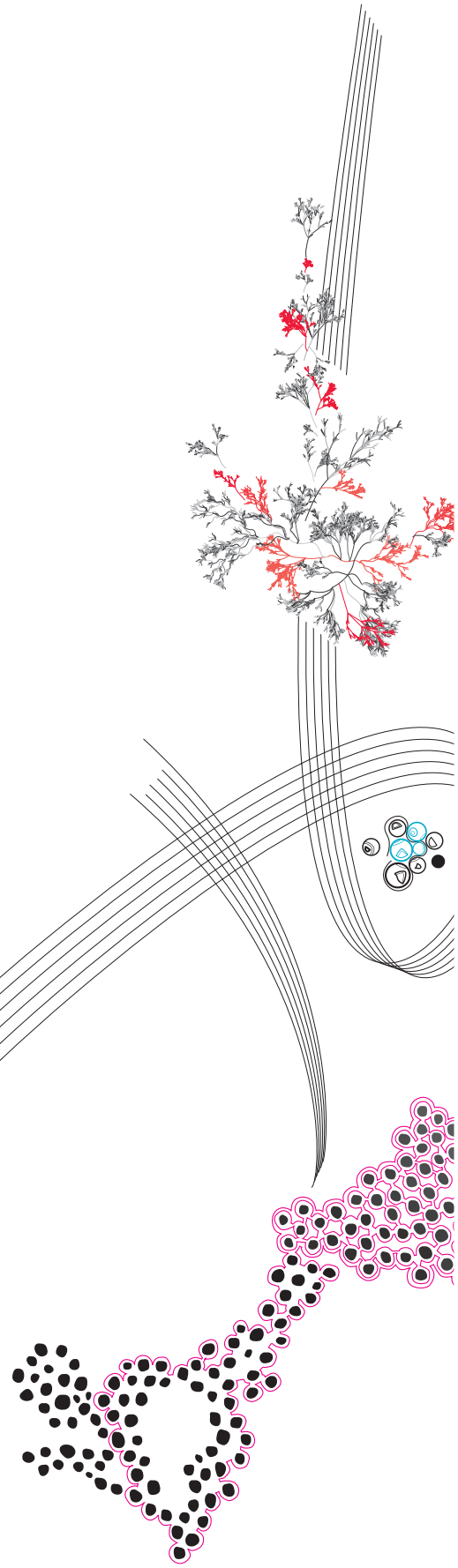


MSc Computer Science
Final Project

How Hard Can It Prompt? Adventures in Cross-model Prompt Transferability

Lola Solovyeva

Supervisor: Dr. ing. Gwenn Englebienne
Dr. ir. Luuk Spreeuwers

August, 2024

Department of Computer Science
Faculty of Electrical Engineering,
Mathematics and Computer Science,
University of Twente

UNIVERSITY OF TWENTE.

Contents

Abstract

Recently, the concept of prompting has been introduced as a mechanism to provide contextual cues to the model, thereby directing its focus towards the downstream task solely through interaction with an input. The concept has evolved into prompt-tuning, a process in which human intervention is eliminated, and the model autonomously adjusts the prompt parameters in pursuit of optimizing its final performance. Furthermore, the advent of prompt transfer has notably advanced performance in prompt-tuning. However, existing prompt transfer methodologies are primarily based on soft prompts, which lack interpretability and exhibit limited transferability due to their reliance on the model's architecture. This study introduces Gradient Guided Discrete Prompt Search (GGDPS), which combines the interpretability and ease of transferability of hard prompts with the expressiveness of soft prompts. GGDPS uses gradients from soft prompts to guide the model in selecting optimal hard tokens, allowing for autonomous prompt generation without human intervention. The resulting hard prompts are easily transferable between models and require no architectural modifications. We evaluate the feasibility and applicability of GGDPS, finding that the resulting hard prompts can match or even surpass baseline results on certain datasets. Additionally, we evaluate the transferability of the generated prompts, revealing their success is depended on the pre-trained knowledge of the model. To address this challenge, we introduce Collaborative GGDPS, which enhances generalizability and robustness by creating shared prompts across various models.

Keywords: hard prompts, soft prompts, gradient-guided search, prompt transfer, interpretability

Chapter 1

Introduction

The emergence of Large Language Models (LLMs) has revolutionized the field of artificial intelligence, advancing the limits of the state-of-the-art results and attaining increased levels of efficiency amidst various domains [53]. Despite its widespread application, conventional fine-tuning through gradient updates remains computationally expensive and time-consuming as it requires updating all the model parameters. As the number of parameters in pre-trained LLM continues to increase, so does the corresponding demand for computational resources. For instance, fully fine-tuning the Falcon model with 180 billion parameters [1] would require a minimum allocation of 5120GB of computational resources [61]. To address the computational burden of full-model tuning, researchers shifted their attention towards identifying parameter-efficient techniques capable of updating solely a subset of parameters, while keeping the rest untouched [17, 61].

The emergence of GPT-3 showcased the capability of LLMs to execute various NLP tasks without any gradient updates, relying solely on textual interactions with the model, forming a new concept of prompting [4]. This includes supplying instructions and examples for the desired task as input to the model. The prompt provided with an additional instruction can alternatively be referred to as a **hard** or **discrete prompt**, as it comprises natural language tokens and exists within the discrete space. The model then generates an output based on these instructions and examples, without necessitating any additional training or modifications to the pre-trained model. Despite their training solely on language modeling objectives, these models demonstrate notable proficiency in novel tasks for which they have not received explicit training. Nonetheless, simple prompting has its drawbacks. The process of writing an instruction that could maximize the model performance is prone to errors and proven to be time-consuming even for experts [24]. Also, the difference has been observed between human and machine comprehension, thereby leaving room for erroneous interpretations of the intended purpose of a given prompt [58].

Hence, the concept of prompting has progressed into prompt-tuning, involving the addition of a small number of parameters to specific parts of the model, with the objective of identifying the most optimal prompt capable of eliciting the desired final output from the model [24, 27, 32]. These continuous vectors, integrated into the input at the model's embedding layer, are referred to as **soft** or **continuous prompts**. They offer enhanced flexibility and adaptability during the fine-tuning process, as they are the only components subjected to gradient updates. Taking gradient steps within the continuous space provides a larger search space, thereby increasing the likelihood of identifying a more optimal set of vectors that can effectively elicit the desired output. In terms of performance, it demonstrated more favorable results when compared with full fine-tuning across the majority of tasks within the GLUE[57] and SuperGLUE[56] benchmarks. Moreover, they consistently

outperform hard prompts according to evaluation metrics such as accuracy and precision. Nevertheless, some challenges remain, as the aforementioned benchmarks merely scratch the surface of all possible tasks.

Prompt-tuning encounters challenges in achieving competitive performance relative to full-model tuning or alternative parameter-efficient methodologies, particularly in scenarios characterized by limited data or the utilization of relatively small models. Transfer learning remains a dominant and preferred approach to deal with those issues, demonstrating substantial improvements, thereby facilitating a shift from full-model tuning towards a more parameter-efficient methodology. The introduction of prompt transfer particularly marked a breakthrough in terms of performance for prompt-tuning. It succeeded in mitigating certain vulnerabilities associated with prompt-tuning, such as sensitivity to prompt initialization [12, 18], high variability in a few-shot setting[12, 54], slow convergence[69], and the ability to generalize across multiple tasks[2].

1.1 Problem statement

Prompt transfer remains a critical area of research due to its focus on reducing the costs associated with model tuning. Given the continual growth of LLMs, transferable prompts can eliminate the need for retraining and the development of new algorithms tailored to each newly introduced model. However, most prompt transfer currently occurs within models of similar characteristics (e.g. from T5-base to T5-XL). Therefore, expanding to cross-model transfer would be significantly impactful. It enhances scalability by enabling new models to leverage the knowledge and capabilities of existing prompts, thereby facilitating quicker deployment and adaptation. This is particularly crucial, as different models may be employed for the same task, with the choice of model dependent on the constraints imposed by the deployment environment. Ideally, prompts would be trained on smaller models but utilized on larger ones, leveraging their greater expressive power and superior performance.

Currently, there are numerous obstacles to achieving cross-model prompt transfer. Hard prompts do not match the performance of full-model tuning. Additionally, identifying the optimal instructions to maximize performance remains labor-intensive. On the other hand, while soft prompts yield better results, their transferability is restricted to the embedding space. Since they are appended to the embedded input, their dimension is dependent on the embedding size of the model, which can vary significantly. For instance, soft prompts trained on a model with a smaller embedding space cannot be transferred to a model with a larger embedding space. Moreover, soft prompts lack interpretability. A work by Bailey et al. [3] demonstrated that interpreting soft prompts through simplistic mappings to natural language prompts can lead to significant misrepresentation, as the resultant natural language prompts fail to encapsulate the effectiveness of the original soft prompts. This issue can facilitate malicious manipulation, resulting in undesirable behaviors LLMs[64, 71]. Hence, understanding the information encoded within soft tokens is crucial, as it would aid in identifying malicious soft prompts.

1.2 Objective

In this study, our goal is to merge the interpretability and transferability of a hard prompt with the expressiveness of a soft prompt. We propose Gradient Guided Discrete Prompt Search (GGDPS), which leverages the gradient from a soft prompt to guide the model in selecting hard tokens that optimize final performance. By utilizing the gradient from a

soft prompt, GGDPS enables the model to autonomously determine the most suitable set of hard tokens for the task at hand, eliminating the need for human prompt engineering. The resulting hard prompt can be easily transferred to homogeneous or heterogeneous models, as it does not necessitate any modifications to the model's architecture and can be seamlessly concatenated with the input data during preprocessing.

1.3 Research Questions

To validate our algorithm, we formulate specific research questions designed to test the feasibility and applicability of the proposed GGDPS algorithm. In addition to testing the algorithm itself, we assess the resulting hard prompts and evaluate their transferability. Furthermore, we introduce an extension to the GGDPS algorithm, termed Collaborative GGDPS, which aims to create a shared prompt across various models, enhancing its generalizability and robustness. Research questions are formulated as follows:

- How can the tokens of soft prompts be mapped or translated into natural language tokens?
- How can a model autonomously produce a hard prompt tailored to a specific task, without requiring human intervention?
- What impact does transferring a hard prompt from one model to another have on performance?
- What are the effects of collaborative shared hard prompt generation between two models on achieving optimal performance for both?

Chapter 2

Literature review & Background

This chapter offers additional insights into the field of this research, aiming to equip the reader with essential background information. It introduces relevant concepts such as prompting, and the distinctions between hard prompts and soft prompts. In addition to providing foundational knowledge, this chapter also reviews related work, thereby situating the current study within the broader research context.

2.1 Background

The pioneering work by Redford et al.[40], has illustrated that large language models can execute downstream tasks in a zero-shot scenario via in-context learning, using the text input of a pre-trained language model as a form of task specification. The model is conditioned on the instruction formed in natural language, called **hard** or **discrete prompt**, and is given a few examples of the input-output pairs, then the model is expected to behave accordingly. Although demonstrating decent performance, the results remained inferior compared to conventional fine-tuning procedures. A work by Brown et al.[4] has noted that the performance gap between in-context learning and fine-tuning could be reduced with scale. This suggests that the performance difference between full-model tuning and prompt tuning diminishes as the model size increases. Their research introduced GPT-3 with 175 billion parameters, showcasing robust performance across various tasks, nearly matching the performance of state-of-the-art fine-tuned models. Nevertheless, it still struggles under one-shot or few-shot setting. It only realizes its full potential under gradient-based training on a handful of labeled examples[44].

The most straightforward approach to enhancing the performance of prompting involves modifying the prompt template based on human intuition regarding what would be logically coherent and could optimize the ultimate outcome. For example, Brown et al.[4] have manually created prefix prompts to support wide variety of NLP tasks. Schick and Schutze[43] proposed to reformulate tasks into a cloze-style questions and use the verbalizer to leverage the knowledge contained in the pre-trained model for assigning soft labels to the large corpora, facilitating a supervised training.

Even though there was a line of work relying on searching for optimal prompt in the discreet space, it was shown that this approach necessitates a heavy dependence on prior knowledge and may pose challenges even for experts [19]. Furthermore, language models may encounter difficulties in comprehending prompts the same way as humans do[58]. With the aim to improve upon the issues, **soft** or **continuous prompts** were introduced, characterized by their trainable embeddings that could condense the knowledge in fewer parameters in comparison to the full model fine-tuning.

Lester et al.[24] introduced the idea of soft prompts, that optimizes model performance on specific tasks by adjusting a small set of prompt embeddings instead of modifying the entire model. This method significantly reduces computational costs and the number of tunable parameters, making it more efficient and scalable compared to traditional fine-tuning.

2.2 Related work

This section reviews previous academic research relevant to the current study. Given that the focus of this work is on conducting hard prompt search using gradients from prompt-tuning, this section is divided into two parts, each dedicated to one of these topics. Subsection 2.2.1 reviews studies that have previously attempted automated hard prompt search, with a focus on the most commonly used approaches. Meanwhile, Subsection 2.2.2 introduces various existing prompt-tuning methods that could be utilized to conduct hard prompt search within the continuous space.

2.2.1 Automated hard prompt search

Automated hard prompt search represents a crucial area within prompting paradigm. Hard prompts, composed of discrete tokens, offer interpretability and ease of transferability across different models. However, they pose a significant challenge in terms of automated generation. Several studies have endeavored to automate the process of hard prompt search through various methodologies. In this study, we categorize the process into two distinct methodologies: gradient-based search and prompt generation.

Gradient-based search. A work by Wallace et al.[55] performed a gradient-based search over tokens to find short trigger sequences that could extract a desired knowledge from the pre-trained LM. The search proceeds iteratively, advancing through the tokens within the prompt in a sequential manner. However, their study focuses on identifying adversarial triggers that elicit negative sentiment from the model, whereas our study aims to find a prompt that maximizes a task performance. A later work by Hambardzumyan et al.[13] was inspired by adversarial reprogramming, that make injections into the input to make the model behave differently from what it was trained for. WARP focuses on searching for ideal prompt in continuous space with respect to the input instances, updating the prompt and verbalizer token embeddings. However, their hard prompt is composed of nearby tokens vectors to the identified embeddings. In contrast, our method updates the gradient with respect to the hard tokens identified by the model.

Prompt generation. Shin et al. [48] introduced an automated approach for prompt generation, termed AutoPrompt. This method synthesizes prompts by combining original task inputs with a collection of learnt trigger tokens according to a template. Essentially, AutoPrompt functions as a search algorithm within the discrete space of words, directed by the training data of the downstream application. Compared to AutoPrompt, our approach searches for prompts in the continuous space, allowing for greater expressiveness. Furthermore, their study was conducted only on BERT models [6]. A later work by Zhang et al. [66] utilized generative model BART[25] to dynamically produce prompts based on the provided input sequence. The original pre-trained LM remains frozen, while PromptGen, comprising an encoder and autoregressive decoder, is subject to training. In contrast, our approach does not require an additional module for prompt generation, thereby reducing the number of trainable parameters during training.

2.2.2 Prompt-tuning

Even though the prompt-tuning method, developed by Lester et al. [24], obtained competitive results to fine-tuning when trained on a single dataset and had access to a large enough model, its performance diminishes in multi-task scenarios. Vue et al. [54] showcased that prompt embeddings acquired for various NLP tasks exhibit significant spatial separation within the embedding space, suggesting the absence of a singular prompt capable of performing equally well across all tasks. This finding suggests the necessity for methodologies that can take into account both the specific task and input at hand to generate more tailored prompts.

Instance-dependent prompt-tuning. Work by Wu et al.[60] introduced a methodology for generating instance-dependent prompts for language understanding tasks employing a lightweight bottleneck architecture. A caveat of their methodology is that their method necessitates two sequential forward passes: firstly, to obtain an input representation from the pre-trained language model, and secondly, to transmit this representation to prompt generation, followed by a conventional classification task pipeline. Jin et al.[20] introduced a more streamlined methodology for instance-aware prompts. They posited that each trainable prompt token exerts distinct influence across various instances, thus they sought to ascertain these contributions by computing the relevance score between an instance and each prompt token.

Meanwhile, a work by Liu et al.[31] sought to address an additional limitation of simple prompt-tuning: slow convergence rate. They conjectured that the poor performance of prompt tuning stems from the extensive propagation path of task-specific information, resulting in significant loss of task-relevant data during propagation within the frozen model, thereby impacting performance. This hypothesis inspired the development of late prompt-tuning, which only inserts the prompt within the intermediate layer of the model as opposed to the input layer. Additionally, the approach harnesses all preceding layers as valuable knowledge, which is utilized for the generation of instance-aware prompts. Although the approach demonstrated improved convergence rate, the resultant enhancement in performance was not substantial, often being surpassed by full-model tuning. Moreover, the process of selecting the appropriate layer may entail prolonged execution times to ascertain its effect on final accuracy.

