

Schritt-für-Schritt Erklärung des Embedding Codes

Verwendete Modelle: sentence-transformers/all-MiniLM-L6-v2, intfloat/e5-small-v2, sentence-transformers/all-mpnet-base-v2, intfloat/e5-base-v2, BAAI/bge-base-en-v1.5, nomic-ai/nomic-embed-text-v1

Erklärung Abkürzungen: BB_T → Bedeutung+ Beispiele und alle Texte

T_B_TE → nur Text, nur Bedeutung und Text + Erklärung

1. JSON laden & Textvarianten erstellen

- Die Funktion lädt eine JSON-Datei und wandelt diese in ein DataFrame um.
- Fehlende Spalten werden automatisch ergänzt, damit es nicht zu Fehlern kommt.
- Es werden verschiedene Textvarianten erzeugt: text_only, meaning_only, text_plus_example, example_plus_meaning und all_texts.
- Diese Varianten ermöglichen alternative Darstellungen für unterschiedliche Embedding-Zwecke.

2. Embeddings berechnen

- Hier werden Embeddings mit einem SentenceTransformer-Modell berechnet.
- Batching sorgt für schnellere Berechnung, die Normalisierung ist optional.
- Der Fortschrittsbalken zeigt an, wie viele Texte bereits verarbeitet wurden.

3. Ergebnisse speichern

- Die erzeugten Texte und Embeddings werden gespeichert.
- meta.csv enthält alle Textvarianten.
- Weitere Dateien enthalten die Embeddings.
- Falls der Zielordner nicht existiert, wird er automatisch erstellt.

4. Argumente einlesen

- Hier werden Kommandozeilenargumente definiert, damit der Code flexibel bleibt.
- Der Benutzer kann z. B. Eingabedatei, Modellname, Batch-Größe oder Zielordner ändern.

5. Hauptlogik

Dies ist der zentrale Ablauf des Skripts:

1. JSON laden
2. Modell laden
3. Embeddings berechnen
4. Ergebnisse speichern (sofern nicht deaktiviert)