Closed domain knowledge injection into general Text-to-Text model

Stefano di Terlizzi

Contributing authors: stefanoditerlizzi96@gmail.com;

Abstract

Dialogue generation has been successfully learned from scratch by neural networks, but tends to produce the same general response, e.g., "what are you talking about?", in many conversations. To reduce this homogeneity, external knowledge is applied as an additional condition to diversify a model's output. The project aims at injecting specific domain knowledge and evaluating the IR system and the model response with and without the injection.

Keywords: Text-to-text generative models, Hallucination, Closed domain injection, Text generation evaluation

1 Introduction

One limitation of Large Language Model (LLM) content generation is hallucination, or false assertions in the generated text. Traditionally, fine-tuning has been a go-to solution to mitigate hallucination in generative models. However, this approach often demands additional computational resources and extensive training, limiting its practicality. Alternatively, an emerging strategy involves injecting domain-specific knowledge directly into the model during the inference process, thus enhancing the model's understanding of the given context and reducing the likelihood of hallucination. In this study, we delve into the efficacy of closed domain injection as a solution to address hallucination in text-to-text generative models. Specifically, we explore the feasibility of augmenting input prompts with domain-specific information obtained from external knowledge retrieval systems. By integrating such knowledge directly into the model's input, we aim to enhance its contextual understanding and improve the relevance and coherence of generated text. To evaluate the effectiveness of this approach, we conduct experiments using two distinct text-to-text generative models, differing in the number of parameters: one with 77 million parameters (flan-t5-small

by Google) and another with 783 million parameters (flan-t5-large by Google). By comparing the performance of these models with and without closed domain injection, we seek to elucidate the impact of knowledge enhancement on mitigating hallucination across varying model complexities. Through our investigation, we aim to provide insights into the potential of closed domain injection as a practical and efficient strategy for improving the robustness and accuracy of text-to-text generative models in real-world applications.

2 Problem Setup and Experiments

Before delving into the details of our experiments and the methods employed, it is imperative to establish the problem context and the experimental framework. In this section, we outline the dataset used for training and evaluation, introduce the information retrieval system utilized for enhancing prompts, detail the methods employed for prompt enhancement, and elucidate the evaluation metrics utilized to assess the performance of our approach. Now, let's proceed to delve deeper into each component of our problem setup and experimental design.

2.1 Dataset

The dataset utilized in this study comprises structured information sourced from various domains. Each entry in the dataset consists of contextual information, questions, and corresponding target answers. A sample entry from the dataset is presented below:

```
[
    {
        "idx": "/wiki/Knox_Cunningham#P39#0",
        "question": "Which position did Knox Cunningham hold from May 1955 to Apr 1956?",
        "context": "Knox Cunningham Sir Samuel Knox Cunningham , 1st Baronet , ...",
        "targets": [
            "Ulster Unionist MP for South Antrim"
        ],
        "level": "easy"
    },
    ...
]
```

It is important to note that our dataset contains duplicates in the contexts, meaning that multiple questions may refer to the same context, resulting in duplicates.

2.2 Information retrieval system for enhancing prompts

To enhance prompts, an information retrieval system is employed, leveraging the Term Frequency-Inverse Document Frequency (TF-IDF) technique. The system utilizes a preprocessed corpus of contexts and questions for vectorization. We choose to organize the contexts as a set of unique contexts due to the presence of duplicate contexts associated with different answers. By organizing the contexts as a set, we ensure that each unique context is considered only once during training and evaluation. This approach prevents redundancy in the dataset and ensures that the models learn from

diverse contextual information without being biased towards repetitive contexts. Subsequently, a TF-IDF vectorizer is fitted on the combined corpus. An excerpt of the code implementation is shown below:

```
from sklearn.feature_extraction.text import TfidfVectorizer

contexts = list(set([e["context"] for e in data]))
dataForFit = contexts + [e["question"] for e in data]
vectorizer = TfidfVectorizer(tokenizer=nltk.word_tokenize)
vectorizer.fit(dataForFit)
corpusTFIDF = vectorizer.transform(contexts)
```

