Large Language Models and Data Streams

Silyu Li

RWTH Aachen University, Aachen, Germany

Abstract. Large language models, such as ChatGPT, have become widely recognized and are extensively utilized across various domains. However, these models are typically trained on static datasets, lacking updates to new data beyond their initial training set. To enable models that can continuously update themselves based on incoming data, it is necessary to have large language models trained and updated on input data streams. In this paper, we begin by outlining the structure and fundamental applications of large language models. Subsequently, we introduce the concept of data streams and provide an overview of current use cases where large language models are adapted to accommodate streaming data. Finally, we summarize the existing challenges associated with integrating large language models with data streams and discuss potential solutions.

Keywords: LLM · data streams · chatGPT.

1 Introduction

Large language models have shown their wide usage and significant competence in many fields according to [1], such as learning and answering users' questions in academic fields, assisting in diagnosing diseases in medical fields, generating text and classifying text data to various categories etc. However, as [24] demonstrates, there is a significant limitation of current large language models: they are trained based on certain static datasets that will not automatically be updated, and this causes the resulting models to only be able to access information from its training datasets. When the models need to be updated as new data becomes available, the only way is to start the training process over again. But in many scenarios such models can't satisfy our needs. For example, to have a better traffic prediction, real-time traffic data is needed [6], Analyzing news and events sentiment can help predict the financial market [27], real-time health data is of great importance when monitoring patients' health condition [3] etc. So to have models that can fulfill those use cases, we need to find and compare useful methods that combine large language models and continuous data input (data streams) together, and also summarize the current major obstacles.

2 Related Works

In this chapter, the following concepts and techniques that are related to this topic will be covered.

2.1 Large Language Model

In this subsection, we will first summarize the evolution of language models, followed by an overview of the current large language models' structure and training process.

The Evolution of Large Language Models

The earliest language models, taking n-grams as an example, are statistical. n-grams refers to an N-characters substring of a longer string and n represents the length of the substring, according to [29]. Taking the sentence "I read a book" as example, a bi-grams composition of this sentence would be "I read", "read a" and "a book". By considering the n previous words, the frequency and probability of each n-grams can be calculated, which makes n-grams model perform well in text classification and word prediction with short documents [29]. However, n-grams performs poorly with long documents due to the rapid growth of dimensionality with large n, and it has only restricted access to the words that appear in the document.

Later models using word embeddings such as Word2Vec solve the problems of earlier models to some extent by representing words in vector spaces, and words with similar meanings such as "walk" and "run" have closer distance in the vector space [30]. Word embeddings successfully reduces the dimensionality of word representations and can work with larger documents by combining techniques such as "Continuous Bags of Words" [30].

With the development of neural networks, more powerful models such as the Seq2Seq model began to take a more important role in many application fields such aspects machine translation [31]. The core idea of the Seq2Seq model is that it uses the Long Short-Term Memory networks as an encoder to map the input to a vector with fixed dimensionality, and then uses another LSTM to decode the output sentence from the vector. The use of LSTM makes it possible for the model to handle long input sequences and the encoder-decoder structure enables the model to manage different lengths for input and output sequences [31].

Large language models start to have a huge leap forward after the introduction of the transformer architecture [15]. [33] demonstrates that the transformer architecture can effectively improve the language understanding ability of language models even with large unlabeled text corpora by first generatively pre-training the model on the text data and then discriminative fine-tuning them on various tasks. Modern models usually own a large number of parameters and large-scale training datasets, but the source of the training datasets can vary from model to model. For example, according to [32], GPT-2 (with 1,5 billion parameters) is trained on massive training datasets including the crawling results from millions of different web pages, called WebText. Where BERT (BERT$_{BASE}$ with 110 million parameters, BERT$_{LARGE}$ with 340 million parameters) is trained on the English Wikipedia (2,500 million words) and BookCorpus (800 million words), a large dataset of books used for machine learning approaches [16]. A more recent model collection called LLaMA (with 7 to 65 billion parameters) is trained on various open-source data such as commoncrawl[1], C4, Github, Wikipedia etc [50]. Table 1 provides a brief overview of the 3 models together with information of the most recent GPT3.5 and GPT4 models according to the OpenAI blog[2].

The parameter size of different models has a growing trend, which can be explained by the relationship between model performance, the number of model parameters N, the size of dataset D and the amount of computing to train the model C. [34] demonstrates the **Smooth Power Laws** which says: "Performance has a power-law relationship with each of the three scale factors N, D, C when not bottlenecked by the other two, with trends spanning more than six orders of magnitude". Nowadays, there are various models that have distinct capabilities, for example, DALL-E can generate and modify images based on text input, whereas ChatGPT 3.5 can generate code and natural language, as [35] in figure 1 summaries.

