Keyword and Semantic Searches with ReRank

Special Topics: Generative AI-Driven Intelligent Apps Development

Ademilton Marcelo da Cruz Nunes (19679)

Links

Github:

https://github.com/ademiltonnunes/Machine-Learning/tree/main/GenerativeAI/Fine-Tuning/Keyword%20and%20Semantic%20Searches%20with%20ReRank

Table of Content

Introduction

This project exemplifies the use of Rerank technology to improve the results of key search and semantic search. To do this project, I gave an example of key search and semantic search in a Dense Retrieval.

Through this exploration, this aims to uncover insights into how Rerank technology contributes to the improvement of information retrieval, particularly in the realms of key search and semantic search.

Introduction - Keyword Search

Keyword search is a type of search methodology used in information retrieval systems to locate and retrieve relevant documents or information. The process involves identifying and matching specific words or terms, known as keywords, within the documents or content.

Keyword search may miss relevant documents if they use different keywords but convey the same meaning.

Introduction - Semantic Search

Unlike keyword search, instead of making comparisons of words in documents, semantic search makes comparisons with embedded chunks of documents that are semantically close.

Embeddings is numerical representations of words or longer text that coordinates to words in a multidimensional space, vectorstore. There, similar sentences or phrases have embeddings indexes that are close semantically. The numerical coordinates capture relationships between words, providing a nuanced understanding of language.

Introduction - Dense Retrieval

Dense retrieval uses semantic search in the context of search engines or question-answering systems. Dense retrieval enhances the accuracy and relevance of results when searching through large datasets.

Introduction - ReRank

In the process of semantic search, multiple potential answers may be retrieved, as more than one result can be very close semantically in the vector space.

ReRank is a crucial step employed to select the most relevant answer from these candidates. It serves to enhance the performance of both Dense Retrieval and Keyword Search by reordering or re-ranking search results, aiming to improve the overall relevance and accuracy of the information retrieved. T

ReRank ensures that the most contextually appropriate results are prioritized, providing users with more accurate and meaningful responses to their queries.

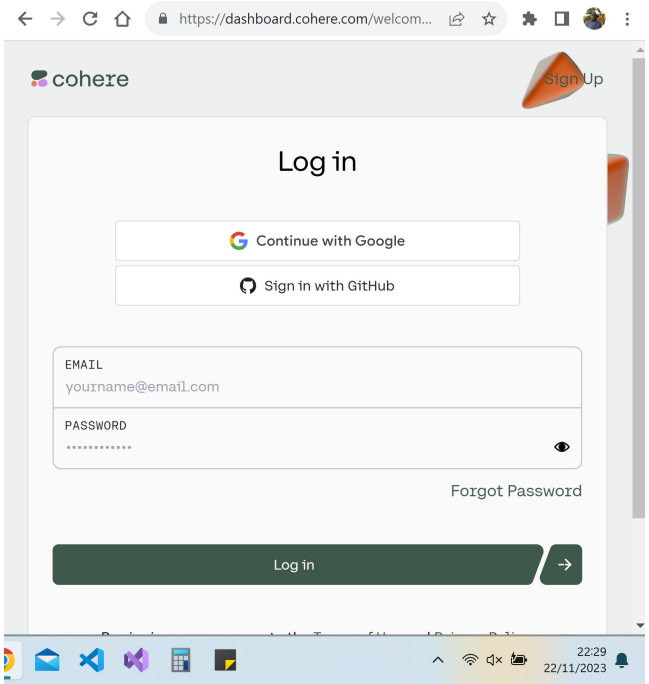
Implementation

This project implemented ReRank's performance in keyword and semantic search. To be able to do this we will use two tools:

- **Cohere**: Cohere is a powerful library that provides features as 'embed' function. This function is designed to generate embeddings for words or phrases.
- **Weaviate**:we need to have a database to make our tests. Weaviate is an open-source database with powerful keyword and vector search capabilities. It houses 10 million Wikipedia-based records across 10 languages, each representing a paragraph.

