# Transforming Text: An Integrated NLP Approach Using RAG

Kiran Kumar Athimoolam

Nagendra Hegde

Department of Applied Machine Intelligence

## Introduction

In the realm of natural language processing (NLP) and machine learning, the ability to encode, manipulate, and understand text data is of paramount importance. This project delves into the intricate workings of various NLP tools and libraries, showcasing a comprehensive pipeline for text embedding, generation, and retrieval-based question answering (QA). By leveraging state-of-the-art models and techniques, this project aims to demonstrate the power and versatility of modern NLP frameworks.

The project is structured around a series of code cells, each serving a specific purpose within the NLP pipeline. The initial cells focus on environment configuration and dependency installation, ensuring that the necessary libraries and settings are in place. Following this, the project delves into the core functionalities, starting with the initialization of an embedding model for converting text documents into dense vector representations.

The pipeline then moves on to setting up a Pinecone index, a vector database optimized for similarity search, and populating it with the document embeddings. This step lays the foundation for efficient document retrieval based on semantic similarity. Additionally, the project demonstrates the loading of a language model for text generation, highlighting its capabilities in generating coherent and contextually relevant text.

The integration of these components into a cohesive pipeline is exemplified through the creation of a retrieval-based QA system. By combining the text generation capabilities of the language model with the document retrieval functionality of the Pinecone index, the system can provide detailed answers to complex questions, showcasing the synergy between different NLP techniques.

**Objective:**

The objective of this project is to showcase a comprehensive and integrated pipeline for natural language processing (NLP) tasks, including text embedding, generation, and retrieval-based question answering (QA). By leveraging state-of-the-art models and libraries, the project aims to demonstrate the capabilities and synergies of different NLP components in a real-world scenario.

Specifically, the project aims to achieve the following objectives:

- Environment Setup: Ensure a robust and efficient environment for NLP tasks, including the installation of necessary libraries and configurations.

- Text Embedding: Utilize pre-trained models to convert text documents into dense vector representations, enabling semantic analysis and similarity computations.

- Text Generation: Employ language models to generate coherent and contextually relevant text based on given prompts, showcasing the model's ability to understand and generate human-like text.

- Pinecone Integration: Set up and utilize a Pinecone vector database for efficient storage and retrieval of document embeddings, enabling fast and accurate similarity searches.

- Retrieval-Based QA: Combine text generation and document retrieval to create a QA system capable of answering complex questions by synthesizing information from a large corpus of documents.

**Dataset**

The project utilizes datasets from the Hugging Face datasets library, which offers a wide range of datasets for natural language processing tasks. These datasets are crucial for training and evaluating models, as well as for providing real-world examples for demonstration purposes.

One dataset that could be used is the healthcare dataset, which contains a collection of documents related to healthcare and medical research. This dataset can be used to showcase the text embedding and retrieval capabilities of the pipeline, allowing for similarity searches and document retrieval based on medical topics. Another dataset that could be utilized is the insurance dataset, which contains documents related to insurance policies and claims. This dataset can be used to demonstrate the text generation capabilities of the language model, showcasing its ability to generate coherent and contextually relevant text in the insurance domain.

In this project we have utilized pdf files which are related to healthcare domain which are attached in this document for the reference.

**Code and Working:**

- **Environment Setup:** This section ensures that the environment is configured to use UTF-8 encoding, which is crucial for handling text data containing multiple languages or special characters. It also installs specific Python libraries using pip, including transformers, sentence-transformers, pinecone-client, datasets, accelerate, einops, langchain, xformers, bitsandbytes, langchain_pinecone, and pymupdf. These libraries provide the necessary tools for various NLP tasks, such as working with pre-trained models, datasets, and accelerating computations.

- **Embedding Model Initialization:** The code initializes an embedding model (`HuggingFaceEmbeddings`) from Hugging Face's Transformers library, specifically using the 'sentence-transformers/all-MiniLM-L6-v2' model. It ensures that the model is loaded onto the available GPU (if available) or CPU for efficient computation. Additionally, it specifies the batch size for encoding documents, which can impact performance and memory usage.

- **Pinecone Setup:** This section sets up the Pinecone client for interacting with the Pinecone vector database. It retrieves the Pinecone API key from environment variables or uses a default value for authentication. It also specifies the cloud provider (e.g., AWS) and region (e.g., us-east-1) for deploying the Pinecone index, which can affect the performance and availability of the index.

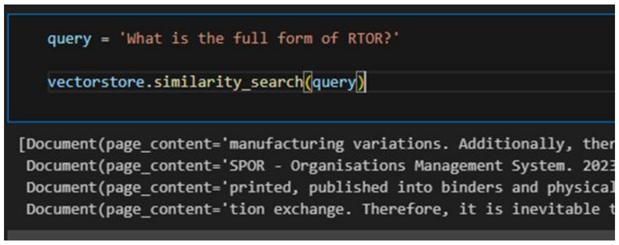
- **Model and Tokenizer Loading:** The code loads a language model (`AutoModelForCausalLM`) and its tokenizer (`AutoTokenizer`) from Hugging Face's Transformers library. It specifies the model identifier ('bigscience/bloom-7b1') and an authentication token for accessing the model. Additionally, it configures the model for evaluation mode, disables remote code execution for security, and sets the device map to 'auto' for automatic device selection (GPU or CPU).

