The 34th Meeting of Computational Linguistics in The Netherlands - Leiden 2024

ecovadis

Case Study: Leveraging Large Language Models in a Hybrid Search System for Improved CSR Information Retrieval

Lorenzo Lazzari, Carlos Villacampa Calvo, Sophia Katrenko, Geoffrey Carbonnel, Svenja Schoe, Frank Soetebeer

Background

At EcoVadis, we specialize in providing comprehensive sustainability ratings for businesses[1], assessing their performance based on 21 Corporate Social Responsibility (CSR) criteria across key areas such as Environment, Labour and Human Rights, Sustainable Procurement, and Fair Business Practices. With a vast network encompassing over 130,000 companies worldwide, our evaluation process involves analyzing a significant amount of unstructured, multilingual data. Each month, thousands of suppliers complete tailored questionnaires and submit accompanying documents to provide evidence to their responses. Our sustainability analysts then review these questionnaires and validate the provided answers against the corresponding documents.

[1]ecovadis.com/our-impact

Objective

An advanced information retrieval (IR) system is essential to assist our analysts by extracting the most relevant pages for supplier answer validation from the provided documents, which can range from a few pages to over hundreds. This system must be adept at understanding and considering the nuanced subtleties of the question and the respective answers in relation to the document.
Developing such a system is crucial for efficiently and accurately validating supplier sustainability, ensuring that our analysts can focus on meaningful evaluation rather than manual document navigation.

1st STEP : RETRIEVAL

In the first step of our proposed retrieval system, we aim to retrieve a pool of relevant candidate pages. To achieve this, we harness the Elasticsearch (ES) search engine, leveraging the Best Matching 25 algorithm (**BM25**) for its effectiveness in ranking documents based on their relevance to the query term. The question and the answer are preprocessed to extract the needed keywords for the query.
The document undergoes text extraction and is subsequently indexed in Elasticsearch using appropriate analyzers to ensure optimal retrieval performance. This indexing process involves the creation of a sparse vector model, which allows ES to efficiently search and rank the documents.

- Fast / Computationally inexpensive
- Highly scalable

- Misses semantic meaning
- Dependency on exact match of query terms

While the initial sparse vector retrieval using BM25 has its limitations, it reliably and efficiently selects a set of viable candidate pages, even for very long documents, with minimal computational expense.

2nd STEP : RERANKING AND FILTERING

In the second step of the retrieval process, we aim to refine the pool of candidate pages retrieved in the first step. Here, we harness the advanced comprehension capabilities of Large Language Models (LLMs). We use the full question and answer text as input for the model, allowing it to grasp the context and nuances necessary for accurate understanding. The LLM is employed to **re-rank** the candidate pages based on their relevance and to **filter out** those that are not pertinent. We experimented with both an unsupervised approach, directly using a commercially available pre-trained LLM, and a supervised approach, fine-tuning an open-source LLM with previous validation data from our database provided by our analysts.

UNSUPERVISED *PROMPT-ENGINEERING* APPROACH

Prompt strategy (Chat History)

SYSTEM ROLE
You are an an ESG expert that helps **reorder** a list of pages according to their relevance to a given query .... The result should be in **JSON** ...
Please remove any irrelevant pages from the list and do not repeat any page. In case all the pages are irrelevant ....

One-shot example

```
EXAMPLE_USER:
{
    "query": "What actions are in place .. Provision of products with an eco-label.. ",
    "pages": {
        "Page 1": "....in accordance with the union of... "
        "Page 2": "... in all the labels we commited...}",
        "Page ...": "....based on the agreement and...",

    }
}

EXAMPLE_ASSISTANT:
{
    "reranked_page_numbers": [4,2]
}
```

User Prompt (fill-in)

```
USER:
{
    "query": {QUESTION} {ANSWER},
    "pages": {
        "Page 1": {PAGE_CONTENT}
        "Page 2": {PAGE_CONTENT},
        "Page ...": {PAGE_CONTENT},
        .........
    }
}
```

OpenAI GPT-4o / Gpt4-Turbo

```
ASSISTANT:
{
    "reranked_page_numbers": [n1,n2,..]
}
```

Model Output

In the unsupervised approach, we use a chat-based pre-trained LLM out-of-the-box to perform the reranking and filtering tasks. We experimented with several models, both open-source and proprietary, ultimately achieving the best results with OpenAI's GPT-4o / Gpt4-Turbo.