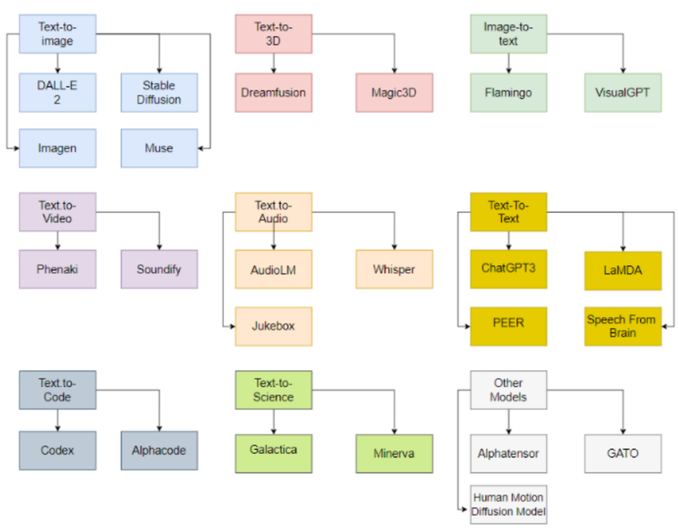


Fig. 1: An overview of the recent major generative AI models [35]

The Architecture and Training Process of Large Language Models
Many of the recent large language models such as ChatGPT, LLaMA and BERT

[1] https://commoncrawl.org/
[2] https://platform.openai.com/docs/models/gpt-base

Model Name	BERT	GPT2	LLaMA	GPT3,5/ 4
Developer	GoogleAI	OpenAI	MetaAI	OpenAI
Release Date	2018	2019	2023	2022/2023
Nr. of Parameters	110 M/ 340 M	1,5 B	7-65 B	175 B/1,7 T
Training Data	Wikipedia(en) & BookCorpus	WebText	Various open-source datasets	WebText
Open-sourced	Yes	No	Yes	No
Major Applications?	QA	Text generation & Translation	Text generation & QA	Content generation & QA

Table 1: Comparison of Large Language Models

are based on transformer-based neural networks. The transformer architecture, as [15] in figure 3 demonstrates, is an encoder-decoder architecture. The encoder takes the input sequence and converts them into dense vectors of fixed size at the input embedding step. Then with the help of the positional encoding step, the input sequence contains information about each token in the sequence. The encoded inputs are then passed through the multi-head attention mechanism so that the model can capture various relationships and features from the input sequence. Then the Add and Normalization layer is applied to stabilize and speed up the training process. In the end, the Feed Forward layer is applied to help in introducing non-linearity and learning complex representations for each token's position in the sequence. Similarly, the decoder uses the vector representation of the input sequence created by the encoder along with the previously generated tokens to produce the next token in the output sequence and the masked multi-head attention mechanism is applied to ensure that the prediction for a particular position depends only on the known outputs at earlier positions.

Various input text datasets must be pre-processed as valid input representations before they can be used for the training process, and the data pre-processing includes the following steps [28]:

- Tokenization involves segmenting text into tokens, which are the basic units of broken text. Tokenization can simplify and standardize the input data and improve model performance [16].
- Subword encoding refers to breaking down the input text into smaller units and helping handle rare or out-of-vocabulary words in the input text.
- Data cleaning is the step where the noisy information in the input data should be removed, which can significantly improve the quality and suitability of the input data for the model.

More specifically, BERT uses the WordPiece embeddings [51] with a 3000 token vocabulary. For each input, a classification token ($[CLS]$) is used to mark the beginning of the sentence and a ($[SEP]$) token is used to mark the end of a sentence. In some cases, several sentences are combined as one single sentence, therefore a new embedding is needed to mark to which sub-sentence each token belongs [16]. As figure 2 shows, an example input sentence "My dog is cute, he likes playing." is first tokenized into smaller units, where ($[CLS]$) marks the start of the input and ($[SEP]$) marks the end of the sentence as well as splits the sentence into 2 parts. A token embedding is added for each token, a segment embedding is added to show to which sentence each token belongs and a position embedding is used to show the position of each token within the sentence. The final input representation is the sum of all embeddings.

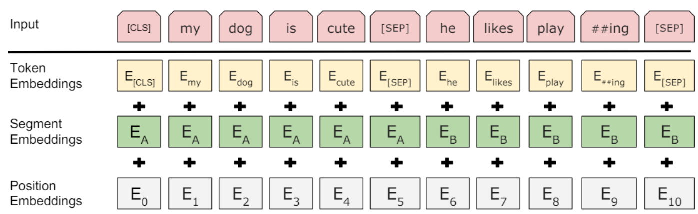


Fig. 2: Input representation of BERT [16]

GPT2 and LLaMA use byte pair encoding (BPE) [52] to tokenize the input text where BPE iteratively count the frequency of the characters or subwords and merge the most frequent ones to form new vocabulary [32], [50]. There are several different implementations of BPE that are moderate for different use cases, the training corpus on which BPE is applied also significantly influences the size of the result vocabulary. GPT2 applies BPE with a base vocabulary of more than 130,000 tokens, whereas the BPE applied on LLaMA creates around 1,4T tokens.

