Automated mobile operator customer service using large language models combined with RAG system

Vladimir A. Lovtsov[1], Maria A. Skvortsova[2]
[1, 2] Bauman Moscow State Technical University
[1, 2] Moscow, Russian Federation
[1]lovtsovva@student.bmstu.ru, [2]magavrilova@bmstu.ru

***Abstract*—Large Language Models (LLMs) have made a real breakthrough in the field of artificial intelligence and have been rapidly integrated into our daily lives, including the telecom domain. The use of Retrieval Augmented Generation (RAG) reduces the likelihood of generating incorrect or outdated data in LLMs and improves the understanding of the query context and the generation of more relevant answers. The aim of this research is to improve the quality of automated customer service for mobile operator customers by using RAG system in combination with different language models. The paper makes a comprehensive analysis of open-source LLMs, the main methods of model adaptation and pre-training. Conclusions are drawn on the applicability of the analyses in the study. The architecture of the RAG system and the deployment diagram are designed. The main stages of system training are described, system results and performance evaluation for different LLMs are given. Major conclusions are drawn on the achievement of the research objective and further development.**

Keywords—RAG, LLM, machine learning, mobile operator, language model adaptation, telecoms

I. INTRODUCTION

The rapid growth of the large language model market in various applications has been accelerated by advances in Transformer Architecture (Generative pre-trained transformers (GPT), Bidirectional Encoder Representations from Transformers (BERT), Bidirectional Encoder Representations from Transformers (LLaMA)) [1,2]. Meanwhile, the artificial intelligence market incorporating LLMs is predicted to reach over $1 trillion by 2030 [2]. The sectors with active LLM adoption have been finance, medicine, telecom, education, and manufacturing. The use of large language models in telecom and mobile operators allows to automate the processing of customer queries, including FAQ and technical support, provide personalised recommendations and improve the quality of service, optimise costs, reducing the load on call centres. But besides the obvious advantages, there are limitations in the form of the risk of "hallucinations" (inaccurate answers), high computational cost, complexity of integrations with CRM and handling sensitive data [3–5]. In addition, not many models allow working with Russian as the main language of the country where the solution is implemented [6,7]. To memorialise the relevance of the problem and to find an optimal solution, it is proposed to analyse modern artificial intelligence tools that address the limitations and complexities described above.

TABLE I. COMPARATIVE ANALYSIS OF LLMS

Metric/Benchmark	Qwen2.5-7B	Mistral-7B	Llama3-8B	Interpretation
MMLU	74,2	64,2	66,6	Qwen2.5-7B significantly outperforms the competition, demonstrating a high level of proficiency across disciplines
MMLU-Pro	45,0	30,9	35,4	Qwen2.5-7B leads again, showing better ability to solve professional problems
MMLU-Redux	71,1	58,1	61,6	A simplified version of MMLU, where Qwen2.5-7B shows good versatility
BBH	70,4	56,1	57,7	Qwen2.5-7B shows the best ability to solve complex non-standard tasks
ARC-C	63,7	60,0	59,3	All models show similar results, but the Qwen2.5-7B maintains a slight advantage
Trurhfulqa	56,4	42,2	44,0	Qwen2.5-7B answers questions most "truthfully," minimising fictitious or erroneous facts
Winogrande	75,9	78,4	77,4	Mistral-7B shows the best performance on the lexico-syntactic ambiguity resolution task
Hellaswag	80,2	83,3	82,1	Mistral-7B is in the lead, but Qwen2.5-7B performs close to the competition
Multi-Exam	59,4	47,1	52,3	Qwen2.5-7B performs better on the multidisciplinary exam tasks
Multi-Understandingt	79,3	63,3	68,6	Qwen2.5-7B understands complex multitasking queries best

The LLM Qwen2.5-7B was the leader on most tasks, leading in the MMLU, MMLU-Pro, BBH, Multi-Exam, TruthfulQA and Multi-Understanding metrics, demonstrating deep knowledge of a wide range of disciplines and the ability to handle professional and complex tasks.

