Project Summary – Document Change Analyzer

In this project, I built a backend service that compares two versions of a document and tells whether the document has changed in a meaningful way.

The main goal was to take two text documents (for example, two versions of Terms & Conditions or a policy document) and analyze how different they are.

Tools and libraries used

- **Python** – main programming language
- **Hugging Face (Sentence Transformers)** – to understand text meaning
- **BentoML** – to turn the logic into a real API service

What Hugging Face was used for

Computers do not understand English text directly.
They understand **numbers**.

So, I used a Hugging Face embedding model:

- **Model used:** `sentence-transformers/all-MiniLM-L6-v2`

This model converts text into numeric vectors (embeddings).

Each document becomes a list of numbers that represents its meaning.
If two documents have similar meaning, their vectors will be close to each other.

What embeddings do in this project

- Document Version 1 → converted into numbers
- Document Version 2 → converted into numbers

Then those two numeric vectors are compared using **cosine similarity**.

- High similarity score → meaning is mostly the same
- Low similarity score → meaning has changed

Based on this score, the project classifies the change as:

- **minor or no change**
- **major change**

What BentoML was used for

BentoML was used to wrap all this logic into a **real service**.

With BentoML:

- The code is written in a `service.py` file
- A service class is defined
- An API endpoint (`analyze`) is exposed
- The service can be started using a single command

This makes the project usable as an API instead of just a script.

How the service works (high level)

1. The service receives two text documents as input
2. Hugging Face converts both documents into embeddings
3. The embeddings are compared
4. A similarity score is calculated
5. The service returns a JSON response with:
    - similarity score
    - type of change (major or minor)

Files in the project

- `service.py`
    - Main production file
    - Contains Hugging Face model loading
    - Contains BentoML service and API logic
- `analysis.ipynb`
    - Used for experimentation and explanation
    - Not required for running the service
- `Policy_v1.txt`, `Policy_v2.txt`
    - Sample input documents
- `output.json`
    - Example output format

What this project demonstrates

- Using Hugging Face pretrained models in a practical way
- Converting text into embeddings for semantic comparison

- Building a production-style API using BentoML
- Separating experimentation from deployment logic

Final note

This project focuses on **semantic document comparison**, not text generation or chatbots. It shows how machine learning models can be used inside a real service to solve a practical problem.