A model starts its training algorithm after taking the pre-processed data as its input. For recent models such as ChatGPT, LLaMA and BERT, their training algorithms share a same process of an unsupervised pre-training phase and a supervised fine-tuning phase [28]. In the pre-training phase, a large amount of unlabeled WebText data is used as the model's training dataset to make it a high-capacity language model. Given an unlabeled corpus of tokens $U = \{u_1, ..., u_n\}$ as a training dataset, the core idea of the pre-training phase is to predict the next token u_i for a sequence $\{u_{i-k}, ..., u_{i-1}\}$. According to [33], this is done by maximizing the likelihood:

$$L_1(U) = \sum_i \log P(u_i | u_{i-k}, ..., u_{i-1}; \Theta) \tag{1}$$

k refers to the context window size and Θ is the parameter of the neural network with which the conditional probability P is modeled.

The supervised fine-tuning phase is used to improve the model's performance on specific tasks by training it on a smaller corpus of labeled data [28]. Given a labeled dataset C, where each instance of C has a sequence of tokens $\{c^1, ..., c^m\}$ and a label y, the goal of the fine-tuning phase is to maximize the following likelihood [33]:

$$L_2(U) = \sum_{(x,y)} \log P(y|x^1, ..., x^m) \tag{2}$$

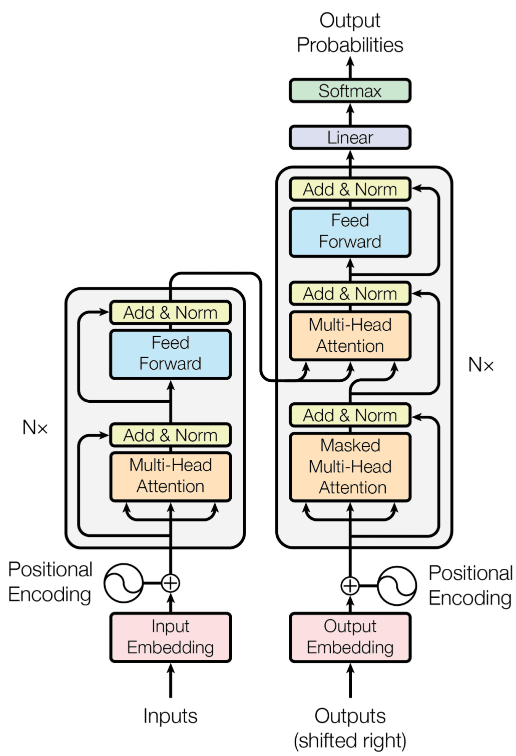


Fig. 3: The transformer structure [15]

In the BERT model, as shown in figure 4, the pre-training phase consists of 2 unsupervised tasks: Masked LM and Next Sentence Prediction (NSP). Masked LM is used to help BERT understands the context of words within a sentence, it masks 15% of all WordPiece tokens in each input sentence and the model predicts only the masked words instead of reconstructing the entire input. NSP is applied to help BERT understand the relationship between 2 sentences. More specifically, when having 2 sentences A and B as input, B is replaced by another random sentence from the training set for half of the time and remains unchanged for other half of the time. The BERT is trained to predict whether the second sentence in the input is the actual sentence B. According to [16], NSP has significant benefits for question answering (QA).

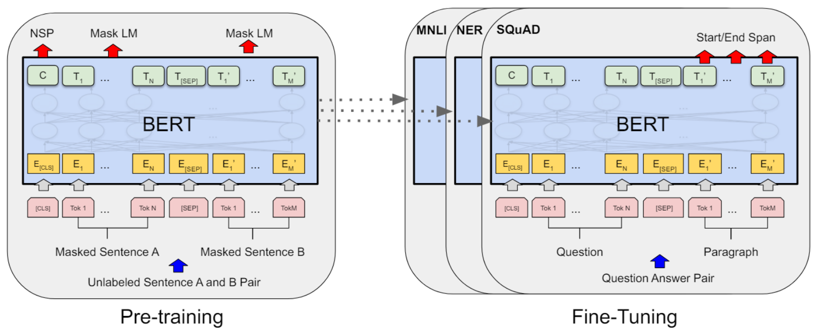


Fig. 4: Training process in the BERT model [16]

2.2 Data Stream

In many application fields nowadays, data is often generated and processed at a very high rate. These include health sensors that track body temperature and blood pressure [13], monitoring of production lines, information on stock trading, posted content on social networks etc. [14]. Due to its near real-time and huge amounts characteristics, traditional data management approaches are often not suitable anymore, a new system called Data Stream Management Systems (DSMS) has evolved to address this issue [14].

Data stream has many informal definitions, [36] describes data stream as "time-varying, volatile, unpredicted and possibly unbounded information". [37] describes data stream as "a data set that is produced incrementally over time, rather than being available in full before its processing begins". According to [13], data stream has the following formal definition:

Definition 1. *A data stream S is an unbounded, potentially infinite multiset of data stream elements (s, τ), where $\tau \in \mathbb{T}$. $\mathbb{T}$ is a timestamp attribute with values from a monotonic, infinite time domain $\mathbb{T}$ with discrete time units.*

There are multiple methods to analyze data streams, [38] categorizes them into data-based and task-based solutions. Data-based solution refers to examining a subset of the data stream and approximating to an overall outcome for the data stream. Task-based solutions, on the other hand, often refer to achieving time and space efficiency in computing by using relevant techniques. We summarize some of the solutions:

- Synopsis is a data-based solution and it refers to the process of creating compact data summaries such as wavelet analysis and histograms, which can efficiently approximate and query large data streams without processing the entire dataset [38].
- Window is a task-based solution and it refers to adjusting a continuous data stream into bounded segments of data (windows), each window is executed over the items of a single data stream and returns a concrete and finite data segment [36].

