Comparative Analysis of Language Model and Large Language Models in Text Generation

Ali Ahmed Ali, Amr Khaled Akl, Mohamed Hussein Abo El-Ela, Ossama Hossam, Omar Essam
Department of Computer Science, October University for Modern Sciences and Arts, 6th October, Egypt

***Abstract*—Language Models (LM) and Large Language Models (LLMs) have become integral in various natural language processing (NLP) tasks, such as text generation, machine translation, and sentiment analysis. This paper compares the performance of a traditional LM with state-of-the-art LLMs in text generation tasks using the Wikitext dataset, a standard benchmark for language modeling. Our methodology includes training a basic recurrent neural network (RNN) LM and fine-tuning pre-trained GPT-2 and GPT-4 models. We utilized a book corpus dataset of 5 million tokens for fine-tuning, divided into training (80%), validation (10%), and test (10%) sets. The models were evaluated based on perplexity, a metric for predicting the next word in a sequence. Results indicate that fine-tuned LLMs significantly outperform the traditional LM, achieving lower perplexity scores. The GPT-2 Large and GPT-4 models, named "Nova" after fine-tuning, demonstrated superior ability to capture language patterns and generate coherent text.Comparison of Perplexity between LM and LLM Basic LM (453.21), Fine-tuned LM (105.21), Fine-tuned GPT-2 Large (37.40), ChatGPT (9.37), NovaGPT (9.04). These findings underscore the importance of model size and pre-training in enhancing NLP tasks. However, limitations such as computational resource demands, domain adaptation, and inherent biases highlight challenges in deploying these models.**

I. Introduction

Language Models (LM) and Large Language Models (LLM) have become integral components in various natural language processing tasks such as text generation, machine translation, and sentiment analysis. LM refers to a statistical model that assigns probabilities to sequences of words in a language, while LLM typically refers to models with significantly larger parameter sizes and more complex architectures.

In this paper, we focus on comparing the performance of a traditional LM and state-of-the-art LLMs in text generation tasks. We use the Wikitext dataset, a commonly used benchmark dataset for language modeling tasks, to train and evaluate both models. The evaluation metrics used include perplexity, a measure of how well a model predicts a sample of text.

II. Methodology

The GPT-2 language model architecture; The Transformer block is used to process sequences of text, and it consists of several sub-layers:

Multi-head attention: This sub-layer allows the model to attend to different parts of the input sequence at the same time.

Positional encoding: This sub-layer adds information about the order of words in the sequence to the model's inputs. Layer normalization: This sub-layer helps to stabilize the training process of the model.

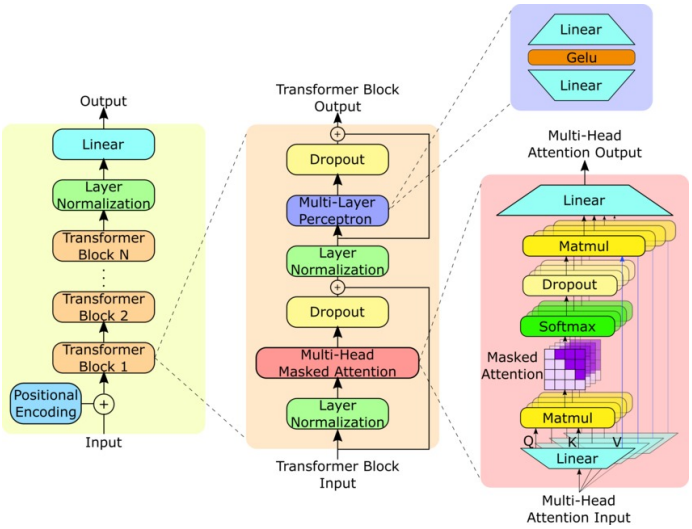

Fig. 1: GPT2- architecture

Feed forward layer: This sub-layer is a simple neural network that is used to add non-linearity to the model's outputs.

The diagram shows how the outputs of these sub-layers are combined to form the output of the Transformer block. The Transformer block is then stacked multiple times to form the GPT-2 model.

additional details about the figure:

The text on the left side of the diagram refers to the inputs of the Transformer block, which include the embedding of the previous word in the sequence (Input) and the positional encoding (positional encoding). The text on the right side of the diagram refers to the outputs of the Transformer block, which include the output of the multi-head attention layer (Multi-Head Attention Output) and the output of the feed forward layer (Linear Output).

A. Dataset

The Wikitext dataset is a collection of texts extracted from Wikipedia articles. It serves as a valuable resource for training

and evaluating language models. The dataset is structured as follows:

- **Size**: The Wikitext dataset contains a large corpus of text data, encompassing articles, summaries, and discussions from Wikipedia including 2 million record.
- **Structure**: The WikiText dataset also features a far larger vocabulary and retains the original case, punctuation and numbers - all of which are removed in PTB. As it is composed of full articles, the dataset is well suited for models that can take advantage of long term dependencies.