2.3 Enhancing methods

The enhancement process involves injecting domain-specific knowledge into the input prompt using the TF-IDF-based information retrieval system. The system identifies relevant context passages based on cosine similarity scores between questions and contexts. Percentile filtering is applied to refine the selection of relevant sentences from the identified context. The enriched prompt is then utilized for text generation. The core functionalities of the enhancement method are illustrated in the provided code snippet.

```
for i,e in enumerate(data):
        #find the context with the IR system
        answer = vectorizer.transform([e["question"]])
        arr = cosine_similarity(answer, corpusTFIDF)[0]
        mx = np.amax(arr)
        contextIndex = (np.where(arr == mx))[0][0]

        #split the context into sentences and score them by cosine similarity
        sentences = sent_tokenize(contexts[contextIndex])
        sent = vectorizer.transform(sentences)
        sentSimilarity = [(i, score) for i,score in enumerate(cosine_similarity(answer, sent)[0])]

        #quantile filtering
        quantiles = [0, 0.5, 0.75, 0.90]
        quant = [np.quantile([e[1] for e in sentSimilarity], q) for q in quantiles]
        for i,q in enumerate(quant):
            #quantile filtering and reduce to a string the result
            searchOn = reduce(
                lambda x, y: f'{x} {y}',
                [ el for i,el in enumerate(sentences) if i in [ e[0] for e in sentSimilarity if e[1]>=q ] ]
            )
            #generate the response enanching the input
            #with the result of previus computation + the question
            response = text2textModel(
                tokenizer=mTokenizer,
                model=mModel,
                input_text = f'{searchOn}. {e["question"]}'
            )
            response = response.replace('<pad> ', '').replace('</s>', '')

            similarityResponseTarget = cosine_similarity(
```

```
                    vectorizer.transform([response]),
                    vectorizer.transform(e["targets"])
                )[0]

                finalResult.append({
                    'question': e["question"],
                    'correctContext': contexts[contextIndex] == e["context"],
                    'scoreContextRetrieval': mx,
                    'quantileLimitSentences': quantiles[i],
                    'targetWithSimilarityScore': [ (el, similarityResponseTarget[i]) for i,el in enumerate(e['targets']) ],
                    'response': response
                })
```

2.4 Evaluation methods

The evaluation methodology relies on cosine similarity metrics to assess the quality and relevance of context retrieved compared to the answer. Each response generated by the model is evaluated based on its cosine similarity score with the corresponding target answer. The evaluation process aims to quantify the degree of semantic similarity between generated responses and ground truth answers.

3 Results

In this chapter, we present the results obtained from our experiments aimed at evaluating the effectiveness of our proposed approach. The results presented herein encompass various aspects of our study, including the performance of the information retrieval system in enhancing prompts, the effectiveness of different enhancement methods, and the evaluation of generated responses against ground truth answers. Now, let us delve into the detailed presentation and analysis of the results obtained from our experiments.

3.1 Information retrieval system

The preliminary analysis of the Information Retrieval system reveals that it exhibits a notable performance in identifying the correct context, with a probability of 0.893. Conversely, the system's ability to identify incorrect contexts is observed to have a probability of 0.107. Additionally, the mean score of the retrieved correct contexts based on the cosine similarity between the answer and the context retrieved is 0.316. These initial findings lay the foundation for further evaluation and analysis of the system's performance in enhancing prompts and guiding the text generation process.

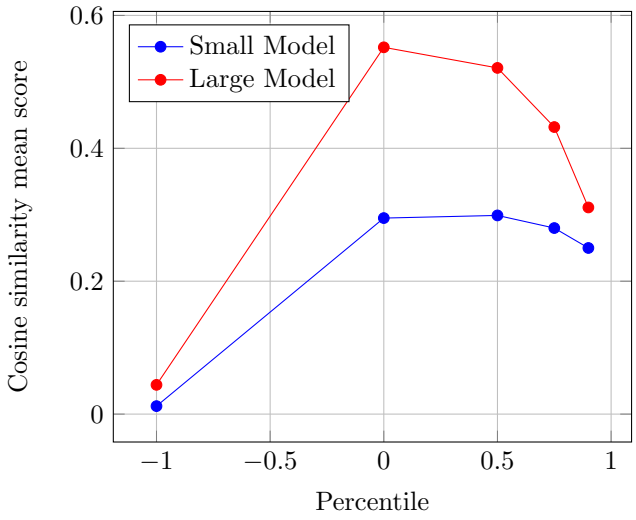
3.2 Model response

Upon conducting an analysis of cosine similarity between the ground truth answers and those generated by the model, notable observations emerge. Initially, for the small model, the mean cosine similarity without injection is recorded at 0.012. Conversely, for the large model, the mean cosine similarity without injection registers at 0.044.