3 LLMs with Data Stream

Although large language models have shown strong capabilities in many application fields, their knowledge remains static as they are pre-trained on static datasets. In some occasions, however, a user query expects up-to-date results. Considering the following questions: *Who is the current president of the USA?* The answer to such a question expires likely every 4 years, which seems a reasonable time for an update of the large language model's training datasets. The question *What are the results of the last Bundesliga match day?* would require the knowledge of large language models to be updated at least every week. The question *What is the current traffic condition in the Königstraße of city Aachen?* would require even a near real-time result. Thus, in many application fields, the performance of a pre-trained large language model is likely to gradually degrade over time [39]. In this section, we will first summarize some use cases where large language models are combined with data streams, then We will give a comparison of large language models using static datasets and those updated with data streams. Finally, we will summarize some common issues and challenges in this field as well as the measurements and solutions.

3.1 Continual Learning

The need for regular updates of large language models brings intense focus to the concept of **Continual Learning**.

Continual Learning of LLMs
According to [40], continual learning refers to the process of accumulating knowledge on non-stationary data. For a data stream of tasks $\mathcal{T} = \{\mathcal{T}_1, \ldots, \mathcal{T}_n\}$, the goal is to have the model learn sequentially based on the input stream, where it only has access to $\mathcal{T}_i$ at time i. [9], in their continual learning framework for large language models, categorizes continual learning into 3 stages as figure 5 shows:

- Continual Pre-training: It refers to incrementally updating a large language model on input data over time without retraining it from scratch.
- Continual Instruction Tuning: It refers to continuously refining the model's ability to follow the instruction.

- Continual Alignment: It refers to adjusting the model so it always produces output that satisfies certain standards and guidelines.

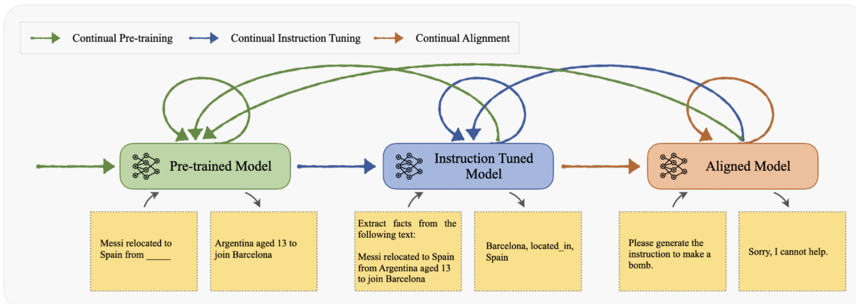


Fig. 5: Stages of continual learning [9]

Furthermore, [39] demonstrates that there are various methods of computing continual learning while mitigating catastrophic forgetting [24] and categorize them as follows:

- Replay-Based Methods: The model keeps a small subset of previous tasks and re-trains them periodically.
- Regularization-Based Methods: The capacity of the model is typically fixed, and the model adds a regularization term (typically a loss function) that reflects the consolidation of previous knowledge.
- Architecture-Based Methods: Change the architecture of the model for each new task and introduce new parameters that fit the task.

Challenge of CL

A critical challenge of continual learning is catastrophic forgetting, which refers to the models forgetting previously learned knowledge when they are updated with new data [24]. Catastrophic forgetting happens in the continual pre-training stage as the model is continuously updated with a data stream. As discussed earlier, there are currently 3 types of methods that can mitigate catastrophic forgetting [39]. The architecture-based methods often face the problem of the growing training cost, as their models' parameter size grows with new tasks linearly[43]. The regularization methods avoid the expanding parameters challenge but the computational cost increases due to the extra computation of the regularization term [44]. Therefore, the replay-based methods are often considered to be a better approach considering large data stream input. Although the choice of selecting the right exemplars from previous tasks can still be difficult [43].

Another challenge of continual learning is concept drift, where the relationship and distribution of input data change over time [40]. Concept drift often causes the large language models' performance to decrease due to the discrepancies between the models' output based on previous data and current factual accuracies, societal norms and standards. Therefore the continual alignment stage of continual learning is essential to mitigate the effect of concept drift, as it continuously refines the model to align with the current standard [45]. However, to ensure the right alignment of the model, huge computational resources and human oversight are often required. Additionally, aligning a model to changes without introducing new bias and the protection of data privacy and security are also challenges to the continuous alignment stage [39].

