Semantic Search Proje Raporu

- Vergi özelgelerinin semantik olarak aranmasını sağlamak, kullanıcıların istediği bilgiye hızlı ve doğru şekilde ulaşmasını kolaylaştırmak adına bir semantic search projesi geliştirildi.Bu süreçte ise özelge içeriklerinin web scraping ile toplanması, işlenmesi, embedding'lerin oluşturulması ve cosine similarity ile semantik arama yapılması gibi adımlar atıldı.

Kullanılan Teknolojiler ve Araçlar

- Kütüphaneler ve Teknolojiler:
    - Web Scraping: Selenium
    - Embedding ve Semantic Search:
        - SentenceTransformers (all-MiniLM-L6-v2)
        - FAISS (Facebook AI Similarity Search)
        - Cosine Similarity
    - Veri Tabanı: MongoDB
    - Arayüz: Streamlit
- Diğer Araçlar: ChromeDriver, dotenv

Çalışmanın Aşamaları

Web Scraping ile Veri Toplama

- Script: link_collection.py
    - Gelir İdaresi Başkanlığı özelge sayfalarından Selenium ile dinamik içeriklerin yüklenmesini bekleyerek sayfa linklerini toplama.
    - Sayfalardaki tüm özelge bağlantılarını ozelge_links.txt dosyasına kaydetme.
    - 220 sayfadan toplam 2200 adet özelge bağlantısı toplandı.

Özelge İçeriklerini Çekme ve Veritabanına Kaydetme

- Script: icerik_cekme.py
    - XPath ile özelge içeriği, tarihi, konusu ve indirme linklerini çekme. Bu xpathler variables.py dosyasından sağlanır.
    - Çekilen verileri MongoDB'ye şu yapı ile kaydetme:

```
{
 "id": "Özelge ID",
 "konu": "Konu",
 "ozelge_linki": "Link",
 "ozelge_tarihi": "Tarih",
 "indirme_linki": "İndirme Linki",
 "icerik": "Metin"
                    }
```

- Tüm özelgeler veritabanına kaydedildi. Eğer bir özelge zaten veritabanında varsa o özelge tekrar kaydedilmedi. Bunu sağlamak adına id üzerinden indexleme(unique) yapıldı.

Embedding Oluşturma

- Script: embedding_olustur.py
  - SentenceTransformers modeli (all-MiniLM-L6-v2) ile özelge metinlerini ve konu başlıklarını embedding'e dönüştürme.
  - MongoDB belgelerine embedding alanı ekleme. Veritabanının yeni yapısı:

```
{
 "id": "Özelge ID",
 "konu": "Konu",
 "ozelge_linki": "Link",
 "ozelge_tarihi": "Tarih",
 "indirme_linki": "İndirme Linki",
 "icerik": "Metin"
 "embedding": "Embedding(içerik için)"
 "embedding_konu": "Embedding Konu"
                    }
```

  - İçerik ve konu embeddingleri oluşturuldu ve MongoDB'ye kaydedildi.

Semantic Search ve Hibrit Arama

- Scriptler:
  - cosine_icerik_search.py (içerik bazlı arama)
  - cosine_konu_search.py (konu bazlı arama)
  - hibrit_search.py (içerik ve konu birleşimiyle arama)

Scriptler ve İşlevleri

1. cosine_icerik_search.py:
   - Kullanıcı sorgusunu alarak özelge içerikleri üzerinden semantik bir arama gerçekleştirir.
   - Sorgu embedding'ini oluşturur ve MongoDB'den alınan içerik embedding'leriyle cosine similarity hesaplayarak en alakalı sonuçları döner.
   - Daha hızlı bir sonuç için embeddingleri ve kullanılan diğer bilgileri bir belgeye(embedding_icerik_cache.npy gibi) kaydeder. Belge oluşmamışsa bilgileri databaseden alır ve belgeyi oluşturur. GPU kullanarak da sorgu hızını arttırır.
2. cosine_konu_search.py:
   - Kullanıcı sorgusunu özelge konuları üzerinden arar.
   - Sorgu embedding'ini oluşturur ve konu başlıklarının embedding'leriyle benzerlik hesaplar.
3. hibrit_search.py:
   - Hem içerik hem de konu embedding'leri üzerinde arama yapar.
   - İçerik ve konu bazlı benzerlik skorlarını birleştirir.

    - Kullanıcı sorgusuna göre her iki skorun bir ağırlık katsayısı ile birleştirilmesi sağlanır.
    - Bu adresten bu yaklaşımı deneyebilirsiniz: https://semanticsearchgib.streamlit.app/

Yöntem

- Kullanıcıdan alınan metin, SentenceTransformers (SBERT) modeli ile bir embedding'e dönüştürülür.
- SBERT modeli, sorgunun anlamsal temsilini (vektörünü) çıkarır, bu sayede metnin semantik anlamı korunur.

. Cosine Similarity ile Benzerlik Hesaplama:

- Sorgu embedding'i ile veri tabanındaki her bir embedding arasındaki cosine similarity hesaplanır.
- Cosine similarity, iki vektörün arasındaki açıya göre bir benzerlik skoru verir.

. FAISS Model:

- Benzerlik hesaplaması FAISS ile de yapılır. İçerik embeddingleri kullanılır.

. Hibrit Model:

- İçerik ve konu embedding'lerinden elde edilen cosine similarity skorları, birleştirilerek toplam bir benzerlik skoru oluşturulur.
- Kullanıcı arayüzünde sonuçlar bu toplam skorlara göre sıralanarak gösterilir.

Streamlit Uygulaması

- Streamlit uygulamasını çalıştırmak için:
  ```
  python -m streamlit run streamlit_app.py
  ```

- requirements.txt dosyasını kullanarak gerekli paketleri yüklemek için:
  ```
  pip install -r requirements.txt
  ```