Prefix-Tuning: Optimizing Continuous Prompts for Generation

Xiang Lisa Li
Stanford University
xlisali@stanford.edu

Percy Liang
Stanford University
pliang@cs.stanford.edu

Abstract

Fine-tuning is the de facto way of leveraging large pretrained language models for downstream tasks. However, fine-tuning modifies all the language model parameters and therefore necessitates storing a full copy for each task. In this paper, we propose prefix-tuning, a lightweight alternative to fine-tuning for natural language generation tasks, which keeps language model parameters frozen and instead optimizes a sequence of *continuous task-specific* vectors, which we call the *prefix*. Prefix-tuning draws inspiration from prompting for language models, allowing subsequent tokens to attend to this prefix as if it were "virtual tokens". We apply prefix-tuning to GPT-2 for table-to-text generation and to BART for summarization. We show that by modifying only 0.1% of the parameters, prefix-tuning obtains comparable performance in the full data setting, outperforms fine-tuning in low-data settings, and extrapolates better to examples with topics that are unseen during training.

1 Introduction

Fine-tuning is the prevalent paradigm for using large pretrained language models (LMs) (Radford et al., 2019; Devlin et al., 2019) to perform downstream tasks (e.g., summarization), but it requires updating and storing all the parameters of the LM. Consequently, to build and deploy NLP systems that rely on large pretrained LMs, one currently needs to store a modified copy of all the LM parameters for each task. This can be prohibitively expensive given the size of current LMs; for example, GPT-2 has 774M parameters (Radford et al., 2019) and GPT-3 has 175B parameters (Brown et al., 2020).

A natural approach to this problem is *lightweight fine-tuning*, which freezes most of the pretrained parameters and only tunes a smaller set of parameters. For example, adapter-tuning (Rebuffi et al., 2017; Houlsby et al., 2019) inserts additional task-specific layers between the layers of pretrained language models. Adapter-tuning has promising performance on natural language understanding and generation benchmarks, attaining comparable performance with fine-tuning while adding only around 2–4% task-specific parameters (Houlsby et al., 2019; Lin et al., 2020).

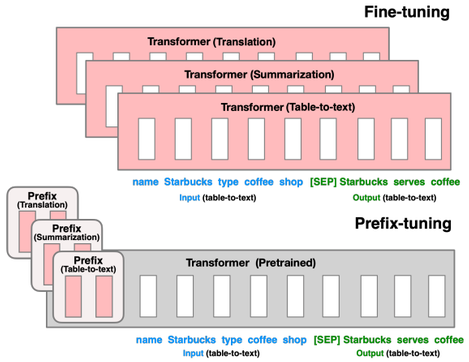


Figure 1: Fine-tuning (top) updates all LM parameters (the red Transformer box) and requires storing a full model copy for each task. We propose prefix-tuning (bottom), which freezes the LM parameters and only optimizes the prefix (the red prefix blocks). Consequently, we only need to store the prefix for each task, making prefix-tuning modular and space-efficient. Note that each vertical block denote transformer activations at one time step.

At the limit, GPT-3 (Brown et al., 2020) can be deployed using in-context learning, which is a form of *prompting*, without modifying any LM parameters. In in-context learning, Brown et al. (2020) prepend a natural language task instruction (e.g., *TL;DR* for summarization) and a few examples to the task input, and then generate the task output from the LM. However, since Transformers can only condition on a bounded-length context (e.g., 2048 tokens for GPT-3), in-context learning is restricted to very small training sets.

Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, pages 4582–4597
August 1–6, 2021.

Điều chỉnh tiền tố: Tối ưu hóa lời nhắc liên tục để tạo

Tư ơ ng Lisa Li
Đại học Stanford
xlisali@stanford.edu

Percy Liang
Đại học Stanford
pliang@cs.stanford.edu

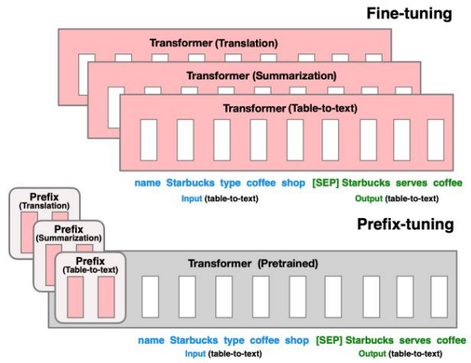
trừu tư ợng

Tinh chỉnh là cách tận dụng trên thực tế các mô hình ngôn ngữ đư ợc huấn luyện trư ớc lớn cho các tác vụ tiếp theo. Tuy nhiên, việc tinh chỉnh sửa đổi tất cả các tham số mô hình ngôn ngữ và do đó cần phải lư u trữ một bản sao đầy đủ cho mỗi tham số nhiệm vụ. Trong bài báo này, chúng tôi đề xuất điều chỉnh tiền tố, một giải pháp thay thế nhẹ nhàng để tinh chỉnh các tác vụ tạo ngôn ngữ tự nhiên, giúp cố định các tham số mô hình ngôn ngữ và thay vào đó tối ư u hóa một chuỗi các tác vụ cụ thể liên tục vectơ mà chúng tôi gọi là tiền tố. Điều chỉnh tiền tố lấy cảm hứng từ việc gợi ý ngôn ngữ mô hình, cho phép các token tiếp theo tham dự với tiền tố này như thể nó là "mã thông báo ảo". Chúng tôi áp dụng tính năng điều chỉnh tiền tố cho GPT-2 để tạo bảng thành văn bản và cho BART để tóm tắt. Chúng tôi chứng minh rằng bằng cách chỉ sửa đổi 0,1% các tham số, tính năng điều chỉnh tiền tố đạt đư ợc hiệu suất tư ơ ng đư ơ ng trong cài đặt dữ liệu đầy đủ, tinh chỉnh dạng vư ợt trội trong cài đặt dữ liệu thấp và ngoại suy tốt hơ n cho các ví dụ có chủ đề không đư ợc nhìn thấy trong quá trình đào tạo.

1. Giới thiệu

Tinh chỉnh là mô hình phổ biến để sử dụng mô hình ngôn ngữ đư ợc huấn luyện trư ớc lớn (LM) (Radford và cộng sự, 2019; Devlin và cộng sự, 2019) để thực hiện các nhiệm vụ xuôi dòng (ví dụ: tóm tắt), như ng nó đòi hỏi cập nhật và lư u trữ tất cả các tham số của LM. Do đó, để xây dựng và triển khai hệ thống NLP dựa vào các LM lớn đã đư ợc huấn luyện trư ớc, một LM hiện đang cần lư u trữ một bản sao sửa đổi của tất cả các tham số LM cho mỗi tác vụ. Điều này có thể bị cấm đắt tiền so với kích thư ớc của LM hiện tại; ví dụ : GPT-2 có 774M tham số (Radford và cộng sự, 2019) và GPT-3 có thông số 175B (Màu nâu và cộng sự, 2020).

Một cách tiếp cận tự nhiên cho vấn đề này là nhẹ nhàng tinh chỉnh, đóng băng hầu hết các dữ liệu đã đư ợc huấn luyện trư ớc tham số và chỉ điều chỉnh một tập hợp tham số nhỏ hơ n . Ví dụ: điều chỉnh bộ chuyển đổi (Rebuffi et al., 2017; Houlsby và cộng sự, 2019) chèn thêm các lớp dành riêng cho nhiệm vụ giữa các lớp đư ợc đào tạo trư ớc các mô hình ngôn ngữ Điều chỉnh bộ điều hợp có nhiều hứa hẹn Hiệu suất hiểu ngôn ngữ tự nhiên và điểm chuẩn thế hệ, đạt đư ợc mức tư ơ ng đư ơ ng hiệu suất với tinh chỉnh trong khi chỉ thêm khoảng 2–4% tham số dành riêng cho nhiệm vụ (Houlsby và cộng sự, 2019; Lin và cộng sự, 2020).



Hình 1: Tinh chỉnh (trên cùng) cập nhật tất cả các tham số LM (hộp Biến áp màu đỏ) và yêu cầu lư u trữ một bản sao mô hình đầy đủ cho mỗi nhiệm vụ. Chúng tôi đề xuất điều chỉnh tiền tố (phía dư ới), giúp cố định các tham số LM và chỉ tối ư u hóa tiền tố (khối tiền tố màu đỏ). Do đó, chúng ta chỉ cần lư u trữ tiền tố cho mỗi nhiệm vụ, làm cho việc điều chỉnh tiền tố trở thành mô-đun và tiết kiệm không gian. Lư u ý rằng mỗi khối dọc biểu thị hoạt động của máy biến áp tại một bư ớc thời gian.

Ở giới hạn, GPT-3 (Brown và cộng sự, 2020) có thể đư ợc triển khai bằng cách sử dụng phư ơ ng pháp học tập trong ngữ cảnh, tức là một hình thức nhắc nhở, không sửa đổi bất kỳ LM nào thông số. Trong học tập trong ngữ cảnh, Brown et al. (2020) thêm vào trư ớc một hư ớng dẫn nhiệm vụ ngôn ngữ tự nhiên (ví dụ: TL;DR để tóm tắt) và một vài ví dụ về đầu vào tác vụ, sau đó tạo tác vụ đầu ra từ LM. Tuy nhiên, kể từ khi Transformers chỉ có thể điều kiện dựa trên bối cảnh có độ dài giới hạn (ví dụ: 2048 mã thông báo cho GPT-3), học tập trong ngữ cảnh bị giới hạn ở các tập huấn luyện rất nhỏ.

Kỷ yếu Hội nghị thư ờng niên lần thứ 59 của Hiệp hội Ngôn ngữ học tính toán và Hội nghị chung quốc tế lần thứ 11 về xử lý ngôn ngữ tự nhiên, trang 4582–4597
Ngày 1–6 tháng 8 năm 2021.

In this paper, we propose *prefix-tuning*, a lightweight alternative to fine-tuning for natural language generation (NLG) tasks, inspired by prompting. Consider the task of generating a textual description of a data table, as shown in Figure 1, where the task input is a linearized table (e.g., "name: Starbucks | type: coffee shop") and the output is a textual description (e.g., "Starbucks serves coffee."). Prefix-tuning prepends a sequence of *continuous task-specific* vectors to the input, which we call a *prefix*, depicted by red blocks in Figure 1 (bottom). To generate each token, the LM can attend to the prefix as if it were a sequence of "virtual tokens", but unlike prompting, the prefix consists entirely of free parameters which do not correspond to real tokens. In contrast to fine-tuning in Figure 1 (top), which updates all LM parameters and thus requires storing a tuned copy of the model for each task, prefix-tuning only optimizes the prefix. Consequently, we only need to store one copy of the large LM and a learned task-specific prefix, yielding a very small overhead for each additional task (e.g., 250K parameters for table-to-text).

In contrast to full fine-tuning, prefix-tuning is also modular: we train an upstream prefix which steers an unmodified LM, and therefore, a single LM can support many tasks at once. In the context of personalization where the tasks correspond to users (Shokri and Shmatikov, 2015; McMahan et al., 2016), we would have a separate prefix for each user trained only on that user's data, thereby avoiding data cross-contamination. Moreover, the prefix-based architecture enables us to even process examples from multiple users/tasks in a single batch, something that is not possible with other lightweight fine-tuning approaches like adapter-tuning.

We evaluate prefix-tuning on table-to-text generation using GPT-2 and abstractive summarization using BART. In terms of storage, prefix-tuning stores 1000x fewer parameters than full fine-tuning. In terms of performance when trained on full datasets, prefix-tuning and fine-tuning are comparable for table-to-text (§6.1), while prefix-tuning suffers a small degradation for summarization (§6.2). In low-data settings, prefix-tuning outperforms fine-tuning on both tasks (§6.3). Prefix-tuning also extrapolates better to tables (for table-to-text) and articles (for summarization) with unseen topics (§6.4).

2 Related Work

Fine-tuning for natural language generation. Current state-of-the-art systems for natural language generation (NLG) are based on fine-tuning pretrained LMs. For table-to-text generation, Kale (2020) fine-tunes a sequence-to-sequence model (T5; Raffel et al., 2020). For extractive and abstractive summarization, researchers fine-tune masked language models (e.g., BERT; Devlin et al., 2019) and encode-decoder models (e.g., BART; Lewis et al., 2020), respectively (Zhong et al., 2020; Liu and Lapata, 2019; Raffel et al., 2020). For other conditional NLG tasks such as machine translation and dialogue generation, fine-tuning is also the prevalent paradigm (Zhang et al., 2020c; Stickland et al., 2020; Zhu et al., 2020; Liu et al., 2020). In this paper, we focus on table-to-text using GPT-2 and summarization using BART, but prefix-tuning in principle can be applied to other generation tasks and pretrained models, such as masked LMs.

Lightweight fine-tuning. Prefix-tuning falls under the broad class of lightweight fine-tuning methods, which freeze most of the pretrained parameters and only tune a smaller set of parameters. The key question is how to augment the LM architecture and decide which subset of pretrained parameters to tune. One line of research learns a task-specific parameter mask (Zhao et al., 2020; Radiya-Dixit and Wang, 2020). Another line of research inserts new modules with trainable parameters. For example, Zhang et al. (2020a) trains a "side" network that is fused with the pretrained model via summation; adapter-tuning inserts task-specific layers (adapters) between each layer of the pretrained LM (Houlsby et al., 2019; Lin et al., 2020; Rebuffi et al., 2017; Pfeiffer et al., 2020). Compared to this line of work, which tunes around 3.6% of the LM parameters, our method obtains a further 30x reduction in task-specific parameters, tuning only 0.1% while maintaining comparable performance on table-to-text tasks.

Prompting. Prompting is a way of leveraging a pretrained LM by prepending instructions and a few examples to the task input and generating the task output from the LM. For autoregressive LMs, the most successful form of prompting is GPT-3's in-context learning (Brown et al., 2020), which uses manually designed prompts to adapt its generation for different tasks in few-shot settings. For masked LMs like BERT and RoBERTa (Liu et al.,

Trong bài báo này, chúng tôi đề xuất điều chỉnh tiền tố, một giải pháp thay thế nhẹ nhàng cho việc tinh chỉnh các tác vụ tạo ngôn ngữ tự nhiên (NLG), lấy cảm hứng từ việc nhắc nhở. Hãy xem xét nhiệm vụ tạo mô tả văn bản của bảng dữ liệu, như trong Hình 1, trong đó đầu vào tác vụ là một bảng tuyến tính hóa (ví dụ: "tên: Starbucks | loại: quán cà phê") và đầu ra là một mô tả bằng văn bản (ví dụ: "Starbucks phục vụ cà phê."). Điều chỉnh tiền tố thêm vào trước một chuỗi các vectơ nhiệm vụ cụ thể liên tục tới đầu vào, chúng tôi gọi một tiền tố, được mô tả bằng các khối màu đỏ trong Hình 1 (đáy). Để tạo ra mỗi mã thông báo, LM có thể xử lý tiền tố như thể nó là một chuỗi "ảo". mã thông báo", nhưng không giống như lời nhắc, tiền tố bao gồm hoàn toàn có các tham số miễn phí không tương ứng thành token thật. Ngược lại với tinh chỉnh trong Hình 1 (trên cùng), cập nhật tất cả các tham số LM và do đó yêu cầu lưu trữ một bản sao đã điều chỉnh của mô hình cho mỗi nhiệm vụ, việc điều chỉnh tiền tố chỉ tối ưu hóa tiền tố. Do đó, chúng ta chỉ cần lưu trữ một bản sao của LM lớn và tiền tố dành riêng cho nhiệm vụ đã học, mang lại chi phí rất nhỏ cho mỗi nhiệm vụ bổ sung (ví dụ: tham số 250K cho chuyển bảng thành văn bản).

Ngược lại với tinh chỉnh đầy đủ, tinh chỉnh tiền tố là cũng theo mô-đun: chúng tôi huấn luyện một tiền tố ngược dòng điều khiển một LM chưa được sửa đổi, và do đó, một LM có thể hỗ trợ nhiều nhiệm vụ cùng một lúc. Trong bối cảnh cá nhân hóa nơi các nhiệm vụ tương ứng tới người dùng (Shokri và Shmatikov, 2015; McMahan et al., 2016), chúng ta sẽ có một tiền tố riêng cho mỗi người dùng chỉ được đào tạo về dữ liệu của người dùng đó, do đó tránh lây nhiễm chéo dữ liệu. Hơn nữa, kiến trúc dựa trên tiền tố cho phép chúng tôi thậm chí xử lý các ví dụ từ nhiều người dùng/tác vụ trong một hàng loạt, điều mà các loại khác không thể làm được các phương pháp tinh chỉnh nhẹ như điều chỉnh bộ chuyển đổi.

Chúng tôi đánh giá việc điều chỉnh tiền tố trong quá trình tạo bảng thành văn bản bằng GPT-2 và tóm tắt trừu tượng bằng BART. Về mặt lưu trữ, điều chỉnh tiền tố lưu trữ ít tham số hơn 1000 lần so với tinh chỉnh đầy đủ. Về hiệu suất khi được đào tạo đầy đủ bộ dữ liệu, điều chỉnh tiền tố và tinh chỉnh có thể so sánh được với tính năng chuyển bảng thành văn bản (§6.1), trong khi việc điều chỉnh tiền tố làm giảm đi một chút khả năng tóm tắt (§6.2). Trong cài đặt dữ liệu thấp, việc tinh chỉnh tiền tố hoạt động tốt hơn việc tinh chỉnh trên cả hai tác vụ (§6.3). Việc điều chỉnh tiền tố cũng ngoại suy tốt hơn cho các bảng (đối với chuyển bảng thành văn bản) và các bài viết (để tóm tắt) với các chủ đề không được nhìn thấy (§6.4)

2 công việc liên quan

Tinh chỉnh để tạo ngôn ngữ tự nhiên. Các hệ thống tiên tiến nhất hiện nay để tạo ra ngôn ngữ tự nhiên (NLG) đều dựa trên việc tinh chỉnh LM được huấn luyện trước. Để tạo bảng thành văn bản, Kale (2020) tinh chỉnh mô hình theo trình tự (T5; Raffel và cộng sự, 2020). Để tóm tắt mang tính khai thác và trừu tượng , các nhà nghiên cứu tinh chỉnh mặt nạ mô hình ngôn ngữ (ví dụ BERT; Devlin và cộng sự, 2019) và các mô hình mã hóa-giải mã (ví dụ BART; Lewis và cộng sự, 2020), tương ứng (Zhong và cộng sự, 2020; Liu và Lapata, 2019; Raffel và cộng sự, 2020). Cho người khác các nhiệm vụ NLG có điều kiện như dịch máy tạo ra các cuộc hội thoại và hội thoại, việc tinh chỉnh cũng là mô hình phổ biến (Zhang và cộng sự, 2020c; Stickland và cộng sự, 2020; Zhu và cộng sự, 2020; Liu và cộng sự, 2020). TRONG bài báo này, chúng tôi tập trung vào việc chuyển bảng thành văn bản bằng GPT-2 và tóm tắt bằng BART, nhưng điều chỉnh tiền tố về nguyên tắc có thể được áp dụng cho các nhiệm vụ tạo khác và các mô hình được huấn luyện trước, chẳng hạn như LM bị che.

Tinh chỉnh nhẹ. Điều chỉnh tiền tố rơi thuộc lớp rộng rãi của tinh chỉnh nhẹ các phương pháp đóng băng hầu hết các dữ liệu đã được huấn luyện trước tham số và chỉ điều chỉnh một bộ tham số nhỏ hơn . Câu hỏi quan trọng là làm thế nào để tăng LM kiến trúc và quyết định tập con nào của dữ liệu được huấn luyện trước các thông số để điều chỉnh. Một dòng nghiên cứu tìm hiểu một mặt nạ tham số dành riêng cho nhiệm vụ (Zhao và cộng sự, 2020; Radiya-Dixit và Wang, 2020). Một dòng khác nghiên cứu chèn các mô-đun mới với khả năng đào tạo được thông số. Ví dụ, Zhang và cộng sự. (2020a) đào tạo một mạng "phụ" được hợp nhất với mô hình được huấn luyện trước thông qua phép tính tổng; điều chỉnh bộ chuyển đổi chèn các lớp (bộ điều hợp) dành riêng cho nhiệm vụ giữa mỗi lớp lớp LM được huấn luyện trước (Houlsby và cộng sự, 2019; Lin và cộng sự, 2020; Rebuffi và cộng sự, 2017; Pfeiffer và cộng sự, 2020). So với dòng công việc này, giai điệu nào khoảng 3,6% tham số LM, phương pháp của chúng tôi được giảm thêm 30 lần trong các nhiệm vụ cụ thể các thông số, chỉ điều chỉnh 0,1% trong khi duy trì hiệu suất tương đương trên các tác vụ chuyển bảng thành văn bản.