Prompt transfer. A work by Vu et al.[54] leverage existing tasks as source tasks, demonstrating the benefits of prompt transfer even in scenarios where dissimilarities exist between the source and target tasks. Similar study was introduced by Asai et al.[2] called ATTEMPT, which is method that employs multiple source prompts for downstream tasks. This approach eliminates the trial-and-error search for the best source task and leverages insights from various related source tasks. They employed a basic attention mechanism aimed at directing attention towards useful information with the respect to the provided input, making the approach instance-dependent. However, their studies primarily focuses on transferring soft prompts within a task, whereas our research investigates the transfer of hard prompts across different models.

2.3 Preliminaries on Soft Prompts

2.3.1 Definition

Initially, prompting was based on the concept of prepending extra information for the model to condition on during the generation of the final output. A classification task could be formalized as $P_\theta(Y|X)$, where X and Y are a sequence of tokens, representing input

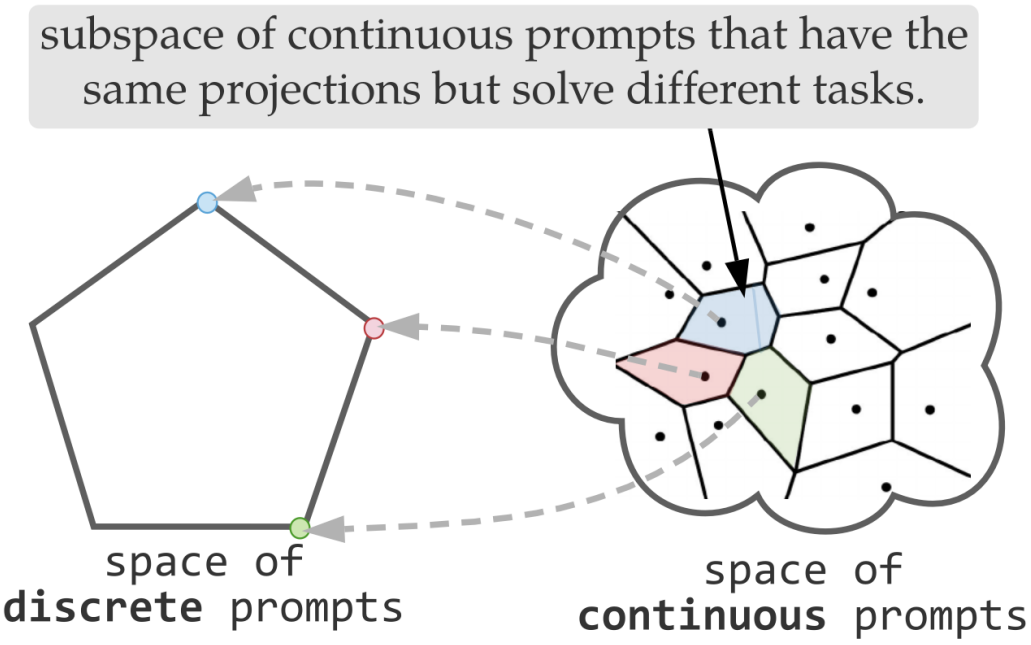


FIGURE 2.1: The projection from the continuous space onto the discrete space results in a clustering effect. Each cluster contains an infinite number of points that are mapped to the same discrete token [22].

and class label respectively, and θ as a set of weights in the chosen model. Prompting is added into the formulation by prepending a series of tokens P, to the input X, such that the model maximizes the likelihood of the correct label Y, while keeping the model parameters θ fixed, resulting in $P_\theta(Y|[P;X])$.

In prompt-tuning, instead of a prompt P being parameterized by model parameters θ, it is assigned its own dedicated parameters θ_P, that are being updated independently. A tunable prompt $P \in \mathbb{R}^{n \times e}$ is seen as a set of embedded tokens $\{p_1, p_2, ..., p_n\}$, where n is a number of tokens and e is the dimension of the embedding space. It is then concatenated to the embedded input, $X \in \mathbb{R}^{l \times e}$ with l being an length of an input sequence, forming a single matrix $[P;X] \in \mathbb{R}^{(n+l) \times e}$. Therefore, the formulation described above for the classification task is restated as $P_{\theta,\theta_P}(Y|[P;X])$ and can be trained to maximize the likelihood of Y via backpropogation, by applying gradient updates solely to θ_P, while the model parameters θ remain frozen.

2.3.2 Mechanics of Prompts

Expressiveness

The success of soft prompting is often attributed to the greater capacity of continuous embeddings compared to discrete tokens. Continuous embeddings can represent a wider and more nuanced range of information compared to the finite set of discrete tokens. Because soft prompts function within the same embedding space as the model's internal representations, they can more seamlessly integrate with the model's pre-trained knowledge, offering greater expressiveness compared to hard prompts. Khashabi et al.[22] have shown that a single discrete prompt can correspond to only one continuous prompt through its embedding, while the reverse does not hold. This indicates that numerous continuous prompts

could potentially map to a given discrete prompt, which can be observed on Figure 2.1. Thus, during the tuning process, the model learns the most suitable representation of the discrete prompt within the continuous space, aligning effectively with the pre-trained knowledge.

Attention Allocation

Soft prompting directs the model towards desired output with minimal parameter updates in contrast to full model tuning. This is achieved by biasing the model with additional information that redirects attention away from less relevant aspects of the input bringing more valuable parts of it to the surface. For a more formal definition, it is necessary to revisit the attention mechanism within the transformer architecture. Each attention block consists of H heads, which is parameterized by query, key and value matrices W_Q^h, W_K^h, W_V^h. The attention matrix $A^h \in \mathbb{R}^{p \times p}$ for head h can be formalized as follows:

$$A_{i,j}^h = \frac{\exp\left((W_Q^h x_i)^T (W_K^h x_j)\right)}{\Sigma_{r=1}^n \exp\left((W_Q^h x_i)^T (W_K^h x_r)\right)}, \tag{2.1}$$

where $A_{i,j}^h$ represents the attention that position i gives to the position j and n is a length of the current input.

To examine the impact of a prompt on attention, let us introduce a prompt p of length one onto position 0 of the input.

$$A_{i,0}^h = \frac{\exp\left((W_Q^h x_i)^T (W_K^h p)\right)}{\exp\left((W_Q^h x_i)^T (W_K^h p)\right) + \Sigma_{r=1}^n \exp\left((W_Q^h x_i)^T (W_K^h x_r)\right)}, \tag{2.2}$$

$$A_{i,j}^h = \frac{\exp\left((W_Q^h x_i)^T (W_K^h x_j)\right)}{\exp\left((W_Q^h x_i)^T (W_K^h p)\right) + \Sigma_{r=1}^n \exp\left((W_Q^h x_i)^T (W_K^h x_r)\right)}, \text{for } j > 0. \tag{2.3}$$

As it can be observed the original formulation of the attention solely changed in the denominator by adding a term $\exp\left((W_Q^h x_i)^T (W_K^h p)\right)$. Therefore, the attention position i gives to the positions $j > 0$ simply scaled down by the attention it now gives to the prefix. Essentially, if a token receives substantial attention from other tokens in a specific context, no prepended prompt can alter this. Hence, soft-tuning cannot modify the relative attention patterns within the content, it can only attenuate their magnitude.

Knowledge Elicitation

Since soft-tuning cannot alter attention patterns, its capacity for task learning is constrained by the pre-trained knowledge of the model. Petrov et al.[39] demonstrated that if the task to be learned is not related to the model's pre-trained knowledge, soft-tuning fails to achieve satisfactory performance, while full-model tuning is successful, ultimately modifying the attention patterns. This is supported by an experiment conducted by Hao et al.[15], which demonstrated that full-model fine-tuning preserves the attention distribution in the lower layers consistent with those of the original model, while adjusting it in the higher layers to adapt the model to specific tasks. However, soft prompts do not seem to significantly modify the attention distribution throughout any layers. This statement

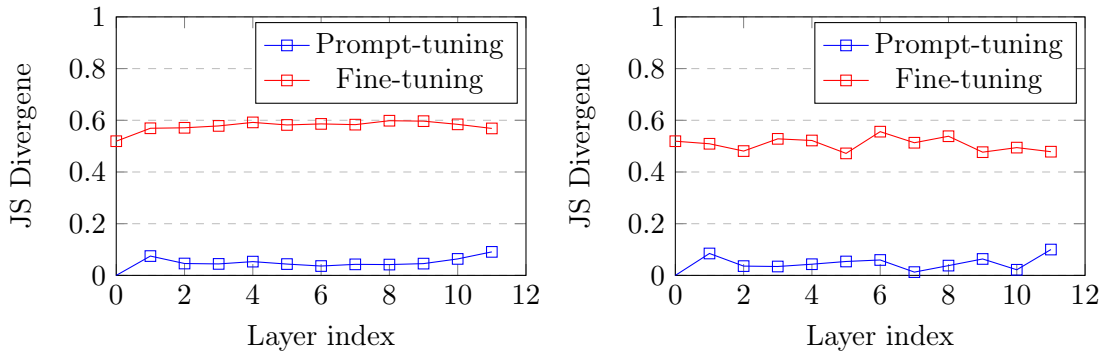


FIGURE 2.2: JS divergence of attention scores of every layer between the original T5 model and the modified model: prompt-tuning and full model tuning. Right figure presents results on MRPC dataset, whilst left on RTE dataset.

could be confirmed using Jensen-Shannon divergence as a metric to measure the difference between two distributions. The JS divergence is defined as follows:

$$D_{JS} = \frac{1}{2}D_{KL}(P||R) + \frac{1}{2}D_{KL}(Q||R), \tag{2.4}$$

where P and Q are two different probability distributions and R is the average between them, with D_{KL} representing a Kullback-Leibler divergence.

Results of the measurements could be observed on Figure 2.2, that illustrates the distance between the probability distributions of the modified model, whether through prompt-tuning or full-model tuning, and the original pre-trained model. It can be observed that the probability distribution of the model utilizing soft-tuning closely resembles that of the original model, whereas the model with all parameters fine-tuned exhibits a noticeably different distribution. Hence, prefix-tuning cannot effectively adapt a model to a new unseen task unless the model possesses related pre-trained knowledge.

Chapter 3

Methodology

This chapter elaborates on the approach to addressing the defined research questions. The first section outlines the methodology, which centers on discretizing soft prompts by leveraging cosine similarity between the embeddings of soft and hard tokens. The second section introduces the algorithm designed to identify a set of hard tokens using gradients obtained through the tuning of soft prompts. The subsequent section provides insights into the setup that enables testing the transferability of the derived hard prompts between different models. Finally, the chapter concludes by defining a configuration that facilitates the discovery of a shared discrete prompt applicable across multiple models.

3.1 Discretization of soft prompts

This section details the approach for identifying a natural language equivalent to soft prompts. It begins by discussing the rationale behind the discretization of soft prompts, highlighting their existing limitations, and concludes with the methodology employed in the experimentation process.

3.1.1 Motivation

Given the notably successful results of prompt-tuning and its relatively low number of tunable parameters, its adaptation to a wide range of machine-learning problems becomes evident. However, their adaptation is not as rapid as might be expected, due to challenges related to their interpretability and explainability[3]. As discussed in the Section 2.3, soft prompts are essentially sequences of numbers that prompt the model to generate a desired output. However, these sequences are not comprehensible to the user, and the impact of altering even a single digit within them is unclear. Specifically, this presents a potential vulnerability for adversarial attacks on LLMs that are challenging to detect and mitigate [64, 71]. The model can be maliciously manipulated to exhibit undesirable behaviors through the strategic use of soft prompts, which are difficult for humans to identify. Therefore, it is desirable to understand the information encoded within the soft tokens, as this would also facilitate the identification of malicious soft prompts. Additionally, since soft prompts are model-specific, with their vector dimensions tied to the model's embedding space, they cannot be easily transferred between different models. The aforementioned drawbacks could be addressed by discretizing the soft prompt. Hard prompts consist of natural language tokens and can be easily appended to the input text to be used across different models.

3.1.2 Approach

The most straightforward approach to interpreting soft prompts involves identifying their nearest neighbors among the embeddings of discrete tokens. This is achieved by utilizing the model's embedding layer to construct a repository of embeddings for all natural tokens in the model's vocabulary, followed by computing the cosine similarity between the soft tokens and the embeddings in this repository. Upon identifying the nearest discrete alternative, which yields a discretized soft prompt, we employ this discrete prompt to assess the model's performance. Specifically, each discrete prompt was appended to the input text, treated as a hard prompt, and then inputted into the model for evaluation. This approach evaluates whether embeddings of natural language tokens can effectively serve as discrete alternatives to soft prompts.

Despite soft prompts being non-interpretable in their raw form to humans, they hold substantial significance for the model. To explore this significance, we analyze their representation in the last layer of the encoder. This comes from the idea that words, similar in meaning, tend to have similar representations within the encoder of a language model [16, 42]. This phenomenon arises because language models are trained to encode semantic and syntactic similarities among words in their embedding spaces. As a result, words with similar meanings often exhibit closer proximity in their vector representations within the encoder. We compute the cosine similarity between the vectors in the encoder output corresponding to the soft tokens and those corresponding to the input. Subsequently, we analyze the resulting discrete prompt from both semantic and syntactic perspectives before incorporating it into the model for further performance evaluation.

3.2 Autonomous hard prompt generation

This section seeks to address the second research question, which focuses on enabling a model to autonomously identify a set of discrete prompts, by introducing an algorithm for Gradient-Guided Discrete Prompt Search (GGDPS). Initially, we elucidate the rationale underpinning the development of this algorithm, followed by a detailed introduction to its structure and functionality.

3.2.1 Rationale behind GGDPS

The previous chapter highlighted characteristics of soft prompts, revealing that they cannot be directly mapped to natural tokens through the embedding space without losing their intrinsic information that is meaningful to the model. However, the primary advantage of soft prompts lies in their ability of being tuned by the model, which means that a model autonomously selects a set of vectors from continuous space that directs its attention to the more significant parts of the input, perfecting the performance. Therefore, rather than directly converting soft prompts into discrete alternatives, we aim to investigate whether the model can autonomously select a set of discrete tokens that optimize its performance to the same extent as soft prompts.

The approach involves allowing the model to identify a set of optimal vectors within the continuous space, leveraging its expressive and flexible search space for optimization algorithms. The main difference from conventional prompt-tuning lies in mapping these vectors to discrete tokens via the model's linear head, rather than directly concatenating them with the input embeddings. Khashabi et al.[22] have identified that numerous continuous prompts can correspond to the same discrete prompt. Consequently, the continuous space can be conceptualized as being divided into n clusters, where n represents

the size of the vocabulary. Thus, instead of looking for a soft prompt, the model searches for a set of vectors that enable the linear head to generate a set of tokens, which then are concatenated into a prompt. A found discrete prompt is supposed to be the most optimal one, maximizing the final performance of the model.

3.2.2 GGDPS

Algorithm 1 Gradient-Guided Discrete Prompt Search

Input: Model $\mathbf{M}_\theta$, vocabulary embedding $\mathbf{E}^{|V|}$, linear head L_H, optimisation step T learning rate γ, dataset D

Sampled from real embeddings:

$\mathbf{P}_S = [\mathbf{e_i}...\mathbf{e_n}] \sim \mathbf{E}^{|V|}$

for 1,..., T **do**

  Retrieve current mini batch $(X, Y) \subseteq D$

  Projection to discrete embedding:

  $\mathbf{P}_D = argmax(L_H(\mathbf{P}_S))$

  Calculate gradient w.r.t. the discrete embedding:

  $g = \nabla_{P_D} \mathcal{L}_{task}(\mathbf{M}_\theta([\mathbf{P}_D; X_i]), Y_i)$

  Apply the gradient to the continuous embedding:

  $\mathbf{P}_S = \mathbf{P}_S - \gamma g$

end for

Final projection to discrete embedding:

$\mathbf{P}_D = argmax(L_H(\mathbf{P}_S))$

return $\mathbf{P}_D$

Above, we present our proposed method in the algorithmic form to enhance clarity and facilitate implementation. The process requires a frozen model M_θ, a sequence of learnable embeddings $\mathbf{P}_S = [\mathbf{e_i}...\mathbf{e_n}], \mathbf{e_i} \in \mathbb{R}^d$, where n is the length of the prompt and d is the dimension of the prompt tokens in the embedding space. Initially, the set of learnable embeddings is initialized from the top n tokens from the vocabulary embedding $\mathbf{E}^{|V| \times d}$, where $|V|$ is a length of the vocabulary. Then, during the training process, each of the learnable embeddings are projected to the discrete space via linear head of the pre-trained model $L_H : \mathbb{R}^{n \times d} \to \mathbb{R}^{n \times |V|}$. The mapping is followed by the application of the $argmax$ function, which identifies the token from the vocabulary with the highest value for each of the projected vectors. Finally, obtained tokens are concatenated with the input tokens, which are subsequently passed to the model for further processing.