However, upon implementing injection techniques, a remarkable improvement in performance is discerned. Specifically, for the small model, the mean cosine similarity experiences a notable increase, ranging from 0.250 to 0.298 across various percentiles. Similarly, substantial enhancements are observed in the large model, where mean cosine similarity values range from 0.310 to 0.551 across differing percentiles.

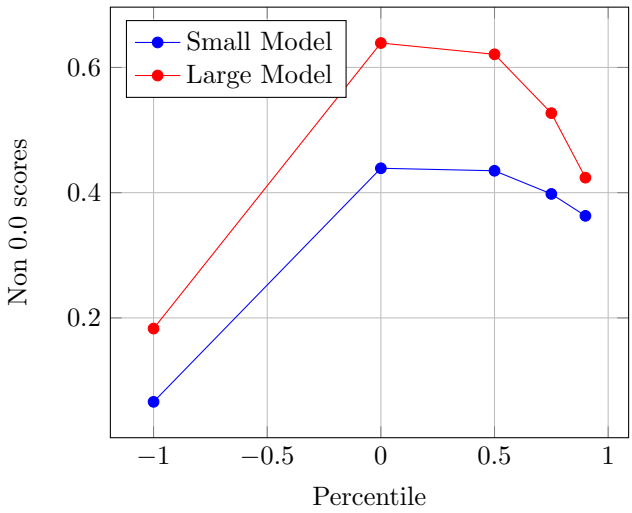
These findings underscore the efficacy of injection techniques in significantly enhancing the quality and relevance of generated responses for both small and large models.

Focusing on the injection of sentences that meet the percentile threshold, we can explore the following plot depicting how the mean score varies based on the analyzed quantile. Note that the value -1 represents the absence of context injection.



percentile	small model	large model
no injection	0.012	0.044
0.0	0.295	0.552
0.5	0.299	0.521
0.75	0.280	0.432
0.90	0.250	0.311

In addition, it is important to analyze the percentage of responses that achieve a similarity different from 0.0, and that may be partially correct.



percentile	small model	large model
no injection	0.066	0.183
0.0	0.439	0.639
0.5	0.435	0.621
0.75	0.398	0.527
0.90	0.363	0.424

As can be seen from the graphs above filtering more context leads to a decrease in the correctness of the response. From our tests we can estimate that an approximate rule of thumb would be to either not filter the context and inject it complete or filter it between 0.0 to 0.5 keeping more than 50% of the context.

4 Conclusion

The experimental findings presented in this study shed light on the efficacy of injecting domain-specific knowledge to enhance the performance of text-to-text generative models. By exploring the impact of percentile-based injection on model performance, we observed significant improvements in mean cosine similarity scores across various percentiles for both small and large models.

Moreover, the analysis of non-zero similarity scores highlights the capability of the injected knowledge to generate responses that deviate from complete hallucination, contributing to the generation of partially correct answers.

Overall, our study underscores the importance of leveraging external knowledge sources to augment text-to-text generative models, offering promising avenues for improving their robustness and relevance in real-world applications. Future research endeavors may delve deeper into refining injection strategies and exploring novel approaches to further enhance model performance and address the challenges of text generation effectively.

References

[1] Ariana Martino, Michael Iannelli Coleen Truong, "Knowledge Injection to Counter Large Language Model (LLM) Hallucination", paper$_M artino_2 023_K nowledge.pdf$

[2] Haziqa Sajid, "Battling LLM Hallucinations in Biomedicine: The Role of Knowledge Graphs in Knowledge Injection Techniques", Battling LLM Hallucinations