Makale Öneri Sistemi

Meryem PAŞAOĞLU
Bilgisayar Mühendisliği
220201123@kocaeli.edu.tr

***Abstract*—Bu doküman Yazlab-II Proje-3 için hazırlanmıştır. makale öneri sistemi projesi deteylarını içerir.**

I. GİRİŞ

Bu proje, kullanıcıların ilgi alanlarına ve okuma geçmişlerine uygun kişiselleştirilmiş makale önerileri sunmayı amaçlayan bir sistem geliştirmeyi hedeflemektedir. Ana hedef, kullanıcılara ilgilendikleri doğru makaleleri yönlendirmektir.

A. Yapılan İşlemler

Projenin ana başlıkları şu şekildedir:

- 1. Veri Kaynağı:
- Hazır makale veri kümesi kullanılacaktır (örneğin, Inspec ve Krapivin2009).
- 2. Ön İşleme ve Veri Analizi:
- Makalelerin ön işlenmesi NLP teknikleri ile yapılacaktır.
- Python ile NLTK veya spaCy kütüphanelerinden biri kullanılarak metin ön işleme gerçekleştirilecektir.
- İngilizce stopwords temizlenmesi, noktalama işaretlerinin çıkarılması, kelime köklerinin bulunması.
- FastText ve SCIBERT modelleri kullanılarak makaleler ve kullanıcı profilleri için vektör temsilleri oluşturulacaktır.
- Kullanıcı profili, ilgi alanlarının vektör temsillerinin ortalaması alınarak oluşturulacaktır.
- Benzerlik değerlendirmesi Cosine Similarity metriği ile yapılacaktır.
- FastText ve SCIBERT için ayrı ayrı 5 öneri sunulacaktır.
- Performans değerlendirmesi Precision ve Recall değerleri ile yapılacak ve arayüzde gösterilecektir.
- 3. Kullanıcı Profil Oluşturma ve Yönetimi:
- Kullanıcıların sisteme kayıt olabilmesi ve demografik bilgileri ile akademik ilgi alanlarının alınması için bir arayüz tasarlanacaktır.
- Kullanıcı profilleri oluşturulacak, güncellenecek ve yönetilecektir.
- 4. Öneri Motoru :
- Başlangıç önerileri, kullanıcıların belirttiği ilgi alanlarına göre oluşturulacaktır.
- Öneriler, kullanıcının ilgi alanları ve okuma geçmişi göz önünde bulundurularak dinamik olarak güncellenecektir.
- Kullanıcı geri bildirimleri toplanarak model güncellenecek ve sonraki öneriler buna göre belirlenecektir.
- Kullanıcıya sunulan önerilerden her biri için uygun veya değil geri bildirimi alınacaktır.
- 5. Kullanıcı Arayüzü :
- Kullanıcı dostu bir web arayüzü geliştirilecektir.
- Kullanıcıların aradıkları makaleleri bulabilmesi için arama çubuğu ve filtreleme seçenekleri sağlanacaktır.
- 6. Kullanılacak Teknolojiler :
- Verilerin senkronizasyonu ve algoritmaların çalışabilmesi için uygun bir veri tabanı (örneğin, MongoDB) kullanılacaktır.
- Yapay zeka ve makine öğrenmesi için Python kullanılacaktır.

II. YÖNTEM

Projede bulunan dosyalar şu şekilde özetlenebilir:

inspecKelimeleriİsleme.py: Inspec veri setinde bulunan makalelerin anahtar kelimeleri ve özet kısımlarını ön işleme tabi tutarak, her bir makalenin kelimelerini bir listede topladım ve MongoDB veritabanında ”kelimeler” koleksiyonuna kaydettim. Ayrıca, bu kelimelerin vektörlerini çıkararak ortalama vektörleri hesaplamak amacıyla bir yapı oluşturdum. İşlemler aşağıda detaylı olarak açıklanmaktadır. Veri Tabanına Bağlantı İlk olarak, MongoDB veritabanına bağlantı kurmak amacıyla pymongo kütüphanesi kullanıldı. Bu bağlantı, MongoClient sınıfı aracılığıyla sağlanarak yerel bir MongoDB sunucusuna bağlanıldı ve yazlab isimli veritabanı ile kelimeler isimli koleksiyon seçildi. Bu adım, verilerin kalıcı olarak saklanabilmesi için gereklidir.