Nhắc nhở. Nhắc nhở là một cách tận dụng LM đã được huấn luyện trước bằng cách chuẩn bị trước các hướng dẫn và một vài ví dụ về việc nhập nhiệm vụ và tạo ra đầu ra nhiệm vụ từ LM. Đối với LM tự hồi quy, hình thức nhắc nhở thành công nhất là GPT-3 học tập trong ngữ cảnh (Brown và cộng sự, 2020), trong đó sử dụng các lời nhắc được thiết kế thủ công để điều chỉnh thế hệ của nó cho các tác vụ khác nhau trong cài đặt vài lần chụp. Vì các LM đeo mặt nạ như BERT và RoBERTa (Liu và cộng sự,

2019), prompt engineering has been explored for natural language understanding tasks (Jiang et al., 2020; Schick and Schütze, 2020). For example, AutoPrompt (Shin et al., 2020) searches for a sequence of discrete trigger words and concatenates it with each input to elicit sentiment or factual knowledge from BERT and RoBERTa. In contrast with AutoPrompt, our method optimizes continuous prefixes, which are more expressive (§7.2); moreover, we focus on language generation tasks.

Continuous vectors have been used to steer LMs; for example, Subramani et al. (2020) showed that a pretrained LSTM language model can reconstruct arbitrary sentences by optimizing a continuous vector for each sentence, making the vector *input-specific*. In contrast, prefix-tuning optimizes a *task-specific* prefix that applies to all instances of that task. As a result, unlike the previous work whose application is limited to sentence reconstruction, prefix-tuning can be applied to NLG tasks.

Controllable generation. Controllable generation aims to steer a pretrained language model to match a sentence-level attribute (e.g., positive sentiment or sports). Such control can happen at training time: Keskar et al. (2019) pretrains the language model (CTRL) to condition on metadata such as keywords or URLs. The control can also happen at decoding time, by weighted decoding (GeDi, Krause et al., 2020) or iteratively updating the past activations (PPLM, Dathathri et al., 2020). However, there is no straightforward way to apply these controllable generation techniques to enforce fine-grained control over generated contents, as demanded by tasks like table-to-text and summarization.

P*-tuning. Prefix tuning is an instance of a new class of methods that has emerged, which we call p*-tuning (since the other prominent instances, p-tuning and prompt-tuning, also start with p), all based on the idea of optimizing a continuous prefix or prompt. Concurrent with our work, Qin and Eisner (2021) learn mixtures of soft fill-in-the-blank prompts to elicit knowledge from LMs such as BERT and BART. Hambardzumyan et al. (2021) learns task-specific embeddings that adapts BERT for sentiment classification. Both works show that tuning soft prompts outperforms previous work, which optimizes over discrete prompts. P-tuning (Liu et al., 2021) shows that jointly updating the prompt embeddings and LM parameters improves GPT-2's performance on natural language understanding tasks, in both few-shot and full data settings. In a followup work, Prompt-tuning (Lester et al., 2021) simplifies our approach and applies it to T5 (Raffel et al., 2020), demonstrating that the performance gap between fine-tuning and p*-tuning vanishes as the model size grows.

3 Problem Statement

Consider a conditional generation task where the input x is a context and the output y is a sequence of tokens. We focus on two tasks, shown in Figure 2 (right): In table-to-text, x corresponds to a linearized data table and y is a textual description; in summarization, x is an article and y is a summary.

3.1 Autoregressive LM

Assume we have an autoregressive neural language model $p_\phi(y \mid x)$ parametrized by ϕ (e.g., GPT-2; Radford et al., 2019). As shown in Figure 2 (top), let $z = [x; y]$ be the concatenation of x and y; let $\mathsf{X}_{\mathsf{idx}}$ denote the sequence of indices that corresponds to x, and $\mathsf{Y}_{\mathsf{idx}}$ denote the same for y.

The activation vector at time step i is $h_i \in \mathbb{R}^d$, where $h_i = [h_i^{(1)}; \cdots; h_i^{(n)}]$ is a concatenation of all activation layers at this time step, and $h_i^{(j)}$ is the activation vector of the j-th layer at time step i.[1]

An autoregressive neural LM computes h_i as a function of z_i and the past activations in its left context, as follows:

$$h_i = \mathrm{LM}_\phi(z_i, h_{<i}), \tag{1}$$

where the last layer of h_i is used to compute the distribution for the next token: $p_\phi(z_{i+1} \mid h_{\le i}) = \mathrm{softmax}(W_\phi\, h_i^{(n)})$ and W_ϕ is a matrix that maps $h_i^{(n)}$ to logits over the vocabulary.

3.2 Encoder-Decoder Architecture

We can also use an encoder-decoder architecture (e.g., BART; Lewis et al., 2020) to model $p_\phi(y \mid x)$, where x is encoded by the bidirectional encoder, and the decoder predicts y autoregressively (conditioned on the encoded x and its left context). We use the same indexing and activation notation, as shown in Figure 2 (bottom): each h_i for $i \in \mathsf{X}_{\mathsf{idx}}$ is computed by the a bidirectional encoder; each h_i for $i \in \mathsf{Y}_{\mathsf{idx}}$ is computed by an autoregressive decoder using the same equation (1).

[1] In GPT-2, $h_i^{(n)}$ consists of a key-value pair, and the dimension of each key and value is 1024.

2019), kỹ thuật nhắc nhở đã đư ợc khám phá cho các nhiệm vụ hiểu ngôn ngữ tự nhiên (Jiang và cộng sự, 2020; Schick và Schutze ¨ , 2020). Ví dụ: AutoPrompt (Shin và cộng sự, 2020) tìm kiếm một chuỗi các từ kích hoạt rời rạc và ghép nó với mỗi đầu vào để khơ i gợi cảm xúc hoặc kiến thức thực tế từ BERT và RoBERTa. Ngư ợc lại với AutoPrompt, phư ơ ng pháp của chúng tôi tối ư u hóa các tiền tố liên tục, mang tính biểu cảm hơ n (§7.2); hơ n nữa, chúng tôi tập trung vào các nhiệm vụ tạo ngôn ngữ.

Các vectơ liên tục đã đư ợc sử dụng để điều khiển LM; ví dụ, Subramani và cộng sự. (2020) đã chỉ ra rằng mô hình ngôn ngữ LSTM đư ợc huấn luyện trư ớc có thể xây dựng lại các câu tùy ý bằng cách tối ư u hóa một vectơ liên tục cho mỗi câu, làm cho vectơ đầu vào trở nên cụ thể. Ngư ợc lại, điều chỉnh tiền tố tối ư u hóa tiền tố dành riêng cho nhiệm vụ áp dụng cho tất cả các phiên bản của nhiệm vụ đó. Kết quả là, không giống như công việc trư ớc đó có ứng dụng giới hạn trong việc tái tạo câu, việc điều chỉnh tiền tố có thể đư ợc áp dụng cho các tác vụ NLG

Thế hệ có thể kiểm soát Thế hệ có thể kiểm soát nhằm mục đích điều khiển mô hình ngôn ngữ đư ợc đào tạo trư ớc để phù hợp với thuộc tính cấp độ câu (ví dụ: tình cảm tích cực hoặc thể thao). Sự kiểm soát như vậy có thể xảy ra trong thời gian đào tạo: Keskar et al. (2019) đào tạo trư ớc mô hình ngôn ngữ (CTRL) để điều chỉnh siêu dữ liệu như từ khóa hoặc URL. Việc kiểm soát cũng có thể diễn ra tại thời điểm giải mã, bằng cách giải mã theo trọng số (GeDi, Krause và cộng sự, 2020) hoặc cập nhật lặp lại các lần kích hoạt trư ớc đây (PPLM, Dathathri và cộng sự, 2020). Tuy nhiên, không có cách nào đơ n giản để áp dụng các kỹ thuật tạo có thể kiểm soát này để thực thi quyền kiểm soát chi tiết đối với nội dung đư ợc tạo , theo yêu cầu của các tác vụ như chuyển bảng thành văn bản và tóm tắt.

Điều chỉnh P*. Điều chỉnh tiền tố là một ví dụ của một lớp phư ơ ng pháp mới đã xuất hiện, mà chúng tôi gọi là p*-tuning (vì các trư ờng hợp nổi bật khác, p- tuning và nhắc điều chỉnh, cũng bắt đầu bằng p), tất cả đều dựa trên ý tư ởng tối ư u hóa tiền tố hoặc dấu nhắc liên tục. Đồng thời với công việc của chúng tôi, Qin và Eisner (2021) tìm hiểu sự kết hợp của các gợi ý điền vào chỗ trống nhẹ nhàng để khơ i gợi kiến thức từ các LM như BERT và BART. Hambardzumyan và cộng sự. (2021) tìm hiểu các phần nhúng dành riêng cho nhiệm vụ giúp điều chỉnh BERT để phân loại cảm xúc. Cả hai tác phẩm đều cho thấy rằng việc điều chỉnh các dấu nhắc mềm hoạt động tốt hơ n công việc trư ớc đó, giúp tối ư u hóa các dấu nhắc riêng biệt. P-tuning (Liu và cộng sự, 2021) cho thấy rằng việc cùng cập nhật các thông số nhúng nhanh chóng và LM cải thiện

Hiệu suất của GPT-2 đối với các tác vụ hiểu ngôn ngữ tự nhiên, ở cả cài đặt dữ liệu ít ảnh và dữ liệu đầy đủ. Trong công việc tiếp theo, Tinh chỉnh nhanh (Lester và cộng sự, 2021) đơ n giản hóa cách tiếp cận của chúng tôi và áp dụng nó cho T5 (Raffel và cộng sự, 2020), chứng minh rằng khoảng cách hiệu suất giữa tinh chỉnh và p*-tinh chỉnh biến mất khi kích thư ớc mô hình tăng lên.

3 Tuyên bố vấn đề

Hãy xem xét một tác vụ tạo có điều kiện trong đó đầu vào x là bối cảnh và đầu ra y là một chuỗi mã thông báo. Chúng tôi tập trung vào hai nhiệm vụ, đư ợc hiển thị trong Hình 2 (phải): Trong chuyển bảng thành văn bản, x tư ơ ng ứng với bảng dữ liệu tuyến tính hóa và y là mô tả văn bản; tóm lại, x là bài viết và y là tóm tắt.

3.1 LM tự hồi quy Giả sử

chúng ta có mô hình ngôn ngữ thần kinh tự hồi quy pφ (y | x) đư ợc tham số hóa bởi φ (ví dụ: GPT-2; Radford và cộng sự, 2019). Như thể hiện trong Hình 2 (trên cùng), đặt z = [x; y] là phần ghép của x và y; gọi Xidx biểu thị chuỗi các chỉ số tư ơ ng ứng với x và Yidx biểu thị tư ơ ng tự cho y.

Vectơ kích hoạt tại bư ớc thời gian i là hi R (1) d , hi = [h · · · ; h (n)] trong đó là sự kết hợp của ; (j) tất cả các lớp kích hoạt ở bư ớc thời gian này và h là vectơ kích hoạt của lớp thứ j tại bư ớc thời gian i.

LM thần kinh tự hồi quy tính hi là hàm của zi và các lần kích hoạt trong quá khứ trong ngữ cảnh bên trái của nó, như sau:

$$\text{chào} = \mathrm{LM}_\varphi(\text{zi , h<i}), \tag{1}$$

trong đó lớp cuối cùng của hi đư ợc sử dụng để tính toán phân phối cho mã thông báo tiếp theo: pφ (zi+1 | h≤i) = softmax(Wφ h (n) tôi) và Wφ là ma trận ánh xạ h để ghi lại từ vựng .

3.2 Kiến trúc bộ mã hóa-giải mã

Chúng ta cũng có thể sử dụng kiến trúc bộ mã hóa-giải mã (ví dụ: BART; Lewis và cộng sự, 2020) để mô hình hóa pφ (y | x), trong đó x đư ợc mã hóa bởi bộ mã hóa hai chiều và bộ giải mã dự đoán y tự động hồi quy (có điều kiện x đư ợc mã hóa và ngữ cảnh bên trái của nó). Chúng tôi sử dụng cùng một ký hiệu chỉ mục và kích hoạt, như trong Hình 2 (phía dư ới): mỗi hi cho i Xidx đư ợc tính toán bằng bộ mã hóa hai chiều; mỗi hi cho i Yidx đư ợc tính toán bằng bộ giải mã tự hồi quy sử dụng cùng phư ơ ng trình (1).

[1] Trong GPT-2, hi (N) bao gồm một cặp khóa-giá trị và di- mỗi khóa và giá trị là 1024.

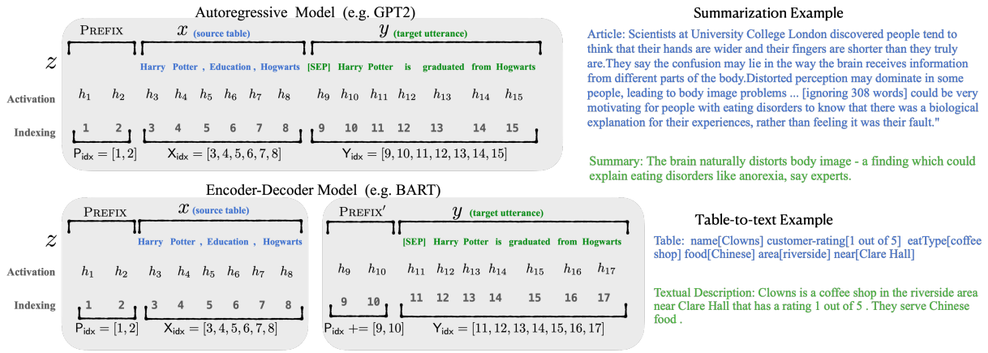


Figure 2: An annotated example of prefix-tuning using an autoregressive LM (top) and an encoder-decoder model (bottom). The prefix activations $\forall i \in \mathsf{P_{idx}}, h_i$ are drawn from a trainable matrix P_θ. The remaining activations are computed by the Transformer.

3.3 Fine-tuning

In the full fine-tuning framework, we initialize with the pretrained parameters ϕ. Here p_ϕ is a trainable language model distribution and we perform gradient updates on the following log-likelihood objective:

$$\max_{\phi} \log p_\phi(y \mid x) = \max_{\phi} \sum_{i \in \mathsf{Y_{idx}}} \log p_\phi(z_i \mid h_{<i}). \tag{2}$$

4 Prefix-Tuning

We propose prefix-tuning as an alternative to full fine-tuning for conditional generation tasks. We first provide intuition in §4.1 before defining our method formally in §4.2.

4.1 Intuition

Prompting has demonstrated that conditioning on a proper context can steer the LM without changing its parameters. For example, if we want the LM to generate a word (e.g., Obama), we can prepend its common collocations as context (e.g., Barack), and the LM will assign much higher probability to the desired word. Extending this intuition beyond generating a single word or sentence, we want to find a context that steers the LM to solve an NLG task. Intuitively, the context could influence the encoding of the task input x by guiding what to extract from x, and it could influence the generation of the task output y by steering the next token distribution. However, it's non-obvious whether such a context exists. Using natural language task instructions (e.g., "summarize the following table in one sentence") for the context might guide a human to solve the task, but this fails for moderately-sized pretrained LMs.[2] Optimizing over the discrete instructions might help, but discrete optimization is computationally challenging.

Instead of optimizing over discrete tokens, we can optimize the instruction as continuous word embeddings, whose effects will be propagated upward to all Transformer activation layers and rightward to subsequent tokens. This is strictly more expressive than a discrete prompt which is constrained to the embeddings of real words. Prefix-tuning goes one step further in increasing expressivity by optimizing the activations of all the layers, not just the embedding layer. As another benefit, prefix-tuning can directly modify representations deeper in the network, therefore, avoiding long computation paths across the depth of the network.

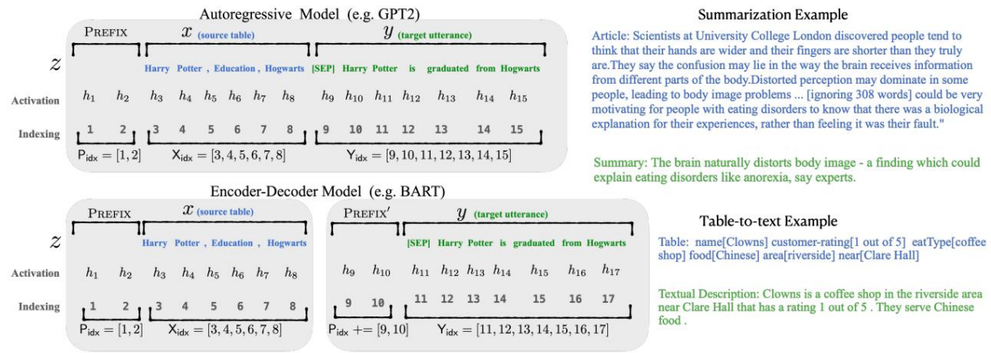
4.2 Method

Prefix-tuning prepends a prefix for an autoregressive LM to obtain $z = [\textsc{Prefix}; x; y]$, or prepends prefixes for both encoder and decoder to obtain $z = [\textsc{Prefix}; x; \textsc{Prefix}'; y]$, as shown in Figure 2. Here, $\mathsf{P_{idx}}$ denotes the sequence of prefix indices, and we use $|\mathsf{P_{idx}}|$ to denote the length of the prefix.

We follow the recurrence relation in equation (1), except that the activations of the prefix indices are free parameters, given by a matrix P_θ (parametrized by θ) of dimension $|\mathsf{P_{idx}}| \times \dim(h_i)$.

$$h_i = \begin{cases} P_\theta[i, :], & \text{if } i \in \mathsf{P_{idx}}, \\ \mathrm{LM}_\phi(z_i, h_{<i}), & \text{otherwise.} \end{cases} \tag{3}$$

[2] In our preliminary experiments, GPT-2 and BART fail in this setting; the only exception is GPT-3.



Hình 2: Một ví dụ có chú thích về điều chỉnh tiền tố bằng cách sử dụng LM tự hồi quy (trên cùng) và mô hình bộ mã hóa-giải mã (dư ới cùng). Các tiền tố kích hoạt i Pidx, hi đư ợc rút ra từ ma trận huấn luyện đư ợc Pθ . Các kích hoạt còn lại đư ợc tính toán bởi Transformer.

3.3 Tinh chỉnh Trong

khung tinh chỉnh đầy đủ, chúng tôi khởi tạo với các tham số đư ợc huấn luyện trư ớc. Ở đây pφ là một phân phối mô hình ngôn ngữ có thể huấn luyện đư ợc và chúng tôi thực hiện cập nhật độ dốc cho mục tiêu khả năng ghi nhật ký sau :

$$\max_{\varphi} \log p_\varphi(y \mid x) = \max_{\varphi} \sum_{i \in \mathsf{Y_{idx}}} \log p_\varphi(z_i \mid h_{<i}). \tag{2}$$

4 Điều chỉnh tiền tố

Chúng tôi đề xuất tinh chỉnh tiền tố như một giải pháp thay thế cho tinh chỉnh hoàn toàn cho các tác vụ tạo có điều kiện. Trư ớc tiên, chúng tôi cung cấp trực giác trong §4.1 trư ớc khi xác định chính thức phư ơng pháp của chúng tôi trong §4.2.

4.1 Trực giác

Việc nhắc nhở đã chứng minh rằng việc điều chỉnh trong một bối cảnh thích hợp có thể điều khiển LM mà không thay đổi các tham số của nó. Ví dụ: nếu chúng ta muốn LM tạo ra một từ (ví dụ: Obama), chúng ta có thể thêm các cụm từ phổ biến của nó làm ngữ cảnh (ví dụ: Barack) và LM sẽ chỉ định xác suất cao hơ n nhiều cho từ mong muốn. Mở rộng trực giác này ngoài việc tạo ra một từ hoặc câu duy nhất, chúng tôi muốn tìm một bối cảnh thúc đẩy LM giải quyết NLG

nhiệm vụ. Theo trực giác, ngữ cảnh có thể ảnh hư ởng đến việc mã hóa đầu vào nhiệm vụ x bằng cách hư ớng dẫn những gì cần trích xuất từ x và nó có thể ảnh hư ởng đến việc tạo đầu ra nhiệm vụ y bằng cách điều khiển việc phân phối mã thông báo tiếp theo . Tuy nhiên, không rõ bối cảnh như vậy có tồn tại hay không. Sử dụng các hư ớng dẫn nhiệm vụ bằng ngôn ngữ tự nhiên (ví dụ: "tóm tắt bảng sau bằng một câu") cho ngữ cảnh có thể hư ớng dẫn con người giải quyết đư ợc nhiệm vụ, như ng điều này không thành công đối với các LM đư ợc huấn luyện trư ớc có kích thư ớc vừa phải.2 Tối ư u hóa theo các hư ớng dẫn riêng biệt có thể hữu ích, như ng tối ư u hóa rời rạc là một thách thức về mặt tính toán.