The loss function $\mathcal{L}_{task}$ is designed to evaluate the performance of P_D, derived through the discretization of continuous embeddings, on the task data. The primary objective is to minimize erroneous outcomes. The gradient obtained from the loss function is applied to the continuous embeddings to direct the search within the continuous space. The process is repeated for each of the optimization step T. A final projection is obtained once the training process concludes and the continuous embeddings have reached their optimal state. The resultant discrete prompt can subsequently be decoded into natural language using a tokenizer and seamlessly applied to any task-related input without requiring modifications to the input, prompt, or model.

3.2.3 Strengths of the GGDPS

The algorithm leverages optimization through continuous space, preserving expressiveness while handling discrete values. Additionally, it employs the linear head of the pre-trained model, removing the need for further parameter tuning for search purposes. Therefore, the number of parameters that need to be learned depends on both the prompt length and the dimensionality of the embedding space. The resulting discrete prompt is presented in natural language, thereby enhancing its explainability and mitigating potential subtle adversarial attacks on language models using soft prompts [64, 71]. The findings suggest that embedding space attacks bypass model alignments and provoke harmful behaviors more efficiently than discrete attacks [45]. The rationale is rooted in the expressive properties of the continuous space, where alterations in the vector can activate any token from the vocabulary. The modification made to soft prompts can go undetected to the human eye, given the enigmatic nature of the underlying meanings embedded within soft prompt values. However, discrete prompt found by the algorithm can be applied seamlessly with any task-related data in natural language form, requiring no modifications to the model.

3.3 Cross-Model Prompt Transfer

This section addresses the third research question, which focuses on the concept of transferring prompts, identified by the GGDPS algorithm, between two different models. We begin by presenting the motivation for the necessity of this research objective, followed by the methodology that facilitates the achievement of this goal.

3.3.1 Motivation

Prompt transfer remains a notable area of research, driven by the goal of optimizing the prompt-tuning process, which is quite sensitive to initialization and the size of the training dataset [21, 26, 50, 70]. Generally, the vast majority of studies focus primarily on transferring prompts between tasks. Consequently, prompt transfer between different models remains an underexplored area of research. Emphasizing the development of prompts that can be transferred between homogeneous and heterogeneous models is advantageous in scenarios with certain constraints. Developing prompts for each model from scratch can be resource-intensive. Transferable prompts can reduce the need for extensive training and tuning, saving computational resources and time [2, 54]. Furthermore, in real-world applications, different models may be preferred for different tasks due to their specific strengths [51, 62]. Transferable prompts allow for greater flexibility in deploying the best-suited model for each task without re-engineering them extensively. This is particularly useful in practical scenarios where different models may be deployed based on specific requirements. For instance, a larger model could be employed to formulate a transferable prompt due to its greater capacity to learn intricate patterns and details from data. However, the smaller model equipped with the formulated prompt could be utilized for inference, given that sending requests to a model with fewer parameters is more cost-effective and energy-efficient [49].

3.3.2 Approach

A significant challenge with soft prompts lies in their dependence on the intrinsic characteristics of the chosen model, such as the embedding size. Thus, the embedding sizes of the models must be aligned for the soft prompt to be transferable. Otherwise, the soft

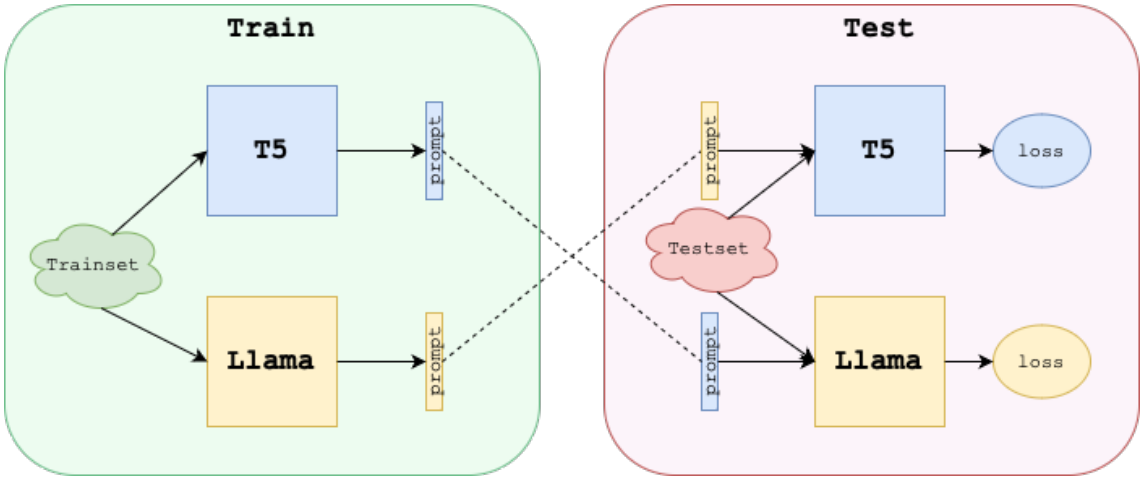


FIGURE 3.1: Approach to evaluate cross-model prompt transfer between two models

prompt must be modified to fit the other model, potentially compromising the integrity of the knowledge it encapsulates. In contrast, the transfer of hard prompts offers greater flexibility for cross-model transfer, as they can simply be appended to the input. Hence, we concentrate on investigating the transfer of hard prompts that are autonomously discovered by the model.

Figure 3.1 presents an approach for the evaluation of cross-model prompt transfer, which can be described as follows: train two models to identify their optimal set of hard prompts, then concatenate the hard prompt trained on the first model with the input for the second model, and similarly, concatenate the hard prompt trained on the second model with the input for the first model. Meanwhile, we focus our research on heterogeneous models to assess the significance of their differences in finding an optimal hard prompt. This approach is justified by real-world environment, where various models with significant internal differences are employed depending on the objective. We evaluate the results on both models using accuracy and F1-score. Furthermore, we evaluate the agreement level between the predictions of the two models on the same dataset, using different hard prompts: one trained on the original model and the other trained on the second model. The agreement level is calculated as the ratio of matching predictions to the total length of the test set. This metric aids in evaluating the consistency, robustness, and generalization of the identified prompts.

3.4 Collaborative Prompt Generation

This section addresses the last research question, which investigates the generation of hard prompts using multiple models. The idea is that the generated hard prompt should achieve satisfactory performance on all models involved in its creation. The section offers the rationale behind collaborative prompt generation and outlines the framework designed to achieve a shared hard prompt.

3.4.1 Motivation

Training heterogeneous models presents unique challenges due to their inherent diversity in architecture, parameterization, and learning dynamics. Unlike homogeneous models,

that share similar structural characteristics and training procedures, heterogeneous models vary significantly in their internal configurations and operational principles. This diversity complicates the training process as it requires adapting methodologies and strategies to accommodate different model behaviors and capabilities [30]. Moreover, integrating diverse models often involves overcoming compatibility issues and ensuring effective communication between components with distinct operational paradigms [28].

Hence, developing a shared prompt that works across various heterogeneous model can be beneficial for the real-world applications. First and foremost, such a prompt is more robust and consistent, since it encapsulates knowledge between various models[2, 68]. Due to the capacity of different models to capture distinct nuances within the same dataset, they collectively provide sufficient information to draw broader conclusions that extend beyond task-specific details. Moreover, with a hard prompt that functions effectively across multiple heterogeneous models, prompt transfer can expand beyond task-specific applications to encompass cross-model compatibility and versatility. This attribute is advantageous particularly in scenarios involving computationally intensive models or environments with constrained resources.

3.4.2 C-GGDPS

Algorithm 2 Collaborative Gradient-Guided Discrete Prompt Search

Input: List of models $[\mathbf{M}_{\theta^1}, ..., \mathbf{M}_{\theta^m}]$, number of models m, where $m \in \mathcal{Z}^+$ vocabulary embedding $\mathbf{E}^{|V|}$, linear head L_H, optimisation step T, learning rate γ, dataset D

Sampled from real embeddings:

$\mathbf{P}_S = [\mathbf{e_i}...\mathbf{e_n}] \sim \mathbf{E}^{|V|}$

for 1,..., T **do**

  Retrieve current mini batch $(X, Y) \subseteq D$

  Projection to discrete embedding:

  $\mathbf{P}_D = argmax(L_H(\mathbf{P}_S))$

  Initialize the variable for accumulation of losses:

  $\mathcal{L}_{avg} = 0$

  Calculate loss for all models:

  for 1,..., m **do**

    $\mathcal{L}_{avg} += \mathcal{L}_{task}(\mathbf{M}_{\theta^m}([\mathbf{P}_D; X_i]), Y_i)$

  Calculate gradient w.r.t. the discrete embedding for the average of all losses:

  $g = \nabla_{P_D} \frac{1}{m} \mathcal{L}_{avg}$

  Apply the gradient to the continuous embedding:

  $\mathbf{P}_S = \mathbf{P}_S - \gamma g$

end for

Final projection to discrete embedding:

$\mathbf{P}_D = argmax(L_H(\mathbf{P}_S))$

return $\mathbf{P}_D$

To facilitate understanding and provide transparency, the proposed method is delineated in algorithmic form above. The algorithm requires a list of frozen models $[\mathbf{M}_{\theta^1}, ..., \mathbf{M}_{\theta^m}]$, where m denotes the number of selected frozen models and must be a positive integer $m \in \mathcal{Z}^+$. Furthermore, it requires a sequence of learnable embeddings $\mathbf{P}_S = [\mathbf{e_i}...\mathbf{e_n}], \mathbf{e_i} \in \mathbb{R}^d$, where n denotes the prompt length and d indicates the dimensionality of the prompt tokens within the embedding space.

As in the previous algorithm, the set of learnable embeddings is initialized from the top n tokens from the vocabulary embedding $\mathbf{E}^{|V| \times d}$, where $|V|$ is a length of the vocabulary. During the training, each of the learnable embeddings are projected to the discrete space via linear head of the pre-trained model $L_H : \mathbb{R}^{n \times d} \rightarrow \mathbb{R}^{n \times |V|}$. Given that we are working with heterogeneous models, the sizes of their embedding layers, hence the dimensionality of the prompt embeddings, and the linear head can differ. The embedding layer for initialization and the linear head for discretization are chosen from one of the models based on overall demonstrated performance of the algorithm.

The $argmax$ function follows the application of the linear head L_H, identifying the token from the vocabulary with the highest value for each of the projected vectors. Finally, the obtained tokens are concatenated with the input tokens, which are subsequently passed to the model for further processing. Each input, along with concatenated prompts, is forwarded to every model in the list, and the losses $\mathcal{L}_{task}$, which assess the performance of P_D, are aggregated during a single optimization step T. After all models have completed their computations, the aggregated losses are averaged by dividing the total value by the number of models in the list.

The gradient derived from the loss function is applied to the continuous embeddings and utilized to guide the exploration within the continuous space. This iterative process occurs over multiple optimization steps denoted by T. Upon completing the training process and achieving optimal states for the continuous embeddings, a final projection is obtained. The resulting discrete prompt can then be transformed into natural language using a tokenizer and seamlessly integrated into any task-related input without necessitating adjustments to the input, prompt, or model structure.

Chapter 4

Experimental setup

In the following section, we describe the experimental setup designed to address the research questions. This section aims to provide detailed information to facilitate the possible replication of the results. Therefore, this chapter provides comprehensive information on the selected datasets, data preprocessing procedures, chosen models, training details, baselines, performance evaluation metrics, and hardware specifications.

4.1 Datasets and Pre-processing

In this study, we utilize five datasets from the GLUE[57] and two from the SuperGLUE[56] benchmarks, which contain the following tasks: natural language inference, paraphrase detection, question answering, and sentiment analysis. Details of the datasets, including domain, size, and task, are provided in Table 4.1. The datasets were selected based on their relevance in the field of prompt-tuning and to facilitate a fair comparison with the chosen baselines. Additionally, we ensured the inclusion of datasets varying in size, domain, and task to provide a comprehensive evaluation of the proposed approach.

Dataset	Category	Task	Domain	Size
RTE	GLUE	natural language inference	News, Wikipedia	5.7K
MRPC	GLUE	paraphrase detection	News	5.8K
WiC	Super GLUE	word sense disambiguation	Lexical databases	8K
BoolQ	Super GLUE	boolean QA	Wikipedia	9.4K
SST2	GLUE	sentiment analysis	Movie Reviews	70K
QNLI	GLUE	natural language inference	Wikipedia	116K
QQP	GLUE	paraphrase detection	Quora	795K

TABLE 4.1: Details of seven datasets used in the experiments for training, testing and validation.

The datasets were obtained from HuggingFace[1] and are pre-split into three categories, which were used as intended: training, validation, and test. All datasets contain two labels, making it two-class classification problem. The initial input data is presented in various formats, that needed to be further pre-processed. To prepare the input data for training and evaluation by the models, input was formatted according to the template illustrated in the Table 4.2. Furthermore, concrete examples of the data can be found in Appendix A in Tables A.1 and A.2.

[1] https://huggingface.co/docs/datasets/en/index

Dataset	Template	Input
RTE	sentence 1: s_1, sentence 2: s_2	$x = (s_1, s_2)$
MRPC	sentence 1: s_1, sentence 2: s_2	$x = (s_1, s_2)$
WiC	sentence 1: s_1, sentence 2: s_2, word: w	$x = (s_1, s_2, w)$
BoolQ	question: q, passage: s	$x = (q, s)$
SST2	sentence: s	$x = (s)$
QNLI	sentence: s, question: q	$x = (s, q)$
QQP	question 1: s_1, question 2: s_2	$x = (s_1, s_2)$

TABLE 4.2: Used templates for each dataset to preprocess the input data before passing it to the model. The initial form of input is data is provided in column **Input**.

4.2 Models

To address the research questions, the study necessitates the involvement of at least two models: T5 and Llama2. T5 was chosen due to its utilization as a backbone language model in the chosen baselines. Llama2 was selected due to its prevalent use in the field of large language models and its open-source nature, which facilitates easier integration. Additionally, the models employ different architectures: T5 is an encoder-decoder model, while Llama is a decoder-only model. This difference is advantageous as it aligns with the research goal of developing a shared prompt that enhances its generalizability and robustness across diverse models. Further, a comprehensive description of each model are provided along with implementation specifics that can be found in Table 4.3. It is evident that Llama is significantly larger, containing 31818 times more parameters than T5. This decision can be attributed to the interest in analyzing the observed behavioral differences between two models that possess distinct natures.

T5-base, Text-To-Text Transfer Transformer, is a variant of the T5 model developed by Google Research [41]. It belongs to the family of transformer models, which have revolutionized natural language processing tasks. T5 Base operates as an encoder-decoder architecture, where it encodes input text and decodes it into an output sequence, making it suitable for a wide range of text generation and understanding tasks. The model is pre-trained on large-scale datasets using a text-to-text approach, treating all NLP tasks as text-to-text problems, which simplifies training and evaluation. T5 Base is known for its versatility, achieving strong performance across various benchmarks in natural language understanding, generation, and translation tasks.

LlamaForSequenceClassification, an advanced variant in the LLaMA (Large Language Model Meta-Learning Approach) series, represents a decoder-only architecture designed for large-scale language modeling tasks with Llama2 backbone [52]. Developed by OpenAI, Llama2 builds on its predecessor's capabilities, emphasizing efficient integration and openness in its design. As a decoder-only model, Llama2 focuses on generating sequences based on input prompts, making it particularly adept for tasks requiring text generation and completion. With its open-source availability and tailored architecture, Llama2 supports a broad range of applications in natural language processing, offering flexibility and robust performance in various language-based tasks. In the current model, the classification head is appended as the final layer following the Llama2 backbone.

Model	nl (el/dl)	ff	dm	kv	nh	#Param
T5-base	12/12	3072	768	64	12	220K
LlamaForSequenceClassification	0/32	11008	4096	32	32	7B

TABLE 4.3: Implementation details for chosen models. Number of transformer blocks: **nl**. Number of transformer blocks in the encoder and decoder: **el** and **dl**. Dimension of intermediate vector within transformer block: **ff**. Dimension of embedding vector: **dm**. Dimension of key/value projection matrix: **kv**. Number of attention heads: **nh**.

4.3 Baselines

The current section of this chapter outlines the selected baselines for evaluating GGDPS and C-GGDPS: prompt-tuning (PT), soft prompt transfer (SPOT), and attentional mixture of prompt tuning (ATTEMPT). Additionally, the performance of the proposed approach is compared to that of the pre-trained model, that does not exhibit any modifications. Each of the baselines was selected based on their novelty, state-of-the-art results, and widespread use in the field of parameter-efficient tuning. The decision was made to compare GGDPS against the performance of soft prompts rather than hard prompts, as previous automated methods for constructing hard prompts have consistently shown inferior performance compared to soft prompts.The chapter is organized by baseline to provide a more comprehensive description of each.