Use Cases

Continual learning is already applied in the field of linguistics, one example is from [48], where they discovered the fact that some large language models are trained on English datasets and their performance in other languages differs from their performance in English. They came up with the idea of developing a large language model that has strong performance in Japanese and decided to apply continual learning methods, especially the continual pre-training stage, on an English LLM Llm2 due to the high computational cost of training a LLM in Japanese from scratch. The idea was to apply continual learning so that the Llma2 model can adapt its knowledge and capabilities to Japanese and to construct a model called Swallow in the end.

According to [48], Swallow preserves the same architecture as Llma2, such as its number of attention heads and its number of layers. To prevent catastrophic forgetting, they applied the replay-based method by selecting 5% of the training data from English RefinedWeb [49], 5% from the English arXiv paper texts and the remaining 90% from Japanese texts. After developing Swallow, they evaluate its performance in various fields such as question answering (QA), reading comprehension (RC), automatic summarization (AS), arithmetic reasoning (AR), commonsense reasoning (CR) and machine translation(MT) and compare its performance to the original Llma2 model. The results showed that despite the large data stream input, Swallow managed to have a better performance in many fields compared with Llma2, especially in question answering task as figure 6 shows.

3.2 Prompt Engineering

Definition

[46] defines a prompt as "a set of instructions provided to an LLM that programs the LLM by customizing it and/or enhancing or refining its capabilities" and prompt engineering as "the means by which LLMs are programmed via prompts". More specifically, Prompt engineering is the process of providing large language models with effective inputs or queries to elicit specific, high-quality responses [6]. It has been proven to have a positive influence on the performance of many downstream tasks such as text summarization, sentiment analysis, and natural language translation [42]. Considering the following prompt:

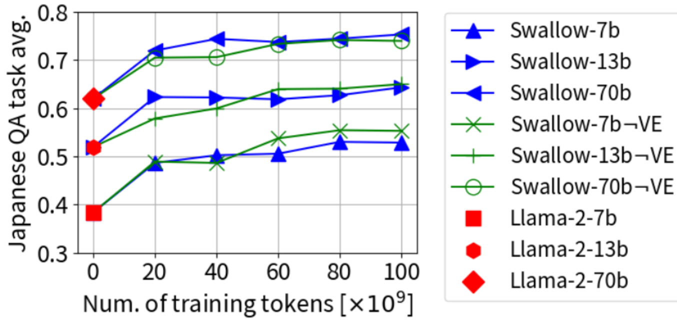
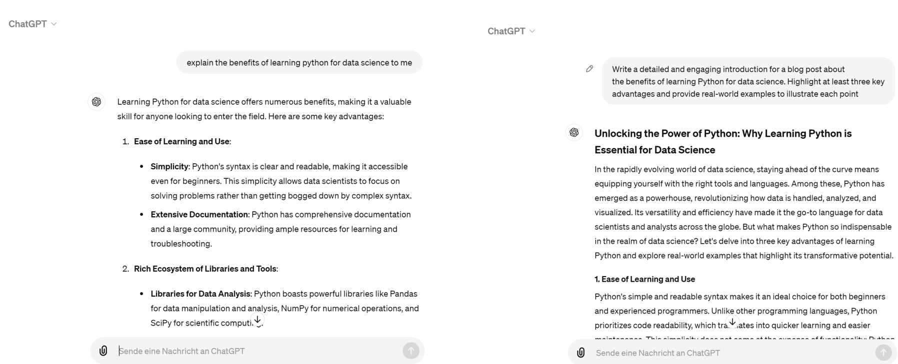


Fig. 6: Performance curve of the Swallow model in question answering task [48]

Example 1. "Write a detailed and engaging introduction for a blog post about the benefits of learning Python for data science. Highlight at least three key advantages and provide real-world examples to illustrate each point."

By giving the prompt, the large language model is required to give an engaging, detailed and structured answer to the topic of the benefits of learning Python for data science. Figure 7 shows an example of how different prompts guide ChatGPT to provide different outputs over the same topic.



(a) Prompt 1 (b) Prompt 2

Fig. 7: 2 different prompts yield different output of ChatGPT

Challenges

[47] demonstrates the quality of the prompt provided by users as the major factor that affects the output quality of large language models using prompt engineering Furthermore, they demonstrate that prompt formulation should follow the

specific prompt pattern and summarize the components of a prompt pattern as follows:

- A name and classification: It uniquely identifies the prompt pattern and indicates the issue being addressed.
- The intent and context: It describes the issue.
- The motivation: It explains the importance of this issue.
- The structure and key ideas: They provide the LLM with important contextual information.
- Example implementation: It provides an example prompt pattern that can be referenced for other pattern formulations.
- Consequences: It explains the possible effect of this prompt pattern.

However, [47] also points out that for different issue categories such as "Alternative Approaches Pattern" (which guides the model to output alternative options) and "Fact Check List Pattern" (which guides the model to provide a list of facts), the specific implementation of each component could differ.