Doğal Dil İşleme Araçlarının Hazırlanması Doğal dil işleme (NLP) görevlerini gerçekleştirebilmek için nltk kütüphanesi kullanıldı. Bu kapsamda, İngilizce stopwords'leri (stopwords.words('english')) ve kelime tokenizasyonu (wordtokenize) için gerekli olan veri setleri indirildi. Aynı zamanda, kelime köklerini bulmak için PorterStemmer kullanıldı. Stopwords, metin içerisindeki anlamsız kelimeleri (örn. ”ve”, ”veya”) çıkarmak için, PorterStemmer ise kelimelerin köklerini bulmak için kullanıldı.

Veri Setinin Hazırlanması Veri setindeki anahtar kelimeler ve özetler, belirtilen klasör yollarından (Inspec/keys ve Inspec/docsutf8) okundu. os.walk fonksiyonu ile bu klasörlerdeki .key dosyaları (anahtar kelimeler) ve ilgili .txt dosyaları (özetler) tarandı.

Ön İşleme Her bir .key dosyasındaki anahtar kelimeler ve .txt dosyasındaki özetler sırasıyla şu ön işleme adımlarından geçirildi:

Küçük harfe dönüştürme (lower()). Noktalama işaretlerinin kaldırılması (translate(str.maketrans(”, ”, string.punctuation))). Kelimelerin tokenizasyonu (wordtokenize). Stopwords'lerin çıkarılması. Kelime köklerinin bulunması (PorterStemmer). Bu

adımlar sonucunda, metinler daha basit ve analiz edilebilir hale getirildi.

Verilerin Birleştirilmesi ve Kaydedilmesi Her bir makale için, anahtar kelimeler ve özet kısımları birleştirildi. Bu birleştirilen kelimeler, her bir makalenin ismi ile birlikte bir sözlük olarak kelimeeSozluguu listesine eklendi. Daha sonra, bu liste MongoDB koleksiyonuna toplu olarak kaydedildi (insertmany).

Verinin Görüntülenmesi Son olarak, oluşturulan veriler terminalde yazdırılarak her bir makalenin adı ve ilgili kelimeler listesi görüntülendi. Bu adım, verilerin doğru bir şekilde işlendiğini ve kaydedildiğini doğrulamak amacıyla yapıldı. Bu yöntem sayesinde, Inspec veri setindeki makalelerin kelimeleri etkili bir şekilde işlenmiş ve veri tabanında depolanmıştır. Bu işlemler, ileri düzey metin analizi ve kelime vektör hesaplamaları için sağlam bir temel oluşturmuştur.

inspecMakaleleri.py : Inspec veri setinde bulunan makalelerin ön işleme tabi tutularak, her bir makalenin numarası, anahtar kelimeleri ve özet bilgilerini MongoDB veritabanında ”inspec” koleksiyonuna kaydettim. Bu işlemler, metinlerin daha etkili bir şekilde analiz edilebilmesi ve saklanabilmesi amacıyla gerçekleştirildi. Aşağıda, bu işlemlerin detayları açıklanmaktadır.

Veri Tabanına Bağlantı İlk adım olarak, pymongo kütüphanesi kullanılarak MongoDB veritabanına bağlantı kuruldu. MongoClient sınıfı aracılığıyla yerel bir MongoDB sunucusuna bağlanıldı ve yazlab isimli veritabanı ile inspec isimli koleksiyon seçildi. Bu adım, verilerin güvenli ve kalıcı bir şekilde saklanabilmesi için gereklidir.

Doğal Dil İşleme Araçlarının Hazırlanması Doğal dil işleme (NLP) görevlerini gerçekleştirebilmek için nltk kütüphanesi kullanıldı. Bu kapsamda, İngilizce stopwords'leri (stopwords.words('english')) ve kelime tokenizasyonu (wordtokenize) için gerekli olan veri setleri indirildi. Stopwords'ler, metinlerde sıkça geçen ancak anlamsız olan kelimelerin (örn. ”ve”, ”veya”) çıkarılması için kullanıldı.

