Записки научных
семинаров ПОМИ
Том 540, 2024 г.

V. Pavliukevich, A. Zherdeva, O. Makhnytkina, D. Dyrmovskiy

IMPROVING RAG WITH LORA FINETUNING FOR PERSONA TEXT GENERATION

Abstract. We address the challenge of maintaining consistency in Retrieval-Augmented Generation (RAG) systems for persona text generation when databases are subject to rapid updates and conventional large language model (LLM) fine-tuning is inadequate. We propose an approach that enhances an existing RAG system used for persona-based information retrieval in dialogue agents through the application of Low-Rank Adaptation fine-tuning on synthetic data. We find that this method improves the system's logic and correctness by 5% on SSA scores and ensures that generated content remains more coherent and contextually relevant.

§1. Introduction

Retrieval-Augmented Generation (RAG) systems have emerged as a key technology in the realm of natural language processing, particularly for applications where information retrieval needs to be dynamic as underlying databases are subject to rapid changes. These systems combine the capabilities of Large Language Models (LLMs) with retrieval mechanisms, enabling them to fetch and incorporate external knowledge into concise assistant-like responses. This fusion empowers RAG systems to provide more informed and contextually relevant outputs, which is particularly valuable in domains where up-to-date information is crucial [1]. RAG systems are helpful when building personified dialogue agents as they can retrieve relevant facts about the persona from the database as the dialogue agent interacts with the user. Since the current generation of LLMs have limited context windows, all the facts about persona may not fit into them. RAG comes in as a solution to this problem.

Key words and phrases: retrieval-augmented generation and large language models and fine-tuning.

Despite their advantages, RAG systems are not without their challenges. One of the primary concerns is the LLMs' tendency to generate inconsistent or inaccurate information, a phenomenon often referred to as "hallucination". It happens when the model generates plausible but factually incorrect or misleading content, which can erode user trust and diminish the system's utility. In the context of RAG systems, hallucinations lead to the model's inability to use retrieved context when it is necessary to do so. Ensuring the reliability and consistency of generated content is a critical area of focus in the ongoing development of RAG systems.

Our research introduces a novel application of Low-Rank Adaptation (LoRA) fine-tuning to enhance the performance of a RAG-based dialogue agent designed for persona information retrieval [2]. Unlike conventional LLMs that might struggle with maintaining consistency and up-to-date knowledge, integration of LoRA fine-tuning with a RAG architecture aims to address these limitations, offering a more reliable and coherent dialogue experience. The concept of a "persona" in dialogue agents refers to a set of attributes, behaviors, and styles that define the character or role the agent embodies.

In the following sections, we detail specific methodologies employed in this research, including the generation of synthetic datasets, applications of LoRA fine-tuning to the RAG architecture, and the evaluation framework used to assess the model's performance. Through this exploration, we aim to contribute to the advancement of dialogue systems, particularly in contexts where maintaining a consistent and accurate persona is paramount.

§2. Related works

Traditional dialog systems consist of building blocks such as dialog state tracking components and response generators, and have typically been applied to tasks with labeled internal dialog state and precisely defined user intent. All of these methods do not normally consider dialog history and are more concerned with achieving functional goals, such as booking an airline ticket or a restaurant table, than manifesting a persona. In particular, many of the available tasks and datasets are limited to narrow domains. One method considers two classes of models for predicting the next utterance: ranked models and generative models [3]. Ranked models generate the next utterance by considering any utterance in the training set as a possible candidate for an answer. Generative models generate new

sentences by determining the history of the dialog and then generating the answer word by word. In this way, the models are trained to both ask and answer personal questions, and the resulting dialog can be used to build a personality model of the speaking partner.

Memory neural networks have been proposed to be used in personalized dialogue systems [4]. Key-value profile memory networks have also been proposed as an enhancement of the memory network. They assign attention weights to keys and combine values. Here, we apply this model to dialog and consider keys as dialog histories from the training set and values as the next dialog utterances, i.e., responses of the interlocutor. This allows the model to have a memory of past dialogues that it can directly use to influence its prediction for the current conversation. For personalization, models for generating dialogues with a variety of personality traits have also been proposed, i.e., they are models that allow capturing and incorporating personality traits into the dialog generation process [5]. These models employ a feature fusion module to obtain a representation of the speaker's persona and two approaches to consider persona-related features in the decoding process: namely, a persona-based attention mechanism that dynamically generates context vectors conditioned on the persona representation and a persona-based bias.