4.3.1 PT

Lester et al. [24] developed a methodology that appends the embedded input with a set of tunable embeddings, known as soft prompts, to the pre-trained model. It aims to leverage the capabilities of large pre-trained models by fine-tuning only a small set of parameters, which influence how the input is processed. The implementation of the method closely followed the training details described in the original paper. The method was implemented on both models T5 and Llama2. The prompts were initialized with embeddings drawn from the model's vocabulary. The rest of the training details are described in the Section 4.4.

4.3.2 SPOT

The concept of SPOT, developed by Vu et al.[54], parallels that of PT, incorporating transfer learning by pre-training soft prompts on a source task that is related to the target task. The method utilizes a pre-trained source prompt to initialize the target prompt, which is then fine-tuned on a downstream task. The core idea of SPOT is to exploit knowledge learned from a source task to improve performance on a target task. The implementation of the method adhered closely to the training protocols outlined in the original paper. The approach was applied to both the T5 and Llama2 models. A source prompt was initialized for each dataset using embeddings drawn from the model's vocabulary and trained over 5 epochs. To determine the appropriate source task for initializing the target task, a methodology outlined in the original paper is adopted, employing cosine similarity between source embeddings to identify the task most closely related to the target task. Subsequently, the source prompt is utilized to initialize the target prompt, which undergoes fine-tuning for 5 epochs. The remaining training specifics are detailed in Section 4.4.

4.3.3 ATTEMPT

ATTEMPT, developed by Asai et al.[2], extends the idea of SPOT by incorporating an attention mechanism to blend multiple source prompts dynamically. Instead of relying on a single source prompt for initializing the target prompt, ATTEMPT integrates information from several source prompts using attention weights. The core innovation in ATTEMPT lies in the attention module, which learns to weight and combine the information from different source prompts based on their relevance to each input instance. An attention module consists of linear layers for down and up projection with a non-linear function in between:

$$H_{down} = W_{down}^{\top}(\hat{X})$$
$$H_{up} = W_{up}^{\top}(NonLinear(\hat{X}))$$
$$H_{out} = LayerNorm(H_{up}),$$

where $W_{down} \in \mathbb{R}^{d \times r} (r < d)$, $W_{up} \in \mathbb{R}^{r \times d}$, d is a dimension of an embeddings, r is a reduced number of dimensions and $\hat{X}$ is a result from the max-pool operation on the input. Projection parameters are updated during training. SiLU[7] is used for the non-linear layer.

The implementation closely followed the training procedures outlined in the original paper, applying the approach to both the T5 and Llama2 models. ATTEMPT involves training four source prompts—MNLI, SST-2, QNLI, and QQP—for 5 epochs each. Each source prompt is initialized by sampling tokens randomly from the top vocabularies. Subsequently, each source prompt is employed to train the attention module alongside a target prompt initialized with the MNLI source prompt. The number of training epochs varies depending on the dataset size, with 20 epochs used for small datasets and 5 epochs used for datasets containing more than 10 thousand training samples. The rest of the training specifics are described in the Section 4.4.

4.4 Training details

The training procedure for each experiment parallels that of PT[24], SPOT[54], and ATTEMPT[2], as these baselines share identical training details. For each baseline, the soft prompt consists of 100 tokens. However, in this study, this number was adjusted to 30 tokens, as GGDPS does not require a long prompt to achieve optimal results. The prompt token embeddings are initialized using the most frequently occurring words in the vocabulary. In the experiments, only parameters of the soft prompts are tuned, which can be computed as follows: 30× `dm`, where `dm` is dimension of embedding vector. Total number of tunable parameters for each model and baseline can be found in Table 5.3.

Pre-trained versions of T5 and Llama2 are utilized. T5 is pre-trained on *The Colossal Clean Crawled Corpus*[41], which contains billions of words from a diverse range of websites. Llama2 is pre-trained on publicly available online data sources [52]. Each of the models used the default set of parameters for training and employed cross entropy loss. They were trained for 5 epochs using the training split of each dataset. The validation split was used to evaluate the model at the end of each epoch, and the set of tokens, or a soft prompt, was saved whenever the model achieved the best performance observed up to that point.

All of the experiments are conducted with a single NVIDIA A40 GPU with 48 GB memory[2]. The implementation is done utilizing Python 3.11[3]. The models are implemented

[2] https://images.nvidia.com/content/Solutions/data-center/a40/nvidia-a40-datasheet.pdf
[3] https://www.python.org/downloads/release/python-3110/

using PyTorch[4] and package `transformers` from HuggingFace[5]. Adam optimizer was used with the learning rate of 0.3. The batch size was set to 16 for T5 and 8 for Llama2, with this difference attributable to the memory constraints of the hardware employed in this study. Due to the same hardware constraints, the batch size for the ensemble model of T5 and Llama2, used to evaluate C-GGDPS, was reduced to 1.

4.5 Performance evaluation

The current section outlines the methodology used to evaluate the research questions. Two approaches are employed for evaluation: quantitative and qualitative. The quantitative approach is employed to quantify the outcomes of both baseline methods and the implementations of proposed methods and hard prompt transfer. However, for a more detailed interpretation of the results, qualitative evaluation is conducted using additional analytical tools.

4.5.1 Quantitative evaluation

We assess the performance of GGDPS, C-GGDPS and baselines using the test sets of the selected datasets. Given that our evaluation focuses on the classification task, we utilize standard metrics: accuracy and F1-score. To ensure reliability of the results, 10-cross validation is also applied. Additionally, we assess the agreement between predictions generated by T5 and Llama. Agreement in predictions aids in evaluating consistency, providing insights into data complexity and whether the approach is specific to the model used. The agreement level is determined by calculating the proportion of instances where both models predict the same outcome, relative to the total number of test instances.

4.5.2 Qualitative evaluation

In the qualitative evaluation, our objective is to assess the extent to which the resulting hard prompts are perceptible to human interpretation and whether their underlying semantics align closely with the task, dataset, or model's pre-trained knowledge. For the evaluation of the resulting prompt from the GGDPS, we visually assess it for its grammatical correctness and coherence. To evaluate the cross-model hard prompt transfer, we employ t-SNE[5] to reduce the dimensionality of hard prompt embeddings for their visualization. By visualizing the hard prompts generated by T5 and Llama, our objective is to assess whether the models have extracted identical sets of features. Additionally, we employ confusion matrices to visualize the distribution of attention patterns in the encoder of T5 and the decoder of Llama, aiming to analyze which semantic information the models prioritize the most.

[4] https://pytorch.org/
[5] https://github.com/huggingface/transformers

Chapter 5

Experiments

In the following chapter, we present the experiments conducted to address the stated research questions. Each section is dedicated to a specific research question, detailing the numerical results and discussing the significance of the findings. Section 5.1 presents findings on the discretization of soft prompts through the mapping of soft prompts to the embeddings of hard tokens extracted from the model's vocabulary. Section 5.2 offers insights into the performance of the newly developed GGDPS algorithm, which is designed to enable the model to autonomously discover an optimal hard prompt. Section 5.3 details the cross-model transferability of the identified prompts, discussing the significant implications of this transfer. Finally, Section 5.4 discusses the collaborative prompt identification algorithm, C-GGDPS, and emphasizes the notable findings.

5.1 On soft prompt discretization

This section outlines the findings related to the first research question, which examines the discretization of soft prompts into natural language. The section begins with a presentation of the numerical results, emphasizing significant insights, and subsequently provides a comprehensive discussion on the feasibility of identifying natural language alternatives to soft prompts by mapping them to the embeddings of hard prompts derived from the model's vocabulary.

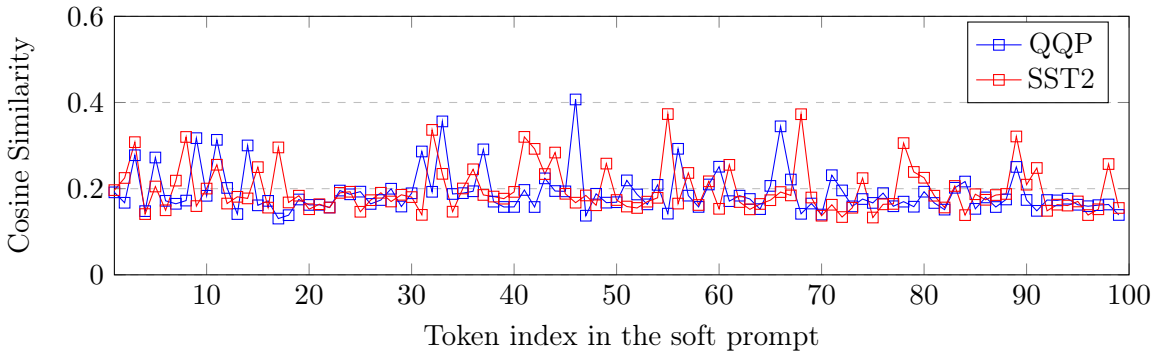


FIGURE 5.1: Cosine similarity between each token in the soft prompt and its nearest natural token embedding.

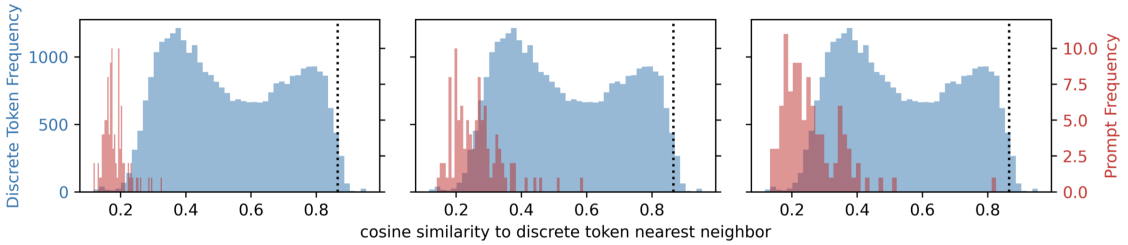


FIGURE 5.2: Histograms depict the highest cosine similarity values between natural tokens (blue) and soft prompt tokens (red) with any natural token embedding in the vocabulary. The vertical line marks a cosine similarity of 0.866, corresponding to $cos(30°)$ [3].

5.1.1 Experimental results

Figure 5.1 illustrates the calculated cosine similarity for two soft prompts each trained on distinct datasets: QQP and SST-2. Each plot represents the relationship between the tokens of the soft prompt and the embeddings of their nearest neighbors. The y-axis represents the cosine similarity values, while the x-axis denotes the token length of the soft prompt. Both soft prompts from the two datasets consist of 100 tokens each. A key observation is that the soft tokens do not closely resemble any specific words in the vocabulary, as evidenced by the highest similarity value, which is only 0.4.

Figure 5.2 the highest cosine similarity values between natural tokens (blue) and soft prompt tokens (red) with any natural token embedding in the vocabulary. The y-axis indicates the frequency of discrete (left-axis) and soft (right-axis) prompts, while the x-axis displays the cosine similarity values. The figure reveals that natural language tokens exhibit higher similarity with one another. In contrast, soft prompt embeddings show lower similarity to any vocabulary token, indicating that the embedded information significantly diverges.

Figure 5.3 presents a word cloud depicting the most frequently occurring tokens in the newly created discrete alternatives of soft prompts trained on four datasets of the GLUE

FIGURE 5.3: Word Cloud of the most common nearest discrete token to the embeddings of the soft tokens.

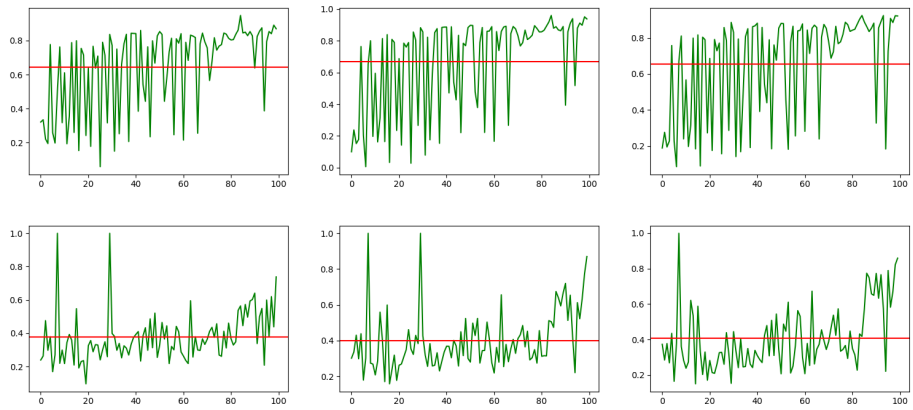


FIGURE 5.4: Cosine similarity between soft prompts and ipnut data in the last layer of the encoder. Top: data gathered with QQP dataset. Bottom: data gathered with RTE dataset.

benchmark[57]. The discrete alternatives were generated by identifying the nearest natural language tokens through cosine similarity between the embeddings of soft prompts and the embeddings of tokens in the model's vocabulary. The word cloud reveals that terms such as *hypothesis*, *query*, *question*, and *summarize* are prominently featured. This is anticipated, as these tokens are commonly utilized when formulating instructions for models across a variety of tasks.

Figure 5.4 demonstrates the results of cosine similarity between soft prompts and ipnut data in the last layer of the encode for some of the inputs in two datasets: RTE and QQP. The selection of these datasets was based on their respective performance metrics with prompt-tuning, with RTE achieving 54% accuracy and QQP achieving 92%. The y-axis denotes the cosine similarity values, while the x-axis indicates the length of the prompt. We observe that the representation of soft tokens in the encoder layer closely resembles the representation of certain segments of the input data, with cosine similarity values reaching approximately 0.85 for QQP and 0.99 for RTE in some cases. So, this indicates that soft tokens indeed carry a similar meaning as some words in the dataset.

5.1.2 Discussion

The results indicate that discrete prompts identified through naive mapping to the nearest natural language tokens, using their embeddings and cosine similarity as a measurement metric, do not exhibit equivalent success in final performance. This may be attributed to the differences in properties between soft prompt embeddings and natural token embeddings, as well as the nature of the embedded information. For example, the geometric characteristics of soft prompt embeddings differ markedly from those of natural token embeddings. Their magnitude significantly exceeds that of natural token embeddings[3]. This difference may be attributed to gradient updates, which cause soft tokens to accumulate larger magnitudes during the training process. Another difference lies in the similarity distribution: the similarity among natural tokens compared to the similarity among soft tokens. Figure 5.2 shows that the cosine similarity among soft tokens is quite low compared to the cosine similarity between natural tokens.

Nonetheless, some tokens from the soft prompts are closer to those that remind of instructive words rather than to arbitrary tokens. For example, Lester et al.[24] suggested that nearest natural neighbors are clustered around task-relevant words, as evidenced by

the word cloud showing frequent tokens such as *hypothesis*, *query*, *question*, and *summarize*. Despite the intuitive appeal of constructing a hard prompt using these tokens in anticipation of a desired output, the final performance was deemed insufficient, resulting in nearly 0% accuracy and notable misclassification errors. Even though some of the soft tokens nearly resemble discrete tokens commonly used for construction of hard prompts, they cannot be straightforwardly mapped to their nearest natural neighbors, as this leads to a significant drop in performance.

In addition to analyzing the straightforward mapping of soft tokens to their closest natural language equivalents, we also investigated the representation of soft tokens in the final layer of the encoder, comparing it to the representation of the input text within the same layer. Despite some soft tokens being perceived similarly by the model as certain input words, these input words typically lack meaningful semantic content. Upon mapping these tokens to their closely perceived natural counterparts, we observed that they predominantly represent: the end token, coordinating conjunctions, numbers, question marks, and padding tokens. Nevertheless, those are the tokens where the model places higher attention during the training [42]. Higher cosine similarity is typically observed for the aforementioned tokens, whereas lower similarity is generally noted for tokens that represent meaningful words. Moreover, it was observed that a higher perceived similarity between soft tokens and input data correlates with better final model performance. For instance, a soft prompt trained on the RTE dataset exhibits a higher perceived similarity to padding tokens, resulting in a final output accuracy of approximately 54%. Conversely, in the case of QQP, there is an average perceived similarity of 0.65 to more semantically meaningful words, contributing to a final performance of 92%.

Overall, based on the aforementioned observations, soft prompts include tokens that the model focuses on most during training, such as tokens representing the beginning and end of a sentence. Moreover, if the model effectively extracts more valuable information for the task, this information will also be encoded within the soft prompt.

5.2 About GGDPS

This section presents the findings related to the second research question, focusing on the proposed method for obtaining hard prompt via gradient-guided search. It begins by presenting numerical findings, emphasizing on notable insights, and subsequently delves into a thorough discussion on the feasibility and practicality of the proposed approach.

