completely shown for some questions to conserve space. However, in our experiments all exemplars include rationales.
The test instance is ratio problem that requires algebra. Subset selected CASE Rank by contains two similar questions(1.,4.), whereas AllRank NeuralNDCG one(2.).

Since addition is commutative, it can be concluded that equations are the same. This means that the information seen for the means is exactly the same for an arm when enough iterations have elapsed. In the case when the number of iterations is not equal to number of queries, there are two possibilities: number of iterations is either larger or lower than number of queries. When the number of iterations is lower than number of queries than it means that there is less information per arm than there are arms. This may introduce uncertainty in mean calculation in first iterations. In the other case, there should not be any uncertainty coming from this approach since there have been enough predictions contributed for an arm mean.

A.5. Inference time speed comparison between CASE Rank and AllRank Neural NDCG

Table A.4: Exemplar subsets selected by AllRank NeuralNDCG and CASE Rank using LLama3.2:3b for a particular AquaRat test instance where CASE Rank had a correct answer and AllRank NeuralNDCG model did not

Question	Glenn gains 10% from selling her scarf at $44. Her mom tells her she'll buy it from her but only after a family discount of 5%. What is Glenn's gain percent if she sells the scarf to her mom?
Method	**Exemplars**
AllRank	**Question:** Dawson and Henry are in a relay race. Dawson runs the first leg of the course in 38 seconds. Henry runs the second leg of the course in 7 seconds. What was the average time they took to run a leg of the course? **Question:** A man can row his boat with the stream at 16 km/h and against the stream in 12 km/h. The man's rate is? **Question:** Can you find the smallest non fractional number such that If the number gets divided by 9 , we get the remainder of 1; If the number gets divided by 10 , we get the remainder of 2 If the number gets divided by 11 , we get the remainder of 3; If the number gets divided by 12 , we get the remainder of 4. **Question:** Ravi purchased 20 dozens of toys at the rate of Rs. 360 per dozen. He sold each one of them at the rate of Rs. 33. What was his percentage profit? **Question:** If w,x,y,z represents the respective results obtained from rounding off the figure 8623.293 to the nearest thousand, hundred, ten and one, which of the following statements is accurate?
CASE Rank	**Question:** If population of certain city increases at the rate of 3%. If population in 1981 was 138915, then population in 1980 was? **Question:** The train travels at an average speed of 25km/h, to the top of the hill where the midpoint of the trip is. Going down hill, train travels at an average speed of 5km/h. Which of the following is the closest approximation of train's average speed, in kilometers per hour, for the round trip? **Question:** For a,w,d are the positive integers, and d|a means that a is divisible by d, if d|aw, which of the following must be true? **Question:** A person purchased a TV set for Rs. 16000 and a DVD player for Rs. 6250. He sold both the items together for Rs. 34710 . What percentage of profit did he make? **Question:** Which number is the odd one out ? 9654 4832 5945 7642 7963 8216 3649

Rationale is not completely shown for some questions to conserve space. However, in our experiments all exemplars include rationales.
The test instance is applied percentage / profitdiscount problem. Subset selected by contains two similarques-tions(3.,4.), whereas AllRank NeuralNDCG one(4.).

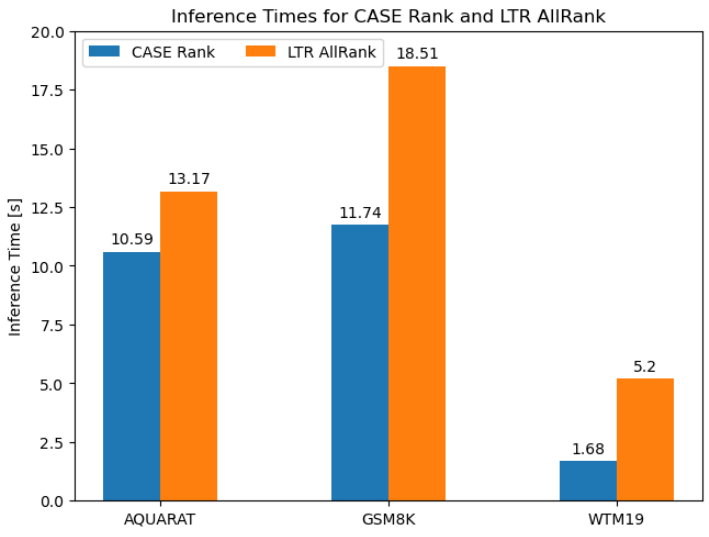


Figure A.4: Inference Time comparison between CASE Rank and AllRank Neural NDCG

Table A.5: Exemplar subsets selected by AllRank NeuralNDCG and CASE Rank using LLama3.2:3b for a particular AquaRat test instance where CASE Rank had a correct answer and AllRank NeuralNDCG model did not

Question	David bought 13 BMW cars for a total price of 1,105,000 dollars. If he wants to make a profit of 39,000 dollars in the deal, at what price should he sell one car?
Method	**Exemplars**
AllRank	**Question:** If the length of a rectangular field is 30 metres more than its breadth and the perimeter of the field is 540 metres, what is the area of the field in square metres? **Question:** Express a speed of 56 kmph in meters per second? **Question:** What is the smallest positive integer x such that 108x is the cube of a positive integer? **Question:** A man purchased 3 blankets @ Rs.100 each, 5 blankets @ Rs.150 each and two blankets at a certain rate which is now slipped off from his memory. But he remembers that the average price of the blankets was Rs.150. Find the unknown rate of two blankets? **Question:** Each of the integers from 0 to 8, inclusive, is written on a separate slip of blank paper and the ten slips are dropped into hat. If the slips are then drawn one at a time without replacement, how many must be drawn to ensure that the numbers on two of the slips drawn will have a sum of 10?
CASE Rank	**Question:** The calender for the year 2007 will be the same for the year: **Question:** A train 100 m long crosses a platform 125 m long in 15 sec; find the speed of the train? **Question:** If x=2y=4z, what is y-z, in terms of x? **Question:** An investor can sell her MicroTron stock for 36$ per share and her Dynaco stock for 60$ per share, If she sells 300 shares altogether, some of each stock, at an average price per share of 40$, how many shares of Dynaco stock has she sold? **Question:** The letters of the word WOMAN are written in all possible orders and these words are written out as in a dictionary ,then the rank of the word 'WOMAN' is

Rationale is not completely shown for some questions to conserve space. However, in our experiments all exemplars include rationales.
The test instance is applied profit/markup problem that requires algebra. Subset selected by CASE Rank contains two similar questions(1.,1.), whereas AllRank NeuralNDCG one(4.).