

Embedding Süreci ve FAISS Entegrasyonu

Bu projede ilk olarak *embedding* kavramını araştırmam gerekti. Başlangıçta bu terime dair hiçbir fikrim yoktu. Araştırmalarım sonucunda embedding işleminin, kelime ya da cümleleri sayısal vektörlere dönüştürme işlemi olduğunu öğrendim. Bu vektör temsilleri sayesinde, anlam olarak benzer cümlelerin sayısal olarak da birbirine yakın olduğu anlaşılabilir. Bu yaklaşımın temelinde ise vektörler arasındaki *dot product* (noktasal çarpım) işlemi yatıyor. Fizikte de bilindiği üzere, iki vektör aynı yönü gösterdiğinde dot product maksimum olur. Bu da doğal dilde anlamca benzer iki cümlenin vektör temsillerinin de benzer yönlerde olacağı anlamına gelir.

Bu işlemi gerçekleştirmek için `sentence-transformers` kütüphanesini kullandım. Bu kütüphane Hugging Face üzerine inşa edilmiş bir yapıdır. Embedding için "`sentence-transformers/paraphrase-multilingual-MiniLM-L12-v2`" modelini seçtim. Bu modeli tercih etme sebebim, 50'den fazla dili desteklemesi; çünkü ileride sistemin çok dilli çalışabilmesi istenebilir.

Ardından, her biri `id`, soru ve cevap alanlarına sahip bir `dict` yapısında sorulardan oluşan bir liste oluşturdum. ID değerlerini, ileride spesifik sorulara ulaşmak için kullanabileceğimi düşündüm. Konu başlıklarını, aldığım makine öğrenmesi dersi doğrultusunda, bu alana uygun seçmeye çalıştım.

Bu aşamada embedding işlemi sorunsuz çalıştı ancak her çalıştırmada yeniden embedding hesaplamanın maliyetli ve zaman alıcı olduğunu fark ettim. Bu nedenle, bir defa oluşturduğum embedding'leri kaydedip gerektiğinde tekrar yükleyerek kullanabilmek istedim. Bunun için NumPy'nin `.npy` formatını ve Python'un `pickle` modülünü kullandım. embedding vektörlerini `.npy` dosyasına, soru metinlerini ise `pickle` dosyasına kaydettim.

Sonraki araştırmamda bu embedding vektörleriyle benzerlik karşılaştırmaları yapmanın birkaç yolu olduğunu öğrendim. Bunlar arasında FAISS ve ChromaDB gibi araçlar yer alıyordu. İlk olarak ChromaDB'yi denemek istedim çünkü daha basit görünüyordu; fakat ortamımda çalışmadığı için FAISS'e yöneldim. FAISS, daha karmaşık bir yapı gibi görünmesine rağmen, vektörler için yüksek performanslı bir indeksleme ve arama imkânı sunuyor.

FAISS kullanırken, öncelikle sistemde daha önce oluşturulmuş embedding, FAISS index ve metadata dosyalarının varlığını `os.path.exists` yöntemiyle kontrol eden bir `if-else` yapısı kurdum. Eğer bu dosyalar mevcut değilse, embedding hesaplaması yapılır ve dosyalar oluşturulur; mevcutsa doğrudan bu dosyalar yüklenerek sistem başlatılır.

Bu işlem sırasında embedding'ler .npy, indeks .index ve metadata ise .pk1 formatında kaydediliyor.

