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, I will talk about the following aspects of large language models:

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. According to [32], large language models such as GPT-2 (with 1,5 billion parameters) are trained on massive training datasets including the crawling results from millions of different web pages, called WebText. As for 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 [35].

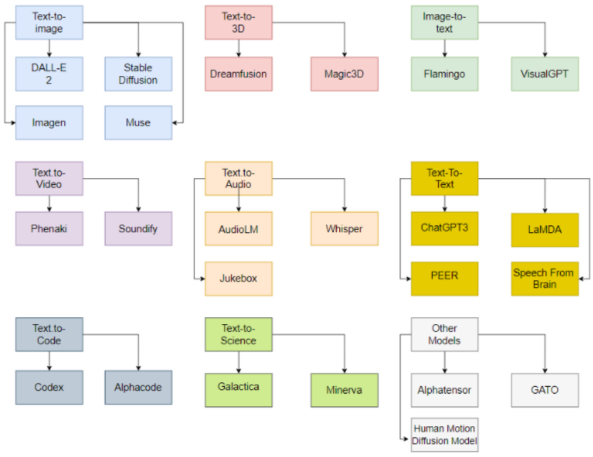

Fig. 1. An overview of the recent major generative AI models [35]

The architecture and training process of large language models
As discussed before, large language models such as ChatGPT and BERT are based on transformer-based neural networks. The transformer architecture, as figure 2 shows, is an encoder-decoder architecture. According to [15], 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 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.

The model starts its training algorithm after taking the pre-processed data as its input. For models such as ChatGPT and BERT, their training algorithm consists 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 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]:

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

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].

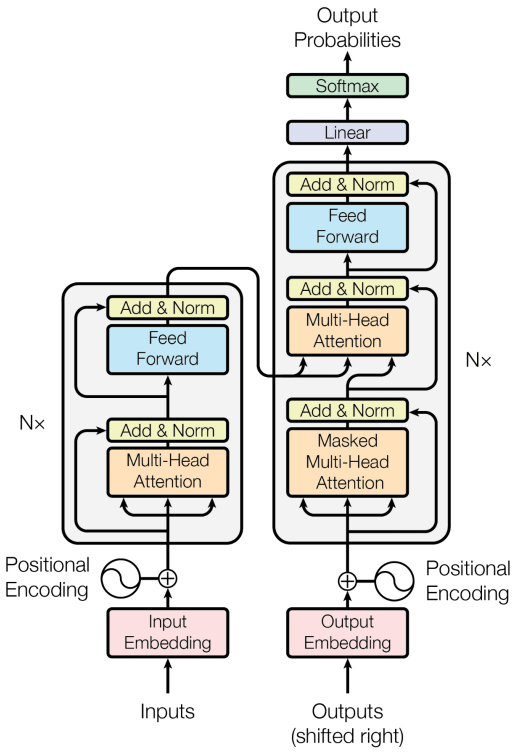

Fig. 2. The transformer structure [15]

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 histograms, which can efficiently approximate and query large data streams without processing the entire dataset [38].
- Sliding window is a task-based solution

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 3 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.

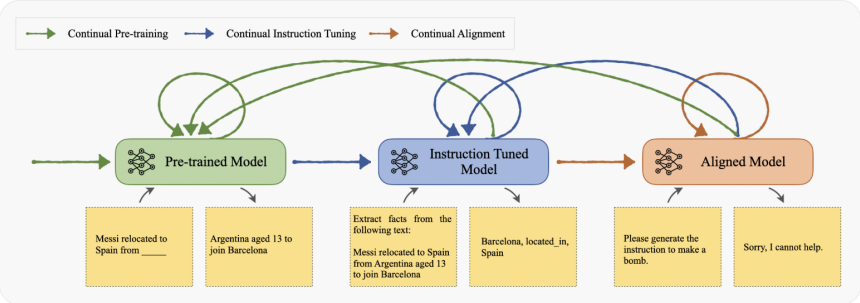

Fig. 3. Stages of continual learning [9]

According to . . . , CL has been proven to be an efficient method . . .

use cases?

3.2 prompt engineering?

promt engineering

use cases

3.3 Use Cases

Following use cases will be covered:

- Social Media Monitoring: Analyzing streaming data from social media platforms can help monitor regional news, sentiment, and trends in real-time.

- Financial Monitoring: Large language models can analyze streaming news feeds to identify important events, trends, and sentiment in real-time. This can be valuable for financial institutions and risk management companies to stay informed about current events and market trends.
- Health Care Monitoring: Large language models can analyze streaming medical data such as patient records, diagnostic reports, and research papers to assist healthcare providers in diagnosing diseases, identifying treatment options, and monitoring public health trends in real-time.
- Traffic Data Monitoring: Large language models can analyze streaming traffic data to monitor traffic conditions in real-time to help reduce traffic accidents and level up transportation efficiency.

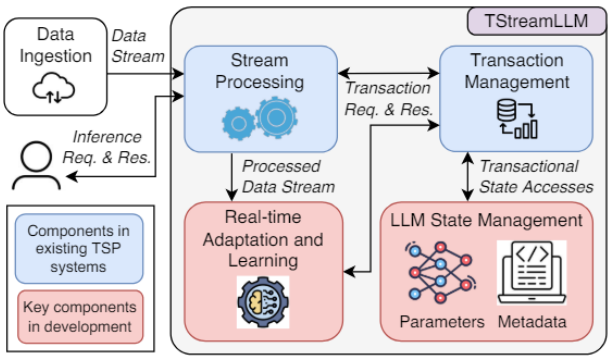

Fig. 4. Architecture of TStreamLLM [6]

3.4 Challenges

Following challenges will be covered:

- Catastrophic forgetting [24]
- Concept drift: if the LLM is trained for a long time, it is possible that the relationship between the inputs and the outputs itself might change.
- Computational resources: continuous learning requires significant computational resources, which may be costly or impractical to scale.
- Privacy and security: streaming data often contains sensitive or private information, raising concerns about data privacy and security.
- Data quality: data streams may contain noisy or unreliable information, leading to incorrect model updates.

3.5 Solutions

Following solutions will be covered:

- Finetuning [16].
- Continual pre-training: continuously pretraining models on new incoming data [24].
- Using data preprocessing techniques to filter out noise and ensure data quality. Use anomaly detection algorithms to identify and remove outliers in the data. Employ quality assurance measures to verify the accuracy of incoming data.
- Continuously monitor model performance and detect concept drift using statistical methods or machine learning algorithms. Implement adaptive learning techniques to update the model in response to concept drift. Periodically retrain the model on recent data batches to maintain accuracy.
- Optimize model architectures and algorithms to reduce computational overhead. Utilize distributed computing frameworks to parallelize model training and inference tasks.
- Implement data anonymization and encryption techniques to protect sensitive information during data transmission and storage

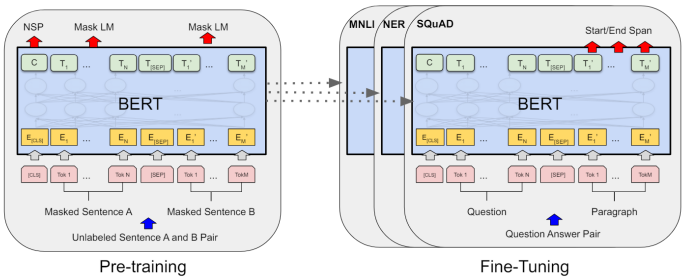

Fig. 5. Finetuning in the BERT model [16]

4 Conclusion

In this section, I will shortly summarize the outline of this paper and also talk about the possible future development of large language models using data streams.

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).