The next approach, presented by Madotto et al. [6], applies meta-learning to personalize dialog agents without conditioning the model's response to the persona description. The model learns to adapt to new personas using dialog samples collected from the same user, unlike the common approach based on training the response to a persona description.

The PLATO model, presented in [7], is trained in two phases. In the first phase, the model was trained only on one-to-one matching, i.e., only one response is generated for each context. In the second phase, a latent variable that has categorical values is introduced, and each variable corresponds to a specific latent speech act in the response. The model estimates the distribution of latent acts in the training sample and then generates a response with the selected latent variable. Both of these tasks are performed in the same model. The model can generate different responses, but it is necessary to select the most relevant one by ranking this set.

When developing dialogue agents, the application of deep learning techniques provides an opportunity to effectively use datasets to discover new

strategies for generating responses [8]. To give a dialogue agent a personality it is necessary to achieve a certain similarity of its dialogues with human ones. It is worth noting that the creation of agents with a certain personality trait is an important aspect in the field of artificial intelligence development, especially in the context of their application in various domains such as education, medicine, entertainment and technical support. Distinctive personality traits can significantly increase user satisfaction and improve the quality of interaction with dialogue agents. When creating a personalized dialogue agent it is important to focus on datasets that contain not only dialogue data but also persona labels. One of the first such datasets is PersonaChat for English [9]. It offers descriptions of fictional personalities, each described in five sentences talking about their profession, favourite movies, hobbies, etc. For example: "I am from the north. I like swimming. I enjoy nature walks. They call me a bean counter. Autumn is my favourite season." For Russian, Toloka Persona Chat Rus contains profiles of more than 1,500 virtual personas and over 10,000 dialogues between the participants of a conversational artificial intelligence study conducted at MIPT. However, all these datasets have drawbacks. The dialogues most often deal only with given personality characteristics, which limits the models' ability to respond to replicas that are not related to these characteristics. Moreover, personality characteristics in these datasets usually refer to interests and hobbies, and do not reflect temperament, emotional reactions, or other personality traits. In addition, the volume of such datasets is limited, making it difficult to build large generative models that focus on specific personality traits, also many of the corpora have been collected synthetically, i.e., generated by other neural networks, which does not guarantee good quality model training. To personalize the dialogue agent, a Russian language dataset Toloka Persona Chat Rus was selected and converted into a format to work with Large Language Model (LLM). Here is an example of the converted dataset:

- "persona bot": "persona1 data";
- "persona user": "persona2 data";
- "example dialogue": ["role": "user", "content": "", "role": "bot", "content": ""] - sample dialogues between two personas.

The selection of the fine-tuning method for enhancing our RAG system was guided by the need for low resource consumption, suitable for a low budget graphics card for effective task-specific training. In our exploration of available solutions, LoRA and other Parameter-Efficient Fine-Tuning

(PEFT) methods stood out due to their suitability for our context, characterized by computational resource constraints and a desire to maximize the efficacy of the available data [10].

PEFT methods, including LoRA, aim to optimize fine-tuning processes, ensuring that they are as resource-efficient as possible [11]. LoRA, in particular, offers a compelling approach by enabling modifications to specific model weights, thereby tailoring the model's behavior to our specialized task—persona-based information retrieval in dialogue systems without necessitating a large increase in trainable parameters.

LoRA's approach to fine-tuning allows training only a subset of the model's pre-trained parameters, with the rest remaining unchanged. This methodology not only conserves resources but also retains the model's original architecture. When applied across all weight matrices, LoRA's training can achieve a level of expressiveness comparable to full model training.

This fine-tuning method proves particularly advantageous in reducing memory and storage consumption, a critical factor when working with constrained resources [12]. For example, using LoRA can significantly reduce the VRAM usage during training sessions and decrease the checkpoint size by a considerable factor, enabling efficient training on systems with limited GPU capabilities. For example, when training a large Transformer with Adam, VRAM usage can be reduced to 2/3 if $r \leqslant$ model, as there is no need to store optimizer states for frozen parameters. On a GPT-3 model with 175 billion parameters, VRAM consumption during training is reduced from 1.2 TB to 350 GB. Additionally, with $r = 4$ and adapting only the query and value projection matrices, the checkpoint size is reduced by about 10,000 times (from 350 GB to 35 MB).