TABLE II. COMPARISON OF THE MAIN LLM ADAPTATION AND PRE-TRAINING METHODS

Adaptation/Retraining Method	Description	Advantages	Disadvantages	Examples of LLM
Fine-Tuning (Full training)	Complete model refinement on specialised data, including updating of all parameters	High accuracy for narrow tasks	Requires large computational resources	GPT, BERT, LLaMA, Qwen2.5-7B-Instruct

2025 7th International Youth Conference on Radio Electronics, Electrical and Power Engineering (REEPE) | 979-8-3315-3183-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/REEPE63962.2025.10971107

Adaptation/Retraining Method	Description	Advantages	Disadvantages	Examples of LLM
LoRA (Low-Rank Adaptation)	Retraining of a small submatrix of parameters, keeping the original parameters unchanged	Full control over the model	Risk of overtraining on small datasets	GPT-3, LLaMA-2, Qwen2.5-7B-Instruct
Adapters	Adding additional modules (adapters) to the model, which are trained independently from the main structure	Suitable for complex domains	Limited ability to make large changes to the model	BERT с адаптерами (AdapterHub), Qwen2.5-7B-Instruct
Prompt Tuning	Retraining special textual hints (prompts) that optimise model predictions	Memory and computational savings	Can degrade performance on complex tasks due to isolated adapter training	GPT-3, GPT-4, T5, Qwen2.5-7B-Instruct
Prefix Tuning	The pre-training of additional “prefixes” in the hidden state space of the model	Easy integration	Less effective for complex tasks	GPT-3, GPT-J, Qwen2.5-7B-Instruct
Few-Shot Learning	Using multiple examples in a prompt to adapt the model to a new task without pre-training	Suitable for multimodal tasks	Dependence on quality of prompts	GPT-3, GPT-4, PaLM, Qwen2.5-7B-Instruct
Zero-Shot Learning	Executing a new task without pre-training or examples in prompt	Minimal changes to the model	Limited flexibility for new domains	GPT-3, GPT-4, Qwen2.5-7B-Instruct
Instruction Tuning	Instruction-based (task-based) pre-training of the model to increase its versatility in problem solving	Resource saving	Requires fine-tuning	GPT-3.5, GPT-4, Flan-T5, Qwen2.5-7B-Instruct
Reinforcement Learning (RLHF)	Machine learning with human feedback to optimise the model's response	Improves responses in terms of human perception. Increases consistency and security	Limited flexibility for complex domains.	ChatGPT (InstructGPT), Claude, Qwen2.5-7B-Instruct
RAG	Integration with search engines or knowledge bases to extract relevant information and use it in the model's answers	Access to up-to-date information Reducing dependence on training data. Interpretability	Dependent on format and quality of samples.	GPT-4+RAG, Qwen2.5-7B-Instruct+RAG
Distillation	Creation of a smaller model based on a larger model trained on its predictions	Reducing the size of the model Maintaining productivity. Suitable for implementation on devices	Often inferior to fine-tuning and few-shot approaches.	DistilBERT, TinyBERT, MobileBERT

As shown in Table 2—Qwen2.5-7B-Instruct is optimised for instructions, few-shot learning and multimodal tasks, and supports adaptation via RAG, LoRA and Instruction Tuning. RAG improves accuracy through external data, while other adaptation methods (Fine-Tuning, RLHF, Distillation) help to balance performance, accuracy and resources. As a further study, it is proposed to select RAG system as a tool for model adaptation and pre-training.

The use of RAG system by itself does not provide clear advantages [9,10], including in telecom, as it is aimed at helping to find the right information in a large amount of disparate information. At the same time, if we use a combination of LLM and RAG-system we can get not just generative AI, but intelligent search with the ability to explain and work with relevant information.

Based on the described relevance and research problems, the aim of the work is to investigate the possibility of improving the efficiency of the mobile operator's support service, reducing the workload and reducing costs by applying.

Different LLMs will be used as the main LLM in combination with RAG.

II. Materials and Methods

A. Data Collection and Training Sample Generation

The adaptor-based pre-training approach requires a labelled training sample on which to improve the quality of the LLM. Due to the initial lack of resource for manual markup of the sample, it was decided to generate the marked-up data also with the help of the LLM. In the process of pre-training language models, especially in the context of automated customer service systems, an important step is the creation of a high-quality dataset. This dataset should contain markup in a question-answer-context-source format, which allows the model in a RAG system to be trained efficiently on examples that are as close as possible to real user interaction scenarios [2,11,12].

