A System of Hierarchical Generative AI Models Trained with nlmate

C. Strommen
LibreCS, May 2023

1. Introduction

Artificial Intelligence (AI) has taken giant strides over the past decade, with generative models playing a significant role in this advancement. These models, particularly in the domain of Natural Language Processing (NLP), have been successful in a wide range of tasks including text generation, translation, summarization, and more. However, training such models requires considerable resources and large-scale datasets. The need for custom datasets for specific use cases led to the development of tools like nlmate, an open-source solution for scraping, preprocessing, and labeling datasets from Wikipedia pages for training NLP models. This white paper presents a novel approach for building a system of many generative AI models, each trained for specific purposes using data collected via nlmate. These models are organized in a hierarchical structure, where smaller models, if unable to adequately respond to a prompt, pass the prompt to a larger model trained on a broader dataset collected with nlmate.

2. nlmate: An Overview

nlmate is an open-source tool designed for the purpose of scraping, preprocessing, and labeling data from Wikipedia. It offers functionality to generate a list of random Wikipedia page names or use specified Wikipedia pages for data collection. The tool then fetches content from these pages, preprocesses it, labels it, and saves the resulting structured dataset as a CSV file. Moreover, nlmate allows for customization of the labeling function. This function takes a text input and assigns a label based on the content of the text. The default implementation uses TextBlob sentiment analysis to assign "positive", "negative", or "neutral" labels based on the sentiment polarity of the text. nlmate is designed as a comprehensive solution for collecting, preparing, and labeling data for use in training natural language processing (NLP) models. The tool performs several functions in its pipeline:

1. **Data Collection:** nlmate offers two ways of collecting data from Wikipedia. Users can either generate a list of random Wikipedia page names or manually specify the pages they want to scrape. This allows for the creation of general-purpose datasets or more domain-specific ones based on the user's needs.
2. **Data Scraping:** Once the list of Wikipedia pages is determined, nlmate uses the Wikipedia API to fetch the content from these pages. It retrieves the raw text from the pages, which often includes a wealth of information on a wide range of topics.
3. **Data Preprocessing:** After fetching the content, nlmate preprocesses the data to prepare it for use in training an NLP model. This preprocessing stage typically involves tasks such as cleaning the text (e.g., removing special characters, correcting encoding issues, etc.), normalizing the text (e.g., lowercasing, stemming, etc.), and tokenizing the text into smaller units (e.g., words, sentences, etc.).
4. **Data Labeling:** The next stage in the pipeline is data labeling. nlmate allows for the customization of the labeling function, making it a flexible tool for various NLP tasks. By default, it uses TextBlob, a popular Python library for processing textual data, to perform sentiment analysis on the text. This sentiment analysis is used to label the data as "positive", "negative", or "neutral" based on the sentiment polarity of the text. However, users can modify the labeling function to suit their specific needs.
5. **Data Export:** Finally, nlmate saves the structured dataset as a CSV file. This file contains the processed and labeled data, ready to be used for training an NLP model.

In essence, nlmate is a powerful tool that streamlines the process of creating datasets for NLP tasks. Its open-source nature allows users to modify and adapt it to their specific needs, making it a versatile solution for a wide range of NLP applications.

3. System Architecture

The proposed system is organized as a hierarchy of generative AI models. Each model in the hierarchy is trained on a dataset that is specific to its intended purpose. The hierarchy is organized such that smaller, more specialized models are at the lower levels, while larger, more general models are at the higher levels.

3.1. Lower-Level Models

The lower-level models are specialized models, each trained on a specific dataset prepared using nlmate. These datasets are created from a curated list of Wikipedia pages relevant to the model's specialized domain. For example, a model designed for medical queries might be trained on data scraped from medical-related Wikipedia pages.

3.2. Higher-Level Models

Higher-level models are trained on broader datasets that encompass a wider range of topics. These datasets are created using nlmate to scrape data from a larger and more diverse list of Wikipedia pages. This ensures that the higher-level models have a general understanding of language and a wide range of topics.

3.3. Model Interaction

The models interact with each other in a hierarchical manner. When a prompt is given to the system, it first passes to the most relevant lower-level model. If this model can generate an adequate response, it does so. However, if it determines (based on a pre-defined confidence threshold) that it cannot provide an adequate response, it passes the prompt up to the next level in the hierarchy.

This process continues until the prompt reaches a model that can provide an adequate response or until it reaches the highest level in the hierarchy. The highest-level model is the model of last resort and is designed to provide a response to any prompt, albeit possibly with less specificity than a lower-level model.