Use Cases

In the field of traffic signal control, [41] proposes a model called LLMLight based on large language models to act as a decision-making agent. As figure 8 shows, An LLMLight agent is responsible for controlling the traffic light at an intersection. Real-time observations about the traffic condition are converted into human-readable text and transferred to the model, followed by the prompt engineering formed by the real-time observations, task descriptions and control action. [41]

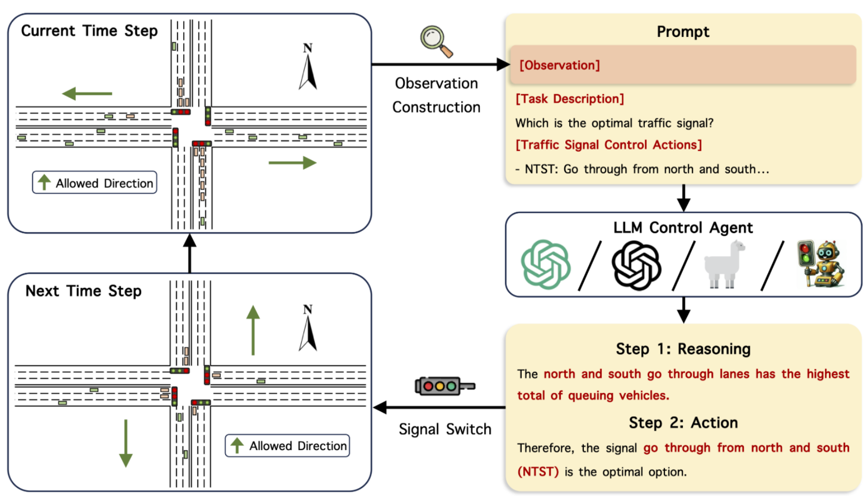


Fig. 8: The workflow of LLMLight [41]

also formulates the workflow as the following equation:

Definition 2. $(Y, a) = \pi_\theta(Prompt(o, d_{scene}, d_{task}, d_{know}, A))$.

Where π_θ refers to the traffic signal control policy, (Y, a) refers to the reasoning trajectory and the action output by the model, based on the real-time traffic condition observation o, traffic scenario description d_{scene}, task descriptions d_{task} , commonsense knowledge d_{know}, and control action space A. Figure 9 shows an example template of the prompt [41].

[Observation]

<traffic condition on the go through lanes>, <traffic condition on the left-turn lanes>

[Traffic Scenario Description]

A traffic light regulates a four-section intersection with northern, southern, eastern, and western sections...

[Task Description]

Which is the optimal traffic signal?

[Commonsense Knowledge]

You should prioritize lanes with long queue length, while pay less attention to vehicles in distant segments.

[Traffic Signal Control Action Space]

NTST (go-through from north and south), NLSL (turn-left from north and south)...

Fig. 9: Prompt template [41]

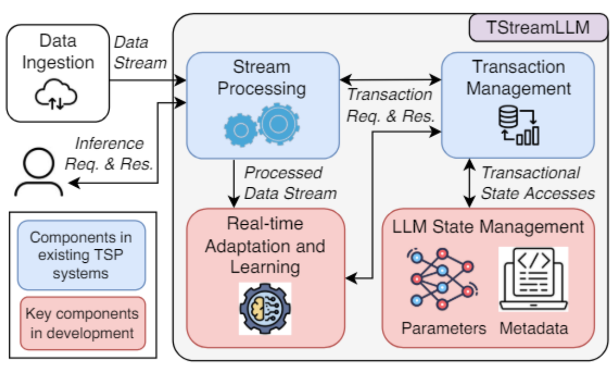


Fig. 10: Architecture of TStreamLLM (draft) [6]

4 Conclusion

This paper discussed the gap between large language models, which are pre-trained on massive static datasets, and data streams, which are continuous and potentially unbounded. Two approaches were presented to mitigate the issue:

- Continual learning of large language models enables the model to continually pre-train over data stream input and by applying continual instruction tuning and continual alignment, the model can adapt itself to the continuous data input. However catastrophic forgetting is likely to occur during this process, so replay-based, regularization-based and architecture-based methods are applied to mitigate the effect of catastrophic forgetting.
- In some application fields, another option is to convert the data stream input into model-readable input and update the model's knowledge via prompt engineering. However, this method is heavily relied on the quality of the prompt formulated by users.