Firstly, it is necessary to select the model that will be used to generate the synthetic data. The Qwen2.5-7B-Instruct model was chosen. This decision is due to the fact that using a different model for generating the markup avoids overtraining on the specific patterns and features of the model that will be used for pre-training (in this case Llama3). Different models may have different architectures and approaches to text processing, which contributes to a more diverse and extensive dataset.

Synthetic data generation is a key step that allows to create the necessary number of samples for training the model. Synthetic data can be created based on pre-trained samples that generate queries to the Qwen2 model [13,14,15]. The advantages of using synthetic partitioning are as follows:

- Time and resource savings.
- Content control: by generating the data independently, the researcher can control the quality and relevance of the generated examples, which is especially important for specific tasks.
- Elimination of bias: the use of synthetic data avoids the bias that can arise when using real data collected from limited sources.

After generating synthetic data, it is necessary to perform data cleaning. Cleaning includes removing unnecessary spaces, line breaks and special characters as well as possible

hallucinations. Cleaning simplifies further processing and analysis of the data, allowing you to focus on the content.

Validation of the resulting markup is an important stage of dataset preparation. It involves checking the quality of the generated questions and answers against the specified criteria. The following methods can be used for this: manual validation, automated metrics (text quality metrics such as BLEU or ROUGE can be used to assess similarity to the reference data).

In order to successfully generate quality questions and answers, it is necessary to follow certain rules when writing prompts [16–18]:

- Clarity: Prompts should be clear and unambiguous.
- Context: a prompt contains sufficient context to generate an adequate response.
- Variety: different formulations of the prompts to create more varied samples.

B. User Feedback

Feedback from users plays an important role in improving customer experience, especially in the context of chatbots. Effective use of this information can significantly improve customer satisfaction and optimise system performance [19–21].

Feedback can come from users through various channels, including surveys after interacting with a chatbot, feedback forms on the website or app, and through messages in the chatbot itself. Collecting this data provides insight into how well the bot is performing, what issues users are having, and what aspects of the service need improvement.

Evaluating feedback starts with systematising it. It is important to create a structure for data collection that allows you to categorise feedback according to different criteria: positive, negative and neutral. Positive feedback can provide information about what users found useful in interacting with the bot, while negative feedback often points to specific problems or flaws. Neutral feedback can help identify areas that need additional attention.

Natural language processing techniques such as tone analysis can be used to analyse the feedback collected. For example, if many users mention problems with getting information about tariffs, this may signal the need to refine the relevant section of the chatbot.

Based on the analysed feedback, it is possible to start generating markup for training the model. This involves creating question-answer pairs, where questions are formulated based on user queries and answers are based on recommendations or corrections suggested as a result of the analysis [22,23,24].

The process of generating the markup can look like this: first, the most frequent questions need to be extracted from user feedback. Then, for each question, an answer is generated, which can be based on existing data or new recommendations to improve the chatbot performance. For example, if users often ask about ways to connect additional services, you can create a response with step-by-step instructions.

This markup will serve as a basis for pre-training the language model. By using real questions and answers from feedback, a more relevant and customised dataset can be created.

After generating the markup, the next step is to pre-train the language model in the RAG system. To do this, the dataset needs to be prepared in a question-answer-context-source format. The context can include information about the current state of the user's query or previous interactions with the bot.

It is important to note that the model should be further trained on a variety of examples so that it can adequately respond to different communication scenarios. For this purpose, both synthetic data (e.g., manually generated) and real examples from collected feedback can be used.

C. Designing the Architecture of the RAG System

To create a RAG system based on a mobile operator's support chatbot, several steps should be followed. First, it is necessary to define a set of documents for the knowledge base, which will include frequently asked questions, instructions for using the operator's services, tariff plans and other useful information. The initial knowledge base, which is a text description of tariff plans, services, etc., is broken down into documents (small pieces of contextual information).

An example document is shown in Fig. 1.

```
   How will the subscriber find out about activation
of automatic renewal? The subscriber will receive a
push notification when the service is activated.
   When switching to a new tariff, the remaining
packages will be retained.
   Distribution of wi-fi traffic from the Internet
package is not available.
```

Fig. 1. An example document.

After that, it is necessary to implement a mechanism for encoding documents into a vector representation. This can be done using BERT model or similar technology. We consider 3 pre-trained models capable of working with Russian language:

- Multilingual-e5-large;
- Paraphrase-multilingual-mpnet-base-v2;
- Ru-en-RoSBERTa.