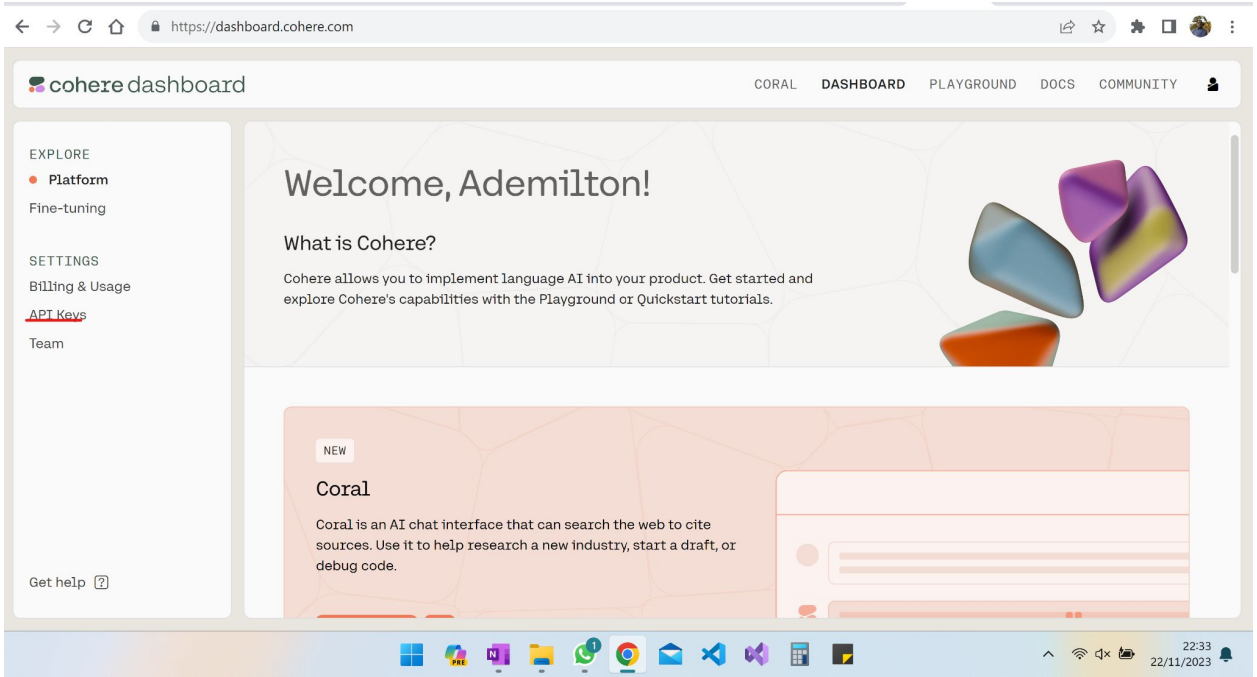
Setting Cohere

Cohere is accessed by the website: https://docs.cohere.com/docs. We can sing up using Google or Github accounts.

