

# Cross-Domain Content Generation with Domain-Specific Small Language Models

Ankit Maloo\*, Abhinav Garg

Clio AI Inc, DE, USA

**Abstract** Generating domain-specific content using small language models poses challenges, especially when dealing with multiple distinct datasets with minimal overlap. In this study, we explore methods to enable a small language model to produce coherent and relevant outputs for two different domains: stories (Dataset A) and recipes (Dataset B). Our initial experiments show that training individual models on each dataset yields satisfactory results, with each model generating appropriate content within its domain. We find that utilizing custom tokenizers tailored to each dataset significantly enhances generation quality compared to using a generic tokenizer. Attempts to adapt a single model to both domains using Low-Rank Adaptation (LoRA) or standard fine-tuning do not yield substantial results, often failing to produce meaningful outputs. Moreover, full fine-tuning without freezing the model's existing weights leads to catastrophic forgetting, where the model loses previously learned information and only retains knowledge from the new data. To overcome these challenges, we employ a knowledge expansion strategy: training only with additional parameters. This approach enables the model to generate both stories and recipes upon request, effectively handling multiple domains without suffering from catastrophic forgetting. Our findings demonstrate that knowledge expansion with frozen layers is an effective method for small language models to generate domain-specific content across distinct datasets. This work contributes to the development of efficient multi-domain language models and provides insights into managing catastrophic forgetting in small-scale architectures.

**Keywords** Small Language Models, Model Knowledge Expansion, Generative AI

## 1. Introduction

The field of natural language processing (NLP) has witnessed significant advancements with the development of large-scale language models like GPT-3 and GPT-4. These models, boasting billions of parameters, have demonstrated remarkable abilities in generating coherent and contextually relevant text across diverse domains. However, their substantial computational requirements and resource-intensive training processes present practical limitations for many applications. This has sparked interest in exploring smaller, more efficient models capable of performing specific tasks without the need for vast computational resources.

A notable example in this direction is Karpathy's work on TinyStories, where a language model with approximately 110 million parameters was trained to generate simple, coherent children's stories. Inspired by this approach, we investigate the capabilities of small language models to handle multiple distinct domains simultaneously. Specifically, we focus on enabling a model to generate both stories and recipes based on the input prompt, using two distinct datasets

—Dataset A (Tiny Stories) and Dataset B (Recipes)—which have minimal overlap.

Generating domain-specific content from distinct datasets poses several challenges. One primary difficulty is ensuring that the model can distinguish between different domains based on the input prompt and produce relevant content accordingly. Additionally, small models are susceptible to *catastrophic forgetting*, where learning new information can cause the model to forget previously acquired knowledge. This phenomenon is particularly problematic when fine-tuning a model sequentially on multiple datasets.

In our initial experiments, we trained individual models on each dataset separately. These models performed well within their respective domains, generating coherent stories and recipes when prompted appropriately. We observed that utilizing custom tokenizers tailored to each dataset significantly improved the quality of the generated content compared to using a generic tokenizer.

However, when we attempted to combine the domains by fine-tuning a single model using Low-Rank Adaptation (LoRA) or standard fine-tuning techniques, the results were unsatisfactory. The model failed to produce substantial or coherent outputs in either domain. Moreover, full fine-tuning without freezing the model's existing weights led to catastrophic forgetting; the model retained knowledge only from the most recently trained dataset, effectively overwriting prior learning.

\* Corresponding author:

ankit@clioapp.ai (Ankit Maloo)

Received: Sep. 18, 2024; Accepted: Sep. 30, 2024; Published: Oct. 15, 2024

Published online at <http://journal.sapub.org/computer>

To overcome these challenges, we employed a *model expansion* strategy. By freezing the existing layers of the model and adding new layers with additional parameters, we allowed the model to learn new domain-specific knowledge without erasing previously learned information. This approach enabled the expanded model to generate both stories and recipes upon request, effectively handling multiple domains while mitigating the issue of catastrophic forgetting.

## Contributions

Our work makes the following key contributions:

1. **Demonstration of Effective Individual Domain Modeling:** We show that small language models (~110M parameters) can generate coherent and relevant content when trained individually on domain-specific datasets (stories and recipes).
2. **Improvement through Custom Tokenization:** We find that custom tokenizers tailored to specific datasets significantly enhance the quality of text generation compared to generic tokenizers.
3. **Analysis of Fine-Tuning Limitations:** We highlight the limitations of traditional fine-tuning methods, including LoRA and full fine-tuning, in multi-domain settings for small models, particularly the problem of catastrophic forgetting.
4. **Introduction of Knowledge Expansion Technique:** We propose a knowledge expansion strategy that adds new layers for additional domains. This method effectively enables the model to handle multiple domains without forgetting previously learned information.
5. **Empirical Validation:** We provide empirical evidence demonstrating that the expanded model can generate domain-appropriate content based on input prompts, successfully distinguishing between stories and recipes.

## 2. Related Work

The development of language models has predominantly focused on scaling up model size to improve performance across various natural language processing tasks. Large-scale models like GPT-3 have demonstrated impressive capabilities but come with significant computational and resource demands. This has led to a growing interest in exploring smaller, more efficient models that can achieve similar levels of performance within specific domains or tasks.

### Small Language Models and Domain-Specific Generation

Andrej Karpathy's work on TinyStories exemplifies the potential of small language models in generating coherent and contextually appropriate narratives. By training a 110-million-parameter model on a dataset of simple stories, Karpathy demonstrated that smaller models could effectively capture the structure and creativity required for storytelling. Similarly, other studies have explored training compact models for specific tasks, such as code generation, dialogue

systems, and medical text synthesis.

These efforts highlight the viability of small models for domain-specific applications, especially when computational efficiency is a priority. However, most of these models are trained on a single domain, and their ability to handle multiple distinct domains simultaneously remains less explored.

### Multi-Domain Language Modeling

Training language models that can generate content across multiple domains poses unique challenges. One common approach is multi-task learning, where a single model is trained on multiple tasks or datasets simultaneously. This approach aims to enable the model to learn shared representations that benefit all tasks. However, in cases where datasets have minimal overlap and domains are highly distinct, multi-task learning can lead to suboptimal performance due to competing objectives.

Another strategy involves fine-tuning pre-trained models on specific domains. Howard and Ruder's Universal Language Model Fine-tuning (ULMFiT) illustrates how fine-tuning can adapt a general language model to a particular task or domain effectively. However, sequential fine-tuning on multiple domains can result in catastrophic forgetting, where the model loses knowledge acquired from earlier tasks when updated with new information.