Veri Setinin Hazırlanması Veri setindeki anahtar kelimeler ve özetler, belirtilen klasör yollarından (inspec/keys ve inspec/docsutf8) okundu. os.walk fonksiyonu ile bu klasörlerdeki .key dosyaları (anahtar kelimeler) ve ilgili .txt dosyaları (özetler) tarandı.

Anahtar Kelimeler ve Özetlerin Okunması Her bir .key dosyasında bulunan anahtar kelimeler ve her bir .txt dosyasında bulunan özetler şu şekilde işlendi:

.key dosyaları, dosya adı ve yolunu birleştirerek tam dosya yolu oluşturuldu ve bu dosyalar açılarak içerikleri okundu. Anahtar kelimeler satır satır okunarak bir listeye alındı ve ardından birleştirildi. Makale adı, dosya adının uzantısız hali (örneğin, ”1234.key” dosyasının adı ”1234” olarak alındı) olarak belirlendi. .txt dosyaları, aynı şekilde tam dosya yolu oluşturularak açıldı ve içerikleri okundu. Verilerin Birleştirilmesi ve Sözlükte Depolanması Her bir makale için, anahtar kelimeler ve özetler birleştirilerek bir sözlükte depolandı. Bu sözlükte, makale adı anahtar olarak kullanıldı ve değer olarak anahtar kelimeler ve özet bilgileri tutuldu. Örneğin, makaleeSozluguu[makaleeAdii] = ”keys”: anahtarrKelimelerr, ”abstract”: abstract şeklinde bir yapı oluşturuldu.

Verilerin MongoDB'ye Kaydedilmesi Oluşturulan veriler, MongoDB koleksiyonuna kaydedildi. Her bir makale için, insertone yöntemi kullanılarak veriler MongoDB'ye tek tek eklendi. Bu adım, verilerin güvenli ve kalıcı bir şekilde saklanmasını sağladı.

Bu yöntem sayesinde, Inspec veri setindeki makalelerin anahtar kelimeleri ve özetleri etkili bir şekilde işlenmiş ve veri tabanında depolanmıştır. Bu işlemler, ileri düzey metin analizi ve bilgi çıkarımı için sağlam bir temel oluşturmuştur.

frekanslar.py : Inspec veri setinde bulunan anahtar kelimelerin frekanslarını hesaplayıp en yüksek değere sahip 300 anahtar kelimeyi belirleyerek bu bilgileri MongoDB veritabanında ”frekanslar” koleksiyonuna kaydettik. Ayrıca, bu kelimelerin fastText ve SciBERT vektörlerini hesaplayarak sırasıyla ”aramafasttext” ve ”aramascibert” koleksiyonlarına ekledik. İşlemler aşağıda detaylı olarak açıklanmaktadır.

Veri Tabanına Bağlantı İlk adım olarak, pymongo kütüphanesi kullanılarak MongoDB veritabanına bağlantı kuruldu. MongoClient sınıfı aracılığıyla yerel bir MongoDB sunucusuna bağlanıldı ve yazlab isimli veritabanı ile frekanslar, aramafasttext ve aramascibert isimli koleksiyonlar seçildi. Bu adım, verilerin güvenli ve kalıcı bir şekilde saklanabilmesi için gereklidir.

Doğal Dil İşleme Araçlarının Hazırlanması Doğal dil işleme (NLP) görevlerini gerçekleştirebilmek için nltk kütüphanesi kullanıldı. Bu kapsamda, İngilizce stopwords'leri (stopwords.words('english')) ve kelime tokenizasyonu (wordtokenize) için gerekli olan veri setleri indirildi. Stopwords'ler, metinlerde sıkça geçen ancak anlamsız olan kelimelerin (örn. ”ve”, ”veya”) çıkarılması için kullanıldı.