5.2.1 Experimental results

Table 5.1 and Table 5.2 demonstrate obtained results from implementing GGDPS on T5 model and Llama model respectively in comparison to the baselines and performance of the model without any added instructions to it (noted as *original* in the table). Average accuracies along with their variances are depicted based on a 10-fold cross validation. A value in parentheses next to the dataset name indicates the size of the training set. As it can be seen, the table is divided according to the dataset size. Specifically, datasets with fewer than 10,000 training instances are grouped together, whereas larger datasets are presented separately. A significant observation from the results on T5 is that GGDPS surpasses the baselines when the training set is extensive, whereas it falls short compared to some baselines on smaller datasets. On the other hand, GGDPS does not exhibit superiority over SPOT and ATTEMPT across any datasets. Nonetheless, it consistently outperforms

vanilla prompt-tuning across all datasets for both models.

Table 5.3 displays the number of parameters required for each baseline and GGDPS to train a prompt. It is evident that GGDPS requires the fewest parameters among all baselines. Although the number of parameters is dependent on the prompt length, GGDPS does not require a long prompt to achieve satisfactory results. PT, SPOT, and ATTEMPT required a 100-token prompt to demonstrate the best performance. Additionally, ATTEMPT necessitates additional parameters for training an attention module, which includes two projection layers. In contrast, GGDPS achieved comparable or superior results in comparison the baselines with a prompt length of only 30 tokens.

Table 5.4 presents the F1-scores for all baselines and the GGDPS implementation on the T5 model. The table is organized in the same manner as previously described. A notable observation is that GGDPS consistently outperforms all baselines on larger datasets. However, on smaller datasets, it remains inferior to SPOT and ATTEMPT, while consistently demonstrating better performance than vanilla prompt-tuning.

Table 5.5 and Table 5.6 display the prompts identified by GGDPS that achieved the highest performance across all datasets for T5 and Llama respectively. As demonstrated by the algorithm, the final projection yielded tokens rather than natural words. However, they were subsequently transformed into natural language using tokenizer.

5.2.2 Discussion

The observed results indicate that the proposed GGDPS algorithm has a potential to guide the model to autonomously identify an optimal hard prompt for a given task. It surpasses the baselines in both accuracy and F1-score on large datasets for T5 model. However, it underperforms compared to SPOT and ATTEMPT on datasets with smaller training sizes. This can be attributed to the fact that both methods use pre-trained soft prompts for the initialization and fine-tuning of target prompts, whereas GGDPS does not incorporate any external knowledge. By supplying task-related knowledge in advance, the prompt can more effectively guide the model towards correct outputs without requiring it to learn the appropriate patterns from scratch [2, 12, 38, 54]. The initial concept behind SPOT

	RTE (2.5K)	MRPC(3.7K)	WiC (6K)	BoolQ (9.4K)
Original	67.39% (±0.35)	85.29% (±0.31)	63.01% (±0.29)	66.42% (±0.23)
PT	54.71% (±0.75)	68.14% (±0.68)	48.92% (±0.51)	61.71% (±0.44)
SPOT	69.83% (±0.38)	79.71% (±0.33)	67.04% (±0.35)	77.22% (±0.22)
ATTEMPT	73.41% (±0.35)	85.73% (±0.29)	66.81% (±0.22)	78.81% (±0.19)
GGDPS	71.22% (±0.30)	86.77% (±0.28)	61.76% (±0.25)	71.68% (±0.21)

	SST2 (67K)	QNLI (105K)	QQP (364K)
Original	85.34% (±0.29)	87.92% (±0.21)	84.12% (±0.23)
PT	90.92% (±0.26)	92.81% (±0.31)	89.74% (±0.51)
SPOT	93.42% (±0.12)	93.04% (±0.09)	90.11% (±0.13)
ATTEMPT	93.23% (±0.11)	90.31% (±0.12)	90.32% (±0.08)
GGDPS	97.51% (±0.12)	93.12% (±0.10)	90.36% (±0.13)

TABLE 5.1: Obtained results from implementing GGDPS on T5 model. Results of GGDPS are compared to the results of original model (no added instructions), vanilla prompt-tuning (PT), soft prompt-transfer (SPOT) and attentional mixture of prompt tuning (ATTEMPT). The values in parentheses denote the standard deviation from the mean accuracy, calculated from 10-fold cross-validation.

	RTE (2.5K)	MRPC (3.7K)	WiC(6K)	BoolQ (9.4K)
Original	74.11% (±0.32)	45.32% (±0.41)	65.71% (±0.12)	62.21% (±0.34)
PT	53.23% (±0.42)	68.14% (±0.52)	45.32% (±0.38)	61.71% (±0.45)
SPOT	76.92% (±0.31)	70.23% (±0.34)	67.02% (±0.17)	65.19% (±0.33)
ATTEMPT	78.32% (±0.29)	72.09% (±0.19)	69.63% (±0.29)	68.32% (±0.29)
GGDPS	70.05% (±0.87)	69.61% (±0.49)	60.64% (±0.28)	63.31% (±0.26)

	SST2 (67K)	QNLI (105K)	QQP (364K)
Original	91.05% (±0.21)	88.21% (±0.45)	82.05% (±0.38)
PT	83.23% (±0.42)	89.03% (±0.27)	84.53% (±0.47)
SPOT	92.32% (±0.28)	91.17% (±0.13)	84.98% (±0.09)
ATTEMPT	94.51% (±0.17)	89.13% (±0.21)	85.63% (±0.14)
GGDPS	88.92% (±0.32)	91.01% (±0.15)	83.62% (±0.07)

TABLE 5.2: Obtained results from implementing GGDPS on Llama model. Results of GGDPS are compared to the results of original model (no added instructions), vanilla prompt-tuning (PT), soft prompt-transfer (SPOT) and attentional mixture of prompt tuning (ATTEMPT). The values in parentheses denote the standard deviation from the mean accuracy, calculated from 10-fold cross-validation.

Method	PT	SPOT	ATTEMPT	GGDPS
Number of parameters on T5	77K	77K	232K	23K
Number of parameters on Llama	409K	409K	1228K	123K

TABLE 5.3: Number of parameters required for each method to train the prompt.

and ATTEMPT arose from the poor performance of prompt-tuning in few-shot learning scenarios, primarily due to the sensitivity of the initialization[2, 54]. Nevertheless, the gap between both methods and GGDPS disappears as the training size increases, allowing the model to encounter sufficient number of examples to develop a more robust understanding of the data and facilitate the creation of a more generalized prompt.

However, it was observed that GGDPS does not perform as well on the Llama model compared to its performance on T5. It remains inferior to SPOT and ATTEMPT and, in some instances, even performs worse than the original model, that has no instructions

	RTE (2.5K)	MRPC (3.7K)	WiC (6K)	BoolQ (9.4K)
Original	69.79% (±0.29)	76.56% (±0.23)	58.45% (±0.33)	57.63% (±0.31)
PT	8.45% (±0.57)	81.39% (±0.42)	23.44% (±0.52)	35.89% (±0.49)
SPOT	57.68% (±0.34)	91.23% (±0.12)	64.21% (±0.32)	68.34% (±0.37)
ATTEMPT	75.67% (±0.20)	92.08% (±0.16)	65.42% (±0.28)	69.86% (±0.29)
GGDPS	73.68% (±0.28)	90.72% (±0.22)	55.32% (±0.38)	60.21% (±0.31)

	SST2 (67K)	QNLI (105K)	QQP (364K)
PT	71.22% (±0.35)	73.75% (±0.42)	76.87% (±0.54)
SPOT	82.33% (±0.24)	83.44% (±0.14)	81.43% (±0.19)
ATTEMPT	89.62% (±0.13)	88.43% (±0.16)	88.12% (±0.09)
GGDPS	94.55% (±0.08)	91.09% (±0.12)	89.21% (±0.08)

TABLE 5.4: The F1-scores for all baselines and the GGDPS implementation on T5 model are presented. The values in parentheses represent the standard deviation from the mean F1-score, derived from 10-fold cross-validation.

Dataset	GGDPS Discrete Prompt
RTE	poultry formularul verteviz Gujaratoulweilawa Böimba Hatalleaga Pilotescu Schwarzime RBIDACintreagaJOcrestereahitIESTotodat tacklenoi"). peisaj teenager
MRPC	corpulbul corpulbul corpulbul Scribulianuetherbul corpul corpul corpulbul corpulbulbulbulighbuludderighbulguardighighighscriptner
WiC	producator LIMITEDezimal omul pielii Yu suprafete baillubachNNdrutindefruititul tractor holders fummul intelege pieleazeice oferaLAYENG ortho preafruitphal
BoolQ	poat imunitar imunitar misconductArticolul imunitarArticolul lucrari imunitar pastra hypothesis adicapremiseMuzeul imunitarTotul imunitar masina lucrari masinapremisepremisepremisepremisepremise sticlapermalink placut impairment
SST-2	strastern Prime statralivitralivi AL vertesivi-ivistebul usor Scotia frunze |mar Star West Stritchski Warner NewTIC. . .)
QNLI	direguardcroncronockVstagre guardcroncronex contufft guarduock Buchuffdeem Fontignak-quimpgg).).
QQP	mbrNKBisericachalnchmbrinkmbrmbrinkinkmbr SfântmbrmbrinkinkinkBisericackmbr Sfântchalmbrckckckckm Question

TABLE 5.5: Found hard prompts by GGDPS on all datasets using T5 model.

added to the data. We can hypothesize that this issue arises due to the larger continuous space of the Llama model, which has 4096 dimensions, in contrast to the T5 model's 768 dimensions. Larger models typically have a more complex optimization landscape with numerous local minima and saddle points[36]. Therefore, Llama may have encountered a local minimum and become stuck, failing to converge. Another possible reason is that the pre-trained data of Llama significantly differs from the given tasks, making it difficult for hard prompts to extract relevant knowledge [52]. In contrast, SPOT and ATTEMPT offer external knowledge sufficient to adjust the model's attention patterns for the target task. GGDPS did not encounter the same problem on T5 model since pre-trained data of T5 is more similar to target tasks [41].

It is also noted that the GGDPS consistently outperforms vanilla prompt-tuning, which is particularly evident on smaller datasets. Upon closer examination of vanilla prompt-tuning, it was observed that the method fails to generalize effectively on smaller datasets, leading to notable overfitting issues. Based on the F1-score results, it is evident that prompt-tuning exhibits overfitting tendencies towards the majority class, leading to lower values. In contrast, GGDPS effectively addresses the overfitting problem observed on smaller datasets. This phenomenon can be rationalized by the nature of soft prompts as vectors positioned closely in the continuous space to the word embeddings that the model places the most attention to. However, these vectors lack inherent semantic meaning, but serve as indicators to the words that the model deems significant [39]. Hence, the model assigns high attention weights to these tokens, causing it to rely heavily on them to make predictions. On the other hand, GGDPS leverages embeddings of natural language tokens. Even though they may still exhibit proximity to important words, they remain positioned at a sufficient distance, due to the sparsity of the continuous space. Since they do not exhibit as high a similarity to important words as soft tokens, the model distributes

Dataset	GGDPS Discrete Prompt
RTE	aille Lav Por Oriental fasttringiloianaatraenthejerstric rörelo Aireserykih Jules Havande colonattice Lag Gestaririn defe
MRPC	VereamoongodbDPCCEistemaadasitzer Loopcgiralrup imin latignebitinenazesv shadowmarkt background Wiederslantztenfredctlosi beskredes
WiC	Graff Tirion moulderium flurstrandic Vesselant graspier Loopstraven darkmatter flowlights scatterwin Schattenwald beskrelo Florrentine whispers.
BoolQ	tzymaison invånansamer puntren LandkreisrettoHSenaiedz MyClass Stutt Ker mediserniteimiterptyucker excel grand Stanisch
SST-2	Mitbag cornerichtcécke Nulliar CSa Valleyleinfeeda Mityen superficIAB Juech
QNLI	blazeflint Cronvock-Stragrenox guardchron deckflare guard quicksilvered contufft guardlock Ferrumshield lopen
QQP	flibberwocky Drift silent chalnchmbrgrinm gentle RID Mysterymbrmurmur Bisericastillm

TABLE 5.6: Found hard prompts by GGDPS on all datasets using Llama model.

attention more judiciously, avoiding the placement of abnormally high attention values on discrete prompt.

Another significant observation regarding the algorithm is that it requires fewer parameters compared to the baselines. It should be stated the number of parameters are dependent on the length of the prompt and the size of embedding space. However, it was observed that the optimal performance of hard prompts was achieved with approximately 30 tokens, whereas soft prompts required at least three times as many tokens [22]. Although GGDPS employs the gradient of the soft prompt to guide the model, the soft prompt tokens are utilized solely to activate the linear head, which then outputs natural language tokens designed to maximize performance. Therefore, GGDPS is computationally more efficient while achieving results that are relatively comparable to the baselines. Moreover, it employs a linear head that is already integrated into the pre-trained model, thereby eliminating the need for additional training or fine-tuning.

Designing an optimal hard prompt can be a time-consuming process, often requiring expert knowledge and extensive trial-and-error to maximize performance [24]. A key advantage of the proposed algorithm is that it eliminates this need by allowing the model to autonomously determine the most effective prompt. Furthermore, the newly discovered prompt does not need any modifications to the model's internal structure, since it can be seamlessly concatenated with the input during data preprocessing. This also improves interpretability and eliminates the potential for implicit adversarial attacks. Such attacks can go unnoticed due to the inherent ambiguity of soft prompts to a human eye. Moreover, the findings indicate that embedding space attacks circumvent model alignments and trigger harmful behaviors more effectively compared to discrete attacks [45].

Although the discovered prompts exist in discrete space and can be applied without altering the model's internal structure, they remain incomprehensible to humans. It might be expected that the discovered prompts would be easily explainable and readable. However, they convey little to no meaning in human language. This suggests that the prompts are highly specific to the model that discovered them and can only be comprehended by

Model	Prompt	Dataset						
		RTE	MRPC	WiC	BoolQ	SST-2	QNLI	QQP
T5	T5	71.22%	86.77%	61.76%	71.68%	97.51%	93.12%	90.36%
	Llama	66.67%	71.07%	56.64%	63.48%	95.89%	90.21%	88.76%
	Agr.	83.34%	69.61%	82.02%	78.92%	96.82%	89.32%	93.29%
Llama	T5	63.84%	69.61%	55.36%	59.82%	85.42%	87.41%	79.35%
	Llama	70.05%	73.04%	60.64%	63.31%	88.92%	91.01%	83.62%
	Agr.	87.62%	95.59%	86.39%	89.54%	94.39%	91.11%	89.44%

TABLE 5.7: Reported accuracy of T5 and Llama models using T5 prompt and Llama prompt for each of the datasets. Abbreviation *Agr* stands for agreement level between model's predictions.

that particular model. Tables 5.5 and 5.6 reveal that the prompts discovered by the T5 and Llama models for a given dataset do not carry any similarity that could be observed by a human eye. This indicates that the features extracted from the text by the T5 model may differ from those extracted by the Llama model. To explore the model-specific nature of these prompts, we conducted cross-model prompt transfers between Llama and T5 to assess possible performance differences.

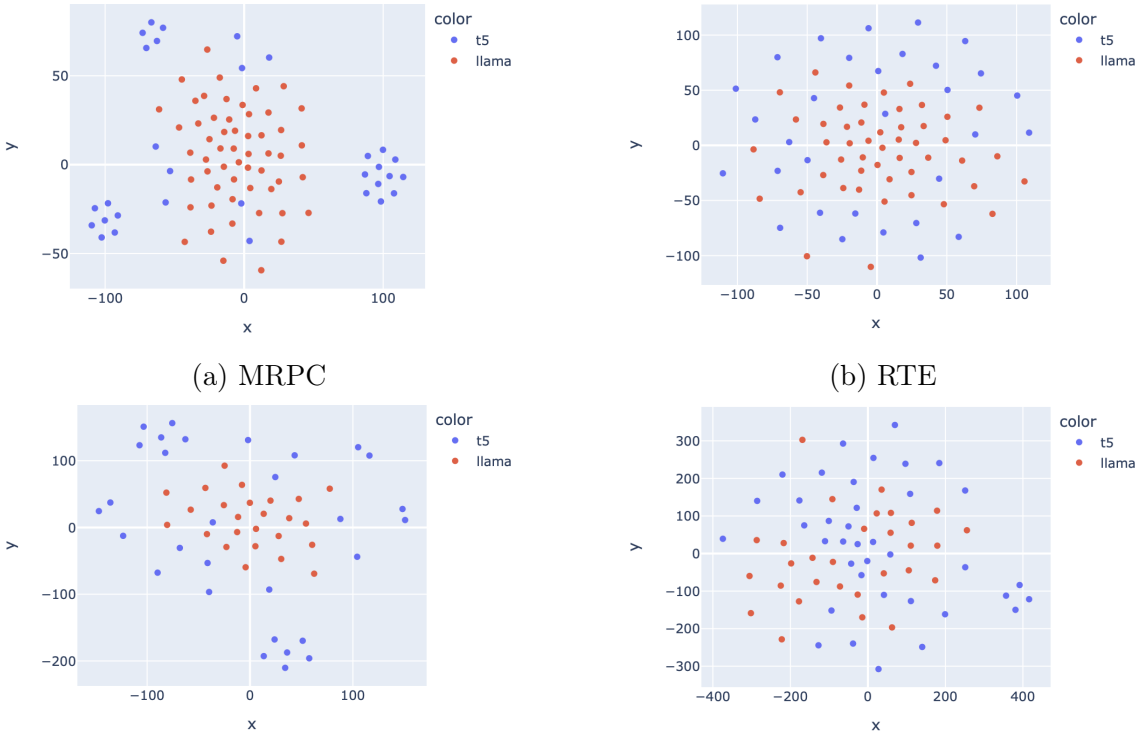
5.3 On cross-model prompt transfer

This section addresses the third research question, which focuses on analyzing the cross-model transferability of prompts identified by the GGDPS on two models: T5 and Llama. It starts off by presenting experimental findings that include cross-model prompt transfer and variations in attention distribution within the encoder of both models. Subsequently, these results are analyzed comprehensively, followed by a detailed evaluation and discussion of significant insights.