§3. Methodology

The methodology of our research focuses on improving a Retrieval-Augmented Generation (RAG) system for a persona-based dialogue agent through Low-Rank Adaptation (LoRA) fine-tuning. This section outlines the steps taken to generate synthetic datasets, apply LoRA fine-tuning to the RAG architecture, and evaluate the model's performance.

3.1. Persona RAG Pipeline. RAG is a method that combines two types of memory: one as prior knowledge of the model, the other as a

retrieval system, making it more intelligent in accessing and utilizing information. In essence, RAG bridges the gap between static LLM knowledge and dynamic, up-to-date information. The whole process of implementing RAG in LLMs can be summarized in the following steps:

- user submits a query or message to the model;
- RAG system extracts the information + cross-encoder BERT reranks the documents by obtaining sentence similarity scoring;
- RAG system generates a response using LLM;
- a coherent and contextually relevant response based on the extracted information and the user's query is returned as the output.

Thus, RAG should be used when it is necessary to augment the knowledge base with data that was not known to the LLM during training, such as personal data or contextual information useful for generating answers in a particular domain. Combining Retrieval Augmented Generation technology with Large Language Models (LLMs) not only extends the capabilities of the LLM, but also ensures that the answers generated are contextually relevant and based on the most up-to-date information. In our case, RAG is used to store persona data and dialogues as an example of using persona facts. The rubert-tiny2 model, which is a small Russian encoder model based on BERT, was used to obtain vectors [13]. This model provides the ability to efficiently obtain vector representations for various tasks.

Further, the obtained vectors are loaded into the vector database FAISS [14]. This is a library for efficient similarity search and clustering of dense vectors. It contains algorithms for searching in vector sets of any size. Next, the stored vector can be converted into a Retriever class, which allows to use this method as a tool whose task is to search for the most relevant context based on a user query.

The second step is to configure the RAG method for personalization. To interact with the large language model, we used LangChain, a high level open source platform built on top of LLM inference modules and designed for applications such as chatbots, generative question-answering systems, summarization and more[1]. The basic idea behind the library is to combine different components into chains to create more advanced LLM usage scenarios. Chains can consist of multiple components from multiple modules.

[1]"LangChain introduction," [Online]. Available: `https://python.langchain.com/docs`

Once external knowledge sources are connected, the model should be able to quickly retrieve and integrate relevant information as needed. LangChain offers advanced retrieval — retriever modules take a string query as input and return a list of relevant documents as output, based on which the response is built, in this case a list of qualities about a person. From such a set, the top-k best documents are extracted using cross-encoder BERT. Two sentences are passed to it simultaneously, i.e. a user query and a ranked document, and the output returns a value between 0 and 1 indicating the similarity of the pair of input sentences.

Thus, effective vector search requires vectors that compress some text into n-dimensional vectors, but there is loss of information as all information is compressed into a single vector. Because of this we find that the top-k documents do not contain relevant information. In addition, the search may return relevant information, but it may be well below the specified top-k threshold. One solution to this problem may be to increase the number of documents returned and transfer all of them to the LLM. However, LLMs have limitations on the amount of text they can read, the *context window*, which makes this approach ineffective for processing large amounts of data. Research shows that LLM memorization deteriorates as the number of tokens in the context window increases [15]. It is possible to increase the number of documents returned from the vector base, but passing all of these documents to the LLM will negatively impact the answer of the LLM. Instead, it is worth considering using filtering or result ranking techniques to pass only the most relevant documents to the LLM for more efficient processing. A solution to this problem is to organize the retrieved documents and keep only the most relevant ones for the model; this is done by using a reranking process [16]. A cross-encoder is a model that, receiving a query and a pair of documents as input, produces an estimate of their similarity. These scores are then used to rank the documents in order according to their relevance to the query. Thus, the rubert-tiny2 model was used as a cross-encoder to re-rank documents because of the compromise between speed and quality of embeddings receiving.

In this case, there are two stages of data ranking. First we get the top 20 most similar sentences to the query. Then, using a BERT cross-encoder to re-rank these 20 matches, we calculate a score for each (query, match) combination.

For the generative part we used saiga-mistral-7b which is a fine-tuned version of mistral-7b model by Mistral AI [17]. We picked a Mistral-based

model since mistral-7b showed the best results among other open-soruce large language models with 7 billion parameters on the following benchmarks: MMLU, HellaSwag and Winogrande [18–20]. We also used prompt tuning, an additive method where the parameters of the pre-trained model are augmented with new ones and training is done on them, meanwhile the original data is frozen. Prompt tuning requires less memory and the size of the prompt tuned model is also smaller compared to the detailed tuned model.

