Generating Synthetic Dataset for RAG

Synthetic Data for RAG Setup

Unfortunately, in the life of a Machine Learning Engineer, there's often a lack of labeled data or very little of it. Typically, upon realizing this, projects embark on a lengthy process of data collection and labeling. Only after a couple of months can one start developing a solution.

However, with the advent of LLM, the paradigm shifted in some products: now one can rely on LLM's generalization ability and test an idea or develop an AI-powered feature almost immediately. If it turns out to work (almost) as intended, then the traditional development process can begin.

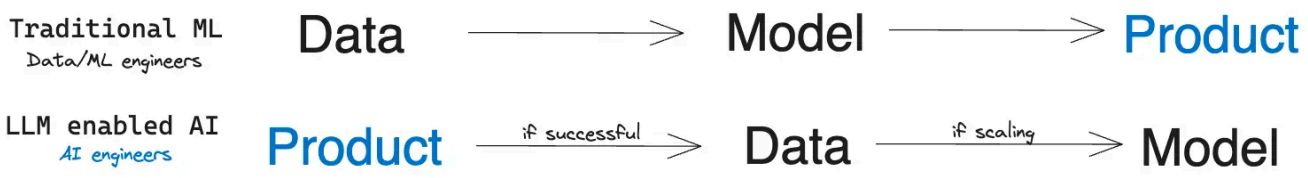

Image Source: The Rise of the AI Engineer, by S. Wang

One of the emerging approaches is Retrieval Augmented Generation (RAG). It's used for knowledge-intensive tasks where you can't solely rely on the model's knowledge. RAG combines an information retrieval component with a text generator model. To learn more about this approach, refer to the relevant section in the guide.

The key component of RAG is a Retrieval model that identifies relevant documents and passes them to LLM for further processing. The better the performance of the Retrieval model, the better the product or feature outcome. Ideally, Retrieval works well right out of the box. However, its performance often drops in different languages or specific domains.

Imagine this: you need to create a chatbot answering questions based on Czech laws and legal practices (in Czech, of course). Or design a tax assistant (a use case presented by OpenAI during the GPT-4 presentation) tailored for the Indian market.

You'll likely find that the Retrieval model often misses the most relevant documents and doesn't perform as well overall, thus limiting the system's quality.

But there's a solution. An emerging trend involves using existing LLMs to synthesize data for the training of new generations of LLMs/Retrievers/other models. This process can be viewed as distilling LLMs into standard-sized encoders via prompt-based query generation. While the distillation is computationally intensive, it substantially reduces inference costs and might greatly enhance performance, particularly in low-resource languages or specialized domains.

In this guide, we will rely on the latest text generation models, like ChatGPT and GPT-4, which can produce vast amounts of synthetic content following instructions. Dai et al. (2022) proposed a method where with only 8 manually labeled examples and a large corpus of unlabeled data (documents for retrieval, e.g., all the parsed laws), one can achieve a near State-of-the-Art performance. This research confirms that synthetically generated data facilitates training task-specific retrievers for tasks where supervised in-domain fine-tuning is a challenge due to data scarcity.

Domain-Specific Dataset Generation

To utilize LLM, one needs to provide a short description and manually label a few examples. It's important to note that different retrieval tasks possess varying search intents, meaning different definitions of "relevance." In other words, for the same pair of (Query, Document), their relevance might differ entirely based on the search intent. For instance, an argument retrieval task might seek supporting arguments, while other tasks require counter-arguments (as seen in ArguAna dataset).

Consider the example below. Though written in English for easier understanding, remember that data can be in any language since ChatGPT/GPT-4 efficiently processes even low-resource languages.

Prompt:

```
Task: Identify a counter-argument for the given argument.
Argument #1: {insert passage X1 here}
A concise counter-argument query related to the argument #1: {insert manually prepared query
Y1 here}
Argument #2: {insert passage X2 here}
A concise counter-argument query related to the argument #2: {insert manually prepared query
Y2 here}
<- paste your examples here ->
```

```
Argument N: Even if a fine is made proportional to income, you will not get the equality of
impact you desire. This is because the impact is not proportional simply to income, but must
take into account a number of other factors. For example, someone supporting a family will
face a greater impact than someone who is not, because they have a smaller disposable income.
Further, a fine based on income ignores overall wealth (i.e. how much money someone actually
has: someone might have a lot of assets but not have a high income). The proposition does not
cater for these inequalities, which may well have a much greater skewing effect, and
therefore the argument is being applied inconsistently.
A concise counter-argument query related to the argument #N:
```

Output:

```
punishment house would make fines relative income
```

In general, such a prompt can be expressed as:

$$(e_{prompt}, e_{doc}(d_1), e_{query}(q_1), ..., e_{doc}(d_k), e_{query}(q_k), e_{doc}(d))$$

, where e_{doc} and e_{query} are task-specific document, query descriptions respectively, e_{prompt} is a task-specific prompt/instruction for ChatGPT/GPT-4, and d is a new document, for which LLM will generate a query.

From this prompt, only the last document d and the generated query will be used for further training of the local model. This approach can be applied when a target retrieval corpus D is available, but the number of annotated query-document pairs for the new task is limited.

The whole pipeline overview:

Image Source: [Dai et al. (2022)]

It's crucial to handle manual annotation of examples responsibly. It's better to prepare more (for instance, 20), and randomly pick 2-8 of them to the prompt. This increases the diversity of generated data without significant time costs in annotation. However, these examples should be representative, correctly formatted, and even detail

specifics such as the target query length or its tone. The more precise the examples and instructions, the better the synthetic data will be for training Retriever. Low-quality few-shot examples can negatively impact the resulting quality of the trained model.

In most cases, using a more affordable model like ChatGPT is sufficient, as it performs well with unusual domains and languages other than English. Let's say, a prompt with instructions and 4-5 examples typically takes up 700 tokens (assuming each passage is no longer than 128 tokens due to Retriever constraints) and generation is 25 tokens. Thus, generating a synthetic dataset for a corpus of 50,000 documents for local model fine-tuning would cost: `50,000 * (700 * 0.001 * $0.0015 + 25 * 0.001 * $0.002) = 55`, where `$0.0015` and `$0.002` are the cost per 1,000 tokens in the GPT-3.5 Turbo API. It's even possible to generate 2-4 query examples for the same document. However, often the benefits of further training are worth it, especially if you're using Retriever not for a general domain (like news retrieval in English) but for a specific one (like Czech laws, as mentioned).

The figure of 50,000 isn't random. In the research by Dai et al. (2022), it's stated that this is approximately the number of manually labeled data needed for a model to match the quality of one trained on synthetic data. Imagine having to gather at least 10,000 examples before launching your product! It would take no less than a month, and the labor costs would surely exceed a thousand dollars, much more than generating synthetic data and training a local Retriever Model. Now, with the technique you learned today, you can achieve double-digit metric growth in just a couple of days!

Image Source: Dai et al. (2022)

And here are prompt templates from the same paper for some of the datasets in BeIR benchmark.

Image Source: Dai et al. (2022)

Last updated on September 19, 2024