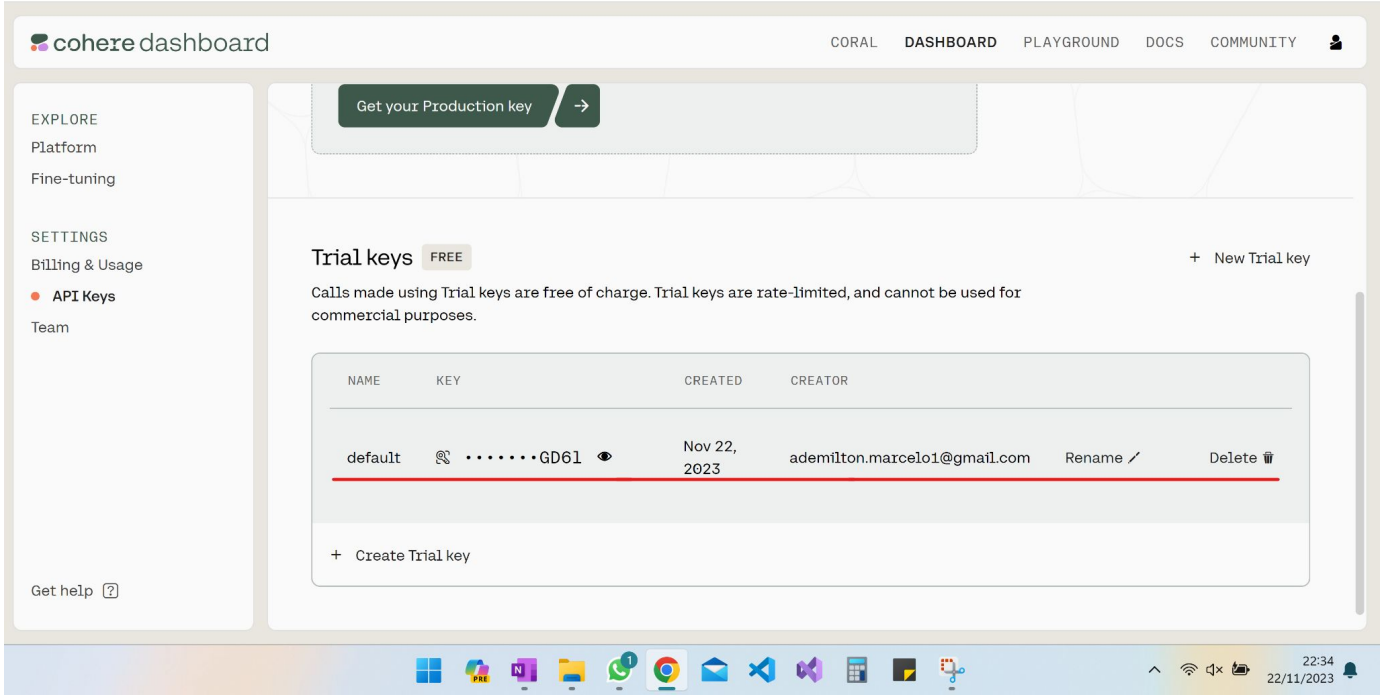


Setting Cohere

After creating our account, we can have access to our API key, this API we are going to use in our application.



Setting Cohere



Setting weaviate

The public weaviate database by:

- API key: 76320a90-53d8-42bc-b41d-678647c6672e
- URL: https://cohere-demo.weaviate.network/

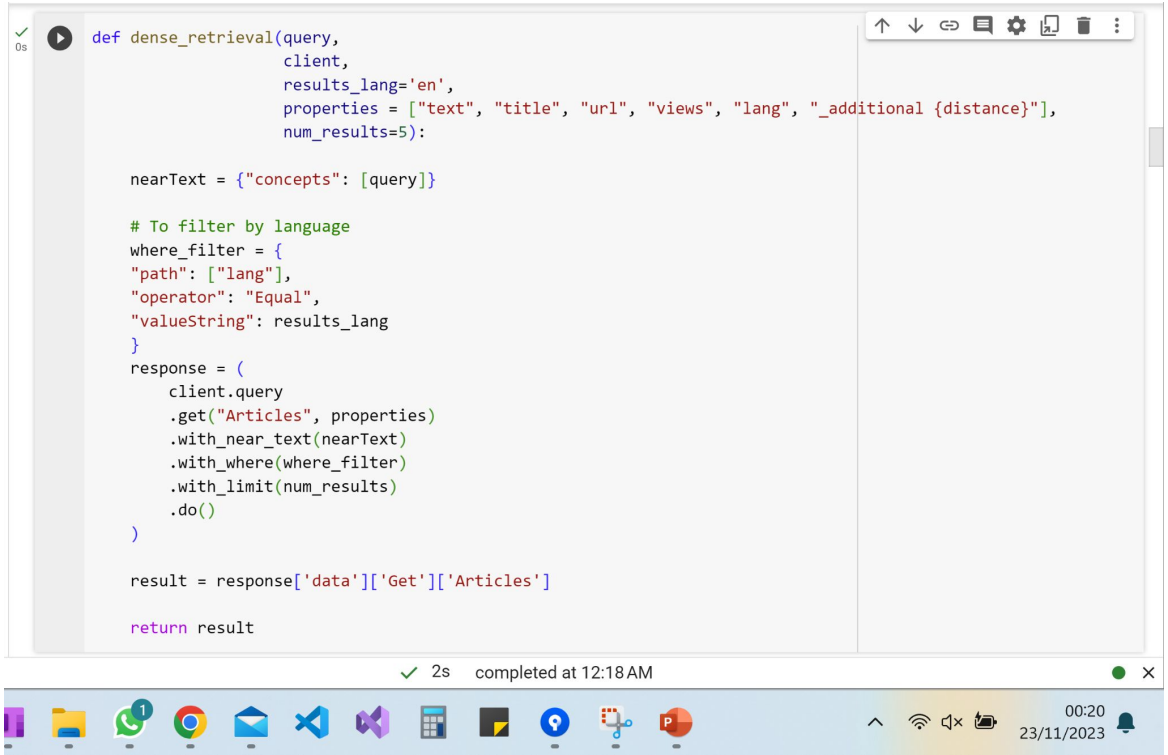
Setting system environment - Cohere and weaviate