Thay vì tối ư u hóa các mã thông báo rời rạc, chúng ta có thể tối ư u hóa hư ớng dẫn dư ới dạng các phần nhúng từ liên tục, hiệu ứng của chúng sẽ đư ợc truyền lên trên tất cả các lớp kích hoạt Máy biến áp và thẳng tới các mã thông báo tiếp theo. Điều này hoàn toàn mang tính biểu đạt hơ n so với một lời nhắc rời rạc vốn bị hạn chế trong việc nhúng các từ thực. Điều chỉnh tiền tố tiến thêm một bư ớc nữa trong việc tăng tính biểu cảm bằng cách tối ư u hóa kích hoạt của tất cả các lớp, không chỉ lớp nhúng. Là một lợi ích khác, việc điều chỉnh tiền tố có thể trực tiếp sửa đổi các biểu diễn sâu hơ n trong mạng, do đó tránh đư ợc các đư ờng tính toán dài xuyên suốt chiều sâu của mạng.

4.2 Phư ơ ng pháp

Điều chỉnh tiền tố thêm tiền tố cho LM tự hồi quy để thu đư ợc z = [PREFIX; x; y] hoặc thêm tiền tố cho cả bộ mã hóa và bộ giải mã để thu đư ợc z = [PREFIX; x; TIẾP ĐẦU NGỮ ; y], như thể hiện trong Hình 2.
Ở đây, Pidx biểu thị chuỗi các chỉ số tiền tố và chúng tôi sử dụng |Pidx| để biểu thị độ dài của tiền tố.

Chúng tôi tuân theo mối quan hệ truy hồi trong phư ơ ng trình (1), ngoại trừ việc kích hoạt các chỉ số tiền tố là các tham số tự do, đư ợc cho bởi ma trận Pθ (đư ợc tham số hóa bởi θ) có thứ nguyên |Pidx| × mờ(xin chào).

$$h_{\text{xin chào}} = \begin{cases} P\theta\,[i, :], \text{ nếu } i \quad Pidx, \\ \mathrm{LM}\varphi\,(zi\ ,\ h{<}i), \text{ ngư ợc lại.} \end{cases} \tag{3}$$

[2] Trong các thử nghiệm sơ bộ của chúng tôi, GPT-2 và BART không thành công trong cài đặt này; ngoại lệ duy nhất là GPT-3.

The training objective is the same as equation (2), but the set of trainable parameters changes: the language model parameters ϕ are fixed and the prefix parameters θ are the only trainable parameters.

Here, each h_i is a function of the trainable P_θ. When $i \in \mathsf{P}_{\mathsf{idx}}$, this is clear because h_i copies directly from P_θ. When $i \notin \mathsf{P}_{\mathsf{idx}}$, h_i still depends on P_θ, because the prefix activations are always in the left context and will therefore affect any activations to the right.

4.3 Parametrization of P_θ

Empirically, directly updating the P_θ parameters leads to unstable optimization and a slight drop in performance.[3] So we reparametrize the matrix $P_\theta[i,:] = \text{MLP}_\theta(P'_\theta[i,:])$ by a smaller matrix (P'_θ) composed with a large feedforward neural network (MLP_θ). Now, the trainable parameters include P'_θ and the parameters of MLP_θ. Note that P_θ and P'_θ has the same number of rows (i.e., the prefix length), but different number of columns.[4]

Once training is complete, these reparametrization parameters can be dropped, and only the prefix (P_θ) needs to be saved.

5 Experimental Setup

5.1 Datasets and Metrics

We evaluate on three standard neural generation datasets for the table-to-text task: E2E (Novikova et al., 2017), WebNLG (Gardent et al., 2017), and DART (Radev et al., 2020), as shown in Table 1. The datasets are ordered by increasing complexity and size. E2E only has 1 domain (i.e. restaurant reviews); WebNLG has 14 domains, and DART is open-domain, using open-domain tables from Wikipedia. For evaluation, we report the metrics using the official evaluation scripts (see details in Appendix A.1).

For the summarization task, we use the XSUM (Narayan et al., 2018) dataset, which is an abstractive summarization dataset on news articles. We report ROUGE-1, ROUGE-2 and ROUGE-L.

5.2 Methods

For table-to-text generation, we compare prefix-tuning with three other methods: full fine-tuning (FT-FULL), fine-tuning only the top 2 layers (FT-TOP2), and adapter-tuning (ADAPTER).[5] We also report the current state-of-the-art results on these datasets: On E2E, Shen et al. (2019) uses a pragmatically informed model without pretraining. On WebNLG, Kale (2020) fine-tunes T5-large. On DART, no official models trained on this dataset version are released.[6] For summarization, we compare against fine-tuning BART (Lewis et al., 2020).

	#examples	input length	output length
E2E	50K	28.5	27.8
WebNLG	22K	49.6	30.7
DART	82K	38.8	27.3
XSUM	225K	473.3	28.1

Table 1: Datasets statistics. The input and output length is the number of BPE tokens per example. For the three table-to-text datasets, the input length is the length of linearized tables (details in Appendix A.1).

5.3 Architectures and Hyperparameters

For table-to-text, we use $\text{GPT-2}_{\text{MEDIUM}}$ and $\text{GPT-2}_{\text{LARGE}}$. For summarization, we use $\text{BART}_{\text{LARGE}}$. Our implementation is based on the Hugging Face Transformers (Wolf et al., 2020).

At training time, we use the AdamW optimizer (Loshchilov and Hutter, 2019) and a linear learning rate scheduler, as suggested by the Hugging Face default setup. The hyperparameters we tune include the number of epochs, batch size, learning rate, and prefix length. Hyperparameter details are in the appendix. The default setting is 10 epochs, batch size 5, learning rate $5 \cdot 10^{-5}$ and prefix length 10. The table-to-text models are trained on TITAN Xp or GeForce GTX TITAN X machines. Prefix-tuning takes 0.2 hours per epoch to train on 22K examples, whereas fine-tuning takes around 0.3 hours per epoch. The summarization models are trained on Tesla V100 machines, taking 1.25 hours per epoch on the XSUM dataset. For *time efficiency*, prefix-tuning is around 30% faster than fine-tuning. For GPU *memory efficiency*, prefix-tuning with batchsize 1 takes 18% of the total GPU memory, whereas fine-tuning takes 50%.

At decoding time, for table-to-text, we use beam search with beam size 5. For summarization, we use beam size 6 and length normalization 0.8. Decoding takes 1.2 seconds per sentence (without

[3] We find in preliminary experiments that directly optimizing the prefix is very sensitive to initialization.

[4] P_θ has dimensions $|\mathsf{P}_{\mathsf{idx}}| \times \dim(h_i)$ while P_θ has dimensions $|\mathsf{P}_{\mathsf{idx}}| \times k$. We choose $k = 512$ for table-to-text and 800 for summarization. MLP_θ maps from k to $\dim(h_i)$.

[5] Same implementation as Lin et al. (2020).

[6] The official benchmark model is trained on v.1.0.0 while the release dataset is v1.1.1.

Mục tiêu huấn luyện giống như phư ơ ng trình (2), như ng tập hợp các tham số có thể huấn luyện đư ợc thay đổi: các tham số mô hình ngôn ngữ φ là cố định và các tham số tiền tố θ là các tham số duy nhất có thể huấn luyện đư ợc.

Ở đây, mỗi hi là một hàm của Pθ có thể huấn luyện đư ợc. Khi i Pidx, điều này rõ ràng vì hi sao chép trực tiếp từ Pθ . Khi i Pidx, hi vẫn phụ thuộc vào Pθ , vì các kích hoạt tiền tố luôn ở ngữ cảnh bên trái và do đó sẽ ảnh hư ởng đến bất kỳ kích hoạt nào ở bên phải.

	#examples	độ dài đầu vào	độ dài đầu ra
E2E	50K	28,5	27,8
WebNLG 22K		49,6	30,7
DART	82K	38,8	27,3
XSUM 225K		473,3	28.1

Bảng 1: Thống kê bộ dữ liệu. Độ dài đầu vào và đầu ra là số lư ợng mã thông báo BPE trên mỗi ví dụ. Đối với ba bộ dữ liệu dạng bảng thành văn bản, độ dài đầu vào là độ dài của bảng tuyến tính hóa (chi tiết tại Phụ lục A.1).

4.3 Tham số hóa của Pθ

Theo kinh nghiệm, việc cập nhật trực tiếp các tham số Pθ dẫn đến tối ư u hóa không ổn định và giảm hiệu suất một chút.3 Vì vậy, chúng tôi tham số hóa lại ma trận Pθ [i, :] = MLPθ (P θ [i, :]) bởi một ma trận nhỏ hơ n (Pθ) đư ợc tạo thành từ một mạng nơ -ron tiếp liệu lớn (MLPθ). Bây giờ, các tham số có thể huấn luyện đư ợc bao gồm Pθ và các tham số của MLPθ . Lư u ý rằng Pθ và P có cùng số hàng (nghĩa là tiền tố θ có độ dài), như ng có số cột khác nhau.4

Sau khi quá trình đào tạo hoàn tất, các tham số đo lại này có thể bị loại bỏ và chỉ cần lư u tiền tố (Pθ) .

5 Thiết lập thử nghiệm

5.1 Bộ dữ liệu và số liệu

Chúng tôi đánh giá dựa trên ba bộ dữ liệu tạo thần kinh tiêu chuẩn cho tác vụ chuyển bảng thành văn bản: E2E (Novikova và cộng sự, 2017), WebNLG (Gardent và cộng sự, 2017) và DART (Radev và cộng sự, 2020), như đư ợc hiển thị trong Bảng 1. Các tập dữ liệu đư ợc sắp xếp theo độ phức tạp và kích thư ớc tăng dần. E2E chỉ có 1 tên miền (tức là đánh giá nhà hàng); WebNLG có 14 miền và DART là miền mở, sử dụng các bảng miền mở từ Wikipedia. Để đánh giá, chúng tôi báo cáo số liệu bằng cách sử dụng tập lệnh đánh giá chính thức (xem chi tiết trong Phụ lục A.1).

Đối với nhiệm vụ tóm tắt, chúng tôi sử dụng bộ dữ liệu XSUM (Narayan và cộng sự, 2018) , đây là bộ dữ liệu tóm tắt trừu tư ợng về các bài báo. Chúng tôi báo cáo ROUGE-1, ROUGE-2 và ROUGE-L.

5.2 Phư ơ ng pháp

Để tạo bảng thành văn bản, chúng tôi so sánh việc tinh chỉnh tiền tố với ba phư ơ ng pháp khác: tinh chỉnh đầy đủ (FT-FULL), chỉ tinh chỉnh 2 lớp trên cùng (FT- TOP2) và điều chỉnh bộ điều hợp (ADAPTER).5 Chúng tôi cũng báo cáo các kết quả tiên tiến hiện tại trên các bộ dữ liệu này: Trên E2E, Shen et al. (2019) sử dụng mô hình có thông tin thực tế mà không cần đào tạo trư ớc. Trên WebNLG, Kale (2020) tinh chỉnh T5-large. Trên DART, không có mô hình chính thức nào đư ợc đào tạo trên phiên bản tập dữ liệu này đư ợc phát hành.6 Để tóm tắt, chúng tôi so sánh với BART tinh chỉnh (Lewis và cộng sự, 2020).

5.3 Kiến trúc và siêu tham số

Đối với tính năng chuyển bảng thành văn bản, chúng tôi sử dụng $\text{GPT-2}_{\text{MEDIUM}}$ và $\text{GPT-2}_{\text{LARGE}}$. Để tóm tắt, chúng tôi sử dụng $\text{BART}_{\text{LARGE}}$. Việc triển khai của chúng tôi dựa trên Máy biến áp ôm mặt (Wolf và cộng sự, 2020).

Tại thời điểm đào tạo, chúng tôi sử dụng trình tối ư u hóa AdamW (Loshchilov và Hutter, 2019) và bộ lập lịch tốc độ học tập tuyến tính, như đư ợc đề xuất bởi thiết lập mặc định Ôm mặt. Các siêu tham số mà chúng tôi điều chỉnh bao gồm số lư ợng kỷ nguyên, kích thư ớc lô, tốc độ học tập và độ dài tiền tố. Chi tiết siêu tham số có trong phần phụ lục. Cài đặt mặc định là 10 kỷ nguyên, kích thư ớc lô 5, tốc độ học tập 5·10 5 và độ dài tiền tố 10. Các mô hình table-to-text đư ợc huấn luyện trên TITAN Máy Xp hoặc GeForce GTX TITAN X. Việc tinh chỉnh tiền tố mất 0,2 giờ mỗi kỷ nguyên để huấn luyện trên các mẫu 22K, trong khi việc tinh chỉnh mất khoảng 0,3 giờ mỗi kỷ nguyên. Các mô hình tóm tắt đư ợc đào tạo trên máy Tesla V100, mất 1,25 giờ mỗi kỷ nguyên trên tập dữ liệu XSUM. Để tiết kiệm thời gian, việc tinh chỉnh tiền tố nhanh hơ n khoảng 30% so với tinh chỉnh. Để đạt hiệu quả bộ nhớ GPU, việc tinh chỉnh tiền tố với kích thư ớc lô 1 chiếm 18% tổng bộ nhớ GPU, trong khi việc tinh chỉnh chiếm 50%.

Tại thời điểm giải mã, đối với bảng thành văn bản, chúng tôi sử dụng tìm kiếm chùm tia với kích thư ớc chùm 5. Để tóm tắt, chúng tôi sử dụng kích thư ớc chùm 6 và chuẩn hóa độ dài 0,8. Việc giải mã mất 1,2 giây mỗi câu (không có

3Chúng tôi nhận thấy trong các thử nghiệm sơ bộ rằng việc tối ư u hóa trực tiếp tiền tố rất nhạy cảm với việc khởi tạo.

4Pθ có các kích thư ớc |Pidx| × dim(hi) trong khi Pθ có thứ nguyên |Pidx| × k. Chúng tôi chọn k = 512 cho bảng thành văn bản và 800 cho tóm tắt. Ánh xạ MLPθ từ k tới dim(hi).

5Thực hiện tư ơ ng tự như Lin et al. (2020).

6Mô hình chuẩn chính thức đư ợc huấn luyện trên v.1.0.0 trong khi tập dữ liệu phát hành là v1.1.1.

batching) for table-to-text, and 2.6 seconds per batch (using a batch size of 10) for summarization.

6 Main Results

6.1 Table-to-text Generation

We find that by updating only 0.1% task-specific parameters,[7] prefix-tuning is effective in table-to-text generation, outperforming other lightweight baselines (ADAPTER and FT-TOP2) even by updating 30x fewer parameters and achieving a comparable performance with (full) fine-tuning. This trend holds for all datasets: E2E, WebNLG,[8] and DART.

If we match the number of parameters for prefix-tuning and adapter-tuning to be 0.1%, Table 2 shows that prefix-tuning is significantly better than ADAPTER (0.1%), attaining 4.1 BLEU improvement per dataset on average. Even when we compare with fine-tuning (100%) and adapter-tuning (3.0%), which update significantly more parameters than prefix-tuning, prefix-tuning still achieves results comparable or better than those two systems. This demonstrates that prefix-tuning is more Pareto efficient than adapter-tuning, significantly reducing parameters while improving generation quality.

Additionally, attaining good performance on DART suggests that prefix-tuning can generalize to tables with diverse domains and a large number of relations. We will delve deeper into extrapolation performance (i.e., generalization to unseen categories or topics) in §6.4.

In summary, prefix-tuning is an effective and space-efficient method to adapt GPT-2 to table-to-text generation. It also maintains the performance gains when scaling up to GPT-2_{LARGE}, suggesting it has the potential to scale to even larger models with a similar architecture, like GPT-3.

6.2 Summarization

As shown in Table 3, with 2% parameters, prefix-tuning obtains slightly lower performance than fine-tuning (36.05 vs. 37.25 in ROUGE-L). With only 0.1% parameters, prefix-tuning underperforms full fine-tuning (35.05 vs. 37.25). There are several differences between XSUM and the three table-to-text datasets which could account for why prefix-tuning has comparative advantage in table-to-text: (1) XSUM contains 4x more examples than the three table-to-text datasets on average; (2) the input articles are 17x longer than the linearized table input of table-to-text datasets on average; (3) summarization is more complex than table-to-text because it requires selecting key contents from an article.

6.3 Low-data Setting

Based on the results from table-to-text (§6.1) and summarization (§6.2), we observe that prefix-tuning has a comparative advantage when the number of training examples is smaller. To explore the low-data setting more systematically, we subsample the full dataset (E2E for table-to-text and XSUM for summarization) to obtain small datasets of size $\{50, 100, 200, 500\}$. For each size, we sample 5 different datasets and average over 2 training random seeds. Thus, we average over 10 models for each low-data setting.[9]

Figure 3 (right) shows that prefix-tuning outperforms fine-tuning in low-data regimes by 2.9 BLEU on average, in addition to requiring much fewer parameters, but the gap narrows as the dataset size increases.

Qualitatively, Figure 3 (left) shows 8 examples generated by both prefix-tuning and fine-tuning models trained on different data levels. While both methods tend to undergenerate (missing table contents) in low data regimes, prefix-tuning tends to be more faithful than fine-tuning. For example, fine-tuning (100, 200)[10] falsely claims a low customer rating while the true rating is average, whereas prefix-tuning (100, 200) generates a description that is faithful to the table.

6.4 Extrapolation

We now investigate extrapolation performance to unseen topics for both table-to-text and summarization. In order to construct an extrapolation setting, we split the existing datasets so that training and test cover different topics. For table-to-text, the WebNLG dataset is labeled with table topics. There are 9 categories that appear in training and dev, denoted as SEEN and 5 categories that only appear at test time, denoted as UNSEEN. So we evaluate extrapolation by training on the SEEN categories and testing on the UNSEEN categories. For summarization, we construct two extrapolation data splits:

[7] 250K for E2E, 250K for WebNLG, and 500K for DART versus 345M GPT-2 parameters.

[8] The S,U,A columns in WebNLG represents SEEN, UNSEEN, and ALL respectively; SEEN categories appear at training time; UNSEEN categories only appears at test time; and ALL is the combination of the two.

[9] We also sample a dev split (with dev size = 30% $\times$ training size) for each training set. We use the dev split to choose hyperparameters and perform early stopping.

[10] The number in the parenthesis refers to the training size.

batching) cho chuyển bảng thành văn bản và 2,6 giây cho mỗi lô (sử dụng cỡ lô là 10) để tóm tắt.

6 kết quả chính

6.1 Tạo bảng thành văn bản

Chúng tôi thấy rằng bằng cách chỉ cập nhật 0,1% các tham số dành riêng cho nhiệm vụ ,7 việc điều chỉnh tiền tố có hiệu quả trong việc chuyển bảng thành văn bản thế hệ mới, vượt trội so với các dòng cơ bản nhẹ khác (ADAPTER và FT-TOP2) ngay cả khi cập nhật Ít tham số hơn 30 lần và đạt được kết quả tương đương hiệu suất với (đầy đủ) tinh chỉnh. Xu hướng này giữ cho tất cả các tập dữ liệu: E2E, WebNLG,8 và DART.

Nếu chúng ta khớp số tham số cho điều chỉnh tiền tố và điều chỉnh bộ điều hợp là 0,1%, Bảng 2 cho thấy việc điều chỉnh tiền tố tốt hơn đáng kể so với ADAPTER (0,1%), đạt mức cải thiện trung bình 4,1 BLEU trên mỗi tập dữ liệu. Ngay cả khi chúng tôi so sánh với tinh chỉnh (100%) và điều chỉnh bộ chuyển đổi (3,0%), cập nhật nhiều tham số hơn đáng kể so với điều chỉnh tiền tố, điều chỉnh tiền tố vẫn đạt được kết quả tương đương hoặc tốt hơn hai hệ thống đó. Điều này chứng tỏ rằng việc điều chỉnh tiền tố thiên về Pareto hơn hiệu quả hơn so với điều chỉnh bộ chuyển đổi, giảm đáng kể các thông số đồng thời cải thiện chất lượng thế hệ.

Ngoài ra, đạt được hiệu suất tốt trên DART gợi ý rằng việc điều chỉnh tiền tố có thể khái quát hóa tới các bảng có miền đa dạng và số lượng lớn của các mối quan hệ. Chúng ta sẽ nghiên cứu sâu hơn về hiệu suất ngoại suy (tức là khái quát hóa cho những danh mục hoặc chủ đề) trong §6.4.

Tóm lại, điều chỉnh tiền tố là một phương pháp hiệu quả và phương pháp tiết kiệm không gian để điều chỉnh GPT-2 cho việc tạo bảng thành văn bản. Nó cũng duy trì hiệu suất đạt được khi mở rộng quy mô lên GPT-2LARGE, gợi ý nó có khả năng mở rộng quy mô sang các mô hình lớn hơn với kiến trúc tương tự, như GPT-3.