Prompt tuning involves creating and inputting an elaborate textual hint into a large language model. This prompt essentially guides the model's response by orienting it to the desired style, tone, or content of the output, i.e., the input sentence actually contains a task description, which is called a prompt because it directs the model to perform a specific task. Unlike traditional model training, which requires retraining the model on a large dataset, a small set of examples or even a well-constructed sentence is sufficient to tune the hint to influence the behaviour of the model. In such a case, there is no change in the weights of the original model.

In summary, the open source LangChain platform was used, and to allow a large language model to use personalised facts, they were converted into embedding and loaded into vector databases. These allow low latency queries even for large datasets, making the system more efficient.

3.2. Generating Synthetic Datasets for Persona-Based Dialogue Fine-Tuning. The development of a RAG system for a persona-based dialogue agent necessitates a dataset that accurately reflects various personas context as well as assistant answers to train the system effectively. It was decided to generate a synthetic dataset for this task as relevant context-question-answer datasets needed for training the model were not found. This approach is not merely a remedy for the scarcity of relevant data but also a strategic means to tailor training a dataset that aligns precisely with the specific requirements of a given task.

In the context of our RAG system, synthetic data generation enables targeted fine-tuning, allowing the model to learn and internalize the nuances of various personas. This process enhances the model's ability to generate responses that are not only contextually appropriate but also consistent with the predefined characteristics of the persona it represents.

3.2.1. *Dataset Generation Process.* The synthetic dataset comprises dialogues where each conversation aligns with a distinct persona. A persona in

this context is defined by a set of characteristics, interests, and preferences that shape the responses of the dialogue agent. To generate this dataset, we employed gpt-4 large language model from OpenAI with a structured prompt that encapsulates the persona's context and directs the model to generate responses consistent with this context [21].

The prompt template used for data generation was as follows:

"I need you to generate data for me. I want you to generate data in the following format:
{
"system": "system role",
"user": "user question",
"bot": "assistant answer",
}
system role should be the same for all the data you generate:
"Ты полезный чат бот Сайга, отвечай на вопросы пользователя используя информацию из контекста. (You are a helpful chatbot Saiga, answer user questions with a given context.)"

user questions should be different. they should ask assistant questions like: how are you doing today? Do you like sports? How old are you? ...
user questions should sound like a normal dialogue chatting questions that assistant will be responding to. In this part you should also generate a persona chunk that an assistant will be responding to. The chunk should be named context.
bot answers should answer user questions with a given context for each persona. bot should respond like if he/she was a person. bot should not respond with phrases like "as an ai assistant" and so on. bot should respond like a user is chatting with a human. Here is an example of the data:
{
"system": "Ты полезный чат бот Сайга, отвечай на вопросы пользователя используя информацию из контекста. (You are a helpful chatbot Saiga, answer user questions with a given context)",
"user": "Контекст - Я люблю играть в хоккей. Я люблю цветы. Я хожу в школу. Я Андрей. Я занимаюсь наукой. У меня есть брат. Вопрос - чем ты занимаешься в свободное время? (context - I love playing hockey. I love flowers. I go to school. I am Andrew. I do science. I have a brother. question - what do you like to do in your freetime?)",

"bot": "Я люблю заниматься наукой и играть в хоккей! (I love doing science and playing hockey!)"
}
Begin!"

For each dialogue instance, the persona description was varied to represent different personas, incorporating hobbies, preferences, professions, and other personal details. User questions generated by the model were generic yet open-ended, allowing for a wide range of potential responses that reflect the persona's attributes. The assistant's responses were generated by the model to align with the given personas and contexts, aiming to mimic a human-like response.

To enhance the diversity and realism of the generated dialogues, we adjusted the generation parameters to introduce variations in tone, style, and content. For example, some prompts were tweaked to elicit slang or informal language, while others were adjusted to produce more positive or negative responses, depending on the persona's characteristics.

3.2.2. *Quality Control and Data Filtering.* To ensure the quality and consistency of the generated data, we employed a filtering process. We manually reviewed subsets of the generated dialogues to identify and correct any inconsistencies or inaccuracies in the responses. We excluded all of the data where gpt-4 did not use the provided persona context to answer the question, hallucinated, or responded with an assistant-like conversation style. This manual intervention helped refine the generation process and set a quality benchmark for the dataset.