Veri Setinin Hazırlanması ve Ön İşleme Anahtar kelimelerin bulunduğu klasör (Inspec/keys) içerisindeki tüm .key dosyaları tarandı. Her bir .key dosyası için şu işlemler gerçekleştirildi:

Dosya içeriği okundu ve satır satır listeye alındı. Bu liste birleştirilerek tek bir metin haline getirildi ve küçük harflere dönüştürüldü (lower()). Noktalama işaretleri kaldırıldı (translate(str.maketrans(”, ”, string.punctuation))). Metin, kelime bazında tokenlara ayrıldı (wordtokenize) ve stopwords'ler çıkarıldı. Anahtar Kelimelerin Frekanslarının Hesaplanması Ön işleme tabi tutulan anahtar kelimeler arasından her bir kelimenin frekansı hesaplandı. Bu frekanslar, bir sözlük (kelimefrekanslarisozlugu) içerisinde tutuldu. Her kelimenin frekansı, o kelimenin kaç kez geçtiğini belirterek sözlükte saklandı.

En Yüksek Frekansa Sahip 300 Kelimenin Belirlenmesi Hesaplanan kelime frekansları, frekans değerlerine göre azalan sırayla sıralandı. En yüksek frekansa sahip olan ilk 300 kelime belirlendi ve bu kelimeler ile frekans değerleri konsola yazdırıldı.

Verilerin MongoDB'ye Kaydedilmesi En yüksek frekansa sahip 300 kelimenin frekans değerleri, MongoDB'deki frekanslar koleksiyonuna kaydedildi. Her bir kelime ve frekans değeri, insertone yöntemi kullanılarak veritabanına tek tek eklendi.

fastText ve SciBERT Modellerinin Kullanılması Anahtar kelimelerin vektör temsillerini elde etmek için iki farklı model kullanıldı:

fastText: Kelimelerin fastText vektörleri hesaplandı ve bu vektörler aramafasttext koleksiyonuna kaydedildi. SciBERT: Kelimelerin SciBERT vektörleri hesaplandı ve bu vektörler aramascibert koleksiyonuna kaydedildi. Her iki model de ilgili kütüphaneler (fasttext ve sentencetransformers) kullanılarak yüklendi ve vektör hesaplamaları gerçekleştirildi.

Bu yöntem sayesinde, Inspec veri setindeki anahtar kelimelerin frekansları etkili bir şekilde hesaplanmış, en yüksek değere sahip 300 kelime belirlenmiş ve bu kelimelerin vektör temsilleri oluşturularak veri tabanında saklanmıştır. Bu işlemler, ileri düzey metin analizi ve bilgi çıkarımı için sağlam bir temel oluşturmuştur.

ilgiVektor.py : ilgialani koleksiyonundaki ilgi alanlarını alarak her bir ilgi alanının ortalama vektör temsillerini hesapladım ve bu temsilleri ilgivektor koleksiyonuna kaydettim. İşlemler aşağıda detaylı olarak açıklanmaktadır.

Scibert Modeli ve Tokenizer'ın Yüklenmesi Öncelikle, ilgi alanlarının vektör temsillerini elde etmek için scibert modeli ve tokenizer yüklendi. transformers kütüphanesinden AutoTokenizer ve AutoModel sınıfları kullanılarak allenai/scibertscivocabuncased modeli indirildi ve kullanıma hazır hale getirildi. Bu model, bilimsel metinlerin işlenmesi için optimize edilmiştir ve her bir ilgi alanı teriminin yüksek kaliteli vektör temsillerini elde etmek için kullanılmıştır.

MongoDB Bağlantısı ve Verilerin Alınması MongoDB veritabanına bağlanmak için pymongo kütüphanesi kullanıldı. Yerel bir MongoDB sunucusuna MongoClient aracılığıyla bağlanıldı ve yazlab isimli veritabanı seçildi. Bu veritabanındaki ilgialani koleksiyonundan tüm ilgi alanları çekildi. Veri sorgulama işlemi sırasında sadece ilgialani alanı alındı ve id alanı hariç tutuldu.