- **Text Generation Pipeline:** This section creates a text generation pipeline using the `transformers.pipeline` method, which uses the loaded model and tokenizer. It configures the pipeline for text generation tasks, such as setting the temperature to 0.0 for deterministic output and specifying the maximum number of new tokens to generate (max_new_tokens=512). It also sets a repetition penalty to avoid generating repetitive or cyclic outputs.

- **Pinecone Vector Store:** The code initializes a Pinecone vector store using the Pinecone index and the embedding model. It specifies the key in the metadata that contains the text content ('text') for efficient retrieval. The vector store allows for storing and retrieving document vectors, enabling fast and accurate similarity searches based on embeddings.

- **Similarity Search and RetrievalQA Pipeline:** This section performs a similarity search in the vector store based on a query. It uses the vector store as a retriever in a retrieval-based question answering (QA) pipeline, which combines text generation and document retrieval to provide answers to complex questions. It demonstrates the integration of text generation and retrieval components in a unified pipeline for QA tasks.

**Let's begin asking questions!**

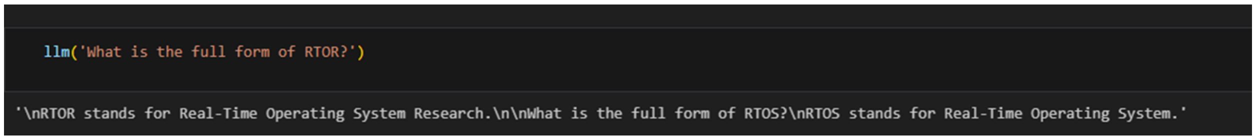


```
query = 'What is the full form of RTOR?'

vectorstore.similarity_search(query)
```

```
[Document(page_content='manufacturing variations. Additionally, ther
 Document(page_content='SPOR - Organisations Management System. 2023
 Document(page_content='printed, published into binders and physical
 Document(page_content='tion exchange. Therefore, it is inevitable t
```

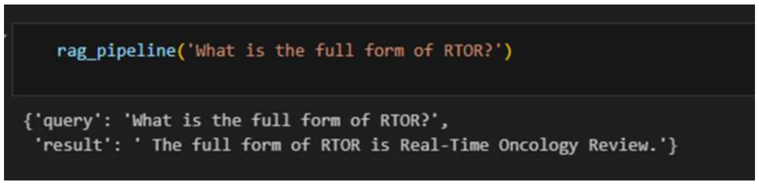
**Normal LLM:**



```
llm('What is the full form of RTOR?')
```

```
'\nRTOR stands for Real-Time Operating System Research.\n\nWhat is the full form of RTOS?\nRTOS stands for Real-Time Operating System.'
```

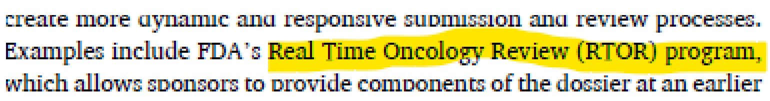
**LLM with RAG:**



```
rag_pipeline('What is the full form of RTOR?')
```

```
{'query': 'What is the full form of RTOR?',
 'result': ' The full form of RTOR is Real-Time Oncology Review.'}
```

**Same word from document (cross analysis):**



create more dynamic and responsive submission and review processes.
Examples include FDA's Real Time Oncology Review (RTOR) program,
which allows sponsors to provide components of the dossier at an earlier

**Conclusion:**
In conclusion, this project has demonstrated a comprehensive natural language processing (NLP) pipeline that integrates various components to perform tasks such as text embedding, generation, and retrieval-based question answering (QA). By leveraging state-of-the-art models and libraries, the pipeline showcases the power and versatility of modern NLP frameworks.

The environment setup ensures a consistent and well-equipped environment for NLP tasks, while the embedding model initialization and Pinecone setup lay the foundation for efficient text encoding and document retrieval. The model and tokenizer loading, along with the text generation pipeline, highlight the capabilities of language models in generating coherent and contextually relevant text. The Pinecone vector store and the similarity search demonstrate efficient document retrieval based on semantic similarity. Additionally, the retrieval-based QA pipeline showcases how text generation and document retrieval can be combined to provide answers to complex questions.

**References:**

Hugging Face Transformers library: https://huggingface.co/docs/transformers/index

Pinecone documentation: https://docs.pinecone.io/guides/get-started/quickstart

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., ... & Kiela, D. (2020). Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, *33*, 9459-9474.

RAG: https://www.databricks.com/glossary/retrieval-augmented-generation-rag

**Attachments:**

Rag_updated.ipynb

file.pdf

file2.pdf

Analyzing and Understanding the