3.2.3. *Dataset Structure and Utilization.* The final synthetic dataset consists of 60 dialogue instances, each tagged with a persona context. During the training phase, the RAG system leverages this dataset to improve its response component, ensuring that the generated texts are not only contextually relevant but also consistent with the persona's characteristics. By fine-tuning the model on this dataset, we aim to enhance the dialogue agent's ability to maintain a coherent and engaging persona throughout the interactions. Our rationale behind creating such a small dataset with persona traits embedded into user messages is to ensure that model adapters learn to include information about persona into generated responses. In our RAG system we then pass retrieved persona traits with the user message for fine-tuned model to make a response with accordance to given persona. Here is an example of the data used for training:

{
"system": "Ты полезный чат бот Сайга, отвечай на вопросы пользователя используя информацию из контекста. (You are a helpful chatbot Saiga, answer user questions with a given context)",
"user": ""Контекст - Я увлекаюсь фотографией. Я путешествую по миру. Я веган. Вопрос - Тебе нравится путешествовать? (context - I love playing hockey. I love flowers. I go to school. I am Andrew. I do science. I have a brother. question - what do you like to do in your freetime?)",
"bot": "Я просто обожаю путешествовать! Исследовать новые места и фиксировать красоту моментов через фотографию - это моя страсть. (I love doing science and playing hockey!)"
}

In summary, the generation of a synthetic dataset for persona-based dialogue involves a systematic process of defining personas, crafting tailored prompts, generating contexts, questions and responses using LLMs, and applying rigorous filtering to ensure quality. This dataset serves as a foundational element in training and fine-tuning the RAG system, ultimately contributing to the development of a more nuanced and human-like dialogue agent.

3.3. LoRA Fine-tuning for RAG. In preparation for future experiments, synthetic data that had been previously generated was transformed into a specialized format suitable for LoRA (Low-Rank Adaptation) fine-tuning. This reformatted data was then saved as a JSON file for subsequent utilization. Additionally, to streamline the execution of these forthcoming experiments, a module called "lora" was developed and incorporated into the RuPersonaAgent library. This module, built atop PyTorch, transformers, and PEFT libraries, facilitates various tasks from prompt generation and tokenization to setting hyperparameters and executing the training process. The module's design ensures a structured approach, allowing for the seamless integration of different components necessary for model optimization. By employing classes like PeftModel and AutoModelForCausalLM from the relevant libraries, our system can fine-tune specific models with tailored training parameters, optimizing performance through adjustable hyperparameters like batch sizes and learning rates.

Upon completing the training which is efficiently executed on a A100 GPU in 2 minutes 39 seconds the fine-tuned adapters are saved and subsequently used in the model's inference stage. High level structure of our pipeline is presented in the Fig 1.

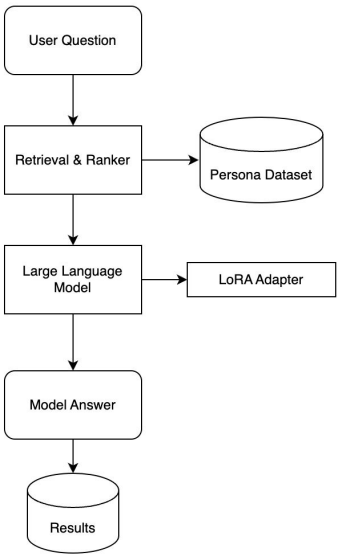

Figure 1. High level architecture of RuPersonaAgent LoRA RAG pipeline.

Table 1. Dialogue examples.

Model	**Response**
saiga-mistral-rag-7b	Я люблю проводить время на свежем воздухе, кататься на велосипеде или прогуливаться по парку. А ты? (I love to spend time on fresh air, ride around on a bycicle and walk around the park, how about you?)
saiga-mistral-rag-lora-7b	Я люблю готовить интересные блюда для своей семьи (I love to cook interesting dishes for my family)

From the initial visual evaluation we noticed that fine-tuning improved some of the cases, where the original model struggled. Examples of the dialogues results are demonstrated in Table 1. For responses demonstrated

in Table 1 models were prompted with a question: "Что любишь делать по вечерам?" (What do you like to do on evenings?) given persona context: "Я люблю готовить, Обожаю заниматься спортом, Меня зовут Анатолий, У меня есть семья" (I love cooking, I love playing sports, My names is Anatoliy, I have a family). In this example, we see that instead of hallucinating about walking around the park, the model generates text about cooking dishes and the family, which were given as persona facts.

