Shahid Beheshti University

Manuscript Template
Advances in Physical Oceanography
honoring Prof. Affonso Mascarenhas
http://dx.doi.org/10.1590/FOR PEER REVIEW
ISSN 2675-2824

Persian Tech Support Assistant

Mohammad Rouintan[1], Reza Mousavi[1] and Amir Hossein Karami[1]

[1]Computer Science Department, Shahid Beheshti University

Abstract

This report describes a project that involves the development of a tech support assistant in Persian language using large language models (LLMs). The project consists of three main parts: initial data gathering, data generation, and model fine-tuning. In the first part, a corpus of texts related to technology and gadgets like mobiles and laptops was collected from Digikala Mag and Digiato Persian blogs. In the second part, ChatGPT 3.5 Turbo via OpenAI API was used to generate question-answer pairs from the crawled corpus. GPT2 was trained on the corpus and fine-tuned on the question-answer data in the third part. Additionally, a Persian Llama named 'persian_llama_7b' was fine-tuned using Hugging Face tools on the question-answer data. Parameter-efficient fine-tuning methods like QLORA were used to enable Colab and Kaggle resources to solve the task. The resulting model can be used as a tech support assistant in Persian language. The paper provides insights into the challenges faced during the project and how they were overcome. Future work that can be done in this field is also discussed.

INTRODUCTION

Large language models (LLMs) have become increasingly popular in recent years due to their ability to perform a variety of natural language processing (NLP) tasks. In this paper, we present a project that involves the development of a tech support assistant in Persian language using LLMs. The project consists of three main parts: initial data gathering, data generation, and model fine-tuning. In the

first part, we collected a corpus of texts related to technology and gadgets like mobiles and laptops from Digikala Mag and Digiato Persian blogs. In the second part, we used ChatGPT 3.5 Turbo via OpenAI API to generate question-answer pairs from the crawled corpus. We generated 7 questions and answers from each piece of the crawled corpus using GPT3.5. In the third part, we trained GPT2 on the corpus and fine-tuned it on the question-answer data. Additionally, we fine-tuned a Persian Llama named 'persian_llama_7b' using Hugging Face tools on our question-answer data. We used parameter-efficient fine-tuning methods like QLORA to enable our Colab and Kaggle resources to solve the task. The resulting model can be used as a tech support assistant in Persian language.

The purpose of this paper is to describe our project in detail, including the data gathering process, data generation process, and model fine-tuning process. We will also discuss the results of our project and how our model can be used as a tech support assistant in Persian language. We will provide insights into the challenges we faced during the project and how we overcame them. Finally, we will conclude with a discussion of the future work that can be done in this field.

CORPUS GATHERING

The first part of our project involved gathering a corpus of texts related to technology and gadgets like mobiles, laptops, tablets, cameras, smart gadgets, and software from Digiato and Digikala Mag Persian blogs. The goal of this part was to collect a Persian corpus in the field of tech that can be used for training our models. We used web scraping techniques to crawl the texts from the blogs and stored them in multiple text files. Web scraping is an automated method used to extract large amounts of data from websites, fast. We then preprocessed the corpus by removing any irrelevant information and cleaning the text data. This step is important to ensure that the data is in a format that can be used for training our models.

The corpus we collected was used to train our models and fine-tune them on the question-answer data generated in the second part of our project. The quality of the corpus is crucial for the performance of the models, and we made sure to collect a diverse set of texts related to technology and gadgets to ensure that our models are robust and can handle a variety of inputs. The corpus can also be used as a useful Persian corpus in the related context.

To ensure that the corpus we collected was of high quality, we made sure to collect a diverse set of texts related to technology and gadgets. We also removed any irrelevant information and cleaned the text data to ensure that the data is in a format that can be used for the proportional works.

DATA GENERATION: AN LLM TRAINS AN LLM!

Fine-tuning LLMs to work as a chatbot can be more effectively done using data with a question-answer format. To generate this data, we used ChatGPT 3.5 Turbo via OpenAI API. We gave pieces of text to ChatGPT and asked it to generate 7 questions and their answers from each piece. We then stored the data in a CSV file with 3 columns containing ‘context’, ‘question’, and ‘answer’. The ‘context’ column contains the piece of text that each question-answer pair is extracted from.

This dataset contains about 31,000 question-answer pairs and can be a useful Persian dataset for performing various NLP tasks. The quality of the data generated in this part is crucial for the performance of our models. We made sure to generate diverse and high-quality question-answer pairs to ensure that our models are robust and can handle a variety of inputs. The generated data was used to fine-tune our models in the third part of our project.

The pipeline described in this section and the previous section, can be used for generating useful datasets in every context.

FINE TUNING MODELS

This part contains fine tuning models. Chosing a good base model is very crucial

Base models

Chosing a good base model is very crucial for this project. Since the goal is to achieve a useful model for Persian language, and according to low ammount of data and resource for full parameter training, we had to use base models with initial kknowledge on Persian language.

Persian Llama 2 7B

Natural language and text processing lab of University of Tehran has trained LLama 2 7B with 7 billion parameters on a collection of Persian text data (translated question-answer datasets and some raw Persian texts), and published this model on HuggingFace.