Prompt strategy We aimed to achieve reranking and filtering with just one model response. To do this, we set the system role to define the task and used a one-shot learning approach, providing a user request and expected model answer in the chat history.
One challenge was structuring the data to prevent the model from hallucinating reranking numbers in the output list. We presented the data in a structured JSON format, which GPT-4 understands well. Each page's content was the value corresponding to a key labeled "Page x," with x representing the page numbers from 1 to n. To avoid confusion with numerical content within the text, we removed instances of "page," "pages," or "pag" during a cleaning step.

- Quick setup
- Adaptability

- High computational resources of OpenAI flagship models
- No explicit numeric score for relevance

SUPERVISED *FINE-TUNING* APPROACH

Training data

```
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 0 }
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 0 }
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 1 }
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 0 }
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 0 }
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 1 }
{"text": {QUESTION} + {ANSWER} + SEP_TOKEN + {PAGE_CONTENT}, "relevance": 1 }
.........
.........
.........
```

The "page-level binary labeled data" (❊), which provides the relevance of a given single page's content to a "question + answer" query, was used to build our training dataset. Using this dataset, we fine-tuned a pre-trained, open-source, decoder-only LLM, specifically the LLama 3-8B model by Meta, available through the HuggingFace framework. Smaller encoder-decoder architectures were unsuitable due to their 512-token limit, whereas our "query + page content" tokenized text samples averaged ~900 tokens with peaks above 3000 tokens.

We adapted the LLM with a classifier architecture by adding a classification layer on top, instead of the standard language modeling layer used for predicting subsequent tokens. The text for classification included the query and page content, separated by the "Beginning of Sentence" token.

The fine-tuned LLM model serves as a binary classifier, assessing the relevance of a single candidate page to the query and assigning a probability score of belonging to the "relevant" class. These scores are used to re-rank and filter the top n candidate pages retrieved by the initial search engine, setting a removal threshold t, with 0 < t < 1, but typically < 0.5 to filter out only pages for which the model has predicted a very low score for relevance.

- Continuous improvement loop: system is learning from feedback
- Open source model
- Numeric score for relevance to the query
- Threshold on score to control filtering

- Careful training
- Manage model deployment and scalability
- Adaptability

Experiments The retrieval performance of the two approaches was evaluated using a dataset of 2194 question-answer pairs, where the pages belonging to English-only relevant documents, identified by the analyst as supporting evidence, were known. The evaluation was conducted using precision and recall metrics at both @1 and @3. For the supervised approach, both GPT-4 and GPT-4 Turbo were tested, with GPT-4 slightly outperforming GPT-4 Turbo at @3. In the unsupervised approach, the fine-tuned LLaMA 3 8B model was evaluated using two filtering thresholds: a more lenient threshold of t=0.15 and a stricter one of t=0.4, which resulted in fewer pages being retrieved.

Takeaways

- ***Convenient Integration of Traditional Search Engines with LLMs:*** Traditional search engines can be effectively and easily combined with LLMs to enhance IR performance.
- ***Efficacy of Smaller, Fine-Tuned LLMs:*** Lightly fine-tuned, task-specific smaller open-source LLMs can achieve comparable results to larger, more resource-intensive models while running on relatively accessible hardware.
- ***Supervised Approach for Continuous Learning:*** A supervised approach allows the LLM to continuously learn and improve over time, closing the MLOps lifecycle, achieving better performance and dynamically adapting to the task.

Which metrics?
- *Precision at k (P@k) measures the proportion of relevant pages among the top k retrieved pages, calculated as the number of relevant pages retrieved in the top k divided by k.*
- *Recall at k (R@k) measures how well the top k retrieved pages capture all relevant pages, calculated as the number of relevant pages retrieved in the top k divided by the total number of relevant pages.*

Fine-tuning process

The fine-tuning of the 8-billion parameter LLaMA model was performed using QLoRA (Quantized Low-Rank Adaptation), employing 4-bit quantization to reduce both the memory (VRAM) footprint and computational requirements while modifying a limited subset of parameters. The LoRA parameters were configured with r=32 and α=16, impacting 83 million (1.10%) of the model's parameters across all linear layers. The weights of the final classification head (4096x2) were fully trained.
The fine-tuning was conducted on a single TESLA T4 GPU with 16GB VRAM, using approximately 36,000 question-answer pairs over one epoch. The instantaneous batch size was 2, with a total training batch size of 4 achieved through gradient accumulation steps of 2. The training speed was 4 seconds per sample, with a consistent reduction in cross-entropy loss observed throughout the whole process. The final LLM-based binary classifier achieved a 75% accuracy on a separate 2000 question-aswer pairs validation set in predicting the right relevant/irrelevant class.