We have to install the modules:

- pip install cohere
- pip install weaviate-client

Dense Retrieval

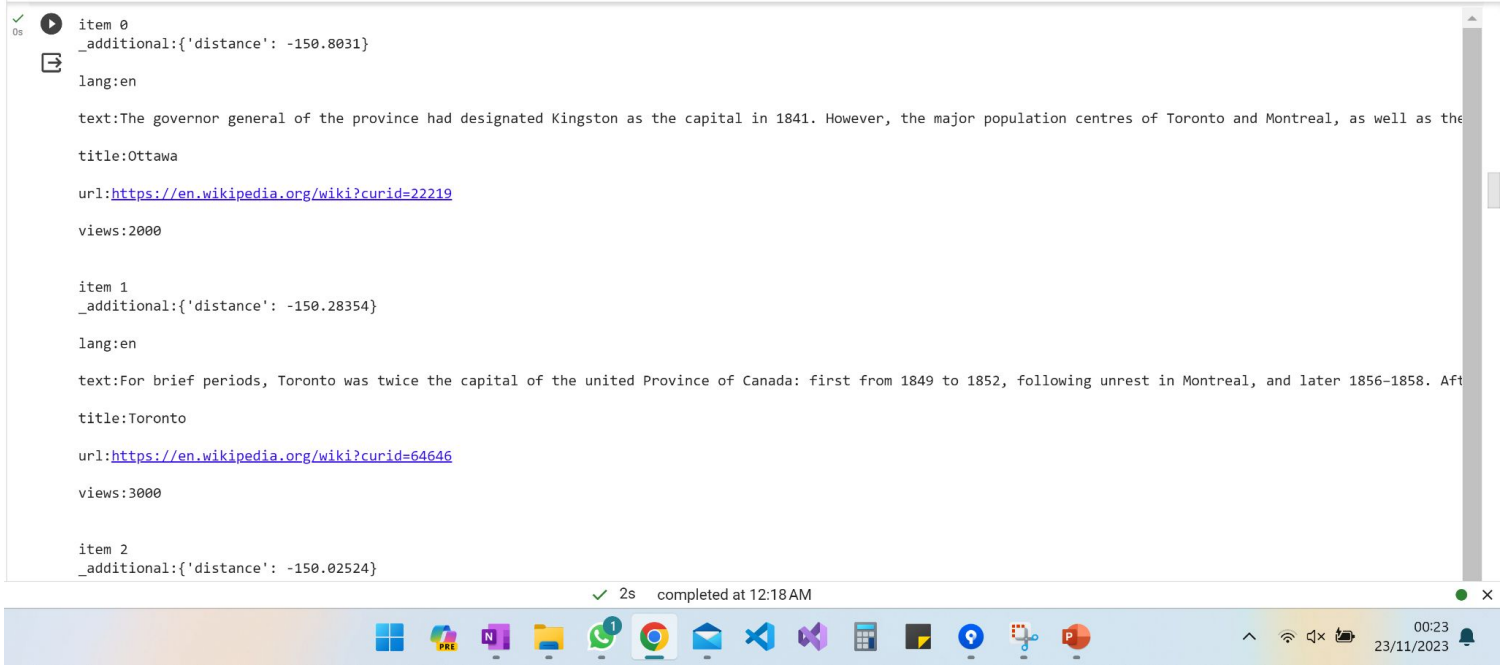
To do dese retrieval, we're going to use the function:



```python
def dense_retrieval(query,
                    client,
                    results_lang='en',
                    properties = ["text", "title", "url", "views", "lang", "_additional {distance}"],
                    num_results=5):

    nearText = {"concepts": [query]}

    # To filter by language
    where_filter = {
    "path": ["lang"],
    "operator": "Equal",
    "valueString": results_lang
    }
    response = (
        client.query
        .get("Articles", properties)
        .with_near_text(nearText)
        .with_where(where_filter)
        .with_limit(num_results)
        .do()
    )

    result = response['data']['Get']['Articles']

    return result
```