4. Training the Models

The tiered nature of the model training in the proposed system represents a hierarchical approach to tackling a wide variety of prompts. By training multiple models of increasing complexity and data capacity, the system is designed to handle a broad spectrum of tasks with various levels of specialization and generalization.

1. Different Sizes of Models: At the lower tiers, smaller models are trained on specific datasets, allowing them to develop a fine-grained understanding of certain topics or tasks. As you move up the hierarchy, the models get larger and are trained on more diverse and expansive datasets. This allows these higher-tier models to have a broader understanding and capability, enabling them to handle more general tasks or prompts that the lower-tier models can't handle.
2. Training Methodologies: Each model is trained using an appropriate generative AI training methodology. This can range from Transformer-based models like GPT and BERT to LSTM-based models. The choice of architecture would depend on various factors, such as the complexity of the task, the size and nature of the dataset, and the computational resources available.
3. Computational Requirements: The tiered nature of the model training has significant implications for the computational power needed to train the models. Smaller models at the lower tiers require less computational resources to train, making them more accessible for smaller-scale applications. On the other hand, larger models at the higher tiers require significant computational resources due to their size and the complexity of the tasks they are designed to handle.
4. Model Validation and Performance Evaluation: Each model is trained until it reaches a specific performance threshold on a validation dataset. The validation dataset is a subset of the training dataset that is used to monitor the model's performance during training and prevent overfitting. The performance of each model is evaluated using relevant NLP metrics like BLEU for translation tasks, ROUGE for summarization tasks, and perplexity for language modeling tasks.

Each model is trained on its respective dataset using a suitable generative AI training methodology. The specific training methodology can vary depending on the requirements of the model and the characteristics of the data. However, it generally involves a process of unsupervised learning where the model learns to generate text similar to its training data. These models can be trained using various architectures such as Transformer-based models like GPT (Generative Pretrained Transformer), BERT (Bidirectional Encoder Representations from Transformers), or LSTM (Long Short-Term Memory) based models, depending on the specific requirements and available resources.

The models are trained until they reach a specified performance threshold on a validation dataset, which is a subset of the training dataset held out for testing the model's performance. The performance of each model is evaluated using appropriate NLP metrics such as BLEU (Bilingual Evaluation Understudy), ROUGE (Recall-Oriented Understudy for Gisting Evaluation), or perplexity.

5. Evaluating the System

The overall system's performance can be evaluated based on its ability to generate adequate responses to a wide range of prompts. This can be measured using a diverse set of evaluation metrics, including but not limited to:

1. **Prompt Coverage:** The percentage of prompts that the system can generate an adequate response to. An adequate response is defined as one that is relevant, coherent, and factually correct.
2. **Response Quality:** The quality of the responses generated by the system, measured using NLP metrics such as BLEU, ROUGE, and perplexity.
3. **Prompt Handling Efficiency:** The level in the model hierarchy that typically handles the prompts. A lower average level indicates that the specialized models are effectively handling their designated prompts, reducing the load on the higher-level models.
4. **System Scalability:** The ability of the system to handle an increased number of prompts or an expanded range of topics without significant degradation in performance.

The introduction of a secondary feedback model presents an opportunity to optimize the performance of the tiered AI system. This feedback model would act as a quality control mechanism, evaluating the responses generated by the AI models at each level of the hierarchy, and playing a crucial role in determining the adequacy of a response and the number of steps required to generate it. Let's delve deeper into this concept:

1. **Quality Assessment:** The feedback model would assess the quality of responses based on various criteria, including relevancy to the prompt, coherence of the response, and factual accuracy. This process might involve sophisticated NLP techniques and potentially human evaluation for more nuanced aspects of quality.
2. **Model Optimization:** Based on the feedback, the system can then optimize the number of steps taken to reach an adequate answer. For example, if a lower-level model consistently generates high-quality responses for a specific type of prompt, the system can be optimized to direct similar prompts to this model, reducing the need to escalate prompts to higher-level models.
3. **System Evaluation:** In addition to optimizing individual responses, the feedback from the secondary model can be used to evaluate the overall performance of the system. This includes metrics like:
    - **Prompt Coverage:** The percentage of prompts the system can adequately respond to. High prompt coverage indicates that the system is capable of handling a diverse range of prompts.
    - **Response Quality:** This measures the overall quality of responses generated by the system. A combination of NLP metrics like BLEU, ROUGE, and perplexity can be used, along with feedback from the secondary model.
    - **Prompt Handling Efficiency:** This metric gauges how efficiently the system handles prompts at different levels of the hierarchy. Ideally, the lower-level models should be able to handle most prompts, with only complex or unfamiliar prompts needing escalation to higher-level models.

    - **System Scalability:** This assesses the system's ability to handle increased demand, such as a larger volume of prompts or a wider range of topics, without a significant drop in performance.
4. **Continuous Improvement:** The feedback from the secondary model can be used for continuous improvement of the AI models in the system. The models can be fine-tuned based on this feedback, improving their performance over time.

By providing a mechanism for ongoing evaluation and improvement, the secondary feedback model enhances the robustness and efficiency of the tiered AI system. It helps ensure that the system is always learning and improving, maximizing its performance and value.

6. Advantages of the System

The proposed hierarchical AI system introduces a unique approach to model training and text generation that brings considerable benefits, notably in computational efficiency during both training and generating phases:

1. **Specialization:** The lower-tier models in the system are trained on specialized datasets, which allows them to efficiently handle specific types of prompts. Training specialized models often requires less computational power compared to training a large, general-purpose model that aims to handle a wide range of prompts. Furthermore, these specialized models can often generate more accurate and nuanced responses for their specific areas of expertise.
2. **Efficiency in Generation:** By distributing prompts across different models based on their areas of specialization, the system can generate responses more efficiently. Smaller models require less computational resources to generate a response, thus the system can save computational power when a lower-tier model can adequately handle a prompt. This approach allows the system to handle a high volume of prompts with a lower computational cost than a system relying solely on larger, more resource-intensive models.
3. **Scalability:** The system is designed to be scalable, both in terms of the number and size of the models. New models can be added at any level of the hierarchy, allowing the system to expand its capabilities and adapt to new requirements over time. This flexibility means that the system can gradually incorporate larger models that require more computational power, while still maintaining the efficiency benefits of the smaller, specialized models.
4. **Robustness:** The hierarchical structure of the system ensures that if a lower-tier model cannot adequately handle a prompt, the prompt is passed up to a higher-tier model. This fail-safe mechanism increases the robustness of the system and ensures it can always generate a response, even for complex or unfamiliar prompts.
5. **Customization of Data Collection:** The use of the nlmate tool allows for customization in the data collection and labeling process. This flexibility in creating tailored datasets for each model can lead to more effective training sessions, potentially reducing the computational power required to reach a desired level of performance.

In summary, the proposed system provides a robust and scalable solution for AI text generation, with the potential to offer significant reductions in computational power requirements during both training and text generation phases. By leveraging the strengths of both specialized and general models, the system can provide high-quality responses across a wide range of prompts, while optimizing computational efficiency.

7. Conclusion

The proposed system of hierarchical generative AI models presents a promising approach to harnessing the power of AI for a wide range of tasks. By combining the strengths of specialized lower-level models with the robustness of general higher-level models, the system offers a powerful and adaptable tool for a variety of NLP applications. With its ability to train models on tailored datasets using nlmate, the system provides an efficient, scalable, and robust solution that can be customized to meet specific requirements. Future work will focus on refining the model training and evaluation process, expanding the system to cover a wider range of topics, and exploring potential applications in various fields. The road to perfecting AI is a continuous journey. However, by leveraging the power of hierarchical models, we are one step closer to creating AI systems that are not only powerful and versatile but also efficient and user-friendly.

Encouraging Open-Source Contribution to the Project

The project is hosted on GitHub at https://github.com/LibreCS/nlmate and is open to contributions from the global open-source community. The project is developed by LibreCS, an open-source organization dedicated to developing software tools and systems that bring enterprise-level performance, reliability, and cutting-edge technology to both professionals and enthusiasts. LibreCS strives to create projects that are inclusive to low resource environments and non-standard process architectures[1].

LibreCS encourages the use of its software in other open-source projects. All of its software, including the nlmate project, is licensed under the GNU GPLv3, promoting freedom, openness, and collaboration in the development process.

As a contributor, you can aid in various ways. This could be by expanding the tool's functionalities, optimizing its existing features, fixing bugs, improving documentation, or even by suggesting new ideas that could enhance the tool. Your contributions will not only be valuable for the project but also for the broader open-source community.

Before starting your contribution, make sure to familiarize yourself with the project's guidelines and code of conduct. This ensures a welcoming and inclusive environment for all contributors. If you're new to open-source contribution or need help understanding the project, don't hesitate to reach out to the maintainers or the community.

Remember, every contribution, no matter how small, is significant in open-source projects. Whether you're a seasoned open-source contributor or just starting, your efforts can make a big difference.

Join us in our journey to advance the field of Natural Language Processing with this project. Your contributions can play a crucial role in shaping its future.