5.3.1 Experimental results

The prompts extracted from Table 5.5 and Table 5.6 were employed to conduct cross-model prompt transfer between the T5 and Llama models. The results of the experiment can be found in Table 5.7. It illustrates the accuracy of the performance on specific dataset for both models using their native prompts and prompts derived from the other model. Furthermore, it also includes an agreement ratio between predictions. The accuracies of T5 and Llama models with their derived prompts are taken from Table 5.1 and Table 5.2 respectively. The notable finding is that cross-model prompt transfer yields unsuccessful results. Specifically, when a model employs a prompt derived from another model, its accuracy shows a decline when compared to its performance with native prompt. In terms of agreement level, we observe an average of approximately 86%, suggesting that models generally exhibit highly similar predictions in the majority of cases no matter which prompt is used. The lowest agreement ratio was noted at 69% for the T5 model on the MRPC dataset, correlating with the largest observed decline in performance from 86.77% to 71.07%. Overall, there is a positive correlation between performance difference and the agreement level.

Figure 5.5 illustrates the visualization of embeddings for hard prompts generated by GGDPS on T5 and Llama models. The embeddings were created using the T5 embedding layer, and the visualization encompasses each dataset for which the T5 and Llama prompts



(a) MRPC

(b) RTE

(c) BoolQ

(d) SST2

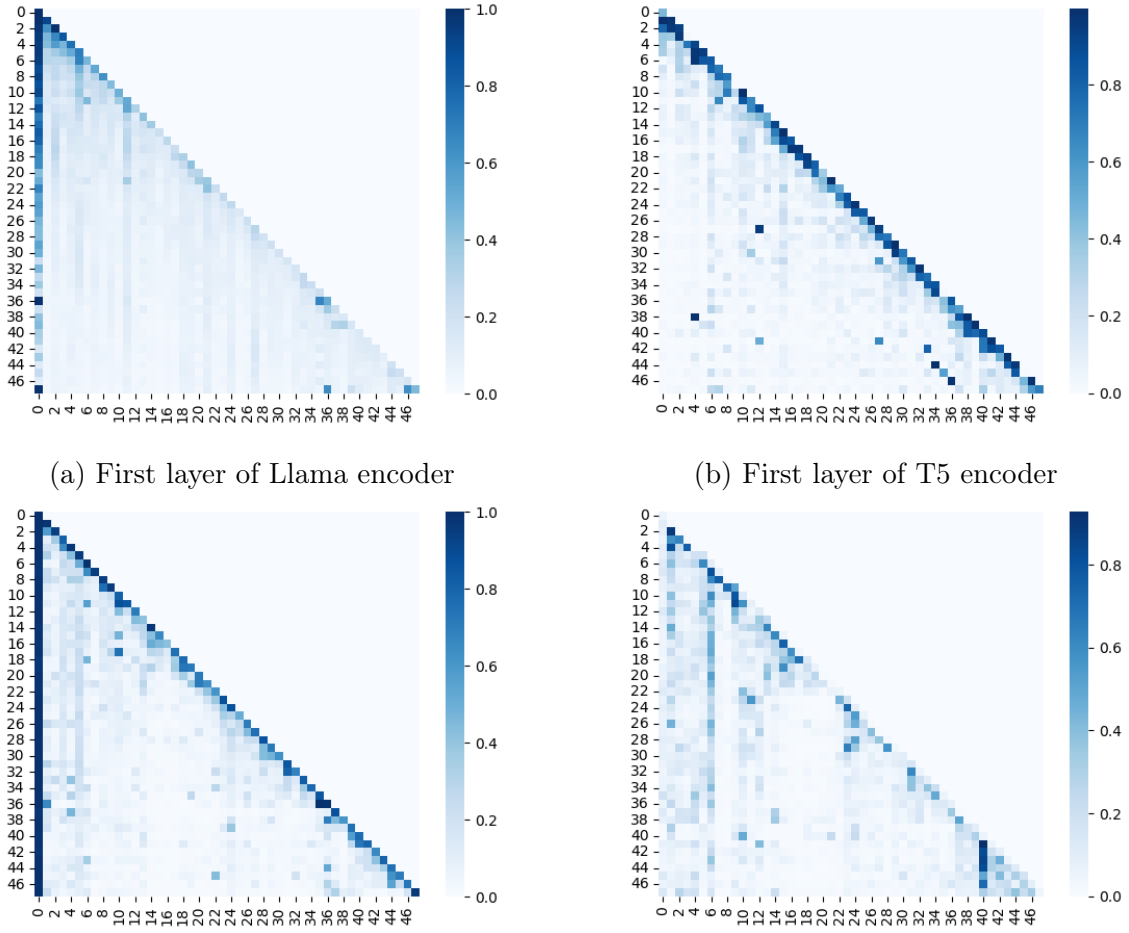
FIGURE 5.5: Embeddings of prompts discovered by T5 and Llama models using GGDPS, with dimensionality reduced to 2 using T-SNE. Each token embedding is color-coded according to the model that generated it. The figures are generated for the datasets where the prompts for both T5 and Llama were identified.

were generated. The dimensionality of the embeddings was reduced using T-SNE[5], and the data points are color-coded according to the model that generated the respective token. Each figure shows how the T5 model perceives the hard prompt tokens generated by GGDPS on T5 and Llama given the same dataset. From the subfigures, clustering is observed in the MRPC and BoolQ datasets, whereas the data points appear more joined for SST2 and RTE. The clustering observed in MRPC and BoolQ corresponds with lower agreement levels as indicated in Table 5.7. This suggests that the prompts discovered by GGDPS for T5 and Llama on these datasets convey different information relative to each model.

Figure 5.6 illustrates the distinctions in attention mechanisms between T5 and Llama models. The confusion matrices were constructed using maximum attention values from each head from the first and last layers of their respective encoders, based on a single data point. The values on the axes denote the token positions within the sampled data. Given the layer, the attention patterns of both models are different. T5 demonstrates a tendency to concentrate on adjacent words in the initial layer, whereas Llama directs its attention towards the beginning of the input. In the final layer, Llama exhibits higher attention towards both the initial part of the input and nearby words, while T5 displays a distinct attention pattern. It should be mentioned that the selection of these layers is based on their critical roles within the encoder framework. The first layer immediately follows the input, initiating the transformation process, while the last layer synthesizes the encoded information to prepare it for task-specific output generation.

5.3.2 Discussion

Overall, the results indicate that cross-model prompt transfer is unsuccessful, with a decline in accuracy observed across all datasets for each model performing the task. The core issue appears to be the task-specific construction of prompts that are tailored exclusively to the model that created them. This finding is evident in Figure 2.1, where the T5 and Llama prompt tokens are clustered based on the model that generated them for the MRPC and BoolQ datasets. The figure illustrates how T5 embeds prompt tokens generated by both T5 and Llama. It is evident that T5 embeds tokens from the Llama prompt differently, indicating that these tokens carry distinct information that is not similar to the T5 prompt tokens. From Table 5.7, there is a notable decline of 15% and 9% in performance when the Llama prompt is used with the T5 model for the aforementioned datasets. This indicates that the information contained in the Llama prompt is not relevant to what T5 requires for successful inference. Even though similar clustering is not explicitly observed for the



(a) First layer of Llama encoder

(b) First layer of T5 encoder

(c) Last layer of Llama encoder

(d) Last layer of T5 encoder

FIGURE 5.6: Attention distribution in the first and last layers of T5 and Llama encoder models, with the numerical values on the axes indicating token positions within the input data. The confusion matrices were generated using a single data point to exemplify the underlying differences in attention mechanisms between the two models. The figures are based on the performance of the original models without any instructions added to the input.

RTE and SST2 datasets, the decline in performance still persists.

Although it is suggested that a prompt discovered by GGDPS is model-specific, the degree of specificity appears to depend on the task and dataset at hand. For instance, the performance decline for RTE and SST2 is merely 5% and 2%, while the agreement of predictions is 83% and 97% respectively. Additionally, by looking at Figure 2.1, T5 and Llama prompt embeddings seem to lay close to each other, indicating that both models extracted similar features from the dataset. It can be hypothesized that the variability in feature extraction by different models may be influenced by the dataset's complexity, which can be quantified as the difference between human performance and machine performance as well as variability in the results on the dataset between various models [8, 9]. For instance, different models exhibit varying results on the MRPC dataset, where the performance gap relative to human is 7%, classifying this dataset as one of the more challenging in the GLUE benchmark [63].

Moreover, the extracted features are influenced by both the pre-trained knowledge and the architecture of the model [23, 35, 65]. For instance, T5 is an encoder-decoder (ED) model, while Llama is a decoder-only (DO) model. Numerous studies have shown that ED models often outperform DO models for certain sequence-to-sequence tasks, with T5 demonstrating superior results compared to Llama [10, 37]. This can be confirmed from the obtained results in this study where T5 outperforms Llama on all the datasets. From Figure 5.6, distinct patterns in attention distribution across the initial and final layers of both models can be observed. This difference in pattern suggests that the attention mechanism of T5 and Llama differ accordingly. It does not imply superiority of one over the other; rather, it suggests that each may capture different relevant details of the input depending on the specific task at hand. Nevertheless, T5 appears to be better suited for the chosen datasets, particularly given that its pre-trained data is more relevant to these datasets compared to that of Llama.

Although cross-model transfer did not yield favorable results, the findings still revealed that two distinct models are capable of extracting different features from the same textual input. Numerous studies have shown that combining the strengths of multiple models can lead to improved outcomes [34, 46, 67]. This suggests that leveraging the unique capabilities of each model could potentially enhance overall performance by capturing a broader range of relevant features. To investigate this, we combined T5 and Llama to identify a prompt that could optimize performance for both. A collaborative prompt is designed to encapsulate the extracted features from both models, thereby enhancing generalization and robustness.

5.4 About Collaborative GGDPS

This section addresses the last research question, which centers on the collaborative generation of a shared prompt that optimizes performance for T5 and Llama using GGDPS. The section begins by presenting the experimental findings, highlighting notable insights, and concludes with a thorough discussion of the results.

5.4.1 Experimental results

Table 5.8 presents the shared hard prompts that achieved the highest performance on T5 and Llama, derived using C-GGDPS across all datasets. The accuracies obtained by applying each shared prompt to T5 and Llama are documented in Table 5.9. Moreover, for the sake of easier comparison, Table 5.9 additionally presents the accuracies achieved by

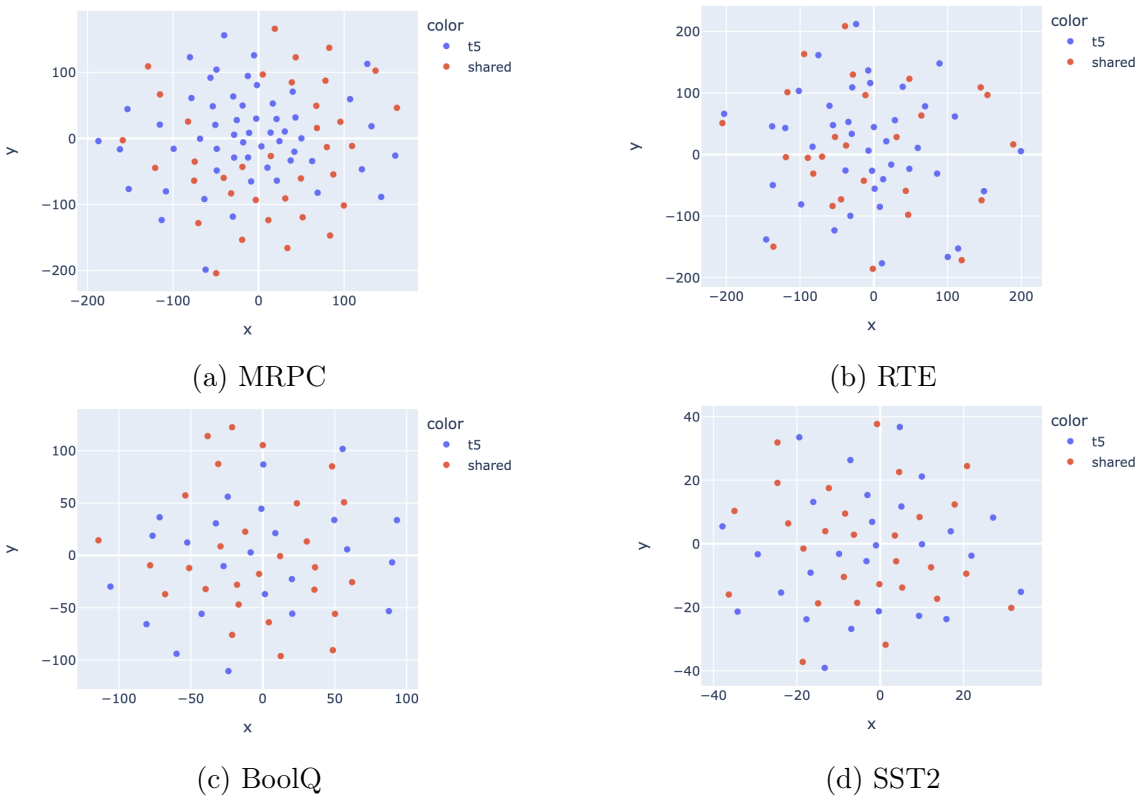
Dataset	C-GGDPS Discrete Prompt
RTE	împărat urge noted NobelRN thinkingDVtiri varfNetUploadedzin headedwatt cogn împărat WattbelTagged bur împărat prevazuttown Doromâniitable împărattons apariți favor
MRPC	givingProfMon Il Gross geboren salary Parish Warner neighbourhoodrison scene reimbursement setizari Dave findingsattach doi teilnehmen ztenfredctlos layerierung RadUnivers plusieurs rezultatul Angel
BoolQ	particlesset DUI continuepas McDonald mal nestcrystalline stuff Découvrez dilemmacal prefer Teil attendantUnion ging WWE anul Güte Serverkeineova familiar bereit wouldn concretția
WiC	drivefall insightsetLED crystalbit ToothEDM bitemoo Experiencemode dilemmafract prefer Kit interlocking ping WWE anul paradoxical linkBoston auction inner foil gamebit purchase tal
SST2	spike statesownedLL October Decicrystalline SuchTEM solidbit Schedule return acid Ro informațiiother Mal contain coincidence cet York bid internal games turmoil cumpăra smartphone
QNLI	ownedDUI Fragmentline dualpower temporpas Starbucks meldbit treeshard Qualität Residcrystal hydratepro Report data Core summary Sourcekeinenova akin await paw concretianalysis
QQP	spiralostatic tekloreRN Phaselink TRtown crestwarcogn scolded GlosarioMinitable Ion Grosszeitborn compensation Void annotationsdoi blend participanți warctfall lamination myriad resultfinal Arc

Table 5.8: The shared hard prompts identified by C-GGDPS using T5 and Llama across all datasets are presented.