In the next step, a search engine should be developed to retrieve relevant documents according to user queries, i.e., documents should be ranked.

When a user asks a question via a chatbot, the system must first process this query and extract relevant snippets of information from the database. These fragments are then combined with the original query to create context before being passed to the generative model.

The retrieval of the fragments of information (document ranking) is done in 2 stages: the first one works fast and allows to browse through the entire knowledge base, returning potential candidates from it. The second stage of re-ranking already uses a heavier model on a limited set of candidates, thus allowing to select the most relevant documents for the context.

The first stage uses the BM25 model, also all documents are indexed using FAISS and vector search is applied using it. Each document obtained from vector search and BM25 is assigned a score based on its rank and method weight. These scores are combined to take into account both the quality of

the results and their order. The most relevant documents that have been evaluated taking into account the information from both approaches are returned, which significantly improves the quality of the output (Fig. 2).

```
   1) With any non-negative account, you can change
the tariff to <name>. The tariff change is available
not more often than once a day.
   2) Upon activation of the tariff <name> new
subscribers are given a package of 10 SMS-messages
as a present
   3) Tariff <name> is a restart of tariff
<old_name>.
```

Fig. 2. Examples of documents retrieved by the query 'Tariff <name>'

The generative model generates a response based on the received context and returns it to the user. Example of model operation (Fig. 3):

```
   Subscriber: I recently changed from Tariff
<name_another> to Tariff <name>, why didn't I get
10 SMS as a gift?
   Assistant: When you activate the tariff <name>
10 SMS messages as a gift are provided only to new
users.
```

Fig. 3. Example of model operation

The architecture of the system is shown in Fig. 4.

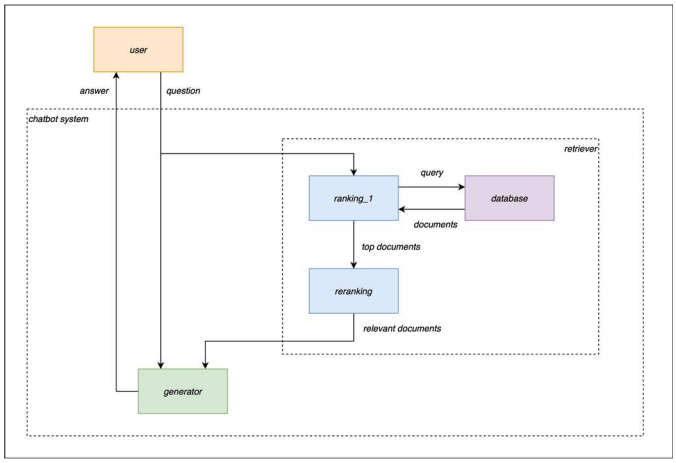

Fig. 4. Architecture of the RAG system.

D. System Deployment

Model deployment can be done through GitLab CI/CD, which automates the process. Each time code is updated in the GitLab repository, a pipline is run, which includes the following steps(see Fig. 5):

- Code conformance checking using flake8, pylint, formatting with black;
- Container build: using Docker, an image with the application and LLM model is created;
- Testing: tests are run to verify the correctness of the application;
- Deploy to Kubernetes: the image is uploaded to the Kubernetes cluster, where the necessary pods and services are created.

Kubernetes can automatically change the number of pods to adapt to load changes. This is achieved with the Horizontal Pod Autoscaler (HPA), which allows the RAG system to efficiently handle changes in the volume of user requests.

How it works:

- Metrics monitoring: the HPA periodically checks metrics, such as GPU and memory usage, that may be related to the load on the LLM models. For example, if the number of bot requests increases, this may lead to an increase in resource utilisation.
- Automatic scaling: if HPA detects that resource utilisation exceeds a given threshold (e.g., 70% of available video memory resources), it automatically increases the number of pods. This allows the system to handle more requests simultaneously.
- Load reduction: when load is reduced (e.g., during nighttime or low activity periods), HPA can reduce the number of pods, freeing up resources for other applications or tasks.

Pods are logically aggregated into services that distribute the load across their pods. In this system, we can distinguish 2 scalable services: the response generator (generator) and the context finder (retriever).