6.2 Tóm tắt

Như được hiển thị trong Bảng 3, với 2% tham số, việc tinh chỉnh tiền tố đạt được hiệu suất thấp hơn một chút so với tinh chỉnh (36,05 so với 37,25 trong ROUGE-L). Chỉ với Thông số 0,1%, điều chỉnh tiền tố hoạt động kém hơn khi đầy đủ tinh chỉnh (35,05 so với 37,25). Có một số sự khác biệt giữa XSUM và ba phương pháp bàn-to-các tập dữ liệu văn bản có thể giải thích tại sao việc điều chỉnh tiền tố lại có lợi thế so sánh trong việc chuyển bảng thành văn bản: (1) XSUM chứa nhiều ví dụ gấp 4 lần so với trung bình ba bộ dữ liệu từ bảng thành văn bản; (2) đầu vào trung bình các bài viết dài hơn 17 lần so với đầu vào bảng tuyến tính hóa của các bộ dữ liệu dạng bảng thành văn bản; (3) việc tóm tắt phức tạp hơn việc chuyển bảng thành văn bản vì nó yêu cầu chọn nội dung chính từ một bài viết.

6.3 Cài đặt dữ liệu thấp

Dựa trên kết quả từ bảng thành văn bản (§6.1) và tóm tắt (§6.2), chúng tôi nhận thấy rằng việc điều chỉnh tiền tố có lợi thế so sánh khi số lượng ví dụ huấn luyện nhỏ hơn. Khám phá cài đặt dữ liệu thấp có hệ thống hơn, chúng tôi lấy mẫu phụ toàn bộ tập dữ liệu (E2E cho chuyển bảng thành văn bản và XSUM để tóm tắt) để thu được các tập dữ liệu nhỏ có kích thước {50, 100, 200, 500}. Đối với mỗi kích thước, chúng tôi lấy mẫu 5 bộ dữ liệu khác nhau và lấy trung bình trên 2 lần huấn luyện. hạt ngẫu nhiên. Vì vậy, chúng tôi tính trung bình trên 10 mô hình cho mỗi cài đặt dữ liệu thấp.9

Hình 3 (bên phải) cho thấy việc tinh chỉnh tiền tố hoạt động tốt hơn tinh chỉnh trong chế độ dữ liệu thấp tới 2,9 BLEU trung bình, ngoài việc yêu cầu ít tham số hơn , nhưng khoảng cách sẽ thu hẹp khi kích thước tập dữ liệu tăng.

Về mặt định tính, Hình 3 (trái) hiển thị 8 ví dụ được tạo ra bởi cả điều chỉnh tiền tố và tinh chỉnh mô hình được đào tạo trên các cấp độ dữ liệu khác nhau. Trong khi cả hai các phương pháp có xu hướng tạo thiếu (thiếu nội dung bảng) trong chế độ dữ liệu thấp, việc điều chỉnh tiền tố có xu hướng trung thực hơn là tinh chỉnh. Ví dụ: tinh chỉnh (100, 200)10 tuyên bố sai về khách hàng thấp xếp hạng trong khi xếp hạng thực sự là trung bình, trong khi điều chỉnh tiền tố (100, 200) tạo mô tả đó là trung thành với bảng.

6.4 Phép ngoại suy

Bây giờ chúng tôi điều tra hiệu suất ngoại suy để các chủ đề chưa được nhìn thấy cho cả dạng bảng thành văn bản và bản tóm tắt. Để xây dựng một thiết lập ngoại suy, chúng tôi chia các tập dữ liệu hiện có để đào tạo và kiểm tra bao gồm các chủ đề khác nhau. Đối với tính năng chuyển bảng thành văn bản, Tập dữ liệu WebNLG được gắn nhãn với các chủ đề bảng. Ở đó là 9 danh mục xuất hiện trong quá trình đào tạo và phát triển, ký hiệu là SEEN và 5 danh mục chỉ xuất hiện ở thời gian kiểm tra, được ký hiệu là UNSEEN. Vì vậy, chúng tôi đánh giá phép ngoại suy bằng cách đào tạo về các danh mục SEEN và thử nghiệm trên các danh mục UNSEEN. Để tóm tắt, chúng tôi xây dựng hai phần tách dữ liệu ngoại suy:

[7] 250K cho E2E, 250K cho WebNLG và 500K cho DART so với thông số 345M GPT-2.

[8] Các cột S,U,A trong WebNLG lần lượt thể hiện SEEN, UN-SEEN và ALL; XEM danh mục xuất hiện tại thời gian huấn luyện; Các hạng mục KHÔNG ĐƯỢC XEM chỉ xuất hiện vào thời điểm thử nghiệm; và ALL là sự kết hợp của cả hai.

[9] Chúng tôi cũng lấy mẫu phân chia nhà phát triển (với kích thước nhà phát triển = 30% × kích thước đào tạo) cho mỗi tập huấn luyện. Chúng tôi sử dụng phân chia nhà phát triển để chọn siêu tham số và thực hiện dừng sớm.

[10] Số trong ngoặc thể hiện quy mô đào tạo.

	E2E					WebNLG									DART					
	BLEU	NIST	MET	R-L	CIDEr	BLEU			MET			TER ↓			BLEU	MET	TER ↓	Mover	BERT	BLEURT
						S	U	A	S	U	A	S	U	A						
									GPT-2$_{\text{MEDIUM}}$											
FT-FULL	68.8	8.71	46.1	71.1	2.43	**64.7**	26.7	45.7	**0.46**	0.30	0.38	**0.33**	0.78	0.54	46.2	**0.39**	**0.46**	**0.50**	**0.94**	**0.39**
FT-TOP2	68.1	8.59	46.0	70.8	2.41	53.6	18.9	36.0	0.38	0.23	0.31	0.49	0.99	0.72	41.0	0.34	0.56	0.43	0.93	0.21
ADAPTER(3%)	68.9	8.71	46.1	71.3	**2.47**	60.5	**47.9**	54.8	0.43	**0.38**	**0.41**	0.35	**0.46**	**0.39**	45.2	0.38	**0.46**	**0.50**	**0.94**	**0.39**
ADAPTER(0.1%)	66.3	8.41	45.0	69.8	2.40	54.5	45.1	50.2	0.39	0.36	0.38	0.40	0.46	0.43	42.4	0.36	0.48	0.47	**0.94**	0.33
PREFIX(0.1%)	**70.3**	**8.82**	**46.3**	**72.1**	2.46	62.9	45.3	**55.0**	0.44	0.37	**0.41**	0.35	0.51	0.42	**46.4**	0.38	**0.46**	**0.50**	**0.94**	**0.39**
									GPT-2$_{\text{LARGE}}$											
FT-FULL	68.5	8.78	46.0	69.9	2.45	**65.3**	43.1	55.5	**0.46**	0.38	**0.42**	**0.33**	0.53	0.42	**47.0**	**0.39**	0.46	**0.51**	**0.94**	**0.40**
Prefix	**70.3**	**8.85**	**46.2**	**71.7**	**2.47**	63.4	**47.7**	**56.3**	0.45	**0.39**	**0.42**	0.34	**0.48**	**0.40**	46.7	**0.39**	**0.45**	**0.51**	**0.94**	**0.40**
SOTA	68.6	8.70	45.3	70.8	2.37	63.9	52.8	57.1	0.46	0.41	0.44	-	-	-	-	-	-	-	-	-

Table 2: Metrics (higher is better, except for TER) for table-to-text generation on E2E (left), WebNLG (middle) and DART (right). With only 0.1% parameters, Prefix-tuning outperforms other lightweight baselines and achieves a comparable performance with fine-tuning. The best score is boldfaced for both GPT-2$_{\text{MEDIUM}}$ and GPT-2$_{\text{LARGE}}$.

Source	name : The Eagle | type : coffee shop | food : Chinese | price : cheap | customer rating : average | area : riverside | family friendly : no | near : Burger King
Prefix (50)	The Eagle is a cheap Chinese coffee shop located near Burger King.
Prefix (100)	The Eagle is a cheap coffee shop located in the riverside near Burger King. It has average customer ratings.
Prefix (200)	The Eagle is a cheap Chinese coffee shop located in the riverside area near Burger King. It has average customer ratings.
Prefix (500)	The Eagle is a coffee shop that serves Chinese food. It is located in the riverside area near Burger King. It has an average customer rating and is not family friendly.
FT (50)	The Eagle coffee shop is located in the riverside area near Burger King.
FT (100)	The Eagle is a cheap coffee shop near Burger King in the riverside area. It has a low customer rating and is not family friendly.
FT (200)	The Eagle is a cheap Chinese coffee shop with a low customer rating. It is located near Burger King in the riverside area.
FT (500)	The Eagle is a cheap Chinese coffee shop with average customer ratings. It is located in the riverside area near Burger King.

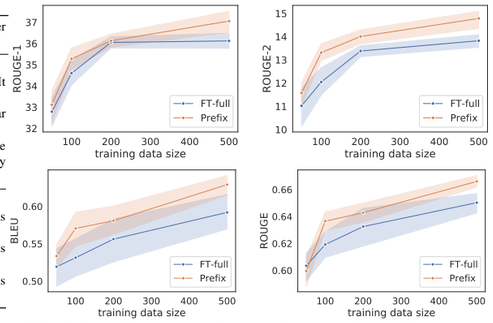


Figure 3: (Left) qualitative examples in lowdata settings. (Right) prefix-tuning (orange) outperforms fine-tuning (blue) in low-data regimes in addition to requiring many fewer parameters. The top two plots correspond to summarization, measured by ROUGE-1 and ROUGE-2. The bottom two plots correspond to table-to-text, measured by BLEU and ROUGE-L. The x-axis is the training size and the y-axis is the evaluation metric (higher is better).

	R-1 ↑	R-2 ↑	R-L ↑
FT-FULL(Lewis et al., 2020)	45.14	22.27	37.25
PREFIX(2%)	43.80	20.93	36.05
PREFIX(0.1%)	42.92	20.03	35.05

Table 3: Performance of methods on the XSUM summarization dataset. Prefix-tuning slightly underperforms fine-tuning in the full-data regime.

	news-to-sports			within-news		
	R-1 ↑	R-2 ↑	R-L ↑	R-1 ↑	R-2 ↑	R-L ↑
FT-FULL	38.15	15.51	30.26	39.20	16.35	31.15
PREFIX	39.23	16.74	31.51	39.41	16.87	31.47

Table 4: Extrapolation performance on XSUM. Prefix-tuning outperforms fine-tuning on both news-to-sports and within-news splits.

In `news-to-sports`, we train on news articles and test on sports articles. In `within-news`, we train on {world, UK, business} news and test on the remaining news categories (e.g., health, tech).

On both table-to-text and summarization, prefix-tuning extrapolates better than fine-tuning under all metrics, as shown in Table 4 and the 'U' columns of Table 2 (middle).

We also find that adapter-tuning achieves good extrapolation performance, comparable with prefix-tuning, as shown in Table 2. This shared trend suggests that preserving LM parameters indeed has a positive impact on extrapolation. However, how prefix-tuning improves extrapolation is an open question and we will discuss this further in §8.

7 Intrinsic Evaluation

We compare different variants of prefix-tuning to study the impact of various design decisions. §7.1 studies the impact of the prefix length. §7.2 studies tuning only the embedding layer, which is more akin to tuning a discrete prompt. §7.3 compares prefixing and infixing, which inserts trainable activations between x and y. §7.4 studies the impact of various prefix initialization strategies. §7.5 further studies the data efficiency of prefix-tuning.

7.1 Prefix Length

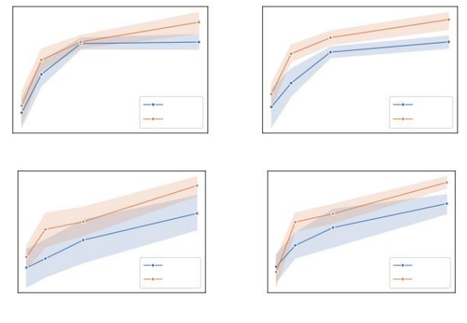
A longer prefix means more trainable parameters, and therefore more expressive power.[11] Figure 4 shows that performance increases as the prefix

[11]Empirically, longer prefixes have a negligible impact on training and inference speed per batch, because attention computation over the entire prefix is parallelized on GPUs.

	E2E					WebNLG									DART					
	BLEU	NIST	MET	RL	CIDEr	BLEU			GẤP			TER			BLEU	MET	TER	Mover	BERT	BLEURT
						SUA	SUA	SUA												
									GPT-2TRUNG BÌNH											
FT-FULL	68,8	8,71	46,1	71,1	2,43	64,7	26,7	45,7	0,46	0,30	0,38	0,33	0,78	0,54	46,2	0,39	0,46	0,50	0,94	0,39
FT-TOP2	0,93	BỘ CHUYỂN ĐỔI	68,1	8,59	46,0	70,8	2,41	53,6	18,9	36,0	0,38	0,23	0,31	0,49	0,99	0,72	41,0	0,34	0,56	BỘ CHUYỂN ĐỔI (0,1%)	66,3	0,43	45,0	0,21
ADAPTER(3%)	69,8	2,40	54,5	45,1	50,2	0,39	0,36	0,38	0,40	0,46	0,43	42,4	0,36	0,48	0,94	TIỀN TỐ(0,1%)	70,3	8,82	46,3	72,1	2,46	62,9	45,3	55,0	0,44	0,37	0,41	0,35	0,51	0,42	46,4	0,39
ADAPTER(0.1%)	0,38	0,46	0,50	0,94																									0,47			0,33
PREFIX(0.1%)																																0,39
									GPT-2LỚN											
FT-FULL	68,5	8,78	46,0	69,9	2,45	65,3	43,1	55,5	0,46	0,38	0,42	0,33	0,53	0,42	47,0	0,39	0,46	70,3	8,85	46,2		0,51	0,94	0,40
Tiếp đầu ngữ	71,7	2,47	63,4	47,7	56.	3	0,45	0,39	0,42	0,34	0,48	0,40	46,7	0,39	0,45							0,51	0,94	0,40
SOTA	68,6	8,70	45,3	70,8	2,37	63,9	52,8	57,1	0,46	0,41	0,44	-			-	-	-	-	-	-

Bảng 2: Số liệu (càng cao càng tốt, ngoại trừ TER) để tạo bảng thành văn bản trên E2E (trái), WebNLG (giữa) và DART (phải). Chỉ với 0,1% tham số, việc điều chỉnh tiền tố vượt trội hơn các đường cơ sở nhẹ khác và đạt được một hiệu suất tương đương với tinh chỉnh. Điểm tốt nhất được in đậm cho cả GPT-2MEDIUM và GPT-2LARGE.

Nguồn	tên : Đại Bàng | loại hình : quán cà phê | món ăn : Trung Quốc | giá : rẻ | khách hàng đánh giá: trung bình | khu vực : ven sông | thân thiện với gia đình: không | gần: Burger King
Tiền tố (50)	The Eagle là quán cà phê Trung Quốc giá rẻ nằm gần Burger King.
Tiền tố (100)	The Eagle là quán cà phê giá rẻ nằm ở ven sông gần Burger King. Nó có xếp hạng trung bình của khách hàng.
Tiền tố (200)	The Eagle là một quán cà phê Trung Quốc giá rẻ nằm ở khu vực ven sông gần Burger King. Nó có xếp hạng khách hàng trung bình.
Tiền tố (500)	The Eagle là quán cà phê phục vụ đồ ăn Trung Quốc. Nó nằm ở ven sông khu vực gần Burger King. Nó có xếp hạng khách hàng trung bình và không phải là gia đình thân thiện.
FT (50)	Quán cà phê The Eagle nằm ở khu vực ven sông gần Burger King.
FT (100)	The Eagle là quán cà phê giá rẻ gần Burger King ở khu vực ven sông. Nó có đánh giá của khách hàng thấp và không thân thiện với gia đình.
FT (200)	The Eagle là một quán cà phê Trung Quốc giá rẻ được khách hàng đánh giá thấp. Nó là nằm gần Burger King ở khu vực ven sông.
FT (500)	The Eagle là quán cà phê Trung Quốc giá rẻ được khách hàng đánh giá ở mức trung bình. Nó là nằm ở khu vực ven sông gần Burger King.



Hình 3: (Trái) các ví dụ định tính trong cài đặt dữ liệu thấp. (Phải) tinh chỉnh tiền tố (màu cam) tốt hơn tinh chỉnh (màu xanh) ở chế độ dữ liệu thấp ngoài việc yêu cầu ít tham số hơn. Hai ô trên cùng tương ứng với tổng hợp, được đo bằng ROUGE-1 và ROUGE-2. Hai ô dưới cùng tương ứng với bảng-to-text, được đo bởi BLEU và ROUGE-L. Trục x là kích thước đào tạo và trục y là thước đo đánh giá (càng cao càng tốt).

	R-1	R-2	RL
FT-FULL(Lewis và cộng sự, 2020)	45,14	22,27	37,25
TIỀN TỐ(2%)	43,80	20,93	36,05
TIỀN TỐ(0,1%)	42,92	20,03	35,05

Bảng 3: Hiệu suất của các phương pháp trên tập dữ liệu tổng hợp XSUM. Việc tinh chỉnh tiền tố kém hơn một chút so với việc tinh chỉnh trong chế độ dữ liệu đầy đủ.

	tin tức thể thao R-1		tin tức nội bộ		
	R-2	RL	R-1	R-2	RL
FT-FULL	38,15	15,51	30,26	39,20	16,35	31,15
TIỀN TỐ	39,23	16,74	31,51	39,41	16,87	31,47

Bảng 4: Hiệu suất ngoại suy trên XSUM. Tinh chỉnh tiền tố hoạt động tốt hơn tinh chỉnh trên cả tin tức và thể thao và sự chia rẽ trong tin tức.

Trong lĩnh vực tin tức đến thể thao, chúng tôi tập luyện dựa trên các bài báo và thử nghiệm trên các bài báo thể thao. Trong tin tức nội bộ, chúng tôi đào tạo về tin tức và bài kiểm tra trên {world, UK, business} các danh mục tin tức còn lại (ví dụ: sức khỏe, công nghệ).

Trên cả tính năng chuyển bảng thành văn bản và tóm tắt, việc điều chỉnh tiền tố sẽ ngoại suy tốt hơn so với việc tinh chỉnh trong tất cả các trường hợp. số liệu, như được hiển thị trong Bảng 4 và cột 'U' của Bảng 2 (giữa).

Chúng tôi cũng nhận thấy rằng việc điều chỉnh bộ chuyển đổi đạt được kết quả tốt hiệu suất ngoại suy, có thể so sánh với tiền tố-điều chỉnh, như thể hiện trong Bảng 2. Xu hướng chung này gợi ý rằng việc bảo toàn các tham số LM thực sự có tác động tích cực đến phép ngoại suy. Tuy nhiên, làm thế nào điều chỉnh tiền tố cải thiện phép ngoại suy là một giải pháp mở câu hỏi và chúng ta sẽ thảo luận thêm về vấn đề này trong §8.

7 Đánh giá nội tại

Chúng tôi so sánh các biến thể khác nhau của việc điều chỉnh tiền tố với nghiên cứu tác động của các quyết định thiết kế khác nhau. §7.1 nghiên cứu tác động của độ dài tiền tố. nghiên cứu §7.2 chỉ điều chỉnh lớp nhúng, còn hơn thế nữa giống như điều chỉnh một dấu nhắc riêng biệt. §7.3 so sánh tiền tố và infixing, chèn các kích hoạt có thể huấn luyện giữa x và y. §7.4 nghiên cứu tác động của các chiến lược khởi tạo tiền tố khác nhau. §7.5 thêm nghiên cứu hiệu quả dữ liệu của việc điều chỉnh tiền tố.

7.1 Độ dài tiền tố

Tiền tố dài hơn có nghĩa là các tham số có thể huấn luyện được nhiều hơn, và do đó có sức biểu cảm cao hơn.11 Hình 4 cho thấy hiệu suất tăng lên khi tiền tố

11Theo kinh nghiệm, tiền tố dài hơn có tác động không đáng kể đến tốc độ đào tạo và suy luận mỗi đợt vì tính toán chú ý trên toàn bộ tiền tố được thực hiện song song trên GPU.

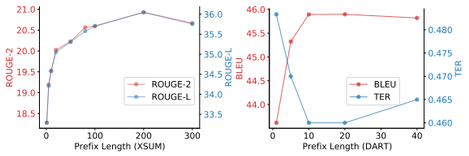


Figure 4: Prefix length vs. performance on summerization (left) and table-to-text (right). Performance increases as the prefix length increases up to a threshold (200 for summarization and 10 for table-to-text) and then a slight performance drop occurs. Each plot reports two metrics (on two vertical axes).

	E2E				
	BLEU	NIST	MET	ROUGE	CIDEr
PREFIX	70.3	8.82	46.3	72.1	2.46
Embedding-only: EMB-{PrefixLength}					
EMB-1	48.1	3.33	32.1	60.2	1.10
EMB-10	62.2	6.70	38.6	66.4	1.75
EMB-20	61.9	7.11	39.3	65.6	1.85
Infix-tuning: INFIX-{PrefixLength}					
INFIX-1	67.9	8.63	45.8	69.4	2.42
INFIX-10	67.2	8.48	45.8	69.9	2.40
INFIX-20	66.7	8.47	45.8	70.0	2.42

Table 5: Intrinsic evaluation of Embedding-only (§7.2) and Infixing (§7.3). Both Embedding-only ablation and Infix-tuning underperforms full prefix-tuning.

length increases up to a threshold (200 for summarization, 10 for table-to-text) and then a slight performance drop occurs. Prefixes longer than the threshold lead to lower training loss, but slightly worse test performance, suggesting that they tend to overfit the training data.