Son olarak basit bir query ("Eksik veriler nasıl doldurulur?") cümlesi ile FAISS arama fonksiyonunu test ettim. Sistem, bu sorguya en çok benzeyen soruları başarıyla getirerek doğru çalıştığını gösterdi. Aldığım çıktıları aşağıya ekliyorum:

```
Command Prompt
1 File(s) 5.675 bytes
2 Dir(s) 551,425,818,624 bytes free

(mlenv) C:\Users\Asus\OneDrive\Masaüstü\student_asistance_chat_bot>python embedding.py
modules.json: 100% | 229/229 [00:00<?, ?B/s]
C:\Users\Asus\lenv\Lib\site-packages\huggingface_hub\file_download.py:144: UserWarning: 'huggingface_hub' cache-system uses symlinks by default to efficiently store duplicated files but your machine does not support them in C:\Users\Asus\cache\huggingface\hub\models--sentence-transformers--paraphrase-multilingual-MiniLM-L12-v2. Caching files will still work but in a degraded version that might require more space on your disk. This warning can be disabled by setting the 'HF_HUB_DISABLE_SYMLINKS_WARNING' environment variable. For more details, see https://huggingface.co/docs/huggingface_hub/how-to-cache#limitations. To support symlinks on Windows, you either need to activate Developer Mode or to run Python as an administrator. In order to activate developer mode, see this article: https://docs.microsoft.com/en-us/windows/apps/get-started/enable-your-device-for-development
warnings.warn(message)
config_sentence_transformers.json: 100% | 122/122 [00:00<?, ?B/s]
README.md: 100% | 3.89k/3.89k [00:00<?, ?B/s]
sentence_bert_config.json: 100% | 53.0/53.0 [00:00<00:00, 53.0kB/s]
tokenizer.json: 100% | 645/645 [00:00<?, ?B/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better performance, install the package with: 'pip install huggingface_hub[hf_xet]' or 'pip install hf_xet'
model.safetensors: 100% | 471M/471M [00:31<00:00, 14.7MB/s]
tokenizer_config.json: 100% | 480/480 [00:00<?, ?B/s]
Xet Storage is enabled for this repo, but the 'hf_xet' package is not installed. Falling back to regular HTTP download. For better performance, install the package with: 'pip install huggingface_hub[hf_xet]' or 'pip install hf_xet'
tokenizer.json: 100% | 9.08M/9.08M [00:00<00:00, 27.6MB/s]
special_tokens_map.json: 100% | 239/239 [00:00<?, ?B/s]
config.json: 100% | 190/190 [00:00<00:00, 190kB/s]
(20, 384)
[[-0.06871915 0.1983286 -0.08697599 -0.11396378 0.13513622 -0.20604458
-0.04982806 0.06114782 -0.17934544 0.10481318 0.00427993 0.01745786
0.3671084 -0.3795869 -0.01921257 0.02035808 -0.10401969 0.14388049
0.01538158 0.07762358 -0.44341776 -0.09340625 -0.1500345 -0.1242678
0.12145871 0.33506706 -0.09562464 -0.00405159 -0.18331447 -0.243055
0.19475842 -0.1608516 0.05854328 0.05895997 0.2007705 -0.03025457
0.07651828 0.15538971 0.05857591 -0.11635134 0.18297787 0.13784
-0.01806576 0.01711403 -0.15330634 -0.16191939 -0.21291976 -0.2919169
-0.2894714 -0.27167112 -0.41335773 0.11818848 0.12191404 -0.18270794
-0.20942596 0.28570466 0.168143 0.06181632 -0.11927591 -0.21634927
0.32341003 0.12296139 -0.06038133 0.02484682 0.24666402 0.38852334
0.05698626 0.08410578 0.464186 -0.03415216 -0.33874747 0.39900327
-0.46489787 0.02967772 -0.11411417 0.11267351 -0.19170582 0.07812851
0.28873047 -0.17330636 0.1816104 0.453761 -0.04079556 -0.0833287
-0.08632369 -0.06880703 0.10274099 0.33492753 -0.08052965 -0.2873036]]

(mlenv) C:\Users\Asus\OneDrive\Masaüstü\student_asistance_chat_bot>python embedding_chromadb.py
✓ Kod başladı.
📁 Embedding'ler FAISS'e eklendi.
🔍 En yakın sonuçlar:
- Eksik verilerle başa çıkmak için hangi yöntemleri kullanabilirim? (ID: 1)
- Sürekli veri nedir ve nasıl işlenir? (ID: 12)
- Veri kümesindeki aykırı değerleri nasıl tespit edebilirim? (ID: 2)

(base) PS C:\Users\Asus\OneDrive\Masaüstü\student_asistance_chat_bot> python embedding_chromadb.py
✓ Kod başladı.
📁 İlk çalıştırma: veriler oluşturuluyor ve kaydediliyor...
✓ Embedding'ler ve index kaydedildi.

🔍 En yakın sonuçlar:
- Eksik verilerle başa çıkmak için hangi yöntemleri kullanabilirim? (ID: 1)
- Sürekli veri nedir ve nasıl işlenir? (ID: 12)
- Veri kümesindeki aykırı değerleri nasıl tespit edebilirim? (ID: 2)

(base) PS C:\Users\Asus\OneDrive\Masaüstü\student_asistance_chat_bot> python embedding_chromadb.py
✓ Kod başladı.
📁 Kayıtlı veriler bulundu, yükleniyor...

🔍 En yakın sonuçlar:
- Eksik verilerle başa çıkmak için hangi yöntemleri kullanabilirim? (ID: 1)
- Sürekli veri nedir ve nasıl işlenir? (ID: 12)
- Veri kümesindeki aykırı değerleri nasıl tespit edebilirim? (ID: 2)
```