Prompt	Prompt	Dataset						
		RTE	MRPC	WiC	BoolQ	SST-2	QNLI	QQP
Native	T5	71.22%	86.77%	61.76%	71.68%	97.51%	93.12%	90.36%
	Llama	70.05%	73.04%	60.64%	63.31%	88.92%	91.01%	83.62%
	Agr.	97.98%	78.23%	95.63%	87.09%	85.12%	92.63%	88.64%
Shared	T5	68.36%	80.21%	60.82%	67.11%	93.47%	90.36%	85.41%
	Llama	71.11%	76.92%	61.12%	64.49%	90.02%	92.01%	87.26%
	Agr.	96.74%	89.41%	97.26%	92.94%	91.52%	94.52%	92.83%

Table 5.9: Reported accuracies from utilizing GGDPS on T5 and Llama to identify hard prompts, along with the changes in results following the application of collaborative alternative C-GGDPS to discover a shared prompt for both models.

each model using its native prompt found by the standard GGDPS method. A significant observation is that the performance of the T5 model decreased across all datasets when the shared hard prompt was employed. On the other hand, Llama model demonstrated moderate improvements compared to its performance with hard prompts identified by the standard GGDPS. Furthermore, an agreement level between predictions of two models both before and after the application of C-GGDPS is presented. An agreement level was instrumental in identifying that the differences in predictions has indeed been minimized. Before the application of C-GGDPS, the average discrepancy was 87%, whereas it improved to 93% after application of collaborative algorithm.



(a) MRPC

(b) RTE

(c) BoolQ

(d) SST2

FIGURE 5.7: Embeddings of prompts discovered by T5 using GGDPS and shared prompts between T5 and Llama identified by C-GGDPS, with dimensionality reduced to 2 using T-SNE. Each token embedding is color-coded according to the model that generated it.

Figure 5.7 illustrates the visualization of reduced embeddings for both shared hard prompts and hard prompts identified by GGDPS for the T5 model. This visualization depicts the similarity between token embeddings as interpreted by the T5 model. The data points are color-coded based on the method by which the hard prompt token was generated: GGDPS on T5 or shared prompt using C-GGDPS. The notable insight is that the data points are more intermixed, showing no clear signs of clustering, as previously observed in Figure 5.5. This suggests that the tokens from the shared prompt and T5's hard prompts are closely positioned in the embedding space, indicating they share similar informational characteristics.

5.4.2 Discussion

Overall, C-GGDPS is an effective method of narrowing the gap between the performances of two models, which can be observed on Table 5.9. These results can be attributed to the creation of a generalized prompt that satisfies the requirements of both models in the experiment. By utilizing a shared prompt during training, a common loss function directs the identification of critical tokens that minimize losses for both models. Hence, the shared prompt incorporates essential features for effective task performance by both models, enhancing generalizability and reducing variance [14, 33, 38].

Ensuring that different models make the same predictions is crucial, since it enhances consistency and reliability, indicating that the results are robust and not model-specific [29,

59]. By comparing an agreement level between models before and after applying C-GGDPS, we can observe an average improvement of 5.5%. This enhancement improves robustness of decision-making, mitigates biases, and serves as a form of cross-validation, verifying the authenticity of observed patterns. Although the average improvement may appear moderate, it is noteworthy that this enhancement also extends to challenging datasets such as MRPC, where agreement in prediction increased by 11%. Such challenging datasets typically exhibit inter-model and cross-model variability, often due to factors such as data inconsistency or mislabeling [9]. Therefore, employing ensemble methods represents a viable strategy for addressing such datasets.

Despite C-GGDPS enhancing prediction consistency across models, it concurrently negatively affected T5's performance across all datasets. According to Table 5.9, T5 exhibited an average decrease in accuracy of 4.5%, whereas Llama, on the other hand, showed a modest improvement of 1.8% across all datasets. We hypothesize that this can be attributed to the differences in the complexity of the continuous space between both models as well as their pre-trained data. Because of its less complex continuous space, T5 could more readily identify a global minimum, potentially aiding Llama in navigating past local minima. Furthermore, the pre-trained data of T5 is more closely aligned with the datasets used in this study, whereas the pre-trained data of Llama is less related. Consequently, during training, Llama may have leveraged T5's pre-trained data to enhance its performance. As observed from SPOT and ATTEMPT, utilizing related pre-trained data from external sources positively influences the final outcome [2, 54]. However, T5 may have been negatively impacted by Llama, given that Llama initially faced challenges with convergence owing to its high-dimensional continuous space. Since the loss function is shared between the two models, T5 might have inherited Llama's convergence difficulties.

Table 5.9 presents the prompts generated by C-GGDPS using T5 and Llama. As shown in Tables 5.5 and 5.6, these prompts are not easily interpretable by humans. Moreover, when comparing the prompts obtained through C-GGDPS to those generated by a standard GGDPS, no clear relationship is visible to the human eye. One might expect that a shared prompt between T5 and Llama would contain similar tokens to the individual prompts of T5 and Llama, but this is not very evident. Nevertheless, Figure 5.7 provides a visualization of the embeddings of tokens from both the shared prompt and the T5 prompt, illustrating how the T5 model perceives these tokens. If the two prompts are significantly different, we would anticipate observing distinct clusters: grouped tokens from shared prompt and grouped tokens from the T5 prompt. However, the visualization does not exhibit any distinct clusters, and the tokens appear to be quite intermixed, indicating that they convey similar underlying information. Thus, C-GGDPS excels at incorporating features extracted by T5 into the shared prompt.

Although the shared prompt appears to contain underlying information similar to the original T5 and Llama prompts, it may still include features that are more specific to these two models. To explore this possibility, an experiment could be designed in which the shared prompt is transferred to a model that was not involved in prompt's creation. An experiment would evaluate the feasibility of this approach in generating a generalized and robust prompt. This approach could be advantageous in resource-constrained settings, where multiple smaller models are employed to create a shared prompt and a larger model is used for inference [34].

Chapter 6

Conclusions and Future Work

This concluding chapter is dedicated to addressing the posed research questions. Furthermore, it aims to identify the limitations of the study to ensure transparency regarding the design choices. The chapter concludes by suggesting potential future research directions and discussing the significance of this contribution to the field.

6.1 Conclusions

How can the tokens of soft prompts be mapped or translated into natural language tokens?

The most straightforward method for translating soft prompts into natural language involves identifying the nearest natural tokens through the embedding space and employing cosine similarity as the metric to measure the distance between neighbours. However, the results indicate that the properties of soft prompt embeddings and natural token embeddings differ significantly upon direct comparison. Soft prompt embeddings exhibit greater magnitudes and lower cosine similarity with natural language tokens. By analyzing their representation in the final layer of the encoder, it is clear that the underlying information of soft tokens is more closely associated with instructive words and dataset-specific tokens rather than with abstract tokens. Nonetheless, the cosine similarity is insufficiently low for effective direct mapping, which has adversely impacted the final performance, leading to a significant decline.

How can a model autonomously produce a hard prompt tailored to a specific task, without requiring human intervention?

In summary, the developed GGDPS algorithm demonstrated that a model can autonomously generate its own hard prompt for a specific task without human intervention. However, the proposed algorithm has both advantages and limitations. While GGDPS surpassed all baseline results on the T5 model, it underperformed on the Llama2 model. The findings suggest that GGDPS excels in leveraging pre-trained knowledge from the model to enhance task performance when the model's pre-trained data is closely related to the target task. However, when the pre-trained knowledge is significantly different, there is insufficient relevant information for the model to extract, hindering performance optimization. This limitation arises because GGDPS does not alter the attention pattern but rather scales the attention values. Moreover, GGDPS may encounter optimization challenges when applied to models with a larger continuous space, potentially resulting in

being trapped in local minima. Therefore, it is advisable to take into account both the pre-trained knowledge and the complexity of the model prior to employing the algorithm.

What impact does transferring a hard prompt from one model to another have on performance?

The results show that cross-model transfer of prompts found by GGDPS is ineffective, as found prompts are highly model-specific. T5 and Llama models extract distinct sets of textual features due to their architectural differences and pre-trained knowledge, resulting in performance declines when prompts are transferred between them. The set of extracted features varies depending on the dataset used. For complex datasets, where there remains a substantial gap between human and machine performance, the features extracted by two models are less similar and more model-specific. Therefore, for successful cross-model transfer, consideration must be given to the complexity of the task, the characteristics of the dataset, and the similarity between the models involved.

What are the effects of collaborative shared hard prompt generation between two models on achieving optimal performance for both?

Overall, C-GGDPS proves effective in bridging difference in predictions between two models providing consistency in results. The primary objective is achieved through the formulation of a prompt that contains features extracted from multiple models. This fusion of features aims to produce a prompt that is more universal and diminishes variability. Enhancing uniformity in predictions across different models enhances reliability and indicates robust results, less prone to model-specific biases. The assessment of agreement levels pre- and post-implementation of C-GGDPS illustrates enhancements, particularly evident in complex datasets. However, there is a risk of negative effects on the model if other models contributing to the ensemble possess undesirable knowledge or stuck in the local minima, potentially introducing disruptions that detracts from optimal performance.

6.2 Limitations

Since this study is simply a proof of concept, it is important to note that the evaluation is done solely on two models across seven datasets, collectively representing five distinct natural language tasks: natural language inference, paraphrase detection, sentiment analysis, word sense disambiguation, and boolean question answering. There exist numerous tasks and datasets that could potentially be employed to offer further insights into the advantages and limitations of the algorithm. Furthermore, our evaluation primarily focuses on classification tasks, and our datasets do not encompass tasks that necessitate the generation of long sequences. This imposes constraints on the study, as each task would need to be reformulated into a classification format to apply the algorithm. Tasks such as summarization, for instance, would be impractical to adapt solely into a classification framework.

Regarding the ensemble model, our exploration of methods for combining the losses of two models was limited. In this study, we opted to use a simple averaging approach to guide the algorithm. Another constraint could be the limited vocabulary size of both models, which is restricted to only 32,000 tokens. Consequently, the generated hard prompt is composed solely of these tokens. In the training process, we experienced a hardware

constraint. Due to the resource-intensive nature of loading two pre-trained models in terms of memory, it required reducing the batch size and precluded exploration of various options for hyperparameter tuning.

6.3 Future work

The suggestions for future work are built upon existing limitations of this study. Given the underexplored fusion techniques for combining two models, an interesting avenue would involve adapting an attention mechanism that governs the contribution of each loss function to the overall model loss. Alternatively, rather than guiding the algorithm with a unified loss, integrating features extracted from both models could offer insights into optimizing feature combination for enhanced performance. The described approach is referred to as early fusion and is recognized for its advantages in multimodal studies[11].

Another interesting direction would involve investigating the possibility of collaboratively creating a shared prompt on a model that was not involved in its creation process. Such a shared prompt holds promise for better robustness, potentially benefiting both prompt initialization for target tasks and direct application. Hence, such experiment would test the feasibility of the approach for creating more generalized and robust prompt.

One intriguing experiment would be to investigate the feasibility of combining more than two models to jointly create a shared prompt among them. Intuitively, this approach could lead to the development of a prompt that is comprehensible to humans, given that all LLMs are originally trained on extensive natural language corpora. This could enhance the interpretability of the resulted prompt as well as its robustness.

Shi et al. [47] introduced a technique employing Langevin dynamics to enhance the readability of hard prompts, resulting in improved performance compared to less readable prompts. Thus, an avenue for exploration could involve integrating aspects of their approach or devising methods to constrain the model in generating more readable prompts within the GGDPS algorithm framework, given its efficiency in terms of parameters and ease of implementation.

Finally, given that the study was conducted using only seven datasets, two models, and five natural language tasks, it is imperative to expand the scope to thoroughly assess the method's applicability and feasibility to the real-world.

Bibliography

[1] Ebtesam Almazrouei, Hamza Alobeidli, Abdulaziz Alshamsi, Alessandro Cappelli, Ruxandra Cojocaru, Mérouane Debbah, Étienne Goffinet, Daniel Hesslow, Julien Launay, Quentin Malartic, Daniele Mazzotta, Badreddine Noune, Baptiste Pannier, and Guilherme Penedo. The falcon series of open language models, 2023. URL: https://arxiv.org/abs/2311.16867, arXiv:2311.16867.

[2] Akari Asai, Mohammadreza Salehi, Matthew Peters, and Hannaneh Hajishirzi. ATTEMPT: Parameter-efficient multi-task tuning via attentional mixtures of soft prompts. In Yoav Goldberg, Zornitsa Kozareva, and Yue Zhang, editors, *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 6655–6672, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.emnlp-main.446, doi:10.18653/v1/2022.emnlp-main.446.

[3] Luke Bailey, Gustaf Ahdritz, Anat Kleiman, Siddharth Swaroop, Finale Doshi-Velez, and Weiwei Pan. Soft prompting might be a bug, not a feature. *OpenReview.net*, 2023. URL: https://openreview.net/forum?id=MHWDdMEJ5s#all.

[4] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL: https://arxiv.org/abs/2005.14165, arXiv:2005.14165.

[5] T. Tony Cai and Rong Ma. Theoretical foundations of t-sne for visualizing high-dimensional clustered data. *J. Mach. Learn. Res.*, 23(1), jan 2022.

[6] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019. URL: https://arxiv.org/abs/1810.04805, arXiv:1810.04805.

[7] Stefan Elfwing, Eiji Uchibe, and Kenji Doya. Sigmoid-weighted linear units for neural network function approximation in reinforcement learning. *Neural Networks*, 107:3–11, 2018. Special issue on deep reinforcement learning. URL: https://www.sciencedirect.com/science/article/pii/S0893608017302976, doi:10.1016/j.neunet.2017.12.012.

[8] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with $\mathcal{V}$-usable information. In Kamalika Chaudhuri, Stefanie Jegelka,

Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR, 17–23 Jul 2022. URL: https://proceedings.mlr.press/v162/ethayarajh22a.html.

[9] Kawin Ethayarajh and Dan Jurafsky. Utility is in the eye of the user: A critique of NLP leaderboards. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4846–4853, Online, November 2020. Association for Computational Linguistics. URL: https://aclanthology.org/2020.emnlp-main.393, doi:10.18653/v1/2020.emnlp-main.393.

[10] Zihao Fu, Wai Lam, Qian Yu, Anthony Man-Cho So, Shengding Hu, Zhiyuan Liu, and Nigel Collier. Decoder-only or encoder-decoder? interpreting language model as a regularized encoder-decoder, 2023. URL: https://arxiv.org/abs/2304.04052, arXiv:2304.04052.

[11] Konrad Gadzicki, Razieh Khamsehashari, and Christoph Zetzsche. Early vs late fusion in multimodal convolutional neural networks. In *2020 IEEE 23rd International Conference on Information Fusion (FUSION)*, pages 1–6, 2020. doi:10.23919/FUSION45008.2020.9190246.

[12] Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. PPT: Pre-trained prompt tuning for few-shot learning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.acl-long.576, doi:10.18653/v1/2022.acl-long.576.

[13] Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. WARP: Word-level Adversarial ReProgramming. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4921–4933, Online, August 2021. Association for Computational Linguistics. URL: https://aclanthology.org/2021.acl-long.381, doi:10.18653/v1/2021.acl-long.381.

[14] L.K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990. doi:10.1109/34.58871.

[15] Yaru Hao, Li Dong, Furu Wei, and Ke Xu. Investigating learning dynamics of BERT fine-tuning. In Kam-Fai Wong, Kevin Knight, and Hua Wu, editors, *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 87–92, Suzhou, China, December 2020. Association for Computational Linguistics. URL: https://aclanthology.org/2020.aacl-main.11.

[16] F. Horn and Klaus-Robert Müller. Learning similarity preserving representations with neural similarity encoders. *ArXiv*, abs/1702.01824, 2017. URL: https://api.semanticscholar.org/CorpusID:127529618.

[17] Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin de Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. Parameter-efficient transfer learning for nlp, 2019. URL: https://arxiv.org/abs/1902.00751, arXiv:1902.00751.

[18] Yukun Huang, Kun Qian, and Zhou Yu. Learning a better initialization for soft prompts via meta-learning. In *International Joint Conference on Natural Language Processing*, 2022. URL: https://api.semanticscholar.org/CorpusID:249062905.

[19] Zhengbao Jiang, Jun Araki, Haibo Ding, and Graham Neubig. How can we know when language models know? on the calibration of language models for question answering. *Transactions of the Association for Computational Linguistics*, 9:962–977, 2021. URL: https://aclanthology.org/2021.tacl-1.57, doi:10.1162/tacl_a_00407.

[20] Feihu Jin, Jinliang Lu, Jiajun Zhang, and Chengqing Zong. Instance-aware prompt learning for language understanding and generation. *ACM Transactions on Asian and Low-Resource Language Information Processing*, 22:1 – 18, 2022. URL: https://api.semanticscholar.org/CorpusID:246036341.

[21] Minji Jung, Soyeon Park, Jeewoo Sul, and Yong Suk Choi. Is prompt transfer always effective? an empirical study of prompt transfer for question answering. In Kevin Duh, Helena Gomez, and Steven Bethard, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 2: Short Papers)*, pages 528–539, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL: https://aclanthology.org/2024.naacl-short.44.