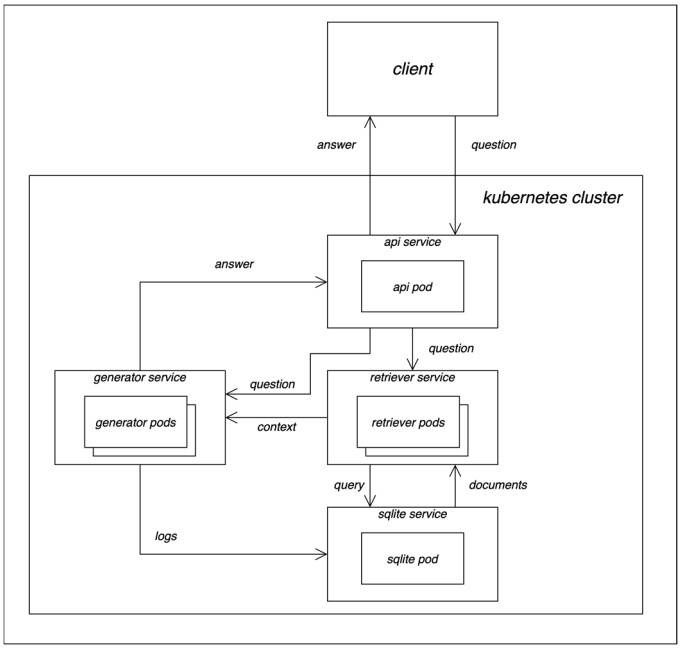

Fig. 5. Shows the diagram of the server part of the system.

III. Results and Discussions

A. Training the Model

The Qwen2.5-7B-Instruct model was applied to generate data for pre-training models to the documents into which the knowledge base was partitioned. The model received as input a query with an instruction to generate questions and answers for a given document. The query specified an instruction describing the requirements for the generated questions and answers (among them completeness, accuracy, complexity, and absence of references).

The model generated 1000 question-answer pairs, which were split 80:20 into training and test samples.

The unsloth library was used to pre-train the LLM models. Unsloth uses LoRA to efficiently pre-train the language models, which significantly reduces memory overhead and speeds up the learning process. The lora_alpha parameter is used to scale the impact of weight updates from low rank

matrices. For example, higher values can speed up convergence but also increase the risk of overtraining.

A plot of Cross Entropy loss when the Llama-3.1-8B-Instruct model is pre-trained is shown in Fig. 6. The error functions decrease smoothly and no significant overtraining is observed.

The pre-training was performed in 4 epochs using SFTPTrainer (a tool designed for efficient pre-training of language models). The optimiser is AdamW, learning rate—0,0001, total number of steps—432.

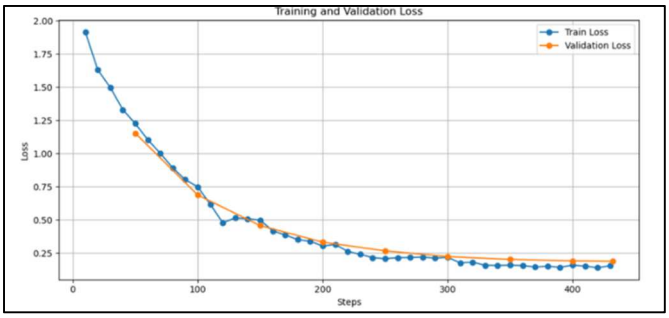

Fig. 6. Cross-Entropy in training.

B. Evaluation of Results

Table 3 shows the comparison of ranking metrics depending on the feature extractor used.

TABLE III. COMPARISON OF RANKING METRICS DEPENDING ON FEATURE EXTRACTOR

	Mean Reciprocal Rank 2 Stage	Mean Reciprocal Rank 1 stage	Hit Rate2 Stage	Hit Rate1 Stage
E5-Large	**0,857253**	**0,841821**	**0,737654**	**0,731481**
Mpnet-Base-v2	0,794753	0,739198	0,643519	0,600309
Ru-En-RoSBERTa	0,841049	0,829475	**0,737654**	0,729938

Based on the above comparison, the best metrics are demonstrated by the e5-large model.

The metrics after stage 2 are higher than after stage 1, indicating the success of document re-ranking.

Table 4 shows the comparison of the three LLMs as well as their versions pre-trained on the training sample (6 models in total).