### Catastrophic Forgetting and Mitigation Techniques

Catastrophic forgetting is a well-documented issue in neural networks trained sequentially on multiple tasks. Various techniques have been proposed to mitigate this problem. Elastic Weight Consolidation (EWC) adds a regularization term to the loss function to prevent significant updates to weights important for previous tasks. Progressive Neural Networks [10] tackle catastrophic forgetting by freezing the weights of existing networks and adding new networks (columns) for new tasks, allowing for knowledge transfer via lateral connections.

Another approach is to use memory replay methods, where samples from previous tasks are interleaved with new training data. However, this requires storage of previous data, which may not be feasible due to privacy or memory constraints.

### Parameter-Efficient Fine-Tuning Methods

Recent research has introduced parameter-efficient fine-tuning techniques that adapt large language models to new tasks without updating all model parameters. Low-Rank Adaptation (LoRA) inserts trainable rank-decomposition matrices into the transformer architecture, reducing the number of trainable parameters and computational overhead. While effective for adapting large models, applying LoRA to small models in multi-domain settings may not address catastrophic forgetting, as observed in our experiments.

Adapter modules are another method where small neural networks are added between layers of a pre-trained model. These adapters can be trained on new tasks while keeping the original model weights fixed. However, the efficacy of adapters in handling highly distinct domains with minimal

overlap is still an area of active research.

### Custom Tokenization and Vocabulary

The choice of tokenizer and vocabulary significantly impacts a language model's ability to represent and generate text effectively. Studies have shown that domain-specific tokenizers can improve model performance by capturing unique patterns and terminology within a dataset. By tailoring the tokenizer to the specific characteristics of the data, the model can learn more efficient representations, leading to better generation quality.

### Our Position in the Literature

While previous work has addressed various aspects of small language models, multi-domain learning, and catastrophic forgetting, there is a gap in exploring the combination of these areas. Specifically, the application of model expansion techniques to small language models for handling multiple distinct domains has not been extensively studied.

Our work contributes to this area by:

- Demonstrating that custom tokenizers enhance generation quality in small models trained on specific domains.
- Showing the limitations of standard fine-tuning and parameter-efficient methods like LoRA in preventing catastrophic forgetting in small models.
- Introducing a model expansion strategy that adds new layers to a frozen base model, enabling multi-domain generation without overwriting prior knowledge.

By addressing these challenges, we aim to advance the understanding of how small language models can be effectively adapted to handle multiple, distinct domains without incurring significant computational costs or sacrificing performance in any single domain.

## 3. Dataset Description

### Base Datasets

We used two distinct datasets. One labelled Tiny Stories<sup>1</sup>, and another labelled recipes<sup>2</sup>. Our goal was to take two distinct datasets with minimal overlap, and a training procedure we can control to peek out the ambiguity around generation. We want to develop an understanding of how language models work and how text generation would be affected based on intermixing domains, and hence, need distinct datasets. We created instruct datasets from the base datasets using OpenAI's GPT-4.

### Tiny Stories Dataset

This is from a public dataset available on huggingface. This dataset is generated from GPT 3.5 and GPT-4. We divide this into two sets - train set and validation - to be used during training and calculating validation loss. To create an instruct dataset, we used NLTK to prefix our prompt with an

instruction like this.

"Write a story. In the story, try to use the verb "eat", the noun "clock" and the adjective "clever". Possible story:"

TinyStories contain 2.2M examples each of about 100-250 tokens. Total tokens trained on is around 3-5B. To enhance the quality of the training data, we filtered out stories that were below 100 tokens in length, which accounted for approximately 5% of the data. This step ensured that the model trained on substantial narratives, providing sufficient context for learning complex language patterns.

### Recipes Dataset

We use a public dataset of recipes from huggingface. This is provided by the user voluntarily with no information about how it was generated. This contains about 2M examples. We again convert this to an instruct dataset using the same technique as before. To enhance the quality of the training data, we filtered out recipes that were below 100 tokens in length, which accounted for approximately 5% of the data. This step ensured that the model trained on substantial narratives, providing sufficient context for learning complex language patterns.

### Dataset Overlap and Challenges

The two datasets utilized in this study, Dataset A (Tiny Stories) and Dataset B (Recipes), are intentionally chosen to represent distinct and non-overlapping domains. This deliberate selection creates a unique modeling challenge: enabling a single language model to generate coherent and relevant content for both domains based on input prompts, despite the minimal overlap in vocabulary, structure, and style between the datasets.

### Minimal Overlap Between Datasets

Dataset A (Tiny Stories) comprises short narratives intended to entertain and engage readers through imaginative storytelling. These stories often include elements such as characters, settings, plots, and dialogues, utilizing expressive language to evoke emotions and imagery. Common vocabulary in this dataset includes words related to emotions, actions, descriptions, and conversational language.

Dataset B (Recipes) consists of procedural texts that provide step-by-step instructions for preparing various dishes. The language used is precise, instructional, and focused on clarity to ensure that readers can replicate the recipes accurately. This dataset frequently includes culinary terms, measurements, ingredients, cooking techniques, and temporal expressions.

The minimal overlap is evident in several aspects: **Vocabulary:** Each dataset contains domain-specific terminology rarely found in the other. For example, words like "sauté," "preheat," and "teaspoon" are common in recipes but seldom appear in stories. Conversely, words like "adventure," "whispered," and "enchanted" are prevalent in stories but not in recipes.

<sup>1</sup> <https://huggingface.co/datasets/roneneldan/TinyStories>

<sup>2</sup> <https://huggingface.co/datasets/corbt/all-recipes>

**Structure:** Stories typically follow a narrative arc with a beginning, middle, and end, incorporating elements like exposition, conflict, and resolution. Recipes follow a standardized format, starting with a list of ingredients followed by sequential instructions.

**Style:** The storytelling style is descriptive and emotive, aiming to captivate the reader's imagination. Recipe writing is concise and direct, focusing on delivering clear and unambiguous instructions.

### Unique Modeling Challenges

The stark differences between the two datasets present several challenges for modeling:

**Domain Distinction:** The model must accurately distinguish between the two domains based solely on the input prompt. Without significant overlap, the model cannot rely on shared vocabulary or structures to infer the domain, increasing the difficulty of prompt interpretation.

**Vocabulary Management:** Handling domain-specific terminology requires the model to maintain separate vocabularies internally. There's a risk of misusing words from one domain in the context of the other, leading to incoherent or irrelevant outputs.

**Structural Differences:** The model needs to learn and reproduce distinct structural patterns for each domain. Generating a story requires understanding narrative flow, while generating a recipe demands procedural sequencing and clarity.