[22] Daniel Khashabi, Xinxi Lyu, Sewon Min, Lianhui Qin, Kyle Richardson, Sean Welleck, Hannaneh Hajishirzi, Tushar Khot, Ashish Sabharwal, Sameer Singh, and Yejin Choi. Prompt waywardness: The curious case of discretized interpretation of continuous prompts. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3631–3643, Seattle, United States, July 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.naacl-main.266, doi:10.18653/v1/2022.naacl-main.266.

[23] David Leake, Zachary Wilkerson, Vibhas Vats, Karan Acharya, and David Crandall. Examining the impact of network architecture on extracted feature quality for cbr. In Stewart Massie and Sutanu Chakraborti, editors, *Case-Based Reasoning Research and Development*, pages 3–18, Cham, 2023. Springer Nature Switzerland.

[24] Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics. URL: https://aclanthology.org/2021.emnlp-main.243, doi:10.18653/v1/2021.emnlp-main.243.

[25] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising

sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics. URL: https://aclanthology.org/2020.acl-main.703, doi:10.18653/v1/2020.acl-main.703.

[26] Junyi Li, Tianyi Tang, Jian-Yun Nie, Ji-Rong Wen, and Xin Zhao. Learning to transfer prompts for text generation. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3506–3518, Seattle, United States, July 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.naacl-main.257, doi:10.18653/v1/2022.naacl-main.257.

[27] Xiang Lisa Li and Percy Liang. Prefix-tuning: Optimizing continuous prompts for generation. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, Online, August 2021. Association for Computational Linguistics. URL: https://aclanthology.org/2021.acl-long.353, doi:10.18653/v1/2021.acl-long.353.

[28] Yuanchun Li, Hao Wen, Weijun Wang, Xiangyu Li, Yizhen Yuan, Guohong Liu, Jiacheng Liu, Wenxing Xu, Xiang Wang, Yi Sun, Rui Kong, Yile Wang, Hanfei Geng, Jian Luan, Xuefeng Jin, Zilong Ye, Guanjing Xiong, Fan Zhang, Xiang Li, Mengwei Xu, Zhijun Li, Peng Li, Yang Liu, Ya-Qin Zhang, and Yunxin Liu. Personal llm agents: Insights and survey about the capability, efficiency and security. *ArXiv*, arXiv:2401.05459, 2024. URL: https://doi.org/10.48550/arXiv.2401.05459.

[29] Chiara Lisciandra and Johannes Korbmacher. Multiple models, one explanation. *Journal of Economic Methodology*, 28(2):186–206, 2021. arXiv:https://doi.org/10.1080/1350178X.2021.1892800, doi:10.1080/1350178X.2021.1892800.

[30] Mengchen Liu, Jiaxin Shi, Kelei Cao, Jun Zhu, and Shixia Liu. Analyzing the training processes of deep generative models. *IEEE Transactions on Visualization and Computer Graphics*, 24(1):77–87, 2018. doi:10.1109/TVCG.2017.2744938.

[31] Xiangyang Liu, Tianxiang Sun, Xuanjing Huang, and Xipeng Qiu. Late prompt tuning: A late prompt could be better than many prompts. *ArXiv*, abs/2210.11292, 2022. URL: https://api.semanticscholar.org/CorpusID:253018816.

[32] Xiao Liu, Kaixuan Ji, Yicheng Fu, Weng Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. P-tuning: Prompt tuning can be comparable to fine-tuning across scales and tasks. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 61–68, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.acl-short.8, doi:10.18653/v1/2022.acl-short.8.

[33] Yaoyao Liu, Bernt Schiele, and Qianru Sun. An ensemble of epoch-wise empirical bayes for few-shot learning. In *European Conference on Computer Vision*, 2019. URL: https://api.semanticscholar.org/CorpusID:212726438.

[34] Xiaoding Lu, Zongyi Liu, Adian Liusie, Vyas Raina, Vineet Mudupalli, Yuwen Zhang, and William Beauchamp. Blending is all you need: Cheaper, better alternative to trillion-parameters llm, 2024. URL: https://arxiv.org/abs/2401.02994, arXiv:2401.02994.

[35] Tharindu Madusanka, Iqra Zahid, Hao Li, Ian Pratt-Hartmann, and Riza Batista-Navarro. Not all quantifiers are equal: Probing transformer-based language models' understanding of generalised quantifiers. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8680–8692, Singapore, December 2023. Association for Computational Linguistics. URL: https://aclanthology.org/2023.emnlp-main.536, doi:10.18653/v1/2023.emnlp-main.536.

[36] Katta G. Murty and Santosh N. Kabadi. Some np-complete problems in quadratic and nonlinear programming. *Mathematical Programming*, 39:117–129, 1987. URL: https://api.semanticscholar.org/CorpusID:30500771.

[37] Varun Nathan, Ayush Kumar, and Digvijay Ingle. Can probing classifiers reveal the learning by contact center large language models?: No, it doesn't! In Shabnam Tafreshi, Arjun Akula, João Sedoc, Aleksandr Drozd, Anna Rogers, and Anna Rumshisky, editors, *Proceedings of the Fifth Workshop on Insights from Negative Results in NLP*, pages 92–100, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL: https://aclanthology.org/2024.insights-1.12.

[38] Xiangyu Peng, Chen Xing, Prafulla Kumar Choubey, Chien-Sheng Wu, and Caiming Xiong. Model ensemble instead of prompt fusion: a sample-specific knowledge transfer method for few-shot prompt tuning. *ArXiv*, abs/2210.12587, 2022. URL: https://api.semanticscholar.org/CorpusID:253098972.

[39] Aleksandar Petrov, Philip H. S. Torr, and Adel Bibi. When do prompting and prefix-tuning work? a theory of capabilities and limitations. *ArXiv*, abs/2310.19698, 2023. URL: https://api.semanticscholar.org/CorpusID:264812826.

[40] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019. URL: https://api.semanticscholar.org/CorpusID:160025533.

[41] Colin Raffel, Noam M. Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2019. URL: https://api.semanticscholar.org/CorpusID:204838007.

[42] Alessandro Raganato and Jörg Tiedemann. An analysis of encoder representations in transformer-based machine translation. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 287–297, Brussels, Belgium, November 2018. Association for Computational Linguistics. URL: https://aclanthology.org/W18-5431, doi:10.18653/v1/W18-5431.

[43] Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In *Conference of the European Chapter of the Association for Computational Linguistics*, 2020. URL: https://api.semanticscholar.org/CorpusID:210838924.

[44] Timo Schick and Hinrich Schütze. It's not just size that matters: Small language models are also few-shot learners, 2021. URL: https://arxiv.org/abs/2009.07118, arXiv:2009.07118.

[45] Leo Schwinn, David Dobre, Sophie Xhonneux, Gauthier Gidel, and Stephan Günnemann. Soft prompt threats: Attacking safety alignment and unlearning in open-source llms through the embedding space. *ArXiv*, abs/2402.09063, 2024. URL: https://api.semanticscholar.org/CorpusID:267657556.

[46] Weizhou Shen, Chenliang Li, Hongzhan Chen, Ming Yan, Xiaojun Quan, Hehong Chen, Ji Zhang, and Fei Huang. Small llms are weak tool learners: A multi-llm agent, 2024. URL: https://arxiv.org/abs/2401.07324, arXiv:2401.07324.

[47] Weijia Shi, Xiaochuang Han, Hila Gonen, Ari Holtzman, Yulia Tsvetkov, and Luke Zettlemoyer. Toward human readable prompt tuning: Kubrick's the shining is a good movie, and a good prompt too? In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10994–11005, Singapore, December 2023. Association for Computational Linguistics. URL: https://aclanthology.org/2023.findings-emnlp.733, doi:10.18653/v1/2023.findings-emnlp.733.

[48] Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts, 2020. URL: https://arxiv.org/abs/2010.15980, arXiv:2010.15980.

[49] Jovan Stojkovic, Esha Choukse, Inigo Goiri Chaojie Zhang, and Josep Torrellas. Towards greener llms: Bringing energy-efficiency to the forefront of llm inference. *ArXiv*, abs/2403.20306, 2024. URL: https://doi.org/10.48550/arXiv.2403.20306.

[50] Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3949–3969, Seattle, United States, July 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.naacl-main.290, doi:10.18653/v1/2022.naacl-main.290.

[51] Shuo Sun, Yuchen Zhang, Jiahuan Yan, Yuze Gao, Donovan Ong, Bin Chen, and Jian Su. Battle of the large language models: Dolly vs LLaMA vs vicuna vs guanaco vs bard vs ChatGPT - a text-to-SQL parsing comparison. In Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 11225–11238, Singapore, December 2023. Association for Computational Linguistics. URL: https://aclanthology.org/2023.findings-emnlp.750, doi:10.18653/v1/2023.findings-emnlp.750.

[52] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Daniel M. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony S. Hartshorn, Saghar Hosseini,

Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. V. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *ArXiv*, abs/2307.09288, 2023. URL: https://api.semanticscholar.org/CorpusID:259950998.

[53] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023. URL: https://arxiv.org/abs/1706.03762, arXiv:1706.03762.

[54] Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. SPoT: Better frozen model adaptation through soft prompt transfer. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland, May 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.acl-long.346, doi:10.18653/v1/2022.acl-long.346.

[55] Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. Universal adversarial triggers for attacking and analyzing NLP. In Kentaro Inui, Jing Jiang, Vincent Ng, and Xiaojun Wan, editors, *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2153–2162, Hong Kong, China, November 2019. Association for Computational Linguistics. URL: https://aclanthology.org/D19-1221, doi:10.18653/v1/D19-1221.

[56] Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems, 2020. URL: https://arxiv.org/abs/1905.00537, arXiv:1905.00537.

[57] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In Tal Linzen, Grzegorz Chrupała, and Afra Alishahi, editors, *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium, November 2018. Association for Computational Linguistics. URL: https://aclanthology.org/W18-5446, doi:10.18653/v1/W18-5446.

[58] Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States, July 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.naacl-main.167, doi:10.18653/v1/2022.naacl-main.167.

[59] Yicheng Wu, Zongyuan Ge, Donghao Zhang, Minfeng Xu, Lei Zhang, Yong Xia, and Jianfei Cai. Mutual consistency learning for semi-supervised medical image segmentation. *Medical Image Analysis*, 81:102530, 2022. URL: https://www.sciencedirect.com/science/article/pii/S1361841522001773, doi:10.1016/j.media.2022.102530.

[60] Zhuofeng Wu, Sinong Wang, Jiatao Gu, Rui Hou, Yuxiao Dong, V.G.Vinod Vydiswaran, and Hao Ma. IDPG: An instance-dependent prompt generation method. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5507–5521, Seattle, United States, July 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.naacl-main.403, doi:10.18653/v1/2022.naacl-main.403.

[61] Lingling Xu, Haoran Xie, Si-Zhao Joe Qin, Xiaohui Tao, and Fu Lee Wang. Parameter-efficient fine-tuning methods for pretrained language models: A critical review and assessment, 2023. URL: https://arxiv.org/abs/2312.12148, arXiv:2312.12148.

[62] Ni Xuanfan and Li Piji. A systematic evaluation of large language models for natural language generation tasks. In Jiajun Zhang, editor, *Proceedings of the 22nd Chinese National Conference on Computational Linguistics (Volume 2: Frontier Forum)*, pages 40–56, Harbin, China, August 2023. Chinese Information Processing Society of China. URL: https://aclanthology.org/2023.ccl-2.4.

[63] Linyi Yang, Shuibai Zhang, Libo Qin, Yafu Li, Yidong Wang, Hanmeng Liu, Jindong Wang, Xing Xie, and Yue Zhang. GLUE-X: Evaluating natural language understanding models from an out-of-distribution generalization perspective. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12731–12750, Toronto, Canada, July 2023. Association for Computational Linguistics. URL: https://aclanthology.org/2023.findings-acl.806, doi:10.18653/v1/2023.findings-acl.806.

[64] Hongwei Yao, Jian Lou, and Zhan Qin. Poisonprompt: Backdoor attack on prompt-based large language models. In *ICASSP 2024 - 2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 7745–7749, 2024. doi:10.1109/ICASSP48485.2024.10446267.

[65] Jian Zhang, Bo Qin, Yufei Zhang, Junhua Zhou, and Hongwei Wang. A knowledge extraction framework for domain-specific application with simplified pre-trained language model and attention-based feature extractor. *Service Oriented Computing and Applications*, 16:1–11, 06 2022. doi:10.1007/s11761-022-00337-5.

[66] Yue Zhang, Hongliang Fei, Dingcheng Li, and Ping Li. PromptGen: Automatically generate prompts using generative models. In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 30–37, Seattle, United States, July 2022. Association for Computational Linguistics. URL: https://aclanthology.org/2022.findings-naacl.3, doi:10.18653/v1/2022.findings-naacl.3.

[67] Yuechen Zhang, Shengju Qian, Bohao Peng, Shu Liu, and Jiaya Jia. Prompt highlighter: Interactive control for multi-modal llms. In *Proceedings of the IEEE/CVF*

Conference on Computer Vision and Pattern Recognition (CVPR), pages 13215–13224, June 2024.

[68] Hang Zhao, Qing Liu, and Yun Yang. Transfer learning with ensemble of multiple feature representations. In *2018 IEEE 16th International Conference on Software Engineering Research, Management and Applications (SERA)*, pages 54–61, 2018. doi:10.1109/SERA.2018.8477189.

[69] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation, 2024. URL: https://arxiv.org/abs/2208.10160, arXiv:2208.10160.

[70] Qihuang Zhong, Liang Ding, Juhua Liu, Bo Du, and Dacheng Tao. Panda: Prompt transfer meets knowledge distillation for efficient model adaptation. *IEEE Transactions on Knowledge and Data Engineering*, pages 1–14, 2024. doi:10.1109/TKDE.2024.3376453.

[71] Andy Zou, Zifan Wang, J. Zico Kolter, and Matt Fredrikson. Universal and transferable adversarial attacks on aligned language models. *ArXiv*, abs/2307.15043, 2023. URL: https://api.semanticscholar.org/CorpusID:260202961.

Appendix A

Complementary material

Dataset	Label	Data Example
SST2	positive	sentence: that loves its characters and communicates something rather beautiful about human nature
	negative	sentence: hide new secretions from the parental units
QNLI	entailment	sentence: When talking about the German language, the term German dialects is only used for the traditional regional varieties. question: When is the term 'German dialects' used in regard to the German language?
	not entailment	sentence: From the 1720s onward, the kingdom was beset with repeated Meithei raids into Upper Myanmar and a nagging rebellion in Lan Na. question: How were the Portuguese expelled from Myanmar?
QQP	duplicate	question 1: How do I control my horny emotions? question 2: How do you control your horniness?
	not duplicate	question 1: What causes stool color to change to yellow? question 2: What can cause stool to come out as little balls?

TABLE A.1: Provided examples of data from the following datasets: SST2, QNLI, and QQP. These examples are formatted according to the template used as input for the models.

Dataset	Label	Data Example
RTE	entailment	sentence 1: Valero Energy Corp., on Monday, said it found "extensive" additional damage at its 250,000-barrel-per-day Port Arthur refinery, sentence 2: Valero Energy Corp. produces 250,000 barrels per day.
	not entailment	sentence 1: No Weapons of Mass Destruction Found in Iraq Yet, sentence 2: Weapons of Mass Destruction Found in Iraq.
MRPC	equivalent	sentence 1: Amrozi accused his brother , whom he called "the witness", of deliberately distorting his evidence, sentence 2: Referring to him as only " the witness ", Amrozi accused his brother of deliberately distorting his evidence .
	not equivalent	sentence 1: The Nasdaq had a weekly gain of 17.27 , or 1.2 percent , closing at 1,520.15 on Friday, sentence 2: The tech-laced Nasdaq Composite.IXIC rallied 30.46 points, or 2.04 percent, to 1,520.15.
WiC	true	sentence 1: We beat the competition, sentence 2: Agassi beat Becker in the tennis championship, word: beat
	false	sentence 1: There's a lot of trash on the bed of the river, sentence 2: I keep a glass of water next to my bed when I sleep, word: bed
BoolQ	true	question: is windows movie maker part of windows essentials, passage: Windows Movie Maker (formerly known as Windows Live Movie Maker in Windows 7) is a discontinued video editing software by Microsoft. It is a part of Windows Essentials software suite and offers the ability to create and edit videos as well as to publish them on OneDrive, Facebook, Vimeo, YouTube, and Flickr.
	false	question: can you use oyster card at epsom station, passage: Epsom railway station serves the town of Epsom in Surrey. It is located off Waterloo Road and is less than two minutes' walk from the High Street. It is not in the London Oyster card zone unlike Epsom Downs or Tattenham Corner stations. The station building was replaced in 2012/2013 with a new building with apartments above the station (see end of article).

TABLE A.2: Provided examples of data from the following datasets: RTE, MRPC, WiC, and BoolQ. These examples are formatted according to the template used as input for the models.