ElasticSearch: Natural Language Interface Using LLMs

Alex Towell

- Making all endpoints on the internet and UIs intelligent with small and fast LLMs.
- As a trial, we are using ElasticSearch as a backend to enable natural language queries (NLQs) on ElasticSearch indexes (databases).
- Key take-aways: GPT-4 is good, GPT-3.5 is reasonable, and small LLMs are not quite there yet.
  - We have some ways to possibly improve them though. More on that later.
  - And, of course, today's large models are tomorrow's small models.
  - We desperately need more compute!

ElasticSearch: What Is It?

- An open source, scalable search engine.
- Supports complex queries, aggregations, and full-text search.
- Can be difficult to use.
- Suppose we have `articles` index with `author` and `title` fields and want to count the number of articles by author:

```
{
  "size": 0,
  "aggs": {
    "articles_by_author": {
      "terms": { "field": "author" }
    }
  }
}
```

FastAPI: What Is It and How Do We Use It?

- A fast web framework for building APIs with Python.
- We are trying two things:

  - Using ElasticSearch backend for storage and search.
  - Using LLMs to convert natural language queries (NLQ) to ElasticSearch queries.
- We expose a single endpoint `/{index}/nlq` that takes an index and an NLQ and returns a result from ElasticSearch.
  - Hopefully the result is useful!
- Later, remind me to open my firewall to allow access.

Structure of Indexes

I populated ElasticSearch with a two example indexes:

- `articles`: A simple index with `author`, `title`, and 'publication_date' fields.
- `gutenberg`: A more complex index with `author`, `publication_date`,`title`, and `content` fields.

Code

Let's look at some code. We'll switch to the code editor. There are a few files we need to look at:

- `main.py`: The FastAPI app.
  - We can probe it using the Swagger UI at `http://lab.metafunctor.com:6789/docs`.
  - There is a crude frontend at `http://lab.metafunctor.com:6789/`.
  - I made the frontend by chatting with ChatGPT-4. By chatting, I mean asked two typo-ridden ill-formed questions and copied its code blocks. See this link: https://chat.openai.com/share/9c95ba2e-94e7-4d9f-ae89-095357fc39bd
- `nlq.py`: The module that handles the NLQ to ElasticSearch query conversion.
- `examples.py`: A crude example database. We'll talk about this in a bit.

Issues

- GPT-4 is good at converting NLQs to ElasticSearch queries, but it's slow and expensive to use at scale.
  - We only need to use an LLM for a relatively narrow task.
    - Maybe we don't need the full power of GPT-4?
- Small LLMs, like `llama2`, did poorly on converting NLQs to ElasticSearch queries.

Idea #1: Use GPT-4 to "Teach" Smaller Models

- Use GPT-4 to generate high-quality examples for smaller LLMs.

- We can feed the examples into the context of the small LLM to do In-Context Learning (ICL).
  - ICL is a technique that allows a model to learn and generalize from examples.
- How? Every now and then, or upon request, we can use GPT-4 to do the task and store its NLQ to ElasticSearch query in an example database.
  - Let's look at the `examples.py` code.

Note on Implementation

- Examples database is just a dictionary that doesn't persist.
  - Didn't have the time to implement a proper database.
  - Ironic considering this is all about how to use ElasticSearch!

Issues

The smaller models, like `llama2:13b` , do not seem to generalize from the examples very well. + They often do better without "polluting" their context with too much information. + More tweaking? Or are these small models simply not up to the task.

Idea #2: RAG (Retrieval-Augmented Generation)

- Maybe the smaller models need to be fed with more *relevant* examples.
- Ue RAG to find relevant examples for the given index and NLQ. This would allow us to find examples that are most relevant to the NLQ.
  - Send the context through a language model to get a dense representation of the context.
  - Store the dense representation of the examples in the database.
  - Find examples closest to the context of the NLQ and sample from them.
- Insert the high-quality examples into the context of the small LLM to do In-Context Learning.

Idea #3: Fine-Tuning

- Fine-tune the smaller models on far more high-quality examples.
- Small LLMs won't have to In-Context Learn as much.
- See my GitHub repo: https://github.com/queelius/elasticsearch-lm
  - Its README has a lot of verbiage.

– I just ran it through GPT-4 and didn't bother to edit it much.