References

1. Liu, Yiheng, Tianle Han, Siyuan Ma, Jiayue Zhang, Yuanyuan Yang, Jiaming Tian, Hao He et al. "Summary of chatgpt-related research and perspective towards the future of large language models." Meta-Radiology (2023): 100017.
2. Kasneci, Enkelejda, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser et al. "ChatGPT for good? On opportunities and challenges of large language models for education." Learning and individual differences 103 (2023): 102274.
3. Thirunavukarasu, Arun James, Darren Shu Jeng Ting, Kabilan Elangovan, Laura Gutierrez, Ting Fang Tan, and Daniel Shu Wei Ting. "Large language models in medicine." Nature medicine 29, no. 8 (2023): 1930-1940.
4. Chang, Yupeng, Xu Wang, Jindong Wang, Yuan Wu, Linyi Yang, Kaijie Zhu, Hao Chen et al. "A survey on evaluation of large language models." ACM Transactions on Intelligent Systems and Technology (2023).
5. Zhang, Shuhao, Xianzhi Zeng, Yuhao Wu, and Zhonghao Yang. "Harnessing scalable transactional stream processing for managing large language models [vision]." arXiv preprint arXiv:2307.08225 (2023).
6. Zhang, Kunpeng, Feng Zhou, Lan Wu, Na Xie, and Zhengbing He. "Semantic understanding and prompt engineering for large-scale traffic data imputation." Information Fusion 102 (2024): 102038.
7. Xu, Xuhai, Bingshen Yao, Yuanzhe Dong, Hong Yu, James Hendler, Anind K. Dey, and Dakuo Wang. "Leveraging large language models for mental health prediction via online text data." arXiv preprint arXiv:2307.14385 (2023).
8. Zhang, Xin, Linhai Zhang, Deyu Zhou, and Guoqiang Xu. "Fine-grainedly Synthesize Streaming Data Based On Large Language Models With Graph Structure Understanding For Data Sparsity." arXiv preprint arXiv:2403.06139 (2024).
9. Wu, Tongtong, Linhao Luo, Yuan-Fang Li, Shirui Pan, Thuy-Trang Vu, and Gholamreza Haffari. "Continual learning for large language models: A survey." arXiv preprint arXiv:2402.01364 (2024).

10. Brown, Tom, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D. Kaplan, Prafulla Dhariwal, Arvind Neelakantan et al. "Language models are few-shot learners." Advances in neural information processing systems 33 (2020): 1877-1901.
11. Gama, Joao, Raquel Sebastiao, and Pedro Pereira Rodrigues. "On evaluating stream learning algorithms." Machine learning 90 (2013): 317-346.
12. Jang, Joel, Seonghyeon Ye, Sohee Yang, Joongbo Shin, Janghoon Han, Gyeonghun Kim, Stanley Jungkyu Choi, and Minjoon Seo. "Towards continual knowledge learning of language models." arXiv preprint arXiv:2110.03215 (2021).
13. Geisler, Sandra. "Data stream management systems." In Dagstuhl Follow-Ups, vol. 5. Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik, 2013.
14. Geisler, Sandra. "A systematic evaluation approach for data stream-based applications." PhD diss., Dissertation, RWTH Aachen University, 2016, 2016.
15. Vaswani, Ashish, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. "Attention is all you need." Advances in neural information processing systems 30 (2017).
16. Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
17. Yenduri, Gokul, M. Ramalingam, G. Chemmalar Selvi, Y. Supriya, Gautam Srivastava, Praveen Kumar Reddy Maddikunta, G. Deepti Raj et al. "GPT (Generative Pre-trained Transformer)–A Comprehensive Review on Enabling Technologies, Potential Applications, Emerging Challenges, and Future Directions." IEEE Access (2024).
18. Lester, Brian, Rami Al-Rfou, and Noah Constant. "The power of scale for parameter-efficient prompt tuning." arXiv preprint arXiv:2104.08691 (2021).
19. So, David, Quoc Le, and Chen Liang. "The evolved transformer." In International conference on machine learning, pp. 5877-5886. PMLR, 2019.
20. Zhao, Feng, Xinning Li, Yating Gao, Ying Li, Zhiquan Feng, and Caiming Zhang. "Multi-layer features ablation of BERT model and its application in stock trend prediction." Expert Systems with Applications 207 (2022): 117958.
21. Ren, Yilong, Yue Chen, Shuai Liu, Boyue Wang, Haiyang Yu, and Zhiyong Cui. "TPLLM: A Traffic Prediction Framework Based on Pretrained Large Language Models." arXiv preprint arXiv:2403.02221 (2024).
22. Liu, Chenxi, Sun Yang, Qianxiong Xu, Zhishuai Li, Cheng Long, Ziyue Li, and Rui Zhao. "Spatial-temporal large language model for traffic prediction." arXiv preprint arXiv:2401.10134 (2024).
23. Yang, Xi, Aokun Chen, Nima PourNejatian, Hoo Chang Shin, Kaleb E. Smith, Christopher Parisien, Colin Compas et al. "A large language model for electronic health records." NPJ digital medicine 5, no. 1 (2022): 194.
24. Gupta, Kshitij, Benjamin Thérien, Adam Ibrahim, Mats L. Richter, Quentin Anthony, Eugene Belilovsky, Irina Rish, and Timothée Lesort. "Continual Pre-Training of Large Language Models: How to (re) warm your model?." arXiv preprint arXiv:2308.04014 (2023).
25. Naga Sanjay, "Continuous Training of ML models. A case-study on how to keep our machine learning models relevant.", Medium, June 25, 2023
26. Prapas, Ioannis, Behrouz Derakhshan, Alireza Rezaei Mahdiraji, and Volker Markl. "Continuous training and deployment of deep learning models." Datenbank-Spektrum 21, no. 3 (2021): 203-212.
27. Araci, Dogu. "Finbert: Financial sentiment analysis with pre-trained language models." arXiv preprint arXiv:1908.10063 (2019).