Dense Retrieval Test

We asked the query: "What is the capital of Canada?". It retrieved 5 different results. For example:

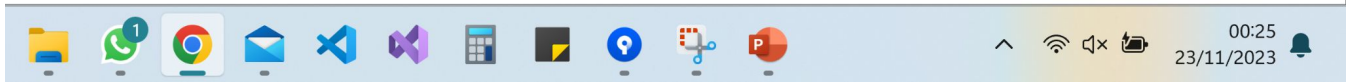


KeyWord Search

```
def keyword_search(query,
                   client,
                   results_lang='en',
                   properties = ["text", "title", "url", "views", "lang", "_additional {distance}"],
                   num_results=3):

    where_filter = {
    "path": ["lang"],
    "operator": "Equal",
    "valueString": results_lang
    }

    response = (
        client.query.get("Articles", properties)
        .with_bm25(
          query=query
        )
        .with_where(where_filter)
        .with_limit(num_results)
        .do()
        )
    result = response['data']['Get']['Articles']
    return result
```

```
[21] query_1 = "What is the capital of Canada?"
```

2s completed at 12:18 AM



Key Word Search Test

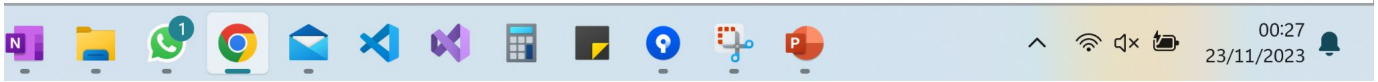
We asked the same query: "What is the capital of Canada?". The result can be huge, since we are using a big database. We are going to make a test with 500 results.

Key Word Search Test

```
# Keyword Search with 500 results
query_1 = "What is the capital of Canada?"
results = keyword_search(query_1,
   client,
   properties=["text", "title", "url", "views",
               "lang",
               "_additional {distance}"],
   num_results=500
   )

for i, result in enumerate(results):
    print(f"i:{i}")
    print(result.get('title'))
    #print(result.get('text'))
```

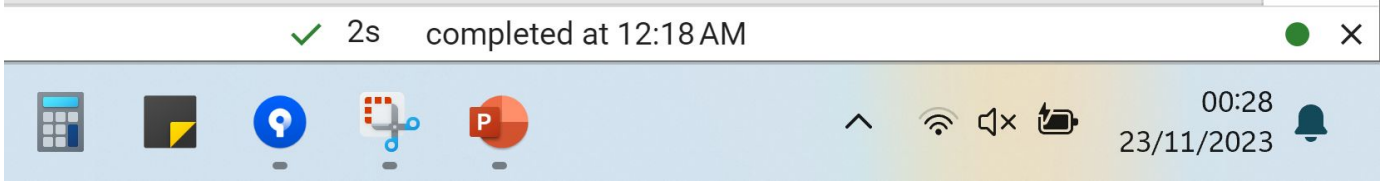
```
i:0
Monarchy of Canada
i:1
Early modern period
i:2
Flag of Canada
i:3
Flag of Canada
i:4
Prime Minister of Canada
i:5
Hamilton, Ontario
i:6
```

✓ 2s completed at 12:18 AM



ReRank

```
# ReRank of the Keyword Search results
def rerank_responses(query, responses,
         num_responses=10):
    reranked_responses = co.rerank(
        model = 'rerank-english-v2.0',
        query = query,
        documents = responses,
        top_n = num_responses,
        )
    return reranked_responses
```

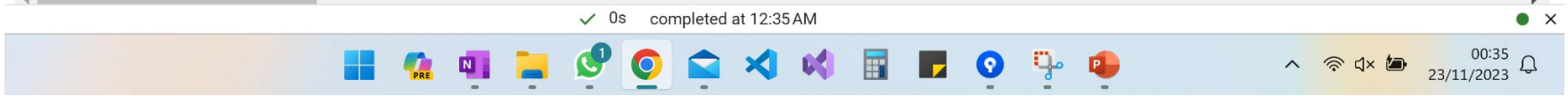


Applying ReRank to Dense Retrieval

Instead of 5 results, after applying ReRank it return 4 results with different combination of semantic search.