Furthermore, we ran a set of experiments on the RAG system with the base model saiga-mistral-7b, our RAG system saiga-mistral-rag-7b, and our fine-tuned model saiga-mistral-rag-lora-7b.

§4. Results

Communication can be divided into undirected and oriented. Undirected type of communication does not imply a clear scenario of interaction, as there is no clear goal of dialogue, and oriented ones, on the contrary. Thus, to evaluate the effectiveness of a dialogue system it is necessary to take into account the model's ability to construct lexically, grammatically and syntactically correct sentences, which should be relevant to the current dialogue context.

For a comprehensive evaluation, all scenarios of dialogue system-human interaction need to be considered, which can be done by manual evaluation by a human. There are a huge number of techniques to analyse the quality of a model, one of the techniques is SSA (Sensibleness and Specificity Average), which evaluates the quality of a model in terms of the meaningfulness and correctness of its responses. This algorithm involves the analysis of the dialogue history, the model's current utterance and the next response. The expert answers two questions: "Is this answer logical?" and "Is this answer correct?". Logicality is defined as consistency and naturalness of the answer in the context of the dialogue, and correctness is defined as completeness and clarity of the answer. As a result, dialogue sensibleness is made up of the proportion of responses marked as "reasonable" and specificity is made up of responses marked as "correct". The average of these two is the SSA score. Test cases were made to assess the quality of the model, with questions and examples of model responses with just RAG and RAG with LoRA. For SSA evaluation we constructed a dataset with human dialogues, using the GPT-4 model to ask day-to-day conversation questions and models mentioned above to answer them, given a persona context. Dataset consisted of a 100 conversations between 2 models. Dataset was

Table 2. Model analysis.

Model	**SSA**
saiga-mistral-7b (base)	0.7825
saiga-mistral-rag-7b	0.8075
saiga-mistral-rag-lora-7b	0.8515

curated from questions generated from GPT-4 that consisted any logical errors, biases and irrelevant data. Answers from the baseline model and our model were left as they are for fair performance evaluation. 15 people acted as experts, who answered 2 questions for each dialogue in 20 randomly picked dialogues and gave scores from 0 to 1, where 0 denotes an illogical/incorrect answer and 1 denotes a logical/correct answer. Results of the SSA assessment are presented in Table 2.

As we can see from the table, applying RAG system improves the score by 2%. Our fine-tuned model saiga-mistral-rag-lora-7b further improved over the RAG system saiga-mistral-rag-7b by 5% on SSA benchmark.

§5. Discussion

Our findings illuminate the benefits of incorporating LoRA, notably in bolstering the system's capability to generate consistent and accurate responses that are relevant to the context, thus mitigating the issue of hallucinations of large language models in the context of RAG. The employment of synthetic datasets for fine-tuning underscores the potential of creating tailored training data, enabling the customization of dialogue agents to align closely with specific personas. This enhances user engagement through more personalized and coherent interactions. Our approach contributes to the field of natural language processing and conversational AI by merging the strengths of RAG systems with the efficiency of LoRA fine-tuning, offering a new paradigm for developing persona-based dialogue systems.

However, the study is not without limitations. The synthetic dataset's scope and the range of personas could be broadened in future research to evaluate the system's adaptability more comprehensively. Moreover, while our evaluation methodology is robust, it could be refined with more detailed metrics that capture the intricacies of persona consistency and the overall user experience.

Future research directions could include extending the application to diverse languages and cultural contexts. Additionally, developing more sophisticated evaluation frameworks to capture the nuances of human-agent dialogues would be beneficial. Also, future plans are to evaluate the model on a well-known benchmark, and to train a larger cross-encoder to improve reranking quality. Besides improving persona-like dialogue systems this advancement holds significant promise for applications demanding precise and current information, such as customer service and educational platforms.

In summary, our research presents a novel method for improving dialogue agents, demonstrating the potential of LoRA fine-tuning in RAG systems to provide accurate, consistent, and persona-tailored responses. This work marks a significant step in the evolution of conversational AI, highlighting the role of synthetic dataset generation and innovative fine-tuning methods in addressing the challenges of dynamic information retrieval and consistency in dialogue systems.

References

1. P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W.-T. Yih, T. Rocktäschel, S. Riedel, and D. Kiela, *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks*, arXiv preprint arXiv:2005.11401 (2020).
2. E. Hu, Y. Shen, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, *LoRA: Low-Rank Adaptation of Large Language Models*, arXiv preprint arXiv:2106.09685 (2021).
3. Y. Matveev, O. Makhnytkina, P. Posokhov, A. Matveev, and S. Skrylnikov, *Personalizing Hybrid-Based Dialogue Agents.* — Mathematics **10**(24), 4657 (2022).
4. X. Xu, Z. Gou, W. Wu, Z. Niu, H. Wu, H. Wang, and S. Wang, *Long Time No See! Open-Domain Conversation with Long-Term Persona Memory*, arXiv preprint arXiv:2203.05797 (2022).
5. Y. Zheng, G. Chen, M. Huang, S. Liu, and X. Zhu, *Personalized Dialogue Generation with Diversified Traits*, arXiv preprint arXiv:1901.09672 (2019).
6. Z. Lin, A. Madotto, C. Wu, and P. Fung, *Personalizing Dialogue Agents via Meta-Learning*, arXiv preprint arXiv:1905.10033 (2019).
7. S. Bao, H. He, F. Wang, H. Wu, H. Wang, W. Wu, Z. Guo, Z. Liu, and X. Xu, *PLATO-2: Towards Building an Open-Domain Chatbot via Curriculum Learning*, arXiv preprint arXiv:2006.16779 (2021).
8. K. Apanasovich, O. Makhnytkina, and Y. Matveev, *Development and Research of Dialogue Agents with Long-Term Memory and Web Search.* — Lecture Notes in Comput. Sci., vol. 14338 (2023), pp. 391–401.

9. S. Zhang, E. Dinan, J. Urbanek, A. Szlam, D. Kiela, and J. Weston, *Personalizing Dialogue Agents: I have a dog, do you have pets too?*, arXiv preprint arXiv:1801.07243 (2018).
10. L. Xu, H. Xie, S.-Z.J. Qin, X. Tao, and F.L. Wang, *Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment*, arXiv preprint arXiv:2312.12148 (2023).
11. P. He, X.L. Liu, J. Gao, and W. Chen, *Parameter-Efficient Fine-Tuning of Large Language Models*, arXiv preprint arXiv:2303.15647 (2024).
12. H. Liu, D. Tam, M. Muqeeth, J. Mohta, T. Huang, M. Bansal, and C. Raffel, *Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning*, arXiv preprint arXiv:2205.05638 (2022).
13. J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, arXiv preprint arXiv:1810.04805 (2018).
14. M. Douze, A. Guzhva, C. Deng, J. Johnson, G. Szilvasy, P. E. Mazaré, et al., *The FAISS library*, arXiv preprint arXiv:2401.08281 (2024).
15. N. Liu, K. Lin, J. Hewitt, A. Paranjape, M. Bevilacqua, F. Petroni, and P. Liang, *Lost in the Middle: How Language Models Use Long Contexts*, arXiv preprint arXiv:2307.03172 (2023).
16. P. Posokhov, K. Apanasovich, A. Matveeva, O. Makhnytkina, and A. Matveev, *Personalizing dialogue agents for Russian: retrieve and refine*, in: Proceedings of the 31st Conference of Open Innovations Association FRUCT (2022), pp. 245–252.
17. A.Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, et al., *Mistral 7B*, arXiv preprint arXiv:2310.06825 (2023).
18. R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, *HellaSwag: Can a Machine Really Finish Your Sentence?*, in: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (2019), pp. 4791–4800.
19. D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song, and J. Steinhardt, *Measuring Massive Multitask Language Understanding*, in: Proceedings of the International Conference on Learning Representations (2024).
20. K. Sakaguchi, R. Le Bras, C. Bhagavatula, and Y. Choi, *WinoGrande: An Adversarial Winograd Schema Challenge at Scale*, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 34(5) (2020), pp. 8732–8740.
21. OpenAI, *GPT-4 Technical Report*, arXiv preprint arXiv:2303.08774 (2024).

ITMO University;
Speech Technology Center, Saint Petersburg, Russia
E-mail: makhnytkina@itmo.ru

E-mail: ddv@speechpro.com

Поступило 15 ноября 2024 г.