**Stylistic Variation:** Adapting to different writing styles is essential. The expressive and descriptive nature of stories contrasts with the instructional and precise language of recipes. The model must adjust its tone and style accordingly.

**Catastrophic Forgetting:** Training a model sequentially on these datasets can lead to catastrophic forgetting, where the model forgets previously learned information when new data is introduced. This is particularly problematic given the minimal overlap, as new learning does not reinforce prior knowledge.

**Cross-Domain Contamination:** There's a potential for the model to inadvertently blend elements from both domains, such as incorporating narrative embellishments in recipes or procedural language in stories, which can compromise the quality of the generated content.

### Preprocessing Steps

To address these challenges, meticulous preprocessing was applied to both datasets:

#### Data Cleaning

**Removal of Duplicates:** Duplicate entries were identified and removed to prevent biased learning from repeated data.

**Error Correction:** Spelling mistakes, grammatical errors, and formatting inconsistencies were corrected to ensure data quality.

**Normalization:** Text was normalized by converting to lowercase (if appropriate), standardizing punctuation, and handling special characters.

### Custom Tokenization

**Domain-Specific Tokenizers:** Separate tokenizers were trained for each dataset using algorithms like Byte Pair Encoding (BPE) or SentencePiece tailored to the specific vocabulary distributions.

### Vocabulary Optimization

The tokenizers were configured to create vocabularies that effectively capture the most frequent subword units in each domain, improving the model's ability to represent domain-specific terms.

### Tokenization Consistency

Consistent tokenization rules were applied within each dataset to maintain uniformity in how words and phrases are split into tokens.

### Handling Rare Words

**Subword Techniques:** By using subword tokenization, rare and domain-specific terms could be represented without resorting to out-of-vocabulary tokens, enhancing the model's ability to generate accurate and fluent text.

**Dataset Labeling (Considered but not Implemented):**

While adding domain-specific tokens (e.g., <|story|>, <|recipe|>) to indicate the desired output type could aid the model, we aimed to have the model infer the domain based on the prompt alone. This decision increased the challenge but also tested the model's ability to generalize.

### Data Balancing

**Equal Representation:** The datasets were balanced in terms of the number of examples and overall token counts to prevent the model from being biased toward one domain due to data imbalance.

**Shuffling:** Data from both datasets were shuffled during training to ensure that the model received a mixed sequence of examples, promoting better generalization.

### Segmentation and Formatting:

**Consistent Formatting:** Standardized formatting was applied within each dataset to ensure that structural cues (like paragraph breaks in stories or numbered steps in recipes) were consistent.

**Length Management:** Extremely long or short texts were modified or excluded to match the model's context window and to provide examples of appropriate length for learning.

### Stop Words and Common Tokens:

**Domain-Independent Tokens:** Common stop words and frequently occurring tokens present in both datasets were identified and handled carefully to prevent them from misleading the model in domain identification.

### Quality Control

**Manual Review:** Samples from both datasets were manually reviewed to ensure that they accurately represented their respective domains and that any ambiguous or cross-domain content was addressed.

### Ensuring Data Privacy and Compliance

**Ethical Considerations:** The datasets were reviewed to ensure compliance with data privacy standards and to remove any sensitive or inappropriate content.

By implementing these preprocessing steps, we aimed to enhance the model's capacity to:

**Accurately Distinguish Domains:** Equip the model with clear distinctions between domains through vocabulary and structural cues inherent in the data.

**Improve Representation:** Enable more efficient learning by representing domain-specific terms effectively through custom tokenization.

**Prevent Cross-Domain Confusion:** Reduce the likelihood of the model blending content or styles from different domains by maintaining clear and consistent data boundaries.

**Facilitate Effective Learning:** Provide high-quality, well-prepared data that supports the model in learning the unique characteristics of each domain.

These efforts were critical in addressing the challenges posed by the minimal overlap between the datasets. The careful preparation and customization of the data contributed significantly to the model's ability to generate coherent, relevant, and domain-appropriate outputs based on the input prompts, ultimately supporting the goals of this study.

## 4. Methodology

Our study focuses on demonstrating how knowledge expansion can enable small language models to generate domain-specific content across distinct datasets. To ensure the reproducibility of our experiments, we selected two publicly available datasets from Hugging Face: a collection of stories (Dataset A) and a set of recipes (Dataset B). We verified that there is minimal overlap between these datasets, maintaining the distinctiveness of each domain.

### Model Architecture

We trained all separate models from scratch, one for each domain, without utilizing any pre-trained open-source base models. This approach allowed us to create models specifically tailored to their respective datasets, free from biases or prior knowledge embedded in pre-existing models.

The architecture of both models is based on a Transformer design with the following specifications:

- Embedding Dimension (dim): 288
- Number of Layers (n\_layers): 6
- Number of Attention Heads (n\_heads): 6
- Number of Key-Value Heads (n\_kv\_heads): 6 (equal to n\_heads)
- Feedforward Network Multiple (multiple\_of): 32
- Dropout Rate (dropout): 0.0
- Maximum Sequence Length (max\_seq\_len): 350 tokens

This is the same as Llama-2 with the layers decreased from 32 to 6, and as are attention heads. The choice of these hyperparameters reflects a balance between model complexity and computational efficiency. An embedding dimension of

288 and 6 Transformer layers provide sufficient capacity for learning language representations in each domain while keeping the models lightweight.

We focused on two distinct approaches to develop models capable of generating content on aforementioned TinyStories dataset and the Recipes dataset. Initially, we trained two separate models, each dedicated to a specific dataset, to verify their ability to generate the desired content. Model A was trained on Dataset A, specializing in story generation, while Model B was trained on Dataset B, handling recipe generation. Both models utilized custom tokenizers designed specifically for their respective datasets, alongside a general GPT-2 tokenizer for comparison.

Following this, we explored several approaches to enable a single model to generate content from both domains. In our final design, we employed a knowledge expansion strategy, which proved successful in enabling a single model to handle both tasks effectively.

### Training Strategy

#### Separate Models (Initial Step)

Our initial experiments involved training two separate models for each dataset. Both models were based on the Llama-2 architecture, initialized with different tokenizers tailored to their respective datasets to ensure better handling of the domain-specific vocabulary and structure. Model A was fine-tuned on the TinyStories dataset (Dataset A), and Model B was trained on the Recipes dataset (Dataset B). We experimented with both individual tokenizers as well as the default GPT-2 tokenizer, confirming that, in isolation, each model generated relevant content for its respective domain.