Llama 2 is a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. The Llama2 model was proposed in “LLaMA: Open Foundation and Fine-Tuned Chat Models” by Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernand. . . . The fine-tuned LLMs, called Llama 2-Chat, are optimized for dialogue use cases. Llama 2 outperforms other open source language models on many external benchmarks, including reasoning, coding,

proficiency, and knowledge tests. Llama 2 is available for free for research and commercial use.

GPT 2

GPT-2 is a large language model developed by OpenAI and the second in their foundational series of GPT models. It was pre-trained on BookCorpus, a dataset of over 7,000 self-published fiction books from various genres, and trained on a dataset of 8 million web pages. GPT-2 is widely used for various natural language processing tasks such as text generation, text classification, and question answering.

Resource challenge: Parameter Efficient Fine Tuning

Parameter-efficient fine-tuning is a technique used to finetune large language models (LLMs) on limited resources such as memory and computational power. QLoRA is an efficient finetuning approach that reduces memory usage enough to finetune a 65B parameter model on a single 48GB GPU while preserving full 16-bit finetuning task performance. QLoRA backpropagates gradients through a frozen, 4-bit quantized pretrained language model into Low Rank Adapters (LoRA). QLoRA introduces a number of innovations to save memory without sacrificing performance, such as 4-bit NormalFloat (NF4), a new data type that is information theoretically optimal for normally distributed weights, double quantization to reduce the average memory footprint by quantizing the quantization constants, and paged optimizers to manage memory spikes.

QLoRA can be leveraged by using bitsandbytes and Hugging Face Parameter Efficient Fine-Tuning (PEFT) library together. HuggingFace Transformer Reinforcement Learning (TRL) library offers a convenient trainer for supervised finetuning with seamless integration for LoRA. LoRA is implemented in the Hugging Face PEFT library, offering ease of use. QLoRA finetuning on a small high-quality dataset leads to state-of-the-art results, even when using smaller models than the previous state-of-the-art. QLoRA can be used to finetune more than 1,000 models, providing a detailed analysis of instruction following and chatbot performance across 8 instruction datasets, multiple model types (LLaMA, T5), and model scales that would be infeasible to run with regular finetuning (e.g. 33B and 65B parameter models).

Fine Tuning

In the third part of our project, we fine-tuned Persian_Llama_2_7B and GPT-2 on our raw text corpus and question-answer data using parameter-efficient fine-tuning techniques and Hugging Face tools. We used QLoRA configurations for quantizing models to enable us to finetune the models on limited resources.

CONTEXT RECOMMENDER FOR USING RAG

Retrieval-augmented generation (RAG)

Retrieval-augmented generation (RAG) is an AI framework for improving the quality of large language model (LLM)-generated responses by grounding the model on external sources of knowledge to supplement the LLM's internal representation of information. RAG is used to retrieve facts from an external knowledge base to ground LLMs on the most accurate, up-to-date information and to give users insight into LLMs' generative process.

RAG has emerged as a promising solution by incorporating knowledge from external databases. This enhances the accuracy and credibility of the models, particularly for knowledge-intensive tasks, and allows for continuous knowledge updates and integration of domain-specific information. RAG can be used to optimize the output of an LLM by referencing an authoritative knowledge base outside of its training data sources before generating a response.

High amount of data to be used for RAG

Assume a business that wants to have an AI-based assistant for your support team to answer the customers questions. Based on the question, you can give some texts to an AI chatbot like ChatGPT to generate the response for that question using knowledge extracted from those texts. Those texts can be come from sources like product specification sheets, FAQs or other documents.

In a typical business there are high amount of documentations able to be used for the mentioned purpose, so a crucial task is to find texts that may contain the most information for answering the given question, and give those suitable texts to the AI chatbot along with the question.

A Solution: Recommender System for RAG

A solution for the mentioned challenge may be a recommmender-like system which recommends the most relevant (maybe similar) texts to a given question.

In our work we have developed a content-based recommender system using ParsBERT model. The system has the stored vector representations of the documents which contains potential information which can be useful for answering the questions. Each coming question is vectorized and using cosine similarity, the most similar documents are recommended to be fed to the AI chatbot along with the given question.

Base model for embedding: ParsBERT

ParsBERT is a monolingual language model based on Google's BERT architecture. This model is pre-trained on large Persian corpora with various writing styles from numerous subjects (e.g., scientific, novels, news) with more than 3.9M documents, 73M sentences, and 1.3B words. ParsBERT is a useful Persian model for performing various NLP tasks such as text classification, sentiment analysis, and named entity recognition. ParsBERT is available for free for research and commercial use.

Most Similar Doc Recommender

This is a system that has stored the vector representation of all documents which contains potentially useful information for answering the questions. The vectorization is performed using ParsBERT model. Each coming question is vectorized using ParsBERT model and using cosine similarity, the most similar documents are recommended to be fed to the AI chatbot along with the given question.

Question-Collaborative-like Recommender

In this case we need to have an initial set of relevant questions (like FAQs for a business) to each document. This system used ParsBERT as the vectorizing model. The system has stored the vector representations of the questions. Each input question, is vectorized using ParsBERT model and is compared to the core vectorized questions. The related text documents to the most similar core questions, are recommended to be fed to the AI chatbot along with the given question.