Applying ReRank to Dense Retrieval

```
i:0
RerankResult<document['text']: Selection of Ottawa as the capital of Canada predates the Confederation of Canada. The selection was contentious and not straightforward,

i:1
RerankResult<document['text']: The Quebec Conference on Canadian Confederation was held in the city in 1864. In 1867, Queen Victoria chose Ottawa as the definite capital

i:2
RerankResult<document['text']: The governor general of the province had designated Kingston as the capital in 1841. However, the major population centres of Toronto and

i:3
RerankResult<document['text']: For brief periods, Toronto was twice the capital of the united Province of Canada: first from 1849 to 1852, following unrest in Montreal,

i:4
RerankResult<document['text']: Until the late 18th century Québec was the most populous city in present-day Canada. As of the census of 1790, Montreal surpassed it with
```



Applying ReRank to KeyWord Search

Applying ReRank to KeyWord search with 500 results, it returns 9 results.

Applying ReRank to KeyWord Search

```
[26]
0s
i:0
RerankResult<document['text']: Selection of Ottawa as the capital of Canada predates the Confederation of Canada. The selection was contentious and not straightforward,

i:1
RerankResult<document['text']: Montreal was the capital of the Province of Canada from 1844 to 1849, but lost its status when a Tory mob burnt down the Parliament buildi

i:2
RerankResult<document['text']: Ottawa is the political centre of Canada and headquarters to the federal government. The city houses numerous foreign embassies, key build

i:3
RerankResult<document['text']: Until the late 18th century Québec was the most populous city in present-day Canada. As of the census of 1790, Montreal surpassed it with

i:4
RerankResult<document['text']: Ottawa was chosen as the capital for two primary reasons. First, Ottawa's isolated location, surrounded by dense forest far from the Canad

i:5
RerankResult<document['text']: Canada is a country in North America. Its ten provinces and three territories extend from the Atlantic Ocean to the Pacific Ocean and nort

i:6
RerankResult<document['text']: Although both rebellions were put down in short order, the British government sent Lord Durham to investigate the causes. He recommended s

i:7
RerankResult<document['text']: Ottawa is headquarters to numerous major medical organizations and institutions such as Canadian Red Cross, Canadian Blood Services, Healt

i:8
RerankResult<document['text']: Ontario ( ; ) is one of the thirteen provinces and territories of Canada. Located in Central Canada, it is Canada's most populous province

i:9
RerankResult<document['text']: With sixty percent of Canada's steel produced in Hamilton by Stelco and Dofasco, the city has become known as the Steel Capital of Canada.

3s  completed at 12:32 AM
```

Conclusion

In conclusion, the implementation of Rerank technology in this project has demonstrated significant enhancements in both keyword search and semantic search within the framework of Dense Retrieval. By employing Rerank, we observed notable improvements in the relevance and accuracy of search results. In keyword search, Rerank contributed to a more nuanced understanding of the user's intent, reducing the likelihood of missing relevant documents due to variations in keyword usage. Similarly, in semantic search, the utilization of Rerank led to a refined retrieval process, ensuring that documents with closer semantic relationships were prioritized.

The project's outcomes strongly indicate that integrating Rerank technology into information retrieval systems, especially within Dense Retrieval, is an effective strategy for optimizing search results. The ability to re-rank and prioritize documents based on semantic understanding significantly contributes to the overall success of both keyword and semantic searches. This not only bolsters the precision of results but also enhances the user experience by delivering more relevant and contextually appropriate information. The positive results observed in this exploration underscore the valuable impact of Rerank technology in advancing the effectiveness of information retrieval processes.