28. Roumeliotis, Konstantinos I., and Nikolaos D. Tselikas. "Chatgpt and open-ai models: A preliminary review." Future Internet 15, no. 6 (2023): 192.
29. Cavnar, William B., and John M. Trenkle. "N-gram-based text categorization." In Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval, vol. 161175, p. 14. 1994.
30. Mikolov, Tomas, Ilya Sutskever, Kai Chen, Greg S. Corrado, and Jeff Dean. "Distributed representations of words and phrases and their compositionality." Advances in neural information processing systems 26 (2013).
31. Sutskever, Ilya, Oriol Vinyals, and Quoc V. Le. "Sequence to sequence learning with neural networks." Advances in neural information processing systems 27 (2014).
32. Radford, Alec, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. "Language models are unsupervised multitask learners." OpenAI blog 1, no. 8 (2019): 9.
33. Radford, Alec, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. "Improving language understanding by generative pre-training." (2018).
34. Kaplan, Jared, 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).
35. Gozalo-Brizuela, Roberto, and Eduardo C. Garrido-Merchan. "ChatGPT is not all you need. A State of the Art Review of large Generative AI models." arXiv preprint arXiv:2301.04655 (2023).
36. Patroumpas, Kostas, and Timos Sellis. "Window specification over data streams." In International Conference on Extending Database Technology, pp. 445-464. Berlin, Heidelberg: Springer Berlin Heidelberg, 2006.
37. Golab, Lukasz, and M. Tamer Özsu. "Issues in data stream management." ACM Sigmod Record 32, no. 2 (2003): 5-14.
38. Gaber, Mohamed Medhat, Arkady Zaslavsky, and Shonali Krishnaswamy. "Mining data streams: a review." ACM Sigmod Record 34, no. 2 (2005): 18-26.
39. Shi, Haizhou, Zihao Xu, Hengyi Wang, Weiyi Qin, Wenyuan Wang, Yibin Wang, and Hao Wang. "Continual Learning of Large Language Models: A Comprehensive Survey." arXiv preprint arXiv:2404.16789 (2024).
40. Biesialska, Magdalena, Katarzyna Biesialska, and Marta R. Costa-Jussa. "Continual lifelong learning in natural language processing: A survey." arXiv preprint arXiv:2012.09823 (2020).
41. Lai, Siqi, Zhao Xu, Weijia Zhang, Hao Liu, and Hui Xiong. "Large language models as traffic signal control agents: Capacity and opportunity." arXiv preprint arXiv:2312.16044 (2023).
42. Liu, Xiao, Kaixuan Ji, Yicheng Fu, Weng Lam Tam, Zhengxiao Du, Zhilin Yang, and Jie Tang. "P-tuning v2: Prompt tuning can be comparable to fine-tuning universally across scales and tasks." arXiv preprint arXiv:2110.07602 (2021).
43. Jovanović, Mlađan. "Towards Incremental Learning in Large Language Models: A Critical Review."
44. Wang, Yifan, Yafei Liu, Chufan Shi, Haoling Li, Chen Chen, Haonan Lu, and Yujiu Yang. "InsCL: A Data-efficient Continual Learning Paradigm for Fine-tuning Large Language Models with Instructions." arXiv preprint arXiv:2403.11435 (2024).
45. Taori, Rohan, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. "Alpaca: A strong, replicable instruction-following model." Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html 3, no. 6 (2023): 7.

46. Liu, Pengfei, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. "Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing." ACM Computing Surveys 55, no. 9 (2023): 1-35.
47. White, Jules, Quchen Fu, Sam Hays, Michael Sandborn, Carlos Olea, Henry Gilbert, Ashraf Elnashar, Jesse Spencer-Smith, and Douglas C. Schmidt. "A prompt pattern catalog to enhance prompt engineering with chatgpt." arXiv preprint arXiv:2302.11382 (2023).
48. Fujii, Kazuki, Taishi Nakamura, Mengsay Loem, Hiroki Iida, Masanari Ohi, Kakeru Hattori, Hirai Shota, Sakae Mizuki, Rio Yokota, and Naoaki Okazaki. "Continual Pre-Training for Cross-Lingual LLM Adaptation: Enhancing Japanese Language Capabilities." arXiv preprint arXiv:2404.17790 (2024).
49. Penedo, Guilherme, Quentin Malartic, Daniel Hesslow, Ruxandra Cojocaru, Alessandro Cappelli, Hamza Alobeidli, Baptiste Pannier, Ebtesam Almazrouei, and Julien Launay. "The RefinedWeb dataset for Falcon LLM: outperforming curated corpora with web data, and web data only." arXiv preprint arXiv:2306.01116 (2023).
50. Touvron, Hugo, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière et al. "Llama: Open and efficient foundation language models." arXiv preprint arXiv:2302.13971 (2023).
51. Wu, Yonghui, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun et al. "Google's neural machine translation system: Bridging the gap between human and machine translation." arXiv preprint arXiv:1609.08144 (2016).
52. Sennrich, Rico, Barry Haddow, and Alexandra Birch. "Neural machine translation of rare words with subword units." arXiv preprint arXiv:1508.07909 (2015).