TABLE IV. COMPARISON OF LLM METRICS ON THE TEST SAMPLE

Model	Rouge1	Rouge2	RougeL	RougeLsum	Bert Score	BLEU
Suzume-Llama-3-8B-Multilingual	0,374876	0,278882	0,340330	0,347897	0,778443	0,234797
Saiga_Llama3_8b	0,347504	0,246291	0,312871	0,322525	0,767656	0,208757
Llama-3.1-8B-Instruct	0,402977	0,293591	0,365334	0,373075	0,779547	0,245728
Tuned Suzume-llama-3-8B-Multilingual	0,394475	0,298895	**0,384866**	0,386778	0,775223	0,260982
Tuned Saiga_Llama3_8b	0,405763	0,288787	0,372295	0,375311	0,772003	0,251227
Tuned Llama-3.1-8B-Instruct	**0,415970**	**0,312779**	0,383478	**0,387015**	**0,788938**	**0,277320**

Based on the above comparison, we can conclude that the pre-training was successful (all models show an increase in quality metrics). The best results are shown by the retrained Llama 3.1.

In the future, it is possible to implement a mechanism for evaluating model responses using another model (LLM Judge).

C. Performance Evaluation

The Llama 3.1 model requires approximately 16 GB of VRAM to run, which is adequate for most modern GPUs.

Multilingual-E5-Large: This model has 24 layers and 560 million parameters, requiring approximately 2 GB of video memory to run text processing and embedding tasks. It is optimised to run on standard GPUs.

BERT-Multilingual-Passage-Ranking-MSMARCO: This model also requires approximately 4 GB of video memory depending on the specific implementation and configuration. It can handle input data with up to 512 tokens in length. Deployment will require a graphics card with 32 GB or more.

Response time will depend on the complexity of processing each request:

- Llama 3.1: Response time can range from 1 to 3 seconds per request due to the high computational load.
- Multilingual-E5-Large: The expected response time is around 200–500 milliseconds, making it fast for most applications.
- BERT-Multilingual-Passage-Ranking-MSMARCO: The response time is around 300–700 milliseconds, which also makes it efficient in responding to user requests.

Thus, with a 32GB graphics card, a RAG system with these models can deliver acceptable performance given the trade-offs between response time and the number of requests processed per second.

IV. CONCLUSION

LLMs are becoming an integral part of the economy, transforming key industries through automation, personalisation and the adoption of new technologies. However, successful implementation requires attention to cost, data quality and security issues.

This work investigated how language models can be adapted in automated customer service systems. A RAG system with a large language model pre-trained on synthetic markup was developed to support mobile network operator subscribers. The key aspects of large language models, their business applications, and methods of adaptation and pre-training were analysed. In the context of the task, it is optimal to use a pre-trained Llama-3.1-8B-Instruct, with pre-training using adapters.

The use of Large Language Models (LLM) in combination with Retrieval-Augmented Generation (RAG) can improve the quality and efficiency of the mobile operator's customer service. RAG provides extraction of relevant data from external sources, while LLM generates accurate, relevant and personalised responses based on this data.

Further improvements to the developed system are planned for the future. One of the directions is the introduction of methods for assessing the quality of responses using LLM Judge, which will allow to measure the model performance more accurately and identify areas for improvement. Also, an important aspect is to guard against "hallucinations," a situation where the model generates non-existent or incorrect information. Another important aspect would be to investigate heavier weight models as well as their quantised versions to improve the generation of training data and system responses.

In summary, Retrieval-Augmented Generation systems with LLMs represent a powerful tool for building high quality chatbots capable of providing accurate and relevant answers to user queries. They provide a significantly improved customer experience by integrating relevant information from external sources and minimising errors in response generation. Despite some implementation challenges, the advantages of RAG make it an attractive choice for companies looking to improve the efficiency of their customer support services.

REFERENCES

[1] J. Vishnevskaya and B. Salyp, "Comparison of the applicability of synergistic models with dense neural networks on the example of mobile device security," *International Conference on Digital Transformation: Informatics, Economics, and Education*, p. 27, Apr. 2023, doi: 10.1117/12.2680842.

[2] S. M. Jammoul, V. V. Syuzev, and A. M. Andreev, "Open Source Software Usage in Education and Research," *Research Anthology on Usage and Development of Open Source Software*, pp. 273–288, 2021, doi: 10.4018/978-1-7998-9158-1.ch015.