7.2 Full vs Embedding-only

Recall in §4.1, we discussed optimizing the continuous embeddings of the "virtual tokens." We instantiate that idea and call it *embedding-only*. The word embeddings are free parameters, and the remaining activation layers are computed by the Transformer. Table 5 (top) shows that the performance drops significantly, suggesting that tuning only the embedding layer is not sufficiently expressive.

Embedding-only upper bounds the performance of discrete prompt optimization (Shin et al., 2020), because discrete prompt restricts the embedding layer to exactly match the embedding of a real word. Consequently, we have this chain of increasing expressive power: discrete prompting < embedding-only < prefix-tuning.

7.3 Prefix-tuning vs Infix-tuning

We also investigate how the trainable activations' position in the sequence affects performance. In

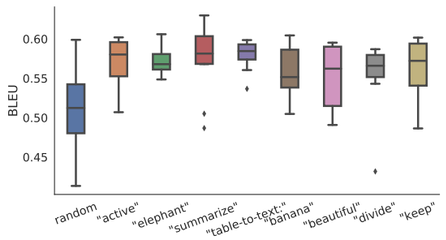


Figure 5: Initializing the prefix with activations of real words significantly outperforms random initialization, in low-data settings.

prefix-tuning, we place them at the beginning $[\text{PREFIX}; x; y]$. We can also place the trainable activations between x and y (i.e. $[x; \text{INFIX}; y]$) and call this infix-tuning. Table 5 (bottom) shows that infix-tuning slightly underperforms prefix-tuning. We believe this is because prefix-tuning can affect the activations of x and y whereas infix-tuning can only influence the activations of y.

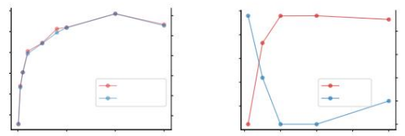
7.4 Initialization

We find that how the prefix is initialized has a large impact in low-data settings. Random initialization leads to low performance with high variance. Initializing the prefix with activations of real words significantly improves generation, as shown in Figure 5. In particular, initializing with task relevant words such as "summarization" and "table-to-text" obtains slightly better performance than task irrelevant words such as "elephant" and "divide", but using real words is still better than random. Moreover, in full data settings, the initialization trick has no impact, and random initialization leads to equally good performance.

Since we initialize the prefix with activations of real words computed by the LM, this initialization strategy is concordant with prefix-tuning's philosophy, which preserves the pretrained LM as much as possible.

7.5 Data Efficiency

We also investigate the data efficiency of prefix-tuning (without initialization trick, a.k.a random initialization) and full fine-tuning by comparing their performance on 5 different data scales of the E2E task (10%, 20%, 40%, 60%, and 80%). Figure 6 shows that prefix-tuning has better performance than fine-tuning when using more than 20% of the data. For data scale of 10%, prefix-tuning with random initialization yields comparable or slightly lower performance than full fine-tuning,



Hình 4: Độ dài tiền tố so với hiệu suất khi hè hóa (trái) và chuyển bảng thành văn bản (phải). Hiệu suất tăng khi độ dài tiền tố tăng đến ngư ỡng (200 cho tóm tắt và 10 cho bảng thành văn bản) và sau đó xảy ra sự sụt giảm hiệu suất nhẹ. Mỗi biểu đồ tái hiện hai số liệu (trên hai trục dọc).

	E2E				
	BLEU	NIST	MET	ROUGE	CIDEr
TIẾP ĐẦU NGỮ	70,3	8,82	46,3	72,1	2,46
Chỉ nhúng: EMB-{PrefixLength}					
EMB-1	48,1	3,33	32,1	60,2	1,10
EMB-10	62,2	6,70	38,6	66,4	1,75
EMB-20	61,9	7,11	39,3	65,6	1,85
Điều chỉnh infix: INFIX-{PrefixLength}					
INFIX-1	67,9	8,63	45,8	69,4	2,42
INFIX-10	67,2	8,48	45,8	69,9	2,40
INFIX-20	66,7	8,47	45,8	70,0	2,42

Bảng 5: Đánh giá nội tại của chỉ nhúng (§7.2) và Infixing (§7.3). Cả hai phư ơ ng pháp cắt bỏ chỉ nhúng và Điều chỉnh infix hoạt động kém hiệu quả hơ n so với điều chỉnh tiền tố đầy đủ.

độ dài tăng lên đến một ngư ỡng (200 cho tóm tắt, 10 cho bảng thành văn bản) và sau đó tăng nhẹ giảm hiệu suất xảy ra. Tiền tố dài hơ n ngư ỡng dẫn đến mất huấn luyện thấp hơ n, như ng hơ i hiệu suất kiểm tra kém hơ n, cho thấy rằng họ có xu hư ớng để phù hợp quá mức với dữ liệu huấn luyện.

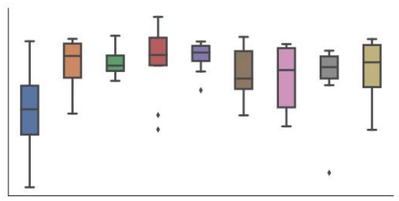
7.2 Đầy đủ và chỉ nhúng

Nhớ lại trong §4.1, chúng tôi đã thảo luận về việc tối ư u hóa việc nhúng liên tục "mã thông báo ảo". Chúng tôi hiện thực hóa ý tư ởng đó và gọi nó là chỉ nhúng. từ phần nhúng là các tham số miễn phí và phần còn lại các lớp kích hoạt đư ợc tính toán bởi Transformer. Bảng 5 (trên cùng) cho thấy hiệu suất giảm đáng kể, cho thấy rằng chỉ điều chỉnh lớp nhúng là không đủ biểu cảm.

Chỉ nhúng giới hạn trên của hiệu suất về tối ư u hóa nhanh chóng rời rạc (Shin và cộng sự, 2020), vì dấu nhắc riêng biệt hạn chế việc nhúng lớp để khớp chính xác với việc nhúng một từ thực. Do đó, chúng ta có chuỗi sức mạnh diễn đạt ngày càng tăng này: nhắc nhở rời rạc < chỉ nhúng < điều chỉnh tiền tố.

7.3 Điều chỉnh tiền tố và điều chỉnh Infix

Chúng tôi cũng điều tra cách các kích hoạt có thể huấn luyện đư ợc' vị trí trong chuỗi ảnh hư ởng đến hiệu suất. TRONG



Hình 5: Khởi tạo tiền tố với các kích hoạt của real từ tốt hơ n đáng kể so với khởi tạo ngẫu nhiên, trong cài đặt dữ liệu thấp.

điều chỉnh tiền tố, chúng tôi đặt chúng ở đầu [TIẾP ĐẦU NGỮ; x; y]. Chúng ta cũng có thể đặt phần có thể huấn luyện đư ợc kích hoạt giữa x và y (tức là [x; INFIX; y]) và gọi đây là điều chỉnh infix. Bảng 5 (phía dư ới) cho thấy điều chỉnh tiền tố hơ i kém hơ n việc điều chỉnh tiền tố. Chúng tôi tin rằng điều này là do việc điều chỉnh tiền tố có thể ảnh hư ởng sự kích hoạt của x và y trong khi điều chỉnh infix có thể chỉ ảnh hư ởng đến sự kích hoạt của y.

7.4 Khởi tạo

Chúng tôi thấy rằng cách khởi tạo tiền tố có một tác động lớn trong cài đặt dữ liệu thấp. Ngẫu nhiên khởi tạo dẫn đến hiệu suất thấp với tốc độ cao phư ơ ng sai. Khởi tạo tiền tố với kích hoạt của những từ thực sự cải thiện đáng kể việc tạo ra, như thể hiện trong Hình 5. Cụ thể, khởi tạo với những từ liên quan đến nhiệm vụ như "tóm tắt" và "chuyển bảng thành văn bản" đạt đư ợc hiệu suất tốt hơ n một chút hơ n những từ không liên quan đến nhiệm vụ như "con voi" và "chia" như ng dùng lời thật vẫn tốt hơ n hơ n ngẫu nhiên. Hơ n nữa, trong cài đặt dữ liệu đầy đủ, thủ thuật khởi tạo không có tác động và ngẫu nhiên khởi tạo dẫn đến hiệu suất tốt như nhau.

Vì chúng tôi khởi tạo tiền tố bằng cách kích hoạt các từ thực đư ợc LM tính toán, việc khởi tạo này chiến lư ợc phù hợp với triết lý điều chỉnh tiền tố , giúp duy trì LM đã đư ợc huấn luyện trư ớc càng nhiều càng tốt. càng tốt.

7.5 Hiệu quả dữ liệu

Chúng tôi cũng điều tra hiệu quả dữ liệu của việc điều chỉnh tiền tố (không có thủ thuật khởi tạo, hay còn gọi là ngẫu nhiên khởi tạo) và tinh chỉnh đầy đủ bằng cách so sánh hiệu suất của họ trên 5 thang dữ liệu khác nhau của Nhiệm vụ E2E (10%, 20%, 40%, 60% và 80%). Hình 6 cho thấy tinh chỉnh tiền tố có hiệu suất tốt hơ n tinh chỉnh khi sử dụng trên 20% của dữ liệu. Đối với quy mô dữ liệu 10%, điều chỉnh tiền tố với việc khởi tạo ngẫu nhiên mang lại kết quả tư ơ ng đư ơ ng hoặc hiệu suất thấp hơ n một chút so với tinh chỉnh đầy đủ,

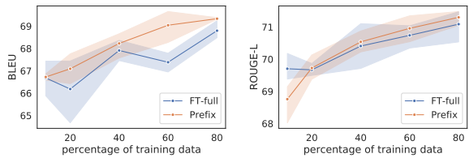


Figure 6: Data efficiency curves: percentage of training set vs. performance on table-to-text (E2E). Prefix-tuning (without the initialization trick) is more data-efficient than fine-tuning when using more than 20% of the data.

necessitating the initialization trick (§6.3) to improve the performance in this low-data regime.

8 Discussion

We will discuss several favorable properties of prefix-tuning and some open problems.

Personalization. As we note in §1, prefix-tuning is advantageous when there are a large number of tasks that needs to be trained independently. One practical setting is user privacy (Shokri and Shmatikov, 2015; McMahan et al., 2016). In order to preserve user privacy, each user's data needs to be separated and a personalized model needs to be trained independently for each user. Consequently, each user can be regarded as an independent task. If there are millions of users, prefix-tuning can scale to this setting and maintain modularity, enabling flexible addition or deletion of users by adding or deleting their prefixes without cross-contamination.

Batching across users. Under the same personalization setting, prefix-tuning allows batching different users' queries even though they are backed by different prefixes. When multiple users query a cloud GPU device with their inputs, it is computationally efficient to put these users in the same batch. Prefix-tuning keeps the shared LM intact; consequently, batching requires a simple step of prepending the personalized prefix to user input, and all the remaining computation is unchanged. In contrast, we can't batch across different users in adapter-tuning, which has personalized adapters between shared Transformer layers.

This batching benefit could also help create efficient ensembles of multiple prefixes trained on the same task (Lester et al., 2021).

Inductive bias of prefix-tuning. Recall that fine-tuning updates all pretrained parameters, whereas prefix-tuning and adapter-tuning preserve them. Since the language models are pretrained on general purpose corpora, preserving the LM parameters might help generalization to domains unseen during training. In concordance with this intuition, we observe that both prefix-tuning and adapter-tuning have significant performance gain in extrapolation settings (§6.4); however, how these methods improve extrapolation is an open question.

While prefix-tuning and adapter-tuning both freeze the pretrained parameters, they tune different sets of parameters to affect the activation layers of the Transformer. Recall that prefix-tuning keeps the LM intact and uses the prefix and the pretrained attention blocks to affect the subsequent activations; adapter-tuning inserts trainable modules between LM layers, which directly add residual vectors to the activations. Moreover, we observe that prefix-tuning requires vastly fewer parameters compared to adapter-tuning while maintaining comparable performance. We think this gain in parameter efficiency is because prefix-tuning keeps the pretrained LM intact as much as possible, and therefore exploits the LM more than adapter-tuning.

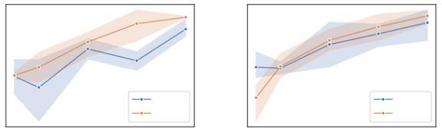
Recent work by Aghajanyan et al. (2020) uses intrinsic dimension to show that there exists a low-dimensional reparameterization that is as effective for fine-tuning as the full parametrization. This explains why good accuracy on downstream tasks can be obtained by updating only a small number of parameters. Our work echoes this finding by showing that good generation performance can also be attained by updating a very small prefix. However, prefix-tuning is not just about the size of trainable parameters, but more importantly, which subset of parameters to modify. Therefore, it would be interesting future work to explore other light-weight fine-tuning methods that achieve an even better accuracy-size tradeoff.

Acknowledgments

We thank the members of p-lambda group as well as anonymous reviewers for valuable feedback. We gratefully acknowledge the support of a PECASE award. XLL is supported by a Stanford Graduate Fellowship.

Reproducibility

Our code is available at https://github.com/XiangLi1999/PrefixTuning.
Experiments and data are available at https://worksheets.codalab.org/worksheets/0x16e0c8e7ab1f4b22aaccddc8b586541f.



Hình 6: Đư ờng cong hiệu quả dữ liệu: tỷ lệ phần trăm của tập huấn luyện so với hiệu suất trên bảng thành văn bản (E2E). Điều chỉnh tiền tố (không có thủ thuật khởi tạo) sẽ tiết kiệm dữ liệu hơ n so với tinh chỉnh khi sử dụng hơ n 20% của dữ liệu.

đòi hỏi thủ thuật khởi tạo (§6.3) để cải thiện hiệu suất trong chế độ dữ liệu thấp này.

8 Thảo luận

Chúng ta sẽ thảo luận về một số tính chất có lợi của điều chỉnh tiền tố và một số vấn đề mở.

Cá nhân hóa. Như chúng tôi đã lư u ý trong §1, điều chỉnh tiền tố thuận lợi khi có số lư ợng lớn nhiệm vụ cần đư ợc đào tạo độc lập.
Một cài đặt thực tế là quyền riêng tư của ngư ời dùng (Shokri và Shmatikov, 2015; McMahan và cộng sự, 2016). theo thứ tự để bảo vệ quyền riêng tư của ngư ời dùng, dữ liệu của mỗi ngư ời dùng cần phải đư ợc tách ra và một mô hình cá nhân hóa cần phải đư ợc đư ợc đào tạo độc lập cho mỗi ngư ời dùng. Do đó, mỗi ngư ời dùng có thể đư ợc coi là một nhiệm vụ độc lập. Nếu như có hàng triệu ngư ời dùng, việc điều chỉnh tiền tố có thể mở rộng quy mô vào cài đặt này và duy trì tính mô-đun, cho phép thêm hoặc xóa linh hoạt ngư ời dùng bằng cách thêm hoặc xóa tiền tố của chúng mà không bị nhiễm chéo.

Hàng loạt trên ngư ời dùng. Trong cùng cài đặt cá nhân hóa, việc điều chỉnh tiền tố cho phép phân nhóm các truy vấn của ngư ời dùng khác nhau ngay cả khi chúng đư ợc hỗ trợ. bởi các tiền tố khác nhau. Khi có nhiều ngư ời dùng truy vấn một thiết bị GPU đám mây với đầu vào của họ, sẽ hiệu quả về mặt tính toán khi đặt những ngư ời dùng này vào cùng một lô hàng. Điều chỉnh tiền tố giữ nguyên LM đư ợc chia sẻ; do đó, việc trộn khối đòi hỏi một bư ớc đơ n giản là thêm tiền tố đư ợc cá nhân hóa vào đầu vào của ngư ời dùng, và tất cả các tính toán còn lại không thay đổi.
Ngư ợc lại, chúng tôi không thể phân nhóm nhiều ngư ời dùng khác nhau trong điều chỉnh bộ điều hợp, có bộ điều hợp đư ợc cá nhân hóa giữa các lớp Transformer đư ợc chia sẻ.

Lợi ích gộp này cũng có thể giúp tạo ra các tập hợp hiệu quả của nhiều tiền tố đư ợc huấn luyện trên nhiệm vụ tư ơ ng tự (Lester và cộng sự, 2021).

Độ lệch quy nạp của điều chỉnh tiền tố. Hãy nhớ lại rằng việc tinh chỉnh cập nhật tất cả các tham số đư ợc huấn luyện trư ớc, trong khi điều chỉnh tiền tố và điều chỉnh bộ điều hợp sẽ bảo tồn chúng.

Do các mô hình ngôn ngữ đư ợc huấn luyện trư ớc trên ngữ liệu có mục đích chung nên việc bảo toàn các tham số LM có thể giúp khái quát hóa các miền không nhìn thấy đư ợc Trong quá trình huấn luyện. Phù hợp với trực giác này, chúng tôi quan sát thấy rằng cả điều chỉnh tiền tố và điều chỉnh bộ điều hợp đều đạt đư ợc hiệu suất đáng kể trong cài đặt ngoại suy (§6.4); tuy nhiên, làm thế nào những phư ơ ng pháp này cải thiện khả năng ngoại suy là một câu hỏi mở.

Trong khi cả điều chỉnh tiền tố và điều chỉnh bộ điều hợp đóng băng các tham số đư ợc huấn luyện trư ớc, chúng điều chỉnh khác nhau tập hợp các tham số ảnh hư ởng đến các lớp kích hoạt của Máy biến áp. Hãy nhớ lại rằng việc điều chỉnh tiền tố sẽ giữ nguyên LM còn nguyên vẹn và sử dụng tiền tố cũng như các khối chú ý đã đư ợc huấn luyện trư ớc để tác động đến các lần kích hoạt tiếp theo; điều chỉnh bộ chuyển đổi chèn các mô-đun có thể đào tạo đư ợc giữa Các lớp LM, bổ sung trực tiếp các vectơ dư vào các kích hoạt. Hơ n nữa, chúng tôi nhận thấy rằng việc điều chỉnh tiền tố yêu cầu ít tham số hơ n rất nhiều so với để điều chỉnh bộ chuyển đổi trong khi vẫn duy trì khả năng so sánh hiệu suất. Chúng tôi cho rằng mức tăng hiệu quả tham số này là do việc điều chỉnh tiền tố giữ cho dữ liệu đư ợc huấn luyện trư ớc LM nguyên vẹn nhất có thể và do đó khai thác LM nhiều hơ n là điều chỉnh bộ chuyển đổi.

Công việc gần đây của Aghajanyan et al. (2020) sử dụng chiều hư ớng nội tại để chỉ ra rằng tồn tại một sự tái tham số hóa theo chiều thấp có hiệu quả để tinh chỉnh như tham số hóa đầy đủ. Cái này giải thích tại sao độ chính xác cao ở các tác vụ tiếp theo có thể thu đư ợc bằng cách chỉ cập nhật một số lư ợng nhỏ các tham số. Công việc của chúng tôi lặp lại phát hiện này bằng cách cho thấy rằng hiệu suất phát điện tốt có thể cũng có thể đạt đư ợc bằng cách cập nhật một tiền tố rất nhỏ. Tuy nhiên, việc điều chỉnh tiền tố không chỉ liên quan đến kích thư ớc của các tham số có thể huấn luyện đư ợc, như ng quan trọng hơ n, tập hợp con các tham số cần sửa đổi. Vì vậy, nó sẽ là công việc thú vị trong tư ơ ng lai để khám phá các phư ơ ng pháp tinh chỉnh gọn nhẹ khác nhằm đạt đư ợc hiệu suất đồng đều sự cân bằng kích thư ớc chính xác tốt hơ n.

Sự nhìn nhận

Chúng tôi cũng xin cảm ơ n các thành viên của nhóm p-lambda với tư cách là ngư ời đánh giá ẩn danh để nhận đư ợc phản hồi có giá trị. Chúng tôi chân thành cảm ơ n sự hỗ trợ của PECASE phần thư ởng. XLL đư ợc hỗ trợ bởi một sinh viên tốt nghiệp Stanford Tình bằng hữu.

Khả năng tái lập

Mã của chúng tôi có sẵn tại https://github.com/XiangLi1999/PrefixTuning.
Các thử nghiệm và dữ liệu có sẵn tại https://worksheets.codalab.org/worksheets/0x16e0c8e7ab1f4b22aaccddc8b586541f.

References

Armen Aghajanyan, Luke Zettlemoyer, and Sonal Gupta. 2020. Intrinsic dimensionality explains the effectiveness of language model fine-tuning.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, Trento, Italy. Association for Computational Linguistics.

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. 2020. Language models are few-shot learners.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. The WebNLG challenge: Generating text from RDF data. In *Proceedings of the 10th International Conference on Natural Language Generation*, pages 124–133, Santiago de Compostela, Spain. Association for Computational Linguistics.