İlgi Alanlarının Vektör Temsillerinin Hesaplanması Çekilen her ilgi alanı terimi için şu işlemler gerçekleştirildi:

Tokenizasyon: İlgi alanı terimleri tokenizer kullanılarak tokenlere dönüştürüldü. Bu işlem, metinlerin modelin anlayabileceği bir formata getirilmesini sağlar. Model Çıktıları: Tokenize edilmiş metinler scibert modeline verildi ve modelden çıktılar alındı. Model çıktıları, ilgi alanı terimlerinin vektör temsillerini içerir. Ortalama Vektör Hesaplama: Modelden elde edilen son gizli durumların ortalaması alındı (outputs.lasthiddenstate.mean(dim=1)) ve bu vektör temsilleri bir listeye eklendi. Bu ortalama, her bir ilgi alanı teriminin 768 boyutlu vektör temsili olarak kullanıldı. Vektörlerin Ortalama Temsillerinin Alınması ve Kaydedilmesi İlgili ilgi alanlarının tüm vektör temsilleri hesaplandıktan sonra, bu vektörlerin ortalaması alındı. Eğer herhangi bir ilgi alanı için vektör temsilleri hesaplanmışsa, bu vektörlerin ortalaması torch.tensor(vektorrlerr).mean(dim=0).tolist() kullanılarak hesaplandı ve liste formatına dönüştürüldü.

Bu ortalama vektör temsilleri, ilgi alanları ile birlikte MongoDB'deki ilgivektor koleksiyonuna kaydedildi. Her bir kayıt, ilgi alanı terimlerini ve bunların ortalama vektör temsillerini içermektedir.

Sonuçların Konsola Yazdırılması İşlem tamamlandığında, her bir ilgi alanı terimi ve buna karşılık gelen ortalama vektör temsili konsola yazdırıldı. Bu, işlemin başarılı bir şekilde tamamlandığını ve verilerin doğru bir şekilde işlendiğini doğrulamak için yapıldı.

Bu yöntem sayesinde, ilgialani koleksiyonundaki ilgi alanları terimlerinin yüksek kaliteli vektör temsilleri elde edilmiş ve bu temsiller ilgivektor koleksiyonuna kaydedilmiştir. Bu işlemler, bilimsel metinlerin analizinde ve ilgi alanları arasında benzerliklerin bulunmasında kullanılabilecek değerli bir veri kaynağı oluşturmuştur.

ilgiVektorFast.py :

ilgialani koleksiyonundaki ilgi alanlarını alarak her bir ilgi alanının ortalama vektör temsillerini hesapladım ve bu temsilleri ilgivektorfast koleksiyonuna kaydettim. İşlemler aşağıda detaylı olarak açıklanmaktadır.

FastText Modelinin Yüklenmesi Öncelikle, ilgi alanlarının vektör temsillerini elde etmek için fastText modeli yüklendi. Bu model, büyük bir metin korpusunda eğitilmiş olup, kelimelerin ve cümlelerin vektör temsillerini elde etmek için kullanılır. Model, "cc.en.300.bin" dosyasından yüklendi ve 300 boyutlu vektörler sağlar. Bu model, her bir ilgi alanı teriminin yüksek kaliteli vektör temsillerini elde etmek için kullanılmıştır.

MongoDB Bağlantısı ve Verilerin Alınması MongoDB veritabanına bağlanmak için pymongo kütüphanesi kullanıldı. Yerel bir MongoDB sunucusuna MongoClient aracılığıyla bağlanıldı ve yazlab isimli veritabanı seçildi. Bu veritabanındaki ilgialani koleksiyonundan tüm ilgi alanları çekildi. Veri sorgulama işlemi sırasında sadece ilgialani alanı alındı ve id alanı hariç tutuldu.

İlgi Alanlarının Vektör Temsillerinin Hesaplanması Çekilen her ilgi alanı terimi için şu işlemler gerçekleştirildi:

Vektör Temsillerinin Elde Edilmesi: İlgi alanı terimleri için fastText modelinin getsentencevector fonksiyonu kullanılarak vektör temsilleri elde edildi. Her ilgi alanı terimi için 300 boyutlu vektörler oluşturuldu. Vektörlerin Listeye Eklenmesi: Elde edilen vektörler, daha sonra ortalamalarının alınabilmesi için bir listeye eklendi. Vektörlerin Ortalama Temsillerinin Alınması ve Kaydedilmesi İlgili ilgi alanlarının tüm vektör temsilleri hesaplandıktan sonra, bu vektörlerin ortalaması alındı. Eğer herhangi bir ilgi alanı için vektör temsilleri hesaplanmışsa, bu vektörlerin ortalaması np.mean(vektorrlerr, axis=0).tolist() kullanılarak hesaplandı ve liste formatına dönüştürüldü.