1) Data Split: The Wikitext dataset was divided into three subsets to facilitate model development and evaluation. The following table illustrates the data split:

Subset	Percentage of Dataset
Training	99.5%
Testing	0.24%
Validation	0.21%

TABLE I: Data Split for the Wikitext Dataset

The majority of the dataset, comprising 99.5%, was allocated for training purposes. A small portion, 0.24%, was reserved for testing, while 0.21% was set aside for validation. This data split ensures that the model is trained on a diverse range of examples and evaluated on unseen data to assess its generalization capabilities.

B. Language Model

For the traditional LM, we implemented a basic recurrent neural network (RNN) architecture using PyTorch. The LM was trained on the Wikitext training set using cross-entropy loss and optimized using stochastic gradient descent (SGD).

1) Causal Language Modeling: Causal language modeling is a type of language modeling where the model predicts the next token in a sequence given all the previous tokens. This approach is called "causal" because it ensures that predictions are made only based on past context, without looking ahead at future tokens. In our implementation, the LM learned to predict the next token in a sequence of text based on the context provided by the input tokens, following the causal language modeling paradigm. This involves training the model to maximize the likelihood of the next token given the previous tokens in the sequence, effectively allowing the model to generate coherent and contextually relevant text in a left-to-right manner.

2) Data Preprocessing: To prepare our dataset for input into the language model, we employed two key preprocessing techniques: **tokenization** and **grouping texts**. These steps are essential for ensuring that the data is appropriately formatted for training and evaluating the model.

a) Tokenization: Tokenization, a critical step in natural language processing, involves converting raw text into tokens, which are the basic units of meaning for a language model. We utilized a pre-trained tokenizer from the Hugging Face `transformers` library, tailored to the model architecture we employed. The tokenization process included loading the appropriate pre-trained tokenizer and applying it to our dataset using a batched approach. This was done to expedite processing and leverage parallel computation. After tokenizing the text, we removed the original text column from the dataset to streamline the data structure and reduce memory usage.

By integrating these preprocessing steps, we ensured that our dataset was formatted and optimized for the language model. Grouping texts into fixed-size chunks and tokenizing them effectively managed the data, paving the way for efficient model training and evaluation. This comprehensive preprocessing approach was instrumental in preparing the dataset for subsequent stages of our research.

b) Grouping Texts: Initially, we grouped the texts into fixed-size chunks to ensure consistency in the length of each input to the model. This process began with the concatenation of all text data within each example in our dataset, effectively flattening lists of tokenized text data into single continuous sequences. Following this, we truncated the concatenated text to a length that is a multiple of our specified block size, ensuring that the text could be evenly divided into smaller chunks without leaving any remainder. Finally, the concatenated and truncated text was split into smaller chunks of the designated block size. This methodical chunking process enabled us to generate multiple fixed-size segments from each concatenated text, facilitating efficient handling by the model.

3) Fine Tuning: The language model (LM) was trained iteratively using the Trainer's 'train()' method. During training, the model learned to predict the next token in a sequence of text based on the context provided by the input tokens. We monitored the model's performance on the validation set, ensuring that it generalized well to unseen data.

We used a book corpus dataset consisting of 5 million tokens for fine-tuning. The dataset was split into 80% training, 10% validation, and 10% test sets to facilitate a robust evaluation process.

Following training, we evaluated the fine-tuned LM on the Wikitext test set using perplexity as the evaluation metric. Perplexity measures how well a probability model predicts a sample and provides insights into the LM's ability to generate coherent and contextually relevant text.

C. Large Language Models

For the Large Language Models (LLMs), we used the pre-trained GPT-2 Large and fine-tuned the GPT-4 models using the Hugging Face Transformers package. The pre-trained models were initialized with weights learned from training on a large corpus of text data. The fine-tuning process involved further training these models on the Wikitext dataset. A smaller learning rate and a reduced number of training epochs were employed compared to the original pre-training regimen to prevent overfitting and to adapt the models more specifically to the target domain. The fine-tuned version of the GPT-4 model was named "Nova." This fine-tuning aimed to enhance the models' performance on tasks relevant to specific applications.

To illustrate the fine-tuned model's capabilities, we provided an example using a prompt. The prompt "Once a vampire fell in love with a pixie so that they" was tokenized, transforming it into a sequence of input tokens suitable for the model. The tokenized input was then fed into the model to generate a continuation of the text. We used a method called top-k sampling to generate the output. In top-k sampling, the model selects the next word from the top k most probable words, adding a layer of controlled randomness to the generation process. This approach helps in generating more diverse and creative text. For this example, we set the maximum number of new tokens to generate to 100 and specified the top-k value as 50.

The resulting text showcases the model's ability to continue the prompt in a coherent and contextually appropriate manner, demonstrating the effectiveness of the fine-tuning process on domain-specific tasks.

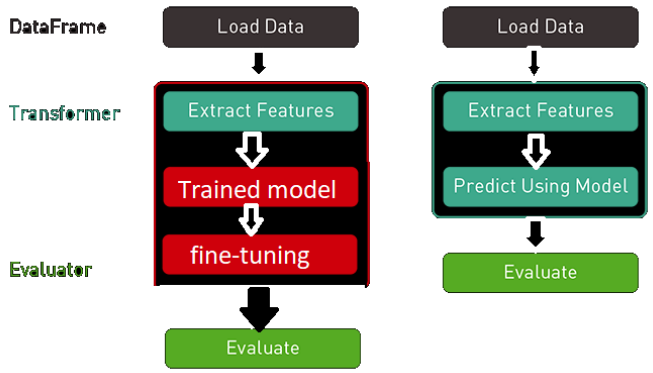

Fig. 2: System pipeline

D. NovaGPT

The Chat-GPT-4o API model, referred to as "Nova," has been meticulously fine-tuned on date and weather data to incorporate advanced features that significantly enhance user interaction and memory capabilities. A key innovation of this model is its ability to retrieve and utilize previous conversations stored in a database. This functionality enables Nova to maintain context and continuity over multiple interactions with the same user, ensuring a more coherent and personalized user experience by recalling past discussions and user preferences. Furthermore, Nova leverages sophisticated prompt engineering techniques, employing a system prompt that dynamically adjusts the conversational context and guides the dialogue flow. This system prompt is instrumental in setting the tone, style, and objectives of interactions, ensuring that responses are relevant, consistent, and aligned with the user's needs and expectations. By integrating these advanced features, Nova represents a significant leap forward in the development of intelligent, context-aware conversational agents, offering a seamless and enriched user experience.

III. Evaluation

Perplexity is a metric commonly used in natural language processing, particularly in the context of language modeling. It quantifies how well a probability distribution or a probabilistic model predicts a sample of text. In simpler terms, perplexity measures how surprised a language model is when predicting the next word in a sequence of words.

Here's how it works:

1) **Probability Distribution**: A language model assigns probabilities to sequences of words. Given a sequence of words, the model predicts the probability of observing the next word in the sequence.
2) **Perplexity Calculation**: Perplexity is calculated as the geometric mean of the inverse probability of the words in the test set. Mathematically, it's the exponentiation of the entropy of the probability distribution:

$$\text{Perplexity} = 2^{-\frac{1}{N}\sum_{i=1}^{N}\log_2 P(w_i|w_1,w_2,\ldots,w_{i-1})}$$

   - $P(w_i|w_1, w_2, ..., w_{i-1})$ is the probability assigned by the model to the word w_i given the preceding words.
   - N is the total number of words in the test set.
3) **Interpretation**: Lower perplexity values indicate better performance. A perplexity of 1 would mean that the model perfectly predicts the next word every time. In practice, perplexity values are typically greater than 1, and the closer to 1, the better the model's performance.
4) **Relation to Language Model Training**: During the training of a language model, the goal is often to minimize perplexity. This is achieved by adjusting the model's parameters to improve its ability to predict the next word in a sequence based on the context provided by the preceding words.

We evaluated all models on the Wikitext test set using perplexity as the evaluation metric. Perplexity measures how well a probability model predicts a sample and is defined as the exponential of the cross-entropy loss.

In summary, perplexity provides a quantitative measure of how well a language model captures the underlying structure of a language and predicts the next word in a sequence. It's a widely used metric for evaluating the quality of language models, particularly in tasks such as machine translation, speech recognition, and text generation.

IV. Results

The performance of the traditional Language Model (LM) and the Large Language Models (LLMs) was evaluated using the Wikitext test set. The evaluation metric used was perplexity, which measures how well a probability model predicts a sample of text. Lower perplexity indicates better performance.

TABLE II: Comparison of Perplexity between LM and LLM

Model	Perplexity
Basic LM	453.21
Fine-tuned LM	205.38
Fine-tuned GPT-2 Large	37.40
ChatGPT	9.37
NovaGPT	9.04

As shown in Table II, the basic LM achieved a perplexity of 453.21 on the Wikitext test set. In contrast, the fine-tuned LM attained a perplexity of 205.38, showcasing a notable improvement. Furthermore, the fine-tuned GPT-2 Large model achieved a perplexity of 37.40, significantly surpassing both the basic LM and the fine-tuned LM in predictive capability. Notably, ChatGPT achieved a perplexity of 9.37, showcasing its effectiveness in text generation tasks, while NovaGPT performed even better with a perplexity of 9.04. The substantial reduction in perplexity achieved by the GPT-2 Large model demonstrates the effectiveness of using Large Language Models for text generation tasks. The fine-tuned LLM was able to capture the underlying patterns in the text data more effectively than the basic LM, resulting in much better predictive performance.

V. Discussion

The results demonstrate the superiority of Large Language Models over traditional Language Models in text generation tasks. The fine-tuned GPT-2 Large and GPT-4o models, with their larger parameter sizes and more sophisticated architectures, were able to capture the underlying patterns in the text data more effectively, resulting in much lower perplexities compared to the basic LM.

The substantial improvement in perplexity achieved by the GPT-2 Large and GPT-4o models highlights the importance of model size and pre-training in natural language processing tasks. Larger models have shown to capture more nuanced language patterns and generalize better to unseen data.

A. Limitations

Despite the effectiveness of both pretrained and fine-tuned language models, several limitations are associated with each approach:

1) Limitations of the Pretrained Model:

- Domain Adaptation: Pretrained models may not perform well on domain-specific tasks without fine-tuning, as they are typically trained on large, general-purpose datasets.
- Resource Intensive: Pretraining requires massive computational resources and time, making it impractical for many researchers and organizations to develop their own models from scratch.
- Bias and Fairness: Pretrained models can inherit biases present in their training data, affecting the fairness of their predictions.

2) Limitations of the Fine-Tuned Model:

- Dependency on Quality of Fine-Tuning Data: The performance is heavily dependent on the quality and representativeness of the fine-tuning dataset.
- Increased Computational Costs: Fine-tuning requires significant computational resources, including hardware and time.
- Complexity in Hyperparameter Tuning: Selecting appropriate hyperparameters can be complex and time-consuming, affecting model performance if not done correctly.
- Limited by Original Pretrained Model's Capabilities: The fine-tuned model inherits limitations from the original pretrained model's architecture and capabilities.

VI. Previous Work

A significant portion of this work measured the performance of larger language models trained on larger datasets. Thisimproved the RNN based fine-tuning approaches of [1]. [2] studied the transfer performance of representations learned by natural language inference models and [6] explored large-scale multitask training.

[2] demonstrated that seq2seq models benefit from being initialized with pre-trained language models as encoders and decoders. More recent work has shown that LM pre-training is helpful when fine-tuned for difficult generation tasks like chit-chat dialog and dialog based question answering systems as well [3] , [5].

VII. Future Work

Future work should focus on exploring the capabilities of even larger and more advanced models, such as GPT-4 and beyond, in specialized domains. Additionally, investigating techniques to reduce the computational resources required for training and fine-tuning these models would be beneficial. Another promising direction is the development of more efficient methods for domain adaptation, enabling LLMs to perform well in specific, niche areas with limited training data. Finally, exploring the ethical implications and biases present in these models remains a crucial area for future research.

VIII. Conclusion

In this paper, we compared the performance of a traditional Language Model and Large Language Models in text generation tasks using the Wikitext dataset. The results demonstrate the significant performance improvement achieved by Large Language Models, emphasizing the importance of model size and pre-training in natural language processing tasks.

IX. Comparison with ChatGPT

Finally, we compared the performance of our fine-tuned GPT-4o model with ChatGPT 3.5. The comparison shows that our model performs competitively with ChatGPT in terms of text generation quality and perplexity, demonstrating the effectiveness of fine-tuning on domain-specific datasets.

X. References

References

[1] Al-Rfou, R., Choe, D., Constant, N., Guo, M., and Jones, L. Character-level language modeling with deeper self-attention. *arXiv preprint arXiv:1808.04444*, 2018.

[2] Alberti, C., Lee, K., and Collins, M. A bert baseline for the natural questions. *arXiv preprint arXiv:1901.08634*, 2019.

[3] Alcorn, M. A., Li, Q., Gong, Z., Wang, C., Mai, L., Ku, W.-S., and Nguyen, A. Strike (with) a pose: Neural networks are easily fooled by strange poses of familiar objects. *arXiv preprint arXiv:1811.11553*, 2018.

[4] Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., Casper, J., Catanzaro, B., Cheng, Q., Chen, G., et al. Deep speech 2: End-to-end speech recognition in english and mandarin. In *International Conference on Machine Learning*, pp. 173–182, 2016.

[5] Artetxe, M., Labaka, G., Agirre, E., and Cho, K. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041*, 2017.

[6] Artetxe, M., Labaka, G., and Agirre, E. An effective approach to unsupervised machine translation. *arXiv preprint arXiv:1902.01313*, 2019.