[3] Hugging Face. Documentation. 2025. [Online]. Available: https://huggingface.co/docs.

[4] OpenI. *Materials on GPT-3/4 capabilities, application of RAG and multimodal models*. 2023. [Online]. Available: https://openai.com/news/.

[5] LangChain. *Documentation*. 2025. [Online]. Available: https://python.langchain.com/docs/introduction/.

[6] P. Lewis, E. Perez, A. Piktus et al., "Retrieval-augmented generation for knowledge-intensive nlp tasks," *Advances in Neural Information Processing Systems*, no. 33, pp. 9459–9474, 2020, doi.org/10.48550/arXiv.2005.11401.

[7] Google AI. *AI Research*. 2025. [Online]. Available: https://ai.google/advancing-ai/research/.

[8] Microsoft Research. *LLM in Customer Support*. 2025. [Online] Available: https://www.microsoft.com/en-us/research/.

[9] S. Roychowdhury, S. Soman, H. G. Ranjani et al., *Evaluation of rag metrics for question answering in the telecom domain*. 2024. arXiv preprint arXiv:2407.12873.

[10] A. L. Bornea, F. Ayed, A. De Domenico et al., *Telco-RAG: Navigating the challenges of retrieval-augmented language models for telecommunications*, 2024, arXiv preprint arXiv:2404.15939.

[11] E. J. Hu, Y. Shen, P. Wallis et al., "Low-rank adaptation of large language models," *arXiv preprint arXiv:2106.09685*, 2021, doi: doi.org/10.48550/arXiv.2106.09685.

[12] The Technology. Media & Telecommunications AI Dossier. 2025. [Online] Available: https://www2.deloitte.com/us/en/pages/consulting/articles/ai-dossier-technology-media-telecommunications.html.

[13] S. I. Suyatinov, T. I. Buldakova, and J. A. Vishnevskaya, "Identification of Situations Based on Synergetic Model," in *2021 3rd International Conference on Control Systems, Mathematical Modeling, Automation and Energy Efficiency*, pp. 509–514, Nov. 2021, doi: 10.1109/summa53307.2021.9632207.

[14] P. Martynyuk, I. Kozlov, and A. Panfilkin, "Applying the Proposed Method for Creating Structural Models to Multilingual Collections of Text Documents Using Multi- and Monolingual BERT Models," *Advances in Automation V*, pp. 334–343, 2024, doi: 10.1007/978-3-031-51127-1_32.

[15] A. Vaswani, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[16] Qwen Team. Qwen2.5: A Party of Foundation Models. Nov. 25, 2024. [Online]. Available: https://qwenlm.github.io/blog/qwen2.5/.

[17] A. Yang et al, "Qwen2 technical report," *arXiv preprint arXiv:2407.10671*, 2024.

[18] D. D. Kosheleva and I. I. Davydov, "Transfer learning in machine learning," *Actual issues of fundamental and applied scientific research*, pp. 239–241, 2023.

[19] .M. Riviere, A. Joulin, P.-E. Mazare et al., "Unsupervised Pretraining Transfers Well Across Languages," in *ICASSP 2020—2020 IEEE International Conference on Acoustics, Speech and Signal Processing*, May 2020, doi: 10.1109/icassp40776.2020.9054548.

[20] Z.-C. Chen, Y.-S. Sung, and H.-Y. Lee, "Chapter: Exploiting Convolutional Neural Network Adapters for Self-Supervised Speech Models," in *2023 IEEE International Conference on Acoustics, Speech, and Signal Processing Workshops*, pp. 1–5, Jun. 2023, doi: 10.1109/icasspw59220.2023.10193427.

[21] J. Guo, Z. Zhang, L. Xu et al., "Adaptive Adapters: An Efficient Way to Incorporate BERT Into Neural Machine Translation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 29, pp. 1740–1751, 2021, doi: 10.1109/taslp.2021.3076863.

[22] Y. Gao et al. "Retrieval-augmented generation for large language models: A survey," 2023, *arXiv preprint arXiv:2312.10997*.

[23] J. White et al., "A prompt pattern catalog to enhance prompt engineering with chatgpt," 2023, *arXiv preprint arXiv:2302.11382*.

[24] K. Shuster et al., "Retrieval augmentation reduces hallucination in conversation," *arXiv preprint arXiv:2104.07567*, 2021.