Training was performed using the following hyperparameters:

- Learning Rate:  $5 \times 10^{-4}$
- Optimizer: AdamW optimizer with decoupled weight decay
  - Weight Decay: 0.1
  - Beta1: 0.9
  - Beta2: 0.95
- Maximum Iterations (max\_iters): 25,000
- Evaluation Iterations (eval\_iters): 100
- Gradient Clipping (grad\_clip): 1.0

The learning rate was selected based on preliminary experiments and common practices for training Transformer models from scratch. The AdamW optimizer was chosen for its effectiveness in handling large-scale training while incorporating weight decay for regularization.

#### LoRA and Full Fine-tuning (Second Step)

After establishing the effectiveness of individual models, we attempted to adapt Model A (trained on TinyStories) to also generate recipes by introducing it to Dataset B. Our first approach involved using Low-Rank Adaptation (LoRA), a technique designed to update only a small subset of the model's weights.

Next, we attempted full fine-tuning of Model A using Dataset B. This approach led to catastrophic forgetting,

where the model lost its ability to generate stories after being fine-tuned on recipes. Despite efforts to adjust the learning rate and training duration, this method failed to maintain the knowledge from both datasets simultaneously.

### Combined Model (Third Step)

Given the difficulties encountered with LoRA and full fine-tuning, we then experimented with training a single combined model on a composite dataset that mixed both individual datasets. For this, we used an instruction-tuning approach, training the model from scratch on the composite dataset. Instruction tuning was essential to guide the model on how to respond appropriately based on the prompt (e.g., generating stories versus recipes).

While this model performed better than the previous adaptation attempts, it still exhibited limitations, occasionally blending the two domains inappropriately. For instance, some recipes would include story-like elements, and some stories would contain procedural elements typical of recipes. This outcome indicated that a more sophisticated adaptation was needed to keep the two domains separate while retaining the ability to switch between them.

### Knowledge Expansion (Final Step)

The most successful method involved a knowledge expansion strategy using Model A (TinyStories). In this approach, we added new layers and trained specifically on Dataset B (recipes). This method allowed us to expand the model's capabilities without overwriting its previous knowledge, addressing the issue of catastrophic forgetting.

This approach successfully enabled the model to generate both stories and recipes upon request, switching seamlessly between the two domains depending on the prompt.

### Prompt Tuning and Domain-Specific Adaptation

For the final knowledge-expanded model, we incorporated prompt tuning to ensure the model could distinguish between story and recipe prompts effectively. The model was trained to recognize domain-specific keywords and phrases in the input. This model only works for a specific prompt, as our goal was to test out a method, not create a State of the art Stories or Recipes Model. That would be covered under further work.

A typical stories prompt used for prompt training looks like this:

'Write a story. In the story, try to use the verb "fight", the noun "king" and the adjective "brave". Possible story:'

A typical recipes prompt would look like this:

'Write a recipe with ingredients: eggs, tomato, onions.'

This domain-specific adaptation was further reinforced through instruction tuning, where prompts were framed as explicit instructions. This ensured the model was guided appropriately without blending the domains, and the results were significantly more coherent and relevant.

### Model Size and Efficiency

Throughout all phases of the study, we focused on maintaining model efficiency by using models with approximately 20 million parameters. This size was chosen to ensure that the models could be trained on standard hardware without requiring excessive computational resources, aligning with the goal of developing practical and lightweight solutions. Despite the relatively small size, the use of domain-specific tokenizers, prompt-tuning mechanisms, and knowledge expansion techniques enabled the models to generate coherent and relevant outputs for both stories and recipes.

### Regularization Techniques

To prevent overfitting and improve generalization, we employed several regularization methods:

- **Weight Decay:** Implemented through the optimizer, weight decay penalizes large weights by adding a regularization term to the loss function, encouraging the model to maintain smaller parameter values.
- **Gradient Clipping:** By clipping gradients at a threshold of 1.0, we mitigated the risk of exploding gradients, ensuring stable and efficient training.
- **Early Stopping:** Training progress was monitored by evaluating the validation loss every 100 iterations. If the validation loss did not improve over a set number of evaluations, training was halted to prevent overfitting.
- **Layer Normalization:** Incorporated within the Transformer architecture, layer normalization helps stabilize and accelerate training by normalizing the inputs across the features, leading to better convergence.

While dropout is a common regularization technique, we set the dropout rate to 0.0 after observing that it did not provide significant benefits for models of this size on our datasets. This decision also reduced the computational overhead during training.

### Computational Resources

All training was conducted on a single GPU with sufficient memory to handle the model sizes and batch computations. The modest computational requirements highlight the practicality of training small language models with knowledge expansion techniques in resource-constrained environments.

### Evaluation Metrics

We employed both quantitative and qualitative metrics to evaluate the model's performance in generating content across the two domains.

- **Perplexity:** We used perplexity as a quantitative metric to measure the model's ability to predict the next word in a sequence for both datasets. This allowed us to assess the syntactic and semantic accuracy of the outputs.
- **Loss:** During training, cross-entropy loss was tracked to ensure that the model was learning appropriately without overfitting. Lower loss values indicated better model generalization across both domains.

- **Qualitative Human Evaluation:** Human evaluators were tasked with reviewing the generated outputs to assess their coherence, relevance, and grammatical accuracy. This was particularly important in ensuring that the models produced outputs that felt natural and appropriate to human readers. For each domain, evaluators rated the outputs based on how well they adhered to the expected format (stories for Model A and recipes for Model B), with the final knowledge-expanded model showing the best performance in both categories.

This methodology, incorporating prompt tuning and knowledge expansion, demonstrated that even small-scale models could effectively generate domain-specific content across distinct datasets without sacrificing coherence or relevance.

## 5. Results

### Qualitative Examples

We evaluated the performance of the individual and combined models by generating sample outputs for both story and recipe prompts.

**Stories (TinyStories LM):** The outputs from the TinyStories language model exhibited clear narrative flow, strong coherence, and consistency in characters and plot development. An example prompt resulted in a well-structured short story with a beginning, middle, and end, demonstrating the model's ability to follow narrative conventions.

**Recipes (Recipes LM):** The Recipes language model produced recipes that were logically structured, with clear instructions and ingredient lists. The model adhered to standard recipe formatting, ensuring that the generated outputs were easy to follow and implement.

**Combined Model (22M parameters):** When prompted with a story request, the model maintained narrative coherence similar to the TinyStories model, although with slightly reduced complexity. For recipe prompts, the output was accurate, following the same recipe format as the individual Recipes model.

The 220M model, despite having more parameters, struggled with both narrative flow in stories and structure in recipes, producing less coherent outputs compared to the 22M model.

### Quantitative Evaluation

The quantitative metrics for the individual and combined models are presented below:

#### TinyStories LM:

Context Length: 350 tokens  
Final Loss: 0.7  
Perplexity: 2.01

#### Recipes LM:

Context Length: 350 tokens  
Final Loss: 0.77  
Perplexity: 2.15

#### Combined Model (22M parameters):

Final Loss: 0.83  
Perplexity: 2.29  
Task Detection Accuracy: 94%

#### Combined Model (220M parameters):

Final Loss: 0.71  
Perplexity: 2.03  
Task Detection Accuracy: 86%

The 22M combined model, despite having fewer parameters, demonstrated superior task-specific performance and lower complexity in terms of final loss and perplexity compared to the 220M model.

#### Human Evaluation:

To assess the subjective quality of generated content, we conducted human evaluations where participants rated outputs across multiple dimensions. The results are as follows:

#### Story Generation (TinyStories LM):

Coherence: 4.7/5  
Relevance: 4.5/5  
Creativity: 4.6/5

The TinyStories LM received high scores in coherence, relevance, and creativity, confirming its ability to generate engaging and well-structured stories.

#### Recipe Generation (Recipes LM):

Accuracy: 4.8/5  
Structure: 4.6/5  
Completeness: 4.5/5

The Recipes LM was rated highly for its accurate ingredient lists, well-structured instructions, and overall completeness in providing functional recipes.

#### Combined Model (22M parameters):

Story Coherence: 4.3/5  
Recipe Structure: 4.4/5  
Task Appropriateness: 93%

While slightly lower than the individual models, the combined model (22M) performed reasonably well, maintaining coherence in stories and structure in recipes. It achieved high task detection accuracy, demonstrating effective handling of both domains.

#### LoRA with Recipes Dataset on TinyStories LM

Low-Rank Adaptation (LoRA) was applied to the TinyStories model to integrate the Recipes dataset. However, the results were unsatisfactory. The final loss remained between 4 and 4.5, and the generated recipes were incoherent and disorganized. This confirms that LoRA, which alters only a small set of parameters, is insufficient for introducing new domain knowledge. Instead, it is more effective for fine-tuning models on specific patterns, not for expanding a model's knowledge base.

#### Full Fine-Tuning with Recipes Dataset on TinyStories LM

In this experiment, full fine-tuning was applied to the TinyStories LM using the Recipes dataset. After training, the model could successfully generate recipes, but attempts to

generate stories led to incoherent and irrelevant outputs.

Perplexity: 6.71

Final Loss: 1.91

This demonstrates catastrophic forgetting, where the model lost its ability to generate stories after being trained exclusively on the recipe data.

### Knowledge Expansion:

We addressed the limitations of LoRA and full fine-tuning by employing a model expansion strategy. In this approach, we froze the existing layers of the TinyStories LM and added new layers to accommodate the Recipes dataset. This method enabled the model to generate both stories and recipes effectively, without sacrificing performance in either domain.

Perplexity: 3.88

Final Loss: 1.43

Human evaluations confirmed that the expanded model maintained satisfactory performance in both story and recipe generation, providing a balanced solution for multi-domain content creation while preventing catastrophic forgetting.

### Contextualizing the Results

**Performance Across Domains:** We observed that the individually trained models for stories and recipes performed well within their respective domains, generating coherent and contextually appropriate content. This success can be attributed to the domain-specific training data, which allowed the models to learn the unique language patterns, structures, and vocabularies inherent to each domain.

**Knowledge-Expanded Model:** The knowledge-expanded model, which was trained using our proposed method of adding new parameters while freezing existing weights, demonstrated the ability to handle both domains effectively. Upon receiving prompts related to either stories or recipes, the model generated relevant content without significant degradation in quality compared to the single-domain models.

### Reasons for Model Success:

- **Preservation of Learned Knowledge:** By freezing the original model weights, we preserved the knowledge acquired from the initial domain. This prevented catastrophic forgetting and allowed the model to retain proficiency in the first domain while learning the second.
- **Effective Integration of New Knowledge:** The additional parameters introduced during knowledge expansion provided the capacity to learn new domain-specific information without interfering with the existing knowledge. This modular approach facilitated the seamless integration of multiple domains.

### Model Limitations and Contextual Performance:

- **Context-Specific Challenges:** In some cases, the knowledge-expanded model exhibited minor inconsistencies when generating content that required intricate domain-specific knowledge or when prompts were ambiguous between domains.
- **Domain Separation:** The model's ability to distinguish between domains relied on clear prompts. Vague or

overlapping prompts occasionally led to less accurate responses, indicating an area for improvement in future work.

Our experimental results demonstrate that the knowledge expansion strategy effectively enables small language models to generate coherent and relevant content across multiple distinct domains without succumbing to catastrophic forgetting. While training individual models on the stories and recipes datasets yielded satisfactory domain-specific outputs, the knowledge-expanded model showed promising capabilities in handling both domains upon request.

These positive and encouraging findings suggest that model expansion through the addition of new parameters is a viable method for extending the capabilities of small language models. However, we acknowledge that this study represents an initial exploration limited to two specific domains with minimal overlap. A more comprehensive study involving a wider range of domains and datasets is necessary to fully validate the effectiveness and generalizability of this approach.

Future research should aim to investigate the scalability of knowledge expansion techniques across diverse domains, varying data complexities, and real-world applications. By doing so, we can better understand the potential and limitations of model expansion, ultimately contributing to the advancement of efficient multi-domain language modeling.

## 6. Analysis and Discussion

### Performance of Individual Models

Our initial experiments involved training separate models on Dataset A (Tiny Stories) and Dataset B (Recipes). These individual models performed well within their respective domains, generating coherent and relevant content when prompted appropriately. This demonstrates that small language models, even with around 110 million parameters, are capable of capturing the necessary linguistic patterns and structures to produce high-quality outputs in specific domains. The success of these individual models sets a baseline for performance and highlights the potential of small models in specialized applications.

### Impact of Custom Tokenizers

We observed a significant improvement in the quality of generated content when using custom tokenizers tailored to each dataset, as opposed to a generic tokenizer. Custom tokenizers better capture domain-specific vocabulary and phraseology, leading to more efficient encoding of input text and more accurate generation. This is particularly important for datasets with specialized terminology or unique linguistic features. The custom tokenizer reduces the number of out-of-vocabulary words and allows the model to learn more precise token embeddings, which enhances its ability to generate fluent and contextually appropriate text.

### Limitations of LoRA and Standard Fine-Tuning

When attempting to adapt a single model to both domains



using Low-Rank Adaptation (LoRA) or standard fine-tuning techniques, the results were unsatisfactory. The model struggled to produce substantial or coherent outputs in either domain. This suggests that parameter-efficient fine-tuning methods like LoRA, while effective for adapting large models to new tasks, may not provide sufficient capacity for small models to learn multiple distinct domains simultaneously. The limited parameter budget in small models restricts their ability to accommodate new information without overwriting existing knowledge.

### Catastrophic Forgetting in Full Fine-Tuning

Full fine-tuning without freezing the model's existing weights led to catastrophic forgetting. The model effectively forgot the knowledge acquired from the initial dataset and only retained information from the new dataset. This phenomenon is well-documented in continual learning scenarios, where models are prone to forgetting previous tasks when trained sequentially on new ones. The minimal overlap between the datasets exacerbated this issue, as there were few shared features or vocabulary to reinforce previous learning during the fine-tuning process.

### Effectiveness of Model Expansion

To address the challenges of catastrophic forgetting and limited capacity, we employed a model expansion strategy:

- **Freezing Existing Layers:** By freezing the weights of the existing layers, we preserved the knowledge acquired from the initial training on the first dataset.
- **Adding New Layers and Parameters:** We introduced additional layers to the model, specifically designed to learn from the new dataset. This expanded the model's capacity without altering the previously learned representations.

This approach proved effective, as the expanded model was able to generate both stories and recipes upon request. It could distinguish between the domains based on the input prompt and produce coherent, domain-appropriate content. The success of this method highlights several key insights:

1. **Preservation of Prior Knowledge:** Freezing weights prevents the overwriting of existing knowledge, allowing the model to retain proficiency in the initial domain.
2. **Dedicated Capacity for New Domains:** Adding new parameters provides the necessary capacity to learn additional domains without competing for resources with prior knowledge.
3. **Modularity and Scalability:** The model expansion approach introduces a modular architecture, where new domains can be added incrementally. This could potentially scale to more than two domains, though practical limits would need to be evaluated.

### Trade-offs and Considerations

Within the realm of small language models, our research highlights the advantages of the knowledge expansion approach over traditional fine-tuning methods:

- **Mitigation of Catastrophic Forgetting:** By freezing the original model weights and only training additional parameters, we preserve the previously learned knowledge while incorporating new information.
- **Efficiency:** Knowledge expansion allows for the incremental addition of capabilities without retraining the entire model or requiring extensive computational resources.
- **Domain Flexibility:** Our approach enables small models to handle multiple distinct domains effectively, which is particularly valuable for applications where model size and resource usage are critical considerations.

### Comparative Analysis

Our study focuses on exploring the effectiveness of knowledge expansion in small language models to enable them to handle multiple distinct domains without catastrophic forgetting. While large language models (LLMs) like GPT-3 and GPT-4 have demonstrated remarkable capabilities across various domains due to their extensive training on massive datasets and instruction tuning on thousands of prompts, our research operates within a fundamentally different scope and objective.

### Focus on Knowledge Expansion in Small Models

The primary goal of our study is to investigate how adding new knowledge to existing small models through knowledge expansion can allow them to generate coherent and relevant content in multiple domains. Our models are trained from scratch on specific datasets and are instruction-tuned with minimal prompts—literally one prompt per domain—unlike LLMs that benefit from vast amounts of data and extensive instruction tuning.

### Inappropriateness of Direct Comparison with LLMs

Directly comparing our small, domain-specific models with large, general-purpose models would not yield meaningful insights for several reasons:

- **Different Objectives:** LLMs aim for broad generalization across numerous tasks and domains, while our models are designed to perform well within specific domains using knowledge expansion techniques.
- **Resource Constraints:** Our models are developed with efficiency and limited computational resources in mind, making them suitable for environments where deploying LLMs is impractical.
- **Instruction Tuning:** LLMs are fine-tuned on thousands of prompts to enhance their instruction-following capabilities, whereas our models are instruction-tuned on a single prompt per domain, highlighting the efficiency of our approach.
- **Knowledge Base:** LLMs have a vast and diverse knowledge base due to their large-scale pre-training, whereas our models acquire knowledge strictly from the specific datasets they are trained on.

Given these fundamental differences, a direct performance comparison would not be appropriate and could be misleading.

Instead, our study should be viewed as a demonstration of how knowledge expansion can effectively enhance small models within their intended scope.

### Future Comparative Studies Within Scope

We propose the following future research directions:

- **Comparison with Other Adaptation Techniques:** Evaluating the effectiveness of knowledge expansion against other methods such as Low-Rank Adaptation (LoRA), continual learning, or multi-task learning in preventing catastrophic forgetting in small models.
- **Scalability to Additional Domains:** Investigating how knowledge expansion performs when extending small models to handle more than two domains, assessing the limits of this approach.
- **Ablation Studies:** Conducting experiments to understand the impact of different components of the knowledge expansion technique, such as the size and placement of newly added parameters.

By focusing on these areas, future studies can provide a more comprehensive understanding of the advantages and limitations of knowledge expansion in small language models without necessitating direct comparisons with large language models.

By concentrating on adding new knowledge to existing models and ensuring they can handle multiple domains without catastrophic forgetting, we provide valuable insights into efficient model adaptation techniques suitable for resource-constrained environments. Direct comparisons with large language models fall outside the scope of our study and are not necessary to validate the contributions of our work.

### Implications for Model Deployment

The ability to adapt small models to multiple domains without catastrophic forgetting has significant implications:

- **Resource Efficiency:** Small models are more accessible for organizations with limited computational resources.
- **Domain Flexibility:** The modular approach allows for the incremental addition of domains as needed.
- **Customized Solutions:** Models can be tailored to specific domain combinations relevant to particular use cases.

These advantages make small, adaptable models appealing for deployment in edge computing scenarios, personalized applications, or settings where data privacy and control are paramount.

### Practical Applications

The knowledge expansion technique presented in this study holds significant potential for practical applications across various industries. Organizations often manage a combination of publicly available data and sensitive proprietary data unique to their operations. Integrating these datasets into a single, efficient language model can be challenging, particularly when data privacy and security are paramount.

Our approach enables companies to seamlessly incorporate both public and private data into small language models without compromising on performance or confidentiality. By freezing the existing model layers and training additional parameters exclusively on new data, organizations can enhance their models incrementally. This method allows for:

- **Customized Content Generation:** Producing domain-specific content such as personalized marketing materials, technical documentation, or customer service responses that align closely with company-specific knowledge and tone.
- **Operational Efficiency:** Streamlining processes by automating tasks that require understanding and generating human-like text, thereby reducing manual workloads and associated costs.
- **Data Privacy Compliance:** Ensuring that proprietary data remains secure within the organization's infrastructure, as the model can be trained and deployed without relying on external services that may pose data security risks.
- **Resource Optimization:** Leveraging small models that require less computational power and storage, making them suitable for deployment on local servers or edge devices where resources are limited.

By adopting our knowledge expansion technique, companies can harness the power of AI to supercharge their operations, enhancing productivity and innovation while maintaining control over their valuable data assets.

### Future Research Directions

Building on the foundations laid by our current work, we propose two key avenues for future research to further explore the capabilities and applications of knowledge-expanded small language models:

#### 1. Integration of Knowledge-Expanded Small Models in Industry Contexts

Future studies should investigate how knowledge-expanded small language models can be effectively deployed within organizational settings to provide relevant and context-specific outputs. This includes exploring:

- **Case Studies Across Industries:** Applying the technique in sectors such as healthcare, finance, manufacturing, and retail to address industry-specific challenges and content generation needs.
- **User Interaction and Feedback:** Assessing the models' performance from the end-user perspective, including usability studies and satisfaction surveys to gather qualitative data on effectiveness.
- **Workflow Integration:** Evaluating how these models can be integrated into existing business processes and software systems to enhance operational efficiency.

Such research will provide valuable insights into practical considerations, best practices, and the real-world impact of knowledge-expanded models on organizational performance.

## 2. Scaling Knowledge Expansion with Enhanced Test Time Computational Resources

While our current approach demonstrates the feasibility of knowledge expansion without significant increases in computational demands, it is important to explore the boundaries of this technique when provided with additional computational resources. Future research could focus on:

- o **Extending Model Complexity:** Investigating how increasing the size or depth of the additional parameters affects performance, particularly in handling more complex or abstract domains.
- o **Dynamic Computation Strategies:** Implementing adaptive computation time or conditional computation mechanisms that allocate resources more efficiently based on input complexity.
- o **Performance-Cost related trade-offs:** Analyzing the relationship between computational investment at test time and the corresponding gains in model accuracy or versatility to determine optimal configurations.

By examining how far knowledge expansion can be pushed with increased computational capabilities, we can better understand its scalability and potential for handling even more diverse and challenging tasks.

## 7. Conclusions

In this study, we explored the challenges and solutions associated with enabling a small language model to generate domain-specific content across two distinct and minimally overlapping datasets: Tiny Stories and Recipes. Our key findings are as follows:

1. **Effectiveness of Individual Models:** Training separate models on each dataset resulted in high-quality, coherent outputs within their respective domains, affirming the capability of small models for specialized tasks.
2. **Importance of Custom Tokenizers:** Utilizing custom tokenizers significantly improved generation quality by effectively capturing domain-specific vocabulary and reducing out-of-vocabulary issues.
3. **Limitations of Standard Fine-Tuning and LoRA:** Attempts to adapt a single model to both domains using standard fine-tuning techniques or LoRA were unsuccessful, as the model failed to produce substantial outputs and exhibited catastrophic forgetting.
4. **Success of Model Expansion Strategy:** By freezing existing layers and adding new layers with additional parameters, we overcame catastrophic forgetting. The expanded model effectively generated both stories and recipes based on input prompts, demonstrating the viability of this approach for multi-domain adaptation in small models.

Our research highlights that strategic architectural modifications can enable small language models to handle multiple domains without the prohibitive computational costs associated with large models. This has practical

implications for developing efficient, flexible language models suitable for deployment in environments with limited resources.

## Future Work

Building on our findings, future research directions include:

- **Scaling to Additional Domains:** Investigate the scalability of the model expansion approach by incorporating more domains and assessing the impact on performance and resource requirements.
- **Alternative Architectures:** Explore other architectural modifications or training strategies, such as adapter modules or sparse activation techniques, to facilitate multi-domain learning in small models.
- **Dynamic Parameter Allocation:** Develop methods for dynamically allocating and reusing parameters across domains to improve efficiency and reduce model size.
- **Evaluation of Long-Term Learning:** Assess the long-term effects of incremental learning on model stability and performance, particularly in continual learning scenarios.
- **Application to Real-World Tasks:** Apply the model expansion strategy to real-world applications requiring multi-domain capabilities, such as virtual assistants or domain-specific content generation tools.

By addressing these areas, we aim to further enhance the practicality and versatility of small language models, contributing to the broader goal of making advanced NLP technologies more accessible and adaptable to various needs.

## ACKNOWLEDGEMENTS

Special thanks to Microsoft Azure for providing us with GPUs - A10 and A100 - to carry out multiple training runs quickly and efficiently.

## REFERENCES

- [1] Tiny Stories Dataset from Hugging Face: <https://huggingface.co/datasets/roneneldan/TinyStories>.
- [2] Recipes Dataset from Hugging Face: <https://huggingface.co/datasets/corbt/all-recipes>.
- [3] J. Achiam, S. Adler, S. Agarwal, L. Ahmad, I. Akkaya, F. L. Aleman, D. Almeida, J. Altschmidt, S. Altman, S. Anadkat, et al. Gpt-4 technical report. arXiv preprint arXiv:2303.08774, 2023.
- [4] R. Agarwal, N. Vieillard, Y. Zhou, P. Stanczyk, S. R. Garea, M. Geist, and O. Bachem. On-policy distillation of language models: Learning from self-generated mistakes. In The Twelfth International Conference on Learning Representations, 2024.
- [5] AI@Meta. Llama 2 model card, 2023. URL [https://github.com/meta-llama/llama3/blob/main/MODEL\\_CARD.md](https://github.com/meta-llama/llama3/blob/main/MODEL_CARD.md).
- [6] J. Ainslie, J. Lee-Thorp, M. de Jong, Y. Zemlyanskiy, F. Lebrón,

- and S. Sanghai. Gqa: Training generalized multi-query transformer models from multi-head checkpoints. arXiv preprint arXiv:2305.13245, 2023.
- [7] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocar, M. Debbah, Étienne Goffinet, D. Hesslow, J. Launay, Q. Malartic, D. Mazzotta, B. Noune, B. Pannier, and G. Penedo. The falcon series of open language models, 2023. R. Anil, A. M. Dai, O. Firat, M. Johnson, D. Lepikhin, A. Passos, S. Shakeri, E. Taropa, P. Bailey, Z. Chen, et al. Palm 2 technical report. arXiv preprint arXiv:2305.10403, 2023.
  - [8] J. Austin, A. Odena, M. I. Nye, M. Bosma, H. Michalewski, D. Dohan, E. Jiang, C. J. Cai, M. Terry, Q. V. Le, and C. Sutton. Program synthesis with large language models. CoRR, abs/2108.07732, 2021. URL <https://arxiv.org/abs/2108.07732>.
  - [9] P. Barham, A. Chowdhery, J. Dean, S. Ghemawat, S. Hand, D. Hurt, M. Isard, H. Lim, R. Pang, S. Roy, B. Saeta, P. Schuh, R. Sepassi, L. E. Shafey, C. A. Thekkath, and Y. Wu. Pathways: Asynchronous distributed dataflow for ml, 2022.
  - [10] I. Bello, H. Pham, Q. V. Le, M. Norouzi, and S. Bengio. Neural combinatorial optimization with reinforcement learning. CoRR, abs/1611.09940, 2016. URL <http://arxiv.org/abs/1611.09940>.
  - [11] I. Beltagy, M. E. Peters, and A. Cohan. Longformer: The long-document transformer. CoRR, abs/2004.05150, 2020b. URL <https://arxiv.org/abs/2004.05150>.
  - [12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei. Language models are few-shot learners. CoRR, abs/2005.14165, 2020. URL <https://arxiv.org/abs/2005.14165>.
  - [13] N. Carlini, D. Ippolito, M. Jagielski, K. Lee, F. Tramèr, and C. Zhang. Quantifying memorization across neural language models. arXiv preprint arXiv:2202.07646, 2022.
  - [14] M. Chen, J. Tworek, H. Jun, Q. Yuan, H. P. de Oliveira Pinto, J. Kaplan, H. Edwards, Y. Burda, N. Joseph, G. Brockman, A. Ray, R. Puri, G. Krueger, M. Petrov, H. Khlaaf, G. Sastry, P. Mishkin, B. Chan, S. Gray, N. Ryder, M. Pavlov, A. Power, L. Kaiser, M. Bavarian, C. Winter, P. Tillet, F. P. Such, D. Cummings, M. Plappert, F. Chantzis, E. Barnes, A. Herbert-Voss, W. H. Guss, A. Nichol, A. Paino, N. Tezak, J. Tang, I. Babuschkin, S. Balaji, S. Jain, W. Saunders, C. Hesse, A. N. Carr, J. Leike, J. Achiam, V. Misra, E. Morikawa, A. Radford, M. Knight, M. Brundage, M. Murati, K. Mayer, P. Welinder, B. McGrew, D. Amodei, S. McCandlish, I. Sutskever, and W. Zaremba. Evaluating large language models trained on code. CoRR, abs/2107.03374, 2021. URL <https://arxiv.org/abs/2107.03374>.
  - [15] Y. Gu, L. Dong, F. Wei, and M. Huang. Minillm: Knowledge distillation of large language models. In The Twelfth International Conference on Learning Representations, 2024.
  - [16] D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt. Measuring massive multitask language understanding. CoRR, abs/2009.03300, 2020. URL <https://arxiv.org/abs/2009.03300>.
  - [17] J. Hoffmann, S. Borgeaud, A. Mensch, E. Buchatskaya, T. Cai, E. Rutherford, D. d. L. Casas, L. A. Hendricks, J. Welbl, A. Clark, et al. Training compute-optimal large language models. arXiv preprint arXiv:2203.15556, 2022.
  - [18] T. Kudo and J. Richardson. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In E. Blanco and W. Lu, editors, Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations, pages 66–71, Brussels, Belgium, Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-2012. URL <https://aclanthology.org/D18-2012>.
  - [19] Ronen Eldan, Yuanzhi Li. TinyStories: How Small Can Language Models Be and Still Speak Coherent English? arXiv:2305.07759.
  - [20] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. Attention is all you need. CoRR, abs/1706.03762, 2017. URL <http://arxiv.org/abs/1706.03762>.
  - [21] Y. Xu, H. Lee, D. Chen, B. A. Hechtman, Y. Huang, R. Joshi, M. Krikun, D. Lepikhin, A. Ly, M. Maggioni, R. Pang, N. Shazeer, S. Wang, T. Wang, Y. Wu, and Z. Chen. GSPMD: general and scalable parallelization for ML computation graphs. CoRR, abs/2105.04663, 2021. URL <https://arxiv.org/abs/2105.04663>.
  - [22] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, et al. Sparks of artificial general intelligence: Early experiments with gpt-4. arXiv preprint arXiv:2303.12712, 2023.
  - [23] Jonathan Frankle and Michael Carbin. The lottery ticket hypothesis: Finding sparse, trainable neural networks. arXiv preprint arXiv:1803.03635, 2018.
  - [24] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.
  - [25] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361, 2020.
  - [26] Brown, T., Mann, B., Ryder, N., et al. (2020). "Language Models are Few-Shot Learners." Advances in Neural Information Processing Systems, 33, 1877–1901.
  - [27] Zhang, Y., Sun, S., Galley, M., Chen, Y.-C., Brockett, C., Gao, X., ... & Dolan, B. (2020). "Dialogpt: Large-scale generative pre-training for conversational response generation." Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations, 270–278.
  - [28] Ruder, S. (2017). "An Overview of Multi-Task Learning in Deep Neural Networks." arXiv preprint arXiv:1706.05098.
  - [29] Howard, J., & Ruder, S. (2018). "Universal Language Model Fine-tuning for Text Classification." Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, 328–339.
  - [30] Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. arXiv preprint arXiv:1709.00103, 2017.

- [31] Guillaume Wenzek, Marie-Anne Lachaux, Alexis Conneau, Vishrav Chaudhary, Francisco Guzmán, Armand Joulin, and Edouard Grave. Ccnet: Extracting high quality monolingual datasets from web crawl data. arXiv preprint arXiv:1911.00359, 2019.
- [32] Aarohi Srivastava, Abhinav Rastogi, Abhishek Rao, Abu Awal Md Shoeb, Abubakar Abid, Adam Fisch, Adam R Brown, Adam Santoro, Aditya Gupta, Adrià Garriga-Alonso, et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. arXiv preprint arXiv:2206.04615, 2022.
- [33] Michael Santacrose, Zixin Wen, Yelong Shen, and Yuanzhi Li. What matters in the structured pruning of generative language models? arXiv preprint arXiv:2302.03773, 2023.
- [34] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1): 5485–5551, 2020.
- [35] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- [36] OpenAI. Gpt-4 technical report, 2023.
- [37] Kirkpatrick, J., Pascanu, R., Rabinowitz, N., et al. (2017). "Overcoming catastrophic forgetting in neural networks." *Proceedings of the National Academy of Sciences*, 114(13), 3521–3526.
- [38] Howard, J., & Ruder, S. (2018). "Universal Language Model Fine-tuning for Text Classification." *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, 328–339.