Karen Hambardzumyan, Hrant Khachatrian, and Jonathan May. 2021. WARP: word-level adversarial reprogramming. *CoRR*, abs/2101.00121.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan, and Sylvain Gelly. 2019. Parameter-efficient transfer learning for NLP. In *Proceedings of the 36th International Conference on Machine Learning*, volume 97 of *Proceedings of Machine Learning Research*, pages 2790–2799, Long Beach, California, USA. PMLR.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know? *Transactions of the Association for Computational Linguistics*, 8:423–438.

Mihir Kale. 2020. Text-to-text pre-training for data-to-text tasks.

N. Keskar, B. McCann, L. R. Varshney, Caiming Xiong, and R. Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *ArXiv*, abs/1909.05858.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2020. GeDi: Generative Discriminator Guided Sequence Generation. *arXiv preprint arXiv:2009.06367*.

Alon Lavie and Abhaya Agarwal. 2007. Meteor: An automatic metric for mt evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, StatMT '07, pages 228–231, Stroudsburg, PA, USA. Association for Computational Linguistics.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, and Pascale Fung. 2020. Exploring versatile generative language model via parameter-efficient transfer learning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 441–459, Online. Association for Computational Linguistics.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too. *arXiv preprint arXiv:2103.10385*.

Yang Liu and Mirella Lapata. 2019. Text summarization with pretrained encoders. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3730–3740, Hong Kong, China. Association for Computational Linguistics.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis, and Luke Zettlemoyer. 2020. Multilingual denoising pre-training for neural machine translation.

Ngư ời giới thiệu

Armen Aghajanyan, Luke Zettlemoyer và Sonal Gupta. 2020. Chiều hư ớng nội tại giải thích hiệu quả của việc tinh chỉnh mô hình ngôn ngữ.

Anja Belz và Ehud Reiter. 2006. So sánh đánh giá tự động và con ngư ời đối với các hệ thống NLG. TRONG Hội nghị lần thứ 11 của Hiệp hội Châu Âu Hiệp hội Ngôn ngữ học tính toán, Trento, Nư ớc Ý. Hiệp hội ngôn ngữ học tính toán.

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 Cờ vua, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever và Dario Amodei. Năm 2020. Các mô hình ngôn ngữ đư ợc học trong vài lần ờ.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hùng, Eric Frank, Piero Molino, Jason Yosinski và Rosanne Liu. 2020. Mô hình ngôn ngữ cắm và chạy: Một cách tiếp cận đơ n giản để tạo văn bản có kiểm soát. Trong Hội nghị quốc tế về đại diện học tập.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, và Kristina Toutanova. 2019. BERT: Đào tạo trư ớc cho biến đổi hai chiều sâu sắc để hiểu ngôn ngữ. Trong Kỷ yếu Hội nghị 2019 của Hiệp hội Bắc Mỹ cho Ngôn ngữ học tính toán: Ngôn ngữ của con ngư ời Công nghệ, Tập 1 (Bài báo dài và ngắn), trang 4171–4186, Minneapolis, Minnesota. Hiệp hội Ngôn ngữ học tính toán.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, và Laura Perez-Beltrachini. 2017. WebNLG thách thức: Tạo văn bản từ dữ liệu RDF. Trong Biên bản Hội nghị quốc tế lần thứ 10 về Tạo ngôn ngữ tự nhiên, trang 124–133, San-tiago de Compostela, Tây Ban Nha. Hiệp hội Ngôn ngữ học tính toán.

Karen Hambardzumyan, Hrant Khachatrian, và Jonathan May. 2021. WARP: lập trình lại đối thủ ở cấp độ từ. CoRR, abs/2101.00121.

Neil Houlsby, Andrei Giurgiu, Stanislaw Jastrzebski, Bruna Morrone, Quentin De Laroussilhe, Andrea Gesmundo, Mona Attariyan và Sylvain Gelly. 2019. Học chuyển giao tham số hiệu quả cho NLP. Trong Kỷ yếu của Hội nghị Quốc tế lần thứ 36 về Học máy, tập 97 của Kỷ yếu của Nghiên cứu Học máy, trang 2790–2799, Bãi Dài, California, Mỹ. PMLR.

Zhengbao Jiang, Frank F. Xu, Jun Araki và Graham Neubig. 2020. Làm sao biết đư ợc ngôn ngữ nào ngư ời mẫu biết không? Giao dịch của Hiệp hội Ngôn ngữ học tính toán, 8:423–438.

Mihir Kale. 2020. Đào tạo trư ớc về chuyển văn bản thành văn bản cho các tác vụ chuyển dữ liệu thành văn bản.

N. Keskar, B. McCann, LR Varshney, Caiming Xiong, và R. Socher. 2019. Ctrl: Mô hình ngôn ngữ chuyển đổi có điều kiện để tạo có thể kiểm soát. ArXiv, abs/1909.05858.

Ben Krause, Akhilesh Deepak Gotmare, Bryan Mc-Cann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher và Nazneen Fatema Rajani. 2020. GeDi: Tạo trình tự có hư ớng dẫn phân biệt đối xử sáng tạo. bản in trư ớc arXiv arXiv:2009.06367.

Alon Lavie và Abhaya Agarwal. 2007. Sao băng: An thư ớc đo tự động để đánh giá mt ở mức cao mối tư ơ ng quan với sự phán xét của con ngư ời. Trong Kỷ yếu Hội thảo lần thứ hai về Máy thống kê Bản dịch, StatMT '07, trang 228–231, Strouds-burg, PA, USA. Hiệp hội tính toán Lin-guistics.

Brian Lester, Rami Al-Rfou và Noah Constant. 2021. Sức mạnh của thang đo cho lời nhắc hiệu quả về tham số điều chỉnh.

Mike Lewis, Yinhan Liu, Naman Goyal, Mar-jan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov và Luke Zettlemoyer. 2020. BART: Khử nhiễu trư ớc khi đào tạo theo trình tự để tạo, dịch, và sự hiểu biết. Trong Kỷ yếu của Hội nghị thư ờng niên lần thứ 58 của Hiệp hội tính toán Ngôn ngữ học, trang 7871–7880, Trực tuyến. Sự kết hợp cho Ngôn ngữ học tính toán.

Chin-Yew Lin. 2004. ROUGE: Gói đánh giá tự động các bản tóm tắt. Trong Text Summariza-tion Branches Out, trang 74–81, Barcelona, Tây Ban Nha. Hiệp hội ngôn ngữ học tính toán.

Triệu Giang Lâm, Andrea Madotto và Pascale Fung. 2020. Khám phá ngôn ngữ sáng tạo đa năng mô hình thông qua học chuyển giao tham số hiệu quả. TRONG Những phát hiện của Hiệp hội Ngôn ngữ học tính toán: EMNLP 2020, trang 441–459, Trực tuyến. Hiệp hội ngôn ngữ học tính toán.

Tiểu Lục, Diên Nam Chính, Chính Tiêu Du, Minh Đỉnh, Yujie Qian, Zhilin Yang và Jie Tang. 2021. Gpt cũng hiểu. bản in trư ớc arXiv arXiv:2103.10385.

Dư ơ ng Lư u và Mirella Lapata. 2019. Tóm tắt văn bản bằng bộ mã hóa đư ợc đào tạo trư ớc. Trong Kỷ yếu của Hội nghị về các phư ơ ng pháp thực nghiệm trong xử lý ngôn ngữ tự nhiên năm 2019 và Hội nghị quốc tế lần thứ 9 Hội nghị chung về xử lý ngôn ngữ tự nhiên (EMNLP-IJCNLP), trang 3730–3740, Hồng Kông, Trung Quốc. Hiệp hội ngôn ngữ học tính toán.

Yinhan Liu, Jiatao Gu, Naman Goyal, Xian Li, Sergey Edunov, Marjan Ghazvininejad, Mike Lewis và Luke Zettlemoyer. 2020. Khử nhiễu đa ngôn ngữ đào tạo trư ớc cho dịch máy thần kinh.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

H. Brendan McMahan, Eider Moore, Daniel Ramage, and Blaise Agüera y Arcas. 2016. Federated learning of deep networks using model averaging. *Proceedings of the 20 th International Conference on Artificial Intelligence and Statistics (AISTATS) 2017*, abs/1602.05629.

Shashi Narayan, Shay B. Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium.

Jekaterina Novikova, Ondrej Dusek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. *CoRR*, abs/1706.09254.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 311–318, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Rücklé, Kyunghyun Cho, and Iryna Gurevych. 2020. Adapterfusion: Non-destructive task composition for transfer learning.

Guanghui Qin and Jason Eisner. 2021. Learning how to ask: Querying LMs with mixtures of soft prompts. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, Mexico City.

Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. 2020. Dart: Open-domain structured data record to text generation.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Evani Radiya-Dixit and Xin Wang. 2020. How fine can fine-tuning be? learning efficient language models. In *Proceedings of the Twenty Third International Conference on Artificial Intelligence and Statistics*, volume 108 of *Proceedings of Machine Learning Research*, pages 2435–2443, Online. PMLR.

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

Sylvestre-Alvise Rebuffi, Hakan Bilen, and Andrea Vedaldi. 2017. Learning multiple visual domains with residual adapters. In *Advances in Neural Information Processing Systems*, volume 30, pages 506–516. Curran Associates, Inc.

Timo Schick and Hinrich Schütze. 2020. Exploiting cloze questions for few shot text classification and natural language inference.

Thibault Sellam, Dipanjan Das, and Ankur Parikh. 2020. BLEURT: Learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7881–7892, Online. Association for Computational Linguistics.

Sheng Shen, Daniel Fried, Jacob Andreas, and Dan Klein. 2019. Pragmatically informative text generation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4060–4067, Minneapolis, Minnesota. Association for Computational Linguistics.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace, and Sameer Singh. 2020. Autoprompt: Eliciting knowledge from language models with automatically generated prompts.

Reza Shokri and Vitaly Shmatikov. 2015. Privacy-preserving deep learning. In *Proceedings of the 22nd ACM SIGSAC Conference on Computer and Communications Security*, CCS '15, page 1310–1321, New York, NY, USA. Association for Computing Machinery.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and Ralph Weischedel. 2006. A study of translation error rate with targeted human annotation. In *In Proceedings of the Association for Machine Transaltion in the Americas (AMTA 2006*.

Asa Cooper Stickland, Xian Li, and Marjan Ghazvininejad. 2020. Recipes for adapting pre-trained monolingual and multilingual models to machine translation.

Nishant Subramani, Samuel R. Bowman, and Kyunghyun Cho. 2020. Can unconditional language models recover arbitrary sentences?

Ramakrishna Vedantam, C. Lawrence Zitnick, and Devi Parikh. 2015. Cider: Consensus-based image description evaluation. In *CVPR*, pages 4566–4575. IEEE Computer Society.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Man-dar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer và Veselin Stoyanov. 2019. Roberta: Một phư ơ ng pháp đào tạo trư ớc BERT đư ợc tối ư u hóa mạnh mẽ . CoRR, abs/1907.11692.

Ilya Loshchilov và Frank Hutter. 2019. Tách rời chính quy hóa giảm cân. Trong Hội nghị quốc tế về đại diện học tập.

H. Brendan McMahan, Eider Moore, Daniel Ramage, và Blaise Aguera y Arcas. 2016. Học liên kết các mạng sâu bằng cách sử dụng mô hình trung bình. Biên bản Hội nghị quốc tế lần thứ 20 về Trí tuệ nhân tạo và thống kê (AISTATS) 2017, abs/1602.05629.

Shashi Narayan, Shay B. Cohen và Mirella Lapata. 2018. Đừng cho tôi chi tiết, chỉ tóm tắt thôi! Mạng lư ới thần kinh tích chập nhận biết chủ đề để tóm tắt cực kỳ . Trong Kỷ yếu năm 2018 Hội thảo về các phư ơ ng pháp thực nghiệm trong xử lý ngôn ngữ tự nhiên, Brussels, Bỉ.

Jekaterina Novikova, Ondrej Dusek và Verena Rieser. 2017. Bộ dữ liệu E2E: Những thách thức mới cho thế hệ end-to-end. CoRR, abs/1706.09254.

Kishore Papineni, Salim Roukos, Todd Ward và Wei-Jing Zhu. 2002. Bleu: Phư ơ ng pháp đánh giá tự động dịch máy. Trong Kỷ yếu của Hội nghị thư ờng niên lần thứ 40 về Hiệp hội Ngôn ngữ học tính toán, ACL '02, trang 311–318, Stroudsburg, PA, Hoa Kỳ . Hiệp hội ngôn ngữ học tính toán.

Jonas Pfeiffer, Aishwarya Kamath, Andreas Ruckl ¨ e,` Cho Kyunghyun và Iryna Gurevych. 2020. Adaptorfusion: Thành phần nhiệm vụ không phá hủy để chuyển giao học tập.

Quang Huy Tần và Jason Eisner. 2021. Học cách để hỏi: Truy vấn LM với sự kết hợp của các lời nhắc nhẹ nhàng. Trong Kỷ yếu Hội nghị phía Bắc năm 2021 Hiệp hội Ngôn ngữ học tính toán Hoa Kỳ : Công nghệ ngôn ngữ của con ngư ời (NAACL-HLT), Thành phố Mexico.

Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Ra-jani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, và Richard Socher. 2020. Dart: Bản ghi dữ liệu có cấu trúc miền mở để tạo văn bản.

A. Radford, Jeffrey Wu, R. Child, David Luan, Dario Amodei và Ilya Sutskever. 2019. Mô hình ngôn ngữ là những ngư ời học đa nhiệm không đư ợc giám sát.

Evani Radiya-Dixit và Xin Wang. 2020. Làm sao có thể ổn đư ợc tinh chỉnh đư ợc? học các mô hình ngôn ngữ hiệu quả. Trong Kỷ yếu của Quốc tế thứ 23 Hội thảo về Trí tuệ nhân tạo và Thống kê, tập 108 của Kỷ yếu nghiên cứu học máy, trang 2435–2443, Trực tuyến. PMLR.

Colin Raffel,Noam Shazeer,Adam Roberts,Kather-ine Lee,Sharan Narang,Michael Matena,Yanqi Chu, Wei Li và Peter J. Liu. 2020. Khám phá các giới hạn của việc học chuyển giao với một công cụ chuyển đổi văn bản thành văn bản thống nhất. Tạp chí Nghiên cứu Học máy, 21(140):1–67.

Sylvestre-Alvise Rebuffi, Hakan Bilen và Andrea Vedaldi. 2017. Học nhiều lĩnh vực trực quan với các bộ điều hợp còn lại. Trong Những tiến bộ trong hệ thống xử lý thông tin thần kinh, tập 30, trang 506–516. Công ty Curran Associates, Inc.

Timo Schick và Hinrich Schutze. 2020. Khai thác đóng các câu hỏi để phân loại văn bản trong vài cảnh quay và suy luận ngôn ngữ tự nhiên.

Thibault Sellam, Dipanjan Das và Ankur Parikh. 2020. BLEURT: Tìm hiểu các số liệu mạnh mẽ cho văn bản thế hệ. Trong Kỷ yếu Hội nghị thư ờng niên lần thứ 58 của Hiệp hội Ngôn ngữ học tính toán, trang 7881–7892, Trực tuyến. Hiệp hội ngôn ngữ học tính toán.

Sheng Shen, Daniel Fried, Jacob Andreas và Dan Klein. 2019. Tạo văn bản mang tính thông tin thực tế . Trong Kỷ yếu Hội nghị 2019 của Hiệp hội Bắc Mỹ cho Ngôn ngữ học tính toán: Ngôn ngữ của con ngư ời Công nghệ, Tập 1 (Bài báo dài và ngắn), trang 4060–4067, Minneapolis, Minnesota. Hiệp hội Ngôn ngữ học tính toán.

Taylor Shin, Yasaman Razeghi, Robert L. Logan IV au2, Eric Wallace và Sameer Singh. 2020. Auto- prompt: Khai thác kiến thức từ các mô hình ngôn ngữ với lời nhắc đư ợc tạo tự động.

Reza Shokri và Vitaly Shmatikov. 2015. Quyền riêng tư - bảo vệ việc học sâu. Trong Kỷ yếu của Hội nghị ACM SIGSAC lần thứ 22 về máy tính và An ninh Truyền thông, CCS '15, trang 1310–1321, New York, NY, Hoa Kỳ . Hiệp hội cho Máy tính.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Lin-nea Micciulla và Ralph Weischedel. 2006. Một nghiên cứu về tỷ lệ lỗi dịch thuật với chú thích của con ngư ời. Trong Kỷ yếu của Hiệp hội Chuyển giao Máy móc ở Châu Mỹ (AMTA 2006.

Asa Cooper Stickland, Xian Li và Marjan Ghazvininejad. 2020. Bí quyết thích nghi mô hình đơ n ngữ và đa ngôn ngữ đư ợc đào tạo trư ớc để dịch máy.

Nishant Subramani, Samuel R. Bowman, và Cho Kyunghyun. 2020. Mô hình ngôn ngữ vô điều kiện có thể phục hồi các câu tùy ý không?

Ramakrishna Vedantam, C. Lawrence Zitnick, và Devi Parikh. 2015. Cider: Hình ảnh dựa trên sự đồng thuận đánh giá mô tả. Trong CVPR, trang 4566–4575. Hiệp hội máy tính IEEE.

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

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas, and Jitendra Malik. 2020a. Side-tuning: A baseline for network adaptation via additive side networks.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020b. BERTScore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2020c. DIALOGPT : Large-scale generative pre-training for conversational response generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 270–278, Online. Association for Computational Linguistics.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi, and Hinrich Schütze. 2020. Masking as an efficient alternative to finetuning for pretrained language models.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer, and Steffen Eger. 2019. MoverScore: Text generation evaluating with contextualized embeddings and earth mover distance. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 563–578, Hong Kong, China. Association for Computational Linguistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. 2020. Extractive summarization as text matching. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 6197–6208, Online. Association for Computational Linguistics.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *International Conference on Learning Representations*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pier-ric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest và Alexander M. Rush. 2020. Transformers: Xử lý ngôn ngữ tự nhiên tiên tiến nhất. Trong Kỷ yếu của Hội nghị 2020 về Phư ơ ng pháp thực nghiệm trong xử lý ngôn ngữ tự nhiên: Trình diễn hệ thống, trang 38–45, Trực tuyến. Hiệp hội Ngôn ngữ học tính toán.

Jeffrey O Zhang, Alexander Sax, Amir Zamir, Leonidas Guibas và Jitendra Malik. 2020a. Điều chỉnh bên: Đư ờng cơ sở cho việc thích ứng mạng thông qua các mạng bên bổ sung.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger và Yoav Artzi. 2020b. Điểm BERTS: Đánh giá việc tạo văn bản với bert. Trong Hội nghị quốc tế về đại diện học tập.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Lư u và Bill Dolan. 2020c. DIALOGPT: Đào tạo trư ớc mang tính khái quát quy mô lớn để tạo phản hồi đàm thoại. Trong Kỷ yếu của Hội nghị thư ờng niên lần thứ 58 của Hiệp hội tính toán Ngôn ngữ học: Trình diễn hệ thống, trang 270–278, Trực tuyến. Hiệp hội Ngôn ngữ học tính toán.

Mengjie Zhao, Tao Lin, Fei Mi, Martin Jaggi và Hin-rich Schutze. 2020. Tạo mặt nạ như một giải pháp thay thế hiệu quả cho việc tinh chỉnh các mô hình ngôn ngữ đư ợc huấn luyện trư ớc.

Wei Zhao, Maxime Peyrard, Fei Liu, Yang Gao, Christian M. Meyer và Steffen Eger. 2019. Điểm Mover: Đánh giá việc tạo văn bản bằng cách nhúng theo ngữ cảnh và khoảng cách di chuyển trên trái đất. Trong thủ tục tố tụng của Hội nghị về phư ơ ng pháp thực nghiệm năm 2019 ở Xử lý ngôn ngữ tự nhiên và Hội nghị chung quốc tế lần thứ 9 về xử lý ngôn ngữ tự nhiên (EMNLP-IJCNLP), trang 563–578, Hong Công, Trung Quốc. Hiệp hội tính toán Lin-guistics.

Ming Zhong, Pengfei Liu, Yiran Chen, Danqing Wang, Xipeng Qiu và Huyền Cảnh Hoàng. 2020. Tóm tắt trích xuất dư ới dạng đối sánh văn bản. Trong thủ tục tố tụng của Hội nghị thư ờng niên lần thứ 58 của Hiệp hội Ngôn ngữ học tính toán, trang 6197–6208, Trực tuyến. Hiệp hội ngôn ngữ học tính toán.

Kim Hoa Chu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Chu Văn Cư ơ ng, Hậu Cư ờng Lý, và Thiết Nham Lư u. 2020. Kết hợp bert vào dịch máy thần kinh. Trong Hội nghị quốc tế về đại diện học tập.

A Supplementary Material

A.1 Datasets and Metrics

We evaluate on three standard neural generation datasets for the table-to-text task: E2E (Novikova et al., 2017), WebNLG (Gardent et al., 2017), and DART (Radev et al., 2020).

The E2E dataset contains approximately 50K examples with 8 distinct fields; it contains multiple test references for one source table, and the average output length is 22.9. We use the official evaluation script,[12] which reports BLEU (Papineni et al., 2002), NIST (Belz and Reiter, 2006), METEOR (Lavie and Agarwal, 2007), ROUGE-L (Lin, 2004), and CIDEr (Vedantam et al., 2015).

The WebNLG (Gardent et al., 2017) dataset consists of 22K examples, and the input x is a sequence of (subject, property, object) triples. The average output length is 22.5. In the training and validation splits, the input describes entities from 9 distinct DBpedia categories (e.g., Monument). The test split consists of two parts: the first half contains DB categories seen in training data, and the second half contains 5 unseen categories. These unseen categories are used to evaluate extrapolation. We use the official evaluation script, which reports BLEU, METEOR and TER (Snover et al., 2006).

DART (Radev et al., 2020) is an open domain table-to-text dataset, with similar input format (entity-relation-entity triples) as WebNLG. The average output length is 21.6. It consists of 82K examples from WikiSQL, WikiTableQuestions, E2E, and WebNLG and applies some manual or automated conversion. We use the official evaluation script[13] and report BLEU, METEOR, TER, MoverScore (Zhao et al., 2019), BERTScore (Zhang et al., 2020b) and BLEURT (Sellam et al., 2020).

For the summarization task, we use the XSUM (Narayan et al., 2018) dataset, which is an abstractive summarization dataset on news articles. There are 225K examples. The average length of the articles is 431 words and the average length of the summaries is 23.3. We report ROUGE-1, ROUGE-2 and ROUGE-L, computed by the python package `rouge-score`.

Data pre-processing. For table-to-text, we linearize a table x in order to fit into a language model context. In the E2E dataset, for example, "(field A, value A), (field B, value B)" is linearized to "field A : value A | field B : value B". Also, in WebNLG and DART, a sequence of triple "(entity1.1, relation1, entity1.2), (entity2.1, relation2, entity2.2)" is linearlized as "entity1.1 : relation1 : entity1.2 | entity2.1 : relation2 : entity2.2".

For summarization, we truncate the articles x to 512 BPE tokens.

Extrapolation data splits. We construct two extrapolation data splits `news-to-sports` and `within-news` from the original XSUM dataset. XSUM dataset is drawn from BBC news, and we identify the topic of each article based on its URL. Since "news" and "sports" are the two domains with the most articles, we create our first train/test split. Additionally, "news" has subdomains such as "UK", "world", and "technology". Consequently, we create a second data split, using the top 3 news subdomains (i.e. {world, UK, business }) as training data and the rest as test data.

A.2 Hyperparameters

In Table 6, we report the hyperparameters used to train the best-performing models documented in the experiment section.

As for the search range of each hyperparameters: the learning rates are selected from {1e-5, 5e-05, 8e-05}; the number of epochs are selected from {5, 10} for table-to-text and {5, 25, 30 } for summarization; We select the largest batch size that can fit into GPU memory and didn't explicitly tune for an optimal batch size. Prefix length are selected from {1, 5, 10, 20, 40} for table-to-text and {1, 10, 20, 50, 80, 100, 200, 300} for summarization. We use perplexity and automatic generation metrics on the validation set to select the best-performing models.

For table-to-text in the low data settings, we use a learning rate of 5e-5, and a batch size of 10. We use a prefix length of 6, since we apply the initialization trick and initialize the prefix with "table-to-text:", which contains 6 BPE tokens. Instead of tuning the number of epochs, we tune the max steps of updates in {100, 200, 400, 600 }, as shown in Table 8. We apply early stopping based on the performance of validation set, where the validation size =30% training size.

For summarization in the low data settings, we use a learning rate of 5e-5 and a warmup step of 100. We use a batch size of 5 for prefix-tuning and 6 for fine-tuning. We apply the initialization trick and use the word "summarize" to initialize

[12] https://github.com/tuetschek/e2e-metrics

[13] https://github.com/Yale-LILY/dart

Tài liệu bổ sung

A.1 Bộ dữ liệu và số liệu

Chúng tôi đánh giá dựa trên ba bộ dữ liệu tạo thần kinh tiêu chuẩn cho tác vụ chuyển bảng thành văn bản: E2E (Novikova và cộng sự, 2017), WebNLG (Gardent và cộng sự, 2017) và DART (Radev và cộng sự, 2020).

Bộ dữ liệu E2E chứa khoảng 50K mẫu với 8 trư ờng riêng biệt; nó chứa nhiều tham chiếu thử nghiệm cho một bảng nguồn và độ dài đầu ra trung bình là 22,9. Chúng tôi sử dụng tập lệnh đánh giá chính thức12 báo cáo BLEU (Papineni và cộng sự, 2002), NIST (Belz và Reiter, 2006), METEOR (Lavie và Agarwal, 2007), ROUGE-L (Lin, 2004), và CIDEr (Vedantam và cộng sự, 2015).

Bộ dữ liệu WebNLG (Gardent et al., 2017) bao gồm 22K ví dụ và đầu vào x là một chuỗi gồm ba (chủ đề, thuộc tính, đối tư ợng). Độ dài đầu ra trung bình là 22,5. Trong phần tách đào tạo và xác thực, đầu vào mô tả các thực thể từ 9 danh mục DBpedia riêng biệt (ví dụ: Monument). Phần phân chia kiểm tra bao gồm hai phần: nửa đầu chứa các danh mục DB đư ợc thấy trong dữ liệu huấn luyện và nửa thứ hai chứa 5 danh mục chư a đư ợc nhìn thấy. Những phạm trù chư a đư ợc nhìn thấy này đư ợc sử dụng để đánh giá phép ngoại suy. Chúng tôi sử dụng tập lệnh đánh giá chính thức, báo cáo BLEU, METEOR và TER (Snover et al., 2006).

DART (Radev và cộng sự, 2020) là một tập dữ liệu chuyển bảng thành văn bản trong miền mở, với định dạng đầu vào tư ơ ng tự (bộ ba thực thể-quan hệ-thực thể) như WebNLG. Độ dài đầu ra trung bình là 21,6. Nó bao gồm 82K ví dụ từ WikiSQL, WikiTableQuestions, E2E và WebNLG và áp dụng một số chuyển đổi thủ công hoặc tự động. Chúng tôi sử dụng đánh giá chính thức

script13 và báo cáo BLEU, METEOR, TER, Mover-Score (Zhao và cộng sự, 2019), BERTScore (Zhang và cộng sự, 2020b) và BLEURT (Sellam và cộng sự, 2020).

Đối với nhiệm vụ tóm tắt, chúng tôi sử dụng bộ dữ liệu XSUM (Narayan và cộng sự, 2018) , đây là bộ dữ liệu tóm tắt trừu tư ợng về các bài báo. Có 225K ví dụ. Độ dài trung bình của các bài viết là 431 từ và độ dài trung bình của phần tóm tắt là 23,3. Chúng tôi báo cáo ROUGE-1, ROUGE- 2 và ROUGE-L, đư ợc tính toán bằng gói python điểm rouge.

Xử lý trư ớc dữ liệu. Đối với chuyển bảng thành văn bản, chúng tôi tuyến tính hóa bảng x để phù hợp với ngữ cảnh mô hình ngôn ngữ. Ví dụ: trong tập dữ liệu E2E, "(trư ờng A, value A), (field B, value B)" đư ợc tuyến tính hóa thành "field A : value A | trư ờng B : giá trị B". Ngoài ra, trong WebNLG và DART, một chuỗi ba "(thực thể1.1, quan hệ1, thực thể1.2), (thực thể2.1, quan hệ2, thực thể2.2)" đư ợc tuyến tính hóa thành "thực thể1.1 : quan hệ1 : thực thể1. 2 | thực thể2.1 : quan hệ2 : thực thể2.2".

Để tóm tắt, chúng tôi cắt bớt các mạo từ x thành 512 mã thông báo BPE.

Phân chia dữ liệu ngoại suy. Chúng tôi xây dựng hai phần dữ liệu ngoại suy từ tin tức đến thể thao và tin tức nội bộ từ tập dữ liệu XSUM ban đầu.
Tập dữ liệu XSUM đư ợc lấy từ tin tức của BBC và chúng tôi xác định chủ đề của từng bài viết dựa trên URL của bài viết đó. Vì "tin tức" và "thể thao" là hai miền có nhiều bài viết nhất nên chúng tôi tạo phần tách đào tạo/kiểm tra đầu tiên. Ngoài ra, "tin tức" còn có các tên miền phụ như "Vư ơ ng quốc Anh", "thế giới" và "công nghệ". Do đó, chúng tôi tạo phần phân tách dữ liệu thứ hai, sử dụng 3 tên miền phụ tin tức hàng đầu (tức là {world, UK, business }) làm dữ liệu đào tạo và phần còn lại làm dữ liệu thử nghiệm.

A.2 Siêu tham số

Trong Bảng 6, chúng tôi báo cáo các siêu tham số đư ợc sử dụng để huấn luyện các mô hình hoạt động tốt nhất đư ợc ghi lại trong phần thử nghiệm.

Đối với phạm vi tìm kiếm của từng siêu tham số: tốc độ học đư ợc chọn từ {1e-5, 5e-05, 8e-05}; số lư ợng kỷ nguyên đư ợc chọn từ {5, 10} cho bảng thành văn bản và {5, 25, 30 } cho tóm tắt; Chúng tôi chọn kích thư ớc lô lớn nhất có thể vừa với bộ nhớ GPU và không điều chỉnh rõ ràng để có kích thư ớc lô tối ư u. Độ dài tiền tố đư ợc chọn từ {1, 5, 10, 20, 40} cho bảng thành văn bản và {1, 10, 20, 50, 80, 100, 200, 300} cho tóm tắt. Chúng tôi sử dụng các số liệu tạo phức tạp và tự động trên bộ xác thực để chọn các mô hình hoạt động tốt nhất.

Đối với tính năng chuyển bảng thành văn bản ở cài đặt dữ liệu thấp, chúng tôi sử dụng tốc độ học tập là 5e-5 và kích thư ớc lô là 10. Chúng tôi sử dụng độ dài tiền tố là 6 vì chúng tôi áp dụng thủ thuật khởi tạo và khởi tạo tiền tố bằng "bảng -to-text:", chứa 6 mã thông báo BPE. Thay vì điều chỉnh số lư ợng kỷ nguyên, chúng tôi điều chỉnh các bư ớc cập nhật tối đa trong {100, 200, 400, 600 }, như trong Bảng 8. Chúng tôi áp dụng tính năng dừng sớm dựa trên hiệu suất của bộ xác thực, trong đó kích thư ớc xác thực =30 % quy mô đào tạo

Để tóm tắt trong cài đặt dữ liệu thấp, chúng tôi sử dụng tốc độ học tập là 5e-5 và bư ớc khởi động là 100. Chúng tôi sử dụng kích thư ớc lô là 5 để tinh chỉnh tiền tố và 6 để tinh chỉnh. Chúng ta áp dụng thủ thuật khởi tạo và dùng từ "tóm tắt" để khởi tạo

12https://github.com/tuetschek/e2e-metrics số liệu điện tử

13https://github.com/Yale-LILY/dart

	learning rate	# epoch	batch size	prefix length
Prefix:				
E2E	8e-05	5	10	5
WebNLG	5e-05	5	5	5
DART	5e-05	10	5	10
XSUM	5e-05	30	14	100
Adapter:				
E2E (3%)	5e-05	5	5	-
E2E (0.1%)	8e-05	10	5	
WebNLG (3%)	5e-05	5	5	-
WebNLG (0.1%)	5e-05	10	5	-
DART (3%)	5e-05	5	5	-
DART (0.1%)	8e-05	5	5	-
Fine-tune:				
E2E	5e-05	5	10	-
WebNLG	1e-05	10	6	-
DART	1e-05	10	6	-
FT-top2:				
E2E	5e-05	5	10	-
WebNLG	5e-05	10	9	-
DART	5e-05	5	5	-
within-news				
Fine-tune	3e-5	5	18	-
Prefix	5e-5	30	36	80
news-to-sports				
Fine-tune	3e-5	5	18	-
Prefix	5e-5	15	36	40

Table 6: Hyperparameter settings for our method and baseline methods.

	R-1 ↑	R-2 ↑	R-L ↑
PREFIX(2%)	43.30	20.35	35.21
PREFIX(0.1%)	41.54	18.56	33.13

Table 7: Metrics for summarization on XSUM validation set.

the prefix, resulting in a prefix length of 1. We tune the number of epochs in {3, 5, 10, 20, 30}, shown in Table 8. We also apply early stopping based on validation performance.

For the extrapolation setting, the hyperparameters for our table-to-text model is the same as the hyperparameters of WebNLG. The hyperparameters for summarization is shown in the last block of Table 6.

A.3 Validation Performance

Table 9 shows the validation performance on the three table-to-text datasets. Table 7 shows the validation performance on XSUM.

	size=50	size=100	size=200	size=500
Prefix (max steps)	200	200	200	400
Finetune (max steps)	100	100	200	400
Prefix (epoch)	30	20	20	20
Finetune (epoch)	30	10	3	3

Table 8: Max # update steps for low data settings.

A.4 Additional Results for Low-data Settings

Figure 7 supplements the low-data performance curves in Figure 3 by plotting the relationship between training size and generation metrics for both prefix-tuning and fine-tuning.

A.5 Additional Results for the Initialization Experiment

Figure 8 supplements Figure 3 by plotting additional metrics for our initialization technique §7.4. It validates that random initialization (from a uniform (0,1) distirbution) significantly underperforms initializing with real words; Additionally, initializing with task-relevant words (e.g., "summarization" and "table-to-text") attains slightly better generation scores than initializing with task-irrelevant words (e.g., "elephant" and "banana").

A.6 Qualitative Examples for Extrapolation

Table 10 contains qualitative examples from both seen and unseen categories in WebNLG. We find that for unseen categories, both prefix-tuning and fine-tuning tend to undergenerate (generated output do not cover full table contents) or generate untruthfully (generated output is inconsistent with table contents). In particular, prefix-tuning tends to undergenerate whereas fine-tuning tends to generate untruthfully. For seen categories, both perform fairly well in terms of coverage and truthfulness.

	tốc độ học tập	# kỷ nguyên	kích thước lô	độ dài tiền tố
Tiếp đầu ngữ:				
E2E	8e-05	5	10	5
WebNLG	5e-05			5
DART	5e-05	5 10	5 5	10
XSUM	5e-05	30	14	100
Bộ chuyển đổi:				
E2E (3%)	5e-05			-
E2E (0,1%)	8e-05	5	5 5	
WebNLG (3%)	5e-05		5	-
WebNLG (0,1%)	5e-05	10	5	-
DART (3%)	5e-05	5 10 5	5	-
DART (0,1%)	8e-05	5	5	-
Tinh chỉnh:				
E2E	5e-05		10	-
WebNLG	1e-05	5		-
DART	1e-05	10 10	6 6	-
FT-top2:				
E2E	5e-05		10	-
WebNLG	5e-05	5		-
DART	5e-05	10 5	9 5	-
tin tức nội bộ				
Tinh chỉnh	3e-5	5	18	-
Tiếp đầu ngữ	5e-5	30	36	80
tin tức-thể thao				
Tinh chỉnh	3e-5		18	-
Tiếp đầu ngữ	5e-5	5 15	36	40

Bảng 6: Cài đặt siêu tham số cho phương pháp của chúng tôi và các phương pháp cơ bản.

	R-1	R-2	RL
TIỀN TỐ(2%)	43,30	20,35	35,21
TIỀN TỐ(0,1%)	41,54	18,56	33,13

Bảng 7: Số liệu tóm tắt về bộ xác thực XSUM.

tiền tố, dẫn đến độ dài tiền tố là 1. Chúng tôi điều chỉnh số kỷ nguyên trong {3, 5, 10, 20, 30}, được hiển thị trong Bảng 8. Chúng tôi cũng áp dụng việc dừng sớm dựa trên hiệu suất xác nhận.

Đối với cài đặt ngoại suy, các siêu tham số cho mô hình chuyển bảng thành văn bản của chúng tôi giống như siêu tham số của WebNLG. Các siêu tham số để tóm tắt được hiển thị trong khối cuối cùng của Bảng 6.

A.3 Hiệu suất xác nhận

Bảng 9 cho thấy hiệu suất xác nhận trên ba bộ dữ liệu từ bảng thành văn bản. Bảng 7 cho thấy giá trị hiệu suất hẹn hò trên XSUM.

	kích thước=50	kích thước=100	kích thước=200	kích thước=500
Tiền tố (bước tối đa)	200	200	200	400
Finetune (bước tối đa)	100	100	200	400
Tiền tố (kỷ nguyên)	30	20	20	20
Finetune (kỷ nguyên)	30	10	3	3

Bảng 8: Số bước cập nhật tối đa cho cài đặt dữ liệu thấp.

A.4 Kết quả bổ sung cho cài đặt dữ liệu thấp

Hình 7 bổ sung hiệu suất dữ liệu thấp các đường cong trong Hình 3 bằng cách vẽ biểu đồ mối quan hệ giữa quy mô đào tạo và số liệu tạo cho cả hai điều chỉnh tiền tố và tinh chỉnh.

A.5 Kết quả bổ sung cho việc khởi tạo

Cuộc thí nghiệm

Hình 8 bổ sung Hình 3 bằng cách vẽ các số liệu bổ sung cho kỹ thuật khởi tạo của chúng tôi §7.4. Nó xác nhận rằng việc khởi tạo ngẫu nhiên (từ phân phối đồng nhất (0,1)) hoạt động kém hơn đáng kể khởi tạo bằng từ thật; Ngoài ra, việc khởi tạo bằng các từ liên quan đến nhiệm vụ (ví dụ: "tóm tắt" và "chuyển bảng thành văn bản") đạt được điểm số thế hệ tốt hơn một chút so với việc khởi tạo bằng các tác vụ không liên quan từ (ví dụ: "con voi" và "chuối").

A.6 Các ví dụ định tính cho phép ngoại suy

Bảng 10 chứa các ví dụ định tính từ cả hai các danh mục đã thấy và chưa thấy trong WebNLG. Chúng ta tìm thấy đối với các danh mục không được nhìn thấy, cả điều chỉnh tiền tố và tinh chỉnh có xu hướng phát sinh kém (đầu ra được tạo không bao gồm toàn bộ nội dung bảng) hoặc tạo ra không trung thực (đầu ra được tạo ra không nhất quán với nội dung bảng). Đặc biệt, việc điều chỉnh tiền tố có xu hướng kém phát triển trong khi việc tinh chỉnh có xu hướng tạo ra không trung thực. Đối với các danh mục đã xem, cả hai đều hoạt động khá tốt về độ bao phủ và độ trung thực.

	E2E					WebNLG			DART					
	BLEU	NIST	MET	R-L	CIDEr	BLEU	MET	TER ↓	BLEU	MET	TER ↓	Mover	BERT	BLEURT
GPT-2$_{\text{MEDIUM}}$														
FT-FULL	74.2	8.76	49.3	76.9	2.66	66.03	0.47	0.30	50.46	0.41	0.44	0.52	0.95	0.41
FT-TOP2	72.7	8.51	48.2	75.3	2.60	54.61	0.39	0.47	48.41	0.39	0.48	0.48	0.94	0.33
ADAPTER(3%)	71.7	8.53	48.4	74.6	2.60	60.63	0.43	0.33	48.56	0.40	0.44	0.51	0.95	0.40
ADAPTER(0.1%)	68.1	8.30	45.9	71.4	2.41	53.24	0.40	0.39	44.72	0.38	0.47	0.47	0.94	0.35
PREFIX(0.1%)	74.8	8.80	49.4	76.8	2.69	64.52	0.46	0.32	51.11	0.41	0.43	0.52	0.95	0.42
GPT-2$_{\text{LARGE}}$														
FT-FULL	72.1	8.62	48.5	75.1	2.56	64.69	0.46	0.31	51.00	0.41	0.44	0.52	0.95	0.43
Prefix	74.8	8.81	49.5	77.0	2.72	64.11	0.46	0.33	50.84	0.41	0.43	0.52	0.95	0.42

Table 9: Metrics on the development set (higher is better, except for TER) for table-to-text generation on E2E (left), WebNLG (middle) and DART (right).

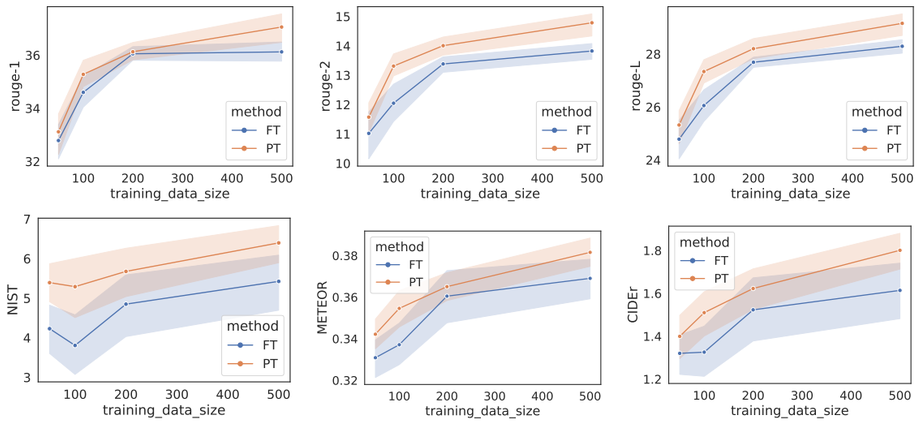


Figure 7: Prefix-tuning (orange) outperforms fine-tuning (blue) in low-data regimes in addition to requiring many fewer parameters. The top three plots correspond to summarization, measured by ROUGE-1, ROUGE-2, and ROUGE-L. The bottom three plots correspond to table-to-text, measured by NIST, METEOR, and CIDEr. The x-axis is the training size and the y-axis is the evaluation metric (higher is better).

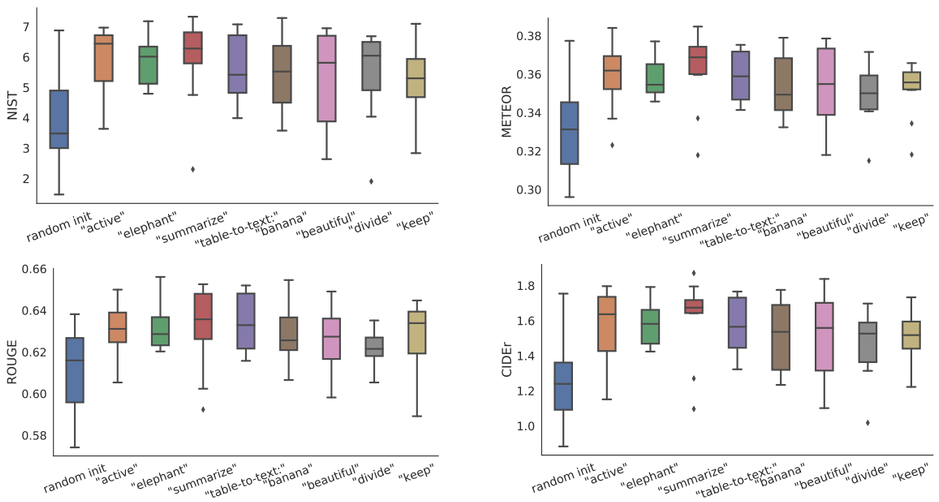
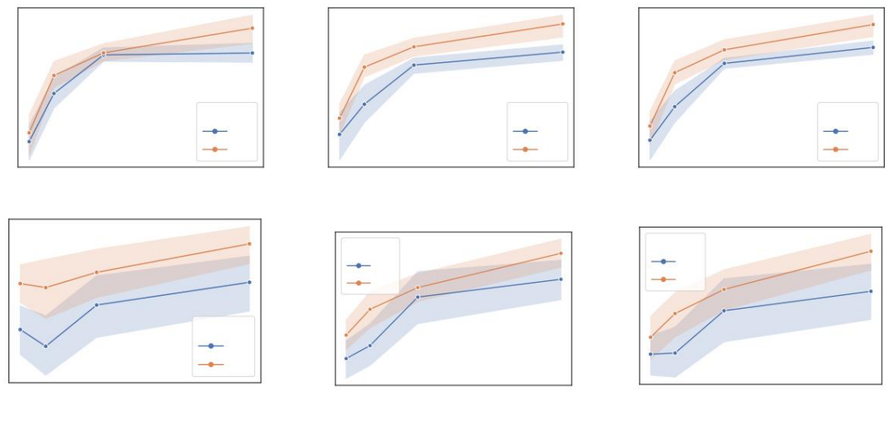


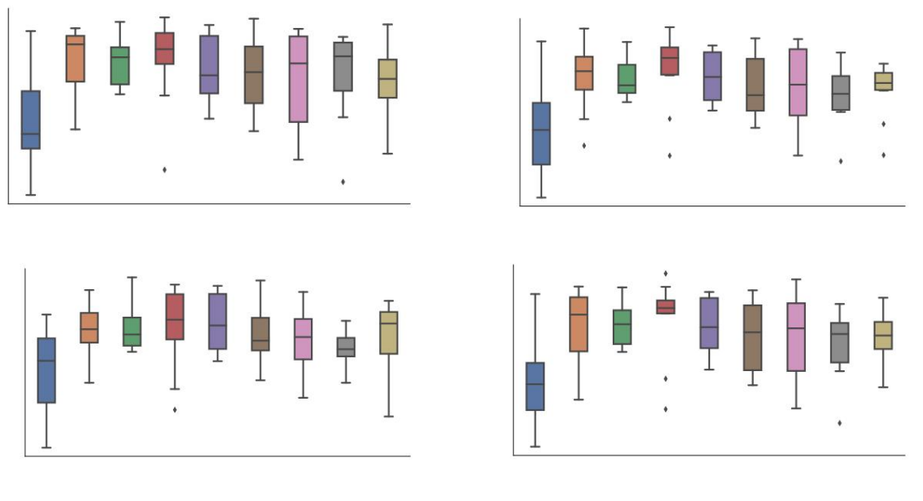
Figure 8: Initializing the prefix with activations of real words significantly outperforms random initialization, in a low-data setting with 100 training data.

	E2E					WebNLG			DART					
	BLEU	NIST	MET	RL	CIDEr	BLEU	MET	TER	BLEU	MET	TER	Mover	BERT	BLEURT
GPT-2TRUNG BÌNH														
FT-FULL	74,2	8,76	49,3	76,9	2,66	66,03	0,47	0,30	50,46	0,41	0,44	0,52	0,95	0,41
FT-TOP2	72,7	8,51	48,2	75,3	2,60	54,61	0,39	0,47	48,41	0,39	0,48	0,48	0,94	0,33
BỘ CHUYỂN ĐỔI(3%)	71,7	8,53	48,4	74,6	2,60	60,63	0,43	0,33	48,56	0,40	0,44	BỘ CHUYỂN ĐỔI(0,1%)	68,1	8,30	45,9	0,51	0,95	0,40
71,4	2,41	53,24	0,40	0,39	44,72	0,38	0,47	TIỀN TỐ(0,1%)	74,8	8,80	49,4	76,8	2,69	64,52	0,46	0,32	51,11	0,41	0,43	0,47	0,94	0,35
																				0,52	0,95	0,42
GPT-2LỚN														
FT-FULL	72,1	8,62	48,5	75,1	2,56	64,69	0,46	0,31	51,00	0,41	0,44	0,52	0,95	0,43
Tiếp đầu ngữ	74,8	8,81	49,5	77,0	2,72	64,11	0,46	0,33	50,84	0,41	0,43	0,52	0,95	0,42

Bảng 9: Các số liệu trên bộ phát triển (càng cao càng tốt, ngoại trừ TER) để tạo bảng thành văn bản trên E2E (trái), WebNLG (giữa) và DART (phải).



Hình 7: Tinh chỉnh tiền tố (màu cam) hoạt động tốt hơ n tinh chỉnh (màu xanh) trong chế độ dữ liệu thấp ngoài việc yêu cầu nhiều ít thông số hơ n. Ba ô trên cùng tư ơ ng ứng với tóm tắt, đư ợc đo bằng ROUGE-1, ROUGE-2 và ROUGE-L. Ba ô dư ới cùng tư ơ ng ứng với bảng thành văn bản, đư ợc đo bằng NIST, METEOR và CIDEr. Các Trục x là kích thư ớc đào tạo và trục y là chỉ số đánh giá (càng cao càng tốt).



Hình 8: Việc khởi tạo tiền tố bằng cách kích hoạt các từ thực tốt hơ n đáng kể so với việc khởi tạo ngẫu nhiên, theo một cách cài đặt dữ liệu thấp với 100 dữ liệu huấn luyện.

Source [Unseen, Athelete]	(Al Kharaitiyat SC, club, Alaa Abdul-Zahra), (Al Khor, ground, Al Kharaitiyat SC), (Shabab Al-Ordon Club, club, Alaa Abdul-Zahra) (Amar Osim, manager, Al Kharaitiyat SC)
Prefix-tuning	Al Kharaitiyat SC are managed by Amar Osim and play at their ground at Al Khor. Al Kharaitiyat SC are also the club for which Alaa Abdul-Zahra is a player.
Fine-tuning	Alaa Abdul-Zahra plays for Al-Kharaitiyat SC and Shabab Al-Ordon Club. He also plays for Al-Khor and manages Al-Kharaitiyat SC.
Reference	Alaa Abdul Zahra plays for Al Kharaitiyat SC which is located at Al Khor and managed by Amar Osim. The Shabab Al-Ordon club is associated with Alaa Abdul-Zahra.
Source [Unseen, Transportation]	(Genoa, location, Costa Crociere), (AIDA Cruises, operator, AIDAstella), (Costa Crociere, owner, AIDAstella)
Prefix-tuning	AID Astella is operated by Aida Cruises and is owned by the Costa Rican tourist resort of Genoa.
Fine-tuning	AID Astella, operated by AIDA-Cruises, is located in Genoa and is owned by the Costa Rican government.
Reference	Costa Crociere is the owner of the AIDAstella and are based in Genoa. The operator of AIDAstella is AIDA Cruises.
Source [Unseen, Politician]	(Euro, currency, Netherlands), (Stellendam, birthPlace, Ab Klink), (Netherlands, nationality, Ab Klink)
Prefix-tuning	Ab Klink was born in Stellendam and is a national of the Netherlands where the currency is the Euro.
Fine-tuning	Ab Klink is a national of the Netherlands where the currency is the Euro. He was born in Stellendam.
Reference	Ab Klink was born in Stellendam in the Netherlands, where the national currency is the euro.
Source [Unseen, Politician]	(Robert E, Lee, commander, Battle of Salem Church), (American Civil War, isPartOfMilitaryConflict, Battle of Salem Church), (Battle of Salem Church, battles, Aaron S. Daggett)
Prefix-tuning	Robert E. Lee was the commander of the Battle of Salem Church which was part of the military conflict in the American Civil war.
Fine-tuning	The Battle of Salem Church is part of the American Civil War and was commanded by Robert E. Lee.
Reference	Robert E Lee was a commander in the Battle of Salem Church, which was one of the military conflicts in the American Civil War. Aaron S Daggett fought in the same battle.
Source [Unseen, Artist]	(Christian alternative rock, musicSubgenre, Alternative rock), (Alternative rock, genre, Andrew White (musician))
Prefix-tuning	Andrew White is a Christian alternative rock musician.
Fine-tuning	Andrew White, a Christian alternative rocker, performs.
Reference	The musician Andrew White's genre is alternative rock, the genre which has the sub genre Christian alternative rock.
Source [Unseen, Artist]	(Hip hop music, genre, Allen Forrest), (solo singer, background, Allen Forrest)
Prefix-tuning	Allen Forrest is a solo singer.
Fine-tuning	Born in
Reference	Allen Forrest is a solo singer whose genre is Hip Hop music.
Source [Seen, ComicsCharacter]	(Americans, nationality, Ducan Rouleau), (Ducan Rouleau, creator, Baymax),(Alan Tudyk, starring, Big Hero 6 (film)), (Steven T Segle, creator, Baymax), (Big Hero 6 (film), serires, Baymax)
Prefix-tuning	Baymax is a character in Big Hero 6 which stars Alan Tudyk. He was created by Steven T. Seagle and the American, Duncan Rouleau.
Fine-tuning	Alan Tudyk stars in the film Big Hero 6 in which Baymax is a character created by Steven T. Seagle and the American, Duncan Rouleau.
Reference	Baymax is a character who appeared in Big Hero 6 starring Alan Tudyk. It was created by Steven T Seagle and the American, Duncan Rouleau.
Source [Seen, City]	(Washington, D.C., capital, United States), (White Americans, ethnicGroup, United States), (United States, country, New Jersey), (New York City, largest City, United States), (New Jersy, isPartOf, Atlantic City)
Prefix-tuning	Washington D.C. is the capital of the United States where the largest city is New York City and the White Americans are an ethnic group. Atlantic City, New Jersey is also part of the United States.
Fine-tuning	Atlantic City, New Jersey is part of New Jersey in the United States. The capital city is Washington D.C. and one of the ethnic groups is White Americans.
Reference	New York City (NYC) is the largest U.S. city. Atlantic City, New Jersey are also part of the United States with its capital as Washington, DC and home to White Americans.

Table 10: Qualitative examples from WebNLG. The first 6 examples are from the unseen categories, labeled next to source; the last two examples are from the seen categories. For unseen categories, both prefix-tuning and fine-tuning tend to undergenerate (generated output do not cover full table contents) or generate untruthfully (generated output is inconsistent with table contents). In particular, prefix-tuning tends to undergenerate more often than generate untruthfully whereas fine-tuning tends to generate untruthfully. For seen categories, both perform fairly well in terms of coverage and truthfulness.

Nguồn [Chư a thấy, Vận động viên]	(Al Kharaitiyat SC, câu lạc bộ, Alaa Abdul-Zahra), (Al Khor, sân, Al Kharaitiyat SC), (Câu lạc bộ Shabab Al-Ordon, câu lạc bộ, Alaa Abdul-Zahra) (Amar Osim, quản lý, Al Kharaitiyat SC)
Điều chỉnh tiền tố	Al Kharaitiyat SC đư ợc quản lý bởi Amar Osim và thi đấu trên sân của họ là Al Khor. Al Kharaitiyat SC cũng
Tinh chỉnh	câu lạc bộ mà Alaa Abdul-Zahra là cầu thủ. Alaa Abdul-Zahra chơ i cho Al-Kharaitiyat SC và Câu lạc bộ Shabab Al-Ordon. Anh ấy cũng chơ i cho Al-Khor và
Thẩm quyền giải quyết	quản lý Al-Kharaitiyat SC. Alaa Abdul Zahra chơ i cho Al Kharaitiyat SC có trụ sở tại Al Khor và đư ợc quản lý bởi Amar Osim. Các Câu lạc bộ Shabab Al-Ordon đư ợc liên kết với Alaa Abdul-Zahra.
Nguồn [Không nhìn thấy, Giao thông vận tải]	(Genoa, địa điểm, Costa Crociere), (AIDA Cruises, nhà điều hành, AIDAstella), (Costa Crociere, chủ sở hữu, AIDAstella)
Điều chỉnh tiền tố	AID Astella đư ợc điều hành bởi Aida Cruises và thuộc sở hữu của khu du lịch Genoa của Costa Rica.
Tinh chỉnh	AID Astella, do AIDA-Cruises điều hành, có trụ sở tại Genoa và thuộc sở hữu của chính phủ Costa Rica.
Thẩm quyền giải quyết	Costa Crociere là chủ sở hữu của AIDAstella và có trụ sở tại Genoa. Nhà điều hành AIDAstella là AIDA Du lịch trên biển.
Nguồn [Không thấy, Chính trị gia]	(Euro, tiền tệ, Hà Lan), (Stellendam, nơ i sinh, Ab Klink), (Hà Lan, quốc tịch, Ab Klink)
Điều chỉnh tiền tố	Ab Klink sinh ra ở Stellendam và là công dân Hà Lan với tiền tệ là Euro.
Tinh chỉnh	Ab Klink là công dân Hà Lan với tiền tệ là Euro. Anh ấy sinh ra ở Stellendam.
Thẩm quyền giải quyết	Ab Klink sinh ra ở Stellendam ở Hà Lan, nơ i có đồng tiền quốc gia là đồng euro.
Nguồn [Không thấy, Chính trị gia]	(Robert E, Lee, chỉ huy, Trận chiến Salem Church), (Nội chiến Hoa Kỳ , isPartOfMilitaryConflict, Trận chiến Nhà thờ Salem), (Trận chiến nhà thờ Salem, trận chiến, Aaron S. Daggett)
Điều chỉnh tiền tố	Robert E. Lee là chỉ huy của Trận chiến Nhà thờ Salem, một phần của cuộc xung đột quân sự ở Nội chiến Hoa Kỳ .
Tinh chỉnh	Trận chiến nhà thờ Salem là một phần của Nội chiến Hoa Kỳ và đư ợc chỉ huy bởi Robert E. Lee.
Thẩm quyền giải quyết	Robert E Lee là ngư ời chỉ huy trong trận chiến ở Nhà thờ Salem, một trong những cuộc xung đột quân sự ở Nội chiến Hoa Kỳ . Aaron S Daggett đã chiến đấu trong trận chiến tư ơ ng tự.
Nguồn [Chư a thấy, Nghệ sĩ]	(Christian alternative rock, musicSubgenre, Alternative rock), (Alternative rock, thể loại, Andrew White (nhạc sĩ))
Điều chỉnh tiền tố	Andrew White là một nhạc sĩ nhạc alternative rock theo đạo thiên chúa.
Tinh chỉnh	Andrew White, một nghệ sĩ nhạc alternative rock theo đạo thiên chúa, biểu diễn.
Thẩm quyền giải quyết	Thể loại của nhạc sĩ Andrew White là alternative rock, thể loại có thể loại phụ là Christian alternative. đá.
Nguồn [Chư a thấy, Nghệ sĩ]	(Nhạc hip hop, thể loại, Allen Forrest), (ca sĩ solo, nền, Allen Forrest)
Điều chỉnh tiền tố	Allen Forrest là một ca sĩ solo.
Tinh chỉnh	Sinh ra ở
Thẩm quyền giải quyết	Allen Forrest là ca sĩ solo thể loại nhạc Hip Hop.
Nguồn [Đã xem, ComicsCharacter]	(Ngư ời Mỹ, quốc tịch, Ducan Rouleau), (Ducan Rouleau, ngư ời sáng tạo, Baymax),(Alan Tudyk, diễn viên chính, Big Hero 6 (phim)), (Steven T Segle, ngư ời sáng tạo, Baymax), (Big Hero 6 (phim), loạt phim, Baymax)
Điều chỉnh tiền tố	Baymax là nhân vật trong Big Hero 6 với sự tham gia của Alan Tudyk. Anh ta đư ợc tạo ra bởi Steven T. Seagle và Ngư ời Mỹ, Duncan Rouleau.
Tinh chỉnh	Alan Tudyk đóng vai chính trong bộ phim Big Hero 6, trong đó Baymax là nhân vật đư ợc tạo ra bởi Steven T. Seagle và Ngư ời Mỹ, Duncan Rouleau.
Thẩm quyền giải quyết	Baymax là nhân vật xuất hiện trong Big Hero 6 với sự tham gia của Alan Tudyk. Nó đư ợc tạo ra bởi Steven T Seagle và ngư ời Mỹ, Duncan Rouleau.
Nguồn [Đã xem, Thành phố]	(Washington, DC, thủ đô, Hoa Kỳ), (Ngư ời Mỹ da trắng, nhóm dân tộc, Hoa Kỳ), (Hoa Kỳ , quốc gia, New Jersey), (Thành phố New Yorx, Thành phố lớn nhất, Hoa Kỳ), (New Jersy, isPartOf, Thành phố Atlantic)
Điều chỉnh tiền tố	Washington DC là thủ đô của Hoa Kỳ với thành phố lớn nhất là thành phố New Yorx và White Ngư ời Mỹ là một nhóm dân tộc. Thành phố Atlantic, New Jersey cũng là một phần của Hoa Kỳ .
Tinh chỉnh	Thành phố Atlantic, New Jersey là một phần của New Jersey ở Hoa Kỳ . Thủ đô là Washington DC và một trong những nhóm dân tộc là ngư ời Mỹ da trắng.
Thẩm quyền giải quyết	Thành phố New Yorx (NYC) là thành phố lớn nhất nư ớc Mỹ. Thành phố Atlantic, New Jersey cũng là một phần của Hoa Kỳ với thủ đô của nó là Washington, DC và là quê hư ơ ng của ngư ời Mỹ da trắng.

Bảng 10: Các ví dụ định tính từ WebNLG. 6 ví dụ đầu tiên thuộc các danh mục chư a đư ợc xem, đư ợc gắn nhãn tiếp theo tới nguồn; hai ví dụ cuối cùng là từ các danh mục đã thấy. Đối với các danh mục không nhìn thấy đư ợc, cả việc tinh chỉnh tiền tố và tinh chỉnh đều có xu hư ớng tạo ra kém hơ n (đầu ra đư ợc tạo ra không bao gồm toàn bộ nội dung bảng) hoặc tạo ra không trung thực (đư ợc tạo ra đầu ra không nhất quán với nội dung bảng). Đặc biệt, việc điều chỉnh tiền tố có xu hư ớng kém phát triển thư ờng xuyên hơ n tạo ra không trung thực trong khi tinh chỉnh có xu hư ớng tạo ra không trung thực. Đối với các danh mục đã xem, cả hai đều hoạt động khá tốt về mức độ bao phủ và tính trung thực.