Bu ortalama vektör temsilleri, ilgi alanları ile birlikte MongoDB'deki ilgivektorfast koleksiyonuna kaydedildi. Her bir kayıt, ilgi alanı terimlerini ve bunların ortalama vektör temsillerini içermektedir.

Sonuçların Konsola Yazdırılması İşlem tamamlandığında, her bir ilgi alanı terimi ve buna karşılık gelen ortalama vektör temsili konsola yazdırıldı. Bu, işlemin başarılı bir şekilde tamamlandığını ve verilerin doğru bir şekilde işlendiğini doğrulamak için yapıldı.

Bu yöntem sayesinde, ilgialani koleksiyonundaki ilgi alanları terimlerinin yüksek kaliteli vektör temsilleri elde edilmiş

ve bu temsiller ilgivektorfast koleksiyonuna kaydedilmiştir. Bu işlemler, bilimsel metinlerin analizinde ve ilgi alanları arasında benzerliklerin bulunmasında kullanılabilecek değerli bir veri kaynağı oluşturmuştur.

tekListe.py : Inspec veri setinden anahtar kelimeleri ve metin dosyalarını işleyerek ön işleme tabii tuttum ve bu kelimeleri "tekdosya" koleksiyonuna kaydettim. İşlemler aşağıda detaylı olarak açıklanmaktadır.

Gerekli Kütüphanelerin Yüklenmesi ve İndirilmesi Öncelikle, gerekli kütüphaneler yüklendi ve İngilizce stopwords ve tokenizasyon için nltk kütüphanesi kullanıldı.

MongoDB Bağlantısı ve Verilerin Hazırlanması MongoDB veritabanına bağlanmak için pymongo kütüphanesi kullanıldı. Yerel bir MongoDB sunucusuna MongoClient aracılığıyla bağlanıldı ve yazlab isimli veritabanı seçildi. Bu veritabanındaki tekdosya koleksiyonu kullanılmak üzere ayarlandı.

Kelime İşleme Fonksiyonları Veri işleme sırasında İngilizce stopwords listesi ve noktalama işaretleri kaldırıldı, kelimeler küçük harfe dönüştürüldü ve kelime köklerini bulmak için PorterStemmer kullanıldı.

Anahtar Kelimelerin ve Metin Dosyalarının İşlenmesi Anahtar kelimelerin bulunduğu dosyalar ve metin dosyaları ayrı ayrı işlendi:

Anahtar Kelimeler:

.key uzantılı dosyalar okundu ve kelimeler küçük harfe dönüştürüldü. Noktalama işaretleri kaldırıldı ve stopwords listesi çıkarıldı. Kelimeler köklerine indirgenerek bir listeye eklendi. Metin Dosyaları:

.txt uzantılı dosyalar okundu ve kelimeler küçük harfe dönüştürüldü. Noktalama işaretleri kaldırıldı ve stopwords listesi çıkarıldı. Kelimeler köklerine indirgenerek bir listeye eklendi. Tekrarlanan Kelimelerin Kaldırılması Tüm kelimeler tek bir listeye eklenerek, tekrarlanan kelimeler set kullanılarak kaldırıldı ve bu işlemden sonra listeye dönüştürüldü.

Kelimelerin MongoDB'ye Kaydedilmesi Elde edilen kelimeler tekdosya koleksiyonuna tek tek kaydedildi ve işlem tamamlandığında konsola bir bilgi mesajı yazdırıldı.

vektorFasttext.py : Bu belgede, "vektorfasttext" koleksiyonuna işlenmiş kelimelerin vektörlerini kaydetmek için çalışır. İşlemler adım adım şu şekilde gerçekleştirilir:

Bağlantı ve Koleksiyon Seçimi İlk olarak, MongoDB veritabanına bağlanılır ve gerekli koleksiyonlar seçilir: "kelimeler" koleksiyonu, "inspec" koleksiyonu ve vektörlerin kaydedileceği "vektorfasttext" koleksiyonu.

