Empowering Conversational Agents using Semantic In-Context Learning

Amin Omidvar, Aijun An
Department of Electrical Engineering and Computer Science, York University, Canada
omidvar@yorku.ca, ann@yorku.ca

Abstract

Language models are one of the biggest game changers in downstream NLP applications, especially in conversational agents. In spite of their awesome capabilities to generated responses to solve the inquiries, there are still some big challenges to using them. One challenge is how to enable the LLMs to use the private internal data to solve inquires. And secondly, how to keep the LLMs updated with newly incoming data without the burden of fine-tuning as it is not only expensive but also not an available option for some commercial LLMs, such as ChatGPT. In this work, we propose Semantic In-Context Learning (S-ICL) to address the aforementioned challenges. Our proposed approach participated in the BEA 2023 shared task[1] and ended up achieving the fourth place in both the development and evaluation phases.

1 Introduction

Conversational agents are one of the most important applications of NLP. If implemented successfully, they can bring tremendous benefits for both organizations and clients, such as improving the efficiency of customer service in terms of support and availability of the services.

With the emergence of powerful large language models (LLMs) such as ChatGPT, there is a lot of interest in leveraging LLMs to develop AI agents. Even though LLMs are capable of answering a broad spectrum of questions, there are still two major bottlenecks for using them as an AI assistant.

First, each organization has some valuable internal knowledge such as FAQs, policies, regulations, etc. that can or should be used to resolve incoming inquiries. However, the LLMs are trained based on public datasets and may not be aware of private knowledge sources that could help them to resolve incoming inquiries more accurately.

Secondly, fine-tuning these LLMs on the organization's internal data is not an easy task due to factors such as the size of the LLMs, cost of training, frequent updates in the internal data, and data privacy. For example, in news media, news articles are published every day that LLMs are not aware of them. If the news media decides to use an LLM as an agent, the agent would be unable to provide users with information about current events or answer their questions about what is happening now. On top of that, the fine-tuning option is not available for certain LLMs (e.g., ChatGPT with the GPT-3.5-turbo engine).

One possible solution to the mentioned problems is In-Context Learning (ICL), as it can enable the LLMs to perform well on the tasks or data that they have never seen before (Brown et al., 2020). In ICL, a prompt containing an instruction, few labeled samples, and an unlabeled sample is given to the LLM. Then, the LLM would be able to label the unlabeled sample without the need for any gradient-based training (Liu et al., 2022).

However, it is infeasible to show all the available samples to the LLM due to the high cost of computation. Also, previous research shows that the format of the prompt, the selection of samples, the number, order, and structure of samples could have not only significant but also unforeseeable effects on LLMs' performance (Min et al., 2022; Sanh et al., 2021; Wei et al., 2023; Liu et al., 2022).

To solve the aforementioned problems, we propose Semantic In-Context Learning (S-ICL) which utilizes a semantic search engine (i.e., an SBERT model (Reimers and Gurevych, 2019)) and an LLM (i.e., ChatGPT with the gpt-3.5-turbo engine) to build a conversational agent. This agent not only benefits from the knowledge of an LLM but also utilizes available private knowledge sources to provide the correct answer to the inquiries. We also propose a flexible architecture that allows experts to apply and compare different approaches for prompt

[1] https://sig-edu.org/sharedtask/2023
Our username and team's are amino and aiitis, respectively.

Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023), pages 766–771

engineering.

The proposed model is developed and participated in the BEA 2023 Shared task (Tack et al., 2023). However, the proposed model is flexible, and the agent can be used in other domains such as news media, customer service, and more.

The rest of the paper is as follows. In Section 2, we describe the proposed architecture along with its components. In section 3, we compare different configurations of the proposed model on the created test set, and we also evaluate the model on the competiton's data. Finally, this paper is wrapped up with the conclusion in section 4.

2 Proposed Model

In this section, we present our proposed approach for generating a response to the inquiry. Our proposed approach uses semantic search (Reimers and Gurevych, 2019) to enable the agent to utilize private domain data. It also uses a large language model not only to provide higher quality answers but also to enable the agent to answer questions that are significantly different from past questions and answers in the private domain data.

2.1 Overview

As shown in Figure 1, the proposed architecture consists of five main components: Data preprocessor, Embedder, Retriever, Prompt builder, and Answer generator. The first three components are related to the semantic search part of the architecture, while the other two are related to the language model.

2.2 Data pre-processor

The data pre-processor receives utterances in JSON format containing a context and a query (i.e., the last utterance). It extracts and transforms the JSON file into the followings:

Concatenation: it's a textual concatenation of all the utterances made by a student and a teacher. The main purpose of transforming data into this format is to enable its use in the semantic search part of the architecture.

Sample: It's a conversational flow between the student and the teacher. Based on who wrote the utterance, either "Teacher: " or "Student: " would be appended in the beginning of the utterance. This format is being used by the prompt builder component as it is more appropriate to be used by the language model.

2.3 Embedder

In this section, we use a state-of-the-art transformer encoder model to convert the concatenation format, which is built in the data pre-precossor part, into the embedding represention. We use the pre-trained model "multi-qa-mpnet-base-dot-v1" to generate embeddings as it has the highest performance in the Hugging Face benchmark [2]. The tokenizer first tokenizes the input text, and then the transformer encoder model infers an embedding vector with a size of 768 for each token of the input text. The embedding vector of the CLS token in the last layer is considered the embedding representation of the whole input text.

2.4 Retriever

The Retriever is responsible for finding the most similar records that exist in the training data to the incoming context. It calculates the cosine similarity between the embedding vector of the context and each embedding vector in the training set. Then, the results would be sorted in descending order based on the cosine similarity score, and the top N results would be passed on to the next step.

This process could be significantly sped up on large datasets by using approximate K-nearest neighbor methods, such as Facebook AI Similarity Search (Faiss) (Johnson et al., 2019). However, due to the small size of our data, we don't need to use any approximate K-NN methods.

2.5 Prompt builder

The prompt builder component creates a prompt based on the selected prompt building approach. Figure 2 shows the structure of the prompt which consists of the following components in order:

Command: It's a first component of the prompt that informs the language model of what is expected to be done.

Sample(s): The retrieved sample(s) from the training set are included to assist the language model in answering the inquiry. This part of the prompt is optional because the number of samples to be used depends on the selected approach.

Inquiry: It contains the last utterance along with the previous utterances (i.e., Context) given to the system.

The command part of the prompt is written by humans, while the other parts are generated auto-

[2] https://www.sbert.net/docs/pretrained_models.html

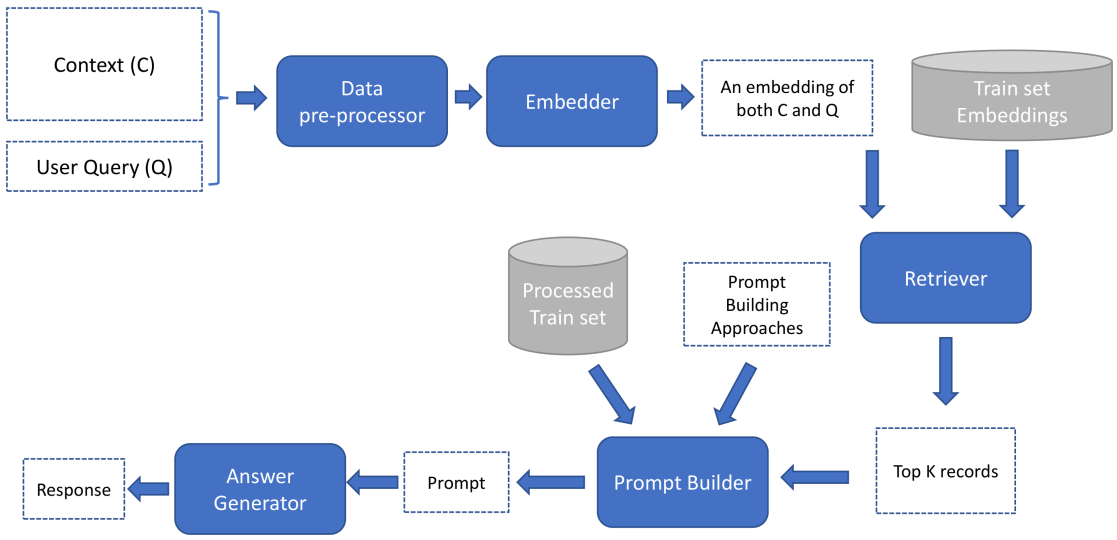

Figure 1: The proposed architecture for the conversational agent uses both semantic search and a large language model.

matically depending on the chosen prompt building approach. So far, four prompt building approaches have been designed, but more could be defined to further improve the agent's performance or adapt it better to different domains of data, such as news.

2.6 Answer Generator

A large language model is used in this part of the architecture. In our experiment, we use ChatGPT [3] API using gpt-3.5-turbo engine. A prompt created in the previous stage would be sent to the language model, and the response would be returned to the end user. To make the result reproducible, we set the temperature value to zero.

In this way, the language model can not only use its knowledge but also have access to the relevant past responses from the private domain knowledge to answer the question. Another advantage is that there is no need to fine-tune the large language model on private internal data, which may not be an option for many models, such as ChatGPT.

3 Experiment

This section has three subsections. In the first subsection, we introduce the dataset used, split the train portion of the data into our created train and test sets, and show how the pre-processing has been done. In the second subsection, we conduct experiments on the proposed architecture using the created test set (i.e., selected from the original training set) and compare the accuracy of the model using different prompt building approaches. In the third subsection, we will demonstrate the model's performance on the development and testing sets of the competition data.

3.1 Data

The data consists of the conversation between a student and a teacher provided by (Caines et al., 2020). The sizes of the provided data and their release dates in the competition are shown in Table 1. We transform the training set using the pre-processor component (subsection 2.2). Then, we use the embedder component (subsection 2.3) to convert the concatenation of the utterances into their embedding representations (i.e., Train set embedding in Figure 1).

Then, we split the train set into customized train and test sets with sizes of 2647 and 100, respectively. We use the customized train and test sets to compare the different prompt generation approaches in subsection 3.2. Since some of the records in the training set have similar utterances (i.e., they overlap), we select the test data in a way that none of the test conversations can be answered directly from the conversations in the training set (i.e., there is no overlap between the utterances of the train and test sets).

3.2 Evaluation of different approaches

We use five different approaches to provide the response to the incoming inquiry. In the first approach, we only use the semantic search. That

[3] https://openai.com/blog/chatgpt

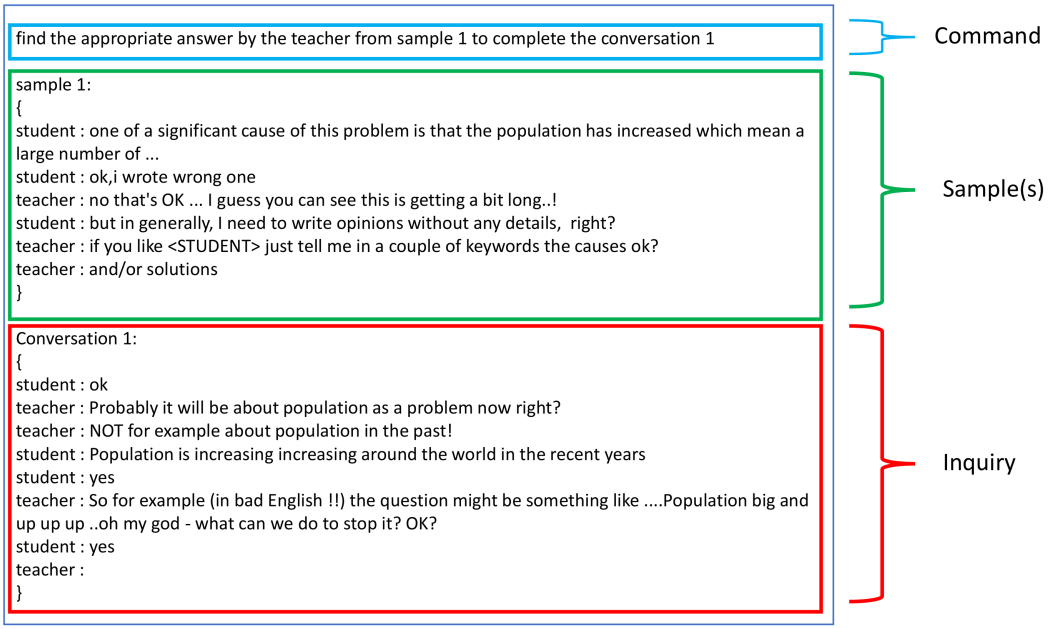

Figure 2: An example of a prompt structure with a single sample.

Set	Size	Release date
Train	2747	March 24, 2023
Dev	305	March 24, 2023
Test	273	May 1, 2023

Table 1: The statistics of the TSCC dataset.

means the last utterance of the most similar retrieved sample is chosen as a response. Next, we are curious to see how good the language model is in completing the conversation without using any samples. The command we use is "*Complete the following conversation by giving an appropriate answer by the teacher*". However, for the third approach, we ask the language model to "*Find the appropriate answer by the teacher from sample 1 to complete the conversation 1*". The provided sample, which has the ID "train_0063", was chosen by us from the training set and has been used for all inquiries.

During the experiments, we observed that ChatGPT tends to generate longer responses than the ground truths. However, we discovered that by formulating our prompt command in a certain way (i.e., find the appropriate answer by the teacher from sample ...), ChatGPT can produce more concise and shorter responses. Therefore, we decided to write the command part of our prompt in this way. We also observed that for some inquiries, ChatGPT mentions "teacher :" in its response, so we wrote a rule to remove it.

The fourth approach includes the top 3 most similar samples in the prompt and the command is "*Find the appropriate answer by the teacher from sample 1, sample 2 and sample 3 to complete the conversation 1*". And the last approach is similar to the third one but instead of using the curated sample, the most similar sample from the training set is being used. The last two approaches are based on S-ICL.

The results of the above approaches on the created test dataset are shown in Table 2 in terms of BERT Score (Zhang et al., 2019) and DialogRPT (Gao et al., 2020). In Table 2, P, R, F, U, HvR, HvM stand for precision, recall, f1-score, updown (the probability that a response receives upvotes), human vs random (the probability that the response is relevant to the given context), human vs machine (the probability that the response was written by a human rather than generated by a machine), respectively. The first three measures belong to BERTScore (Zhang et al., 2019), and the rest of them belong to DialogRPT (Gao et al., 2020). We use "roberta-large" model [4] for the BERTScore as we do not know which model the competition is using. We then compare the generated responses with their ground-truths using BERTScore in terms of precision, recall, and f1-score. Each of the first three measures of DialogRPT (i.e., U, HvR, and HvM) [5] has its own pre-trained model. Each model

[4] https://huggingface.co/roberta-large
[5] https://github.com/golsun/DialogRPT

Approach	BERTScore			DialogRPT				
	P	R	F	U	HvR	HvM	best	avg
1	0.824	0.823	0.823	0.433	0.789	0.983	0.999	0.735
2	0.835	**0.837**	0.836	0.490	**0.922**	**0.999**	**0.999**	**0.804**
3	**0.839**	0.836	**0.837**	0.464	0.877	0.997	0.998	0.779
4	0.828	0.832	0.830	**0.574**	0.767	0.998	0.999	0.780
5	0.835	0.831	0.833	0.481	0.839	0.997	0.999	0.773

Table 2: The comparison between different approaches used on the created test set in terms of BERT Score.

receives the generated responses and their corresponding contexts (i.e., the previous utterances of each conversation) to calculate a score.

Interestingly, the model that uses the fixed sample for all the inquiries (third approach) gained the best BERTscore in terms of f1-score. This observation is inline with the results of other studies such as (Min et al., 2022) that they concluded replacing the sample labels randomly would barely hurts the performance of the LLMs. In terms of DialogRPT, the second approach gained the best results. However, when we examined the generated answers, we found out the answers of the fifth approach are both more reasonable and preferable in comparison with the other approaches.

3.3 BEA Workshop's evaluation

Our proposed approach ranked fourth both in development and evaluation phases. We used our third approach (using the fixed sample) for the development phase as we noticed the majority of utterances in development data have overlap with the training set. If we use either the fifth or fourth approach, the model would recognize the similarity between the sample and the conversation and produce a response so similar to the existing utterance in the sample that it would inflate the performance of the system. However, we discovered that the test data is different in a way that none of its conversations could have their responses directly obtained from any utterances in either the training or development sets. Therefore, for the evaluation set, we used the fifth approach. Another reason that why we used the fifth approach in the evaluation phase is that the top three models would be evaluated by the human evaluators, and we already noticed in subsection 3.2 that the results of the fifth approach are more desirable from humans' point of view.

The evaluation phase was started on May 1st and ended on May 5th. Due to an unprecedented emergency, we were unable to continue working on the test data and our last submission was on May 1st. Our model ended up ranking fourth in the evaluation phase and could not pass to the human evaluation phase. However, we think that the proposed model has a high potential for improvement, especially if more efforts would be put on the prompt engineering part of the architecture.

4 Conclusion

We proposed a Semantic In-Context Learning (S-ICM) approach for conversational agents using the combination of a semantic search and a large language model (i.e., ChatGPT). We also implemented an architecture enabling users to apply and compare different approaches for prompt engineering. We applied our proposed method on the BEA 2023 shared task and our approach ended up ranking fourth in both the development and evaluation phases.

Acknowledgements

This work is funded by Natural Science and Engineering Research Council of Canada (NSERC).

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Andrew Caines, Helen Yannakoudakis, Helena Edmondson, Helen Allen, Pascual Pérez-Paredes, Bill Byrne, and Paula Buttery. 2020. The teacher-student chatroom corpus. *arXiv preprint arXiv:2011.07109*.

Xiang Gao, Yizhe Zhang, Michel Galley, Chris Brockett, and Bill Dolan. 2020. Dialogue response ranking training with large-scale human feedback data. *arXiv preprint arXiv:2009.06978*.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Haokun Liu, Derek Tam, Mohammed Muqeeth, Jay Mohta, Tenghao Huang, Mohit Bansal, and Colin A Raffel. 2022. Few-shot parameter-efficient fine-tuning is better and cheaper than in-context learning. *Advances in Neural Information Processing Systems*, 35:1950–1965.

Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. 2022. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Anaïs Tack, Ekaterina Kochmar, Zheng Yuan, Serge Bibauw, and Chris Piech. 2023. The BEA 2023 Shared Task on Generating AI Teacher Responses in Educational Dialogues. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications*, page to appear, Toronto, Canada. Association for Computational Linguistics.

Jerry Wei, Jason Wei, Yi Tay, Dustin Tran, Albert Webson, Yifeng Lu, Xinyun Chen, Hanxiao Liu, Da Huang, Denny Zhou, et al. 2023. Larger language models do in-context learning differently. *arXiv preprint arXiv:2303.03846*.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.