FastText Modelinin Yüklenmesi FastText modeli yüklenir. Bu model, önceden eğitilmiş bir dil modelidir ve kelime vektörlerini döndürebilir.

Belgelerin İşlenmesi ve Vektörlerin Oluşturulması Tüm belgeler için aşağıdaki işlemler gerçekleştirilir:

Belgeler MongoDB'den alınır. Belgelerin başlıkları ve içerikleri (abstract) alınır. Her bir kelimenin FastText modelinden vektörü alınır. Belgelerin vektörleri, belge içerisindeki tüm kelime vektörlerinin ortalaması alınarak oluşturulur. Oluşturulan belge vektörleri, MongoDB'deki "vektorfasttext" koleksiyonuna kaydedilir. Sonuçların Bildirilmesi İşlem tamamlandığında, konsola "Vektörler başarıyla kaydedildi." mesajı yazdırılır.

Bu kod parçacığı, belgelerin içerdiği kelimelerin FastText modelinden geçirilerek vektörlerinin oluşturulmasını ve bu vektörlerin MongoDB veritabanına kaydedilmesini sağlar. Bu sayede, belgelerin vektörel temsilini elde etmiş oluruz.

vektorScibert.py :

Bu belgede, "vektorscibert" koleksiyonuna belgelerin içerdiği kelimelerin SciBERT modelinden geçirilerek elde edilen vektörlerin kaydedilmesini sağlar. Bilgi işlem aşağıdaki adımlarla gerçekleşir:

Bağlantı ve Koleksiyon Seçimi: İlk olarak, MongoDB veritabanına bağlanılır ve gereken koleksiyonlar belirlenir. "kelimeler" koleksiyonu, "inspec" koleksiyonu ve nihayetinde vektörlerin saklanacağı "vektorscibert" koleksiyonu seçilir.

SciBERT Modelinin Yüklenmesi: SciBERT, bilimsel metinler için özel olarak eğitilmiş bir dil modelidir. Bu kod parçasında, önceden eğitilmiş bir SciBERT modeli yüklenir.

Belgelerin İşlenmesi ve Vektörlerin Oluşturulması: Her bir belge için aşağıdaki işlemler gerçekleştirilir:

Belgeler, MongoDB'den alınır. Belgelerin başlıkları ve içerikleri (abstract) alınır. Her bir belgenin içerdiği kelimeler, SciBERT modelinden geçirilerek kelime vektörleri elde edilir. Belgelerin vektörleri, içerdikleri kelimelerin vektörlerinin ortalaması alınarak oluşturulur. Oluşturulan belge vektörleri, MongoDB'deki "vektorscibert" koleksiyonuna kaydedilir. Sonuçların Bildirilmesi: İşlem tamamlandığında, konsola "Vektörler başarıyla kaydedildi." mesajı yazdırılır.

Bu kod parçası, bilimsel metinlerin içeriğini temsil eden vektörlerin elde edilmesi için SciBERT modelinin kullanımını içerir.

benzerlik.py :

Gerekli Kütüphanelerin İçe Aktarılması ve Flask Uygulamasının Oluşturulması:

Flask web uygulamasını oluşturmak ve çalıştırmak için Flask kütüphanesi içe aktarılır. Veritabanı işlemleri için pymongo kütüphanesi kullanılır. Bilimsel hesaplamalar için numpy ve sklearn kütüphaneleri kullanılır. 2. En Yakın Belgelerin Bulunması İçin Yardımcı Fonksiyonların Tanımlanması:

enn-yakiinn-bess-makalee-scibert ve enn-yakiinn-bess-makalee-fasttext fonksiyonları, belirli bir sorgu vektörüne en yakın belgelerin bulunmasını sağlar. Bu fonksiyonlar, belgeler arasındaki benzerlik skorunu hesaplar. Benzerlik skorlarına göre belgeler sıralanır ve en yakın belgelerin adları, başlıkları ve benzerlik skorları listelenir. 3. SciBERT ve FastText İçin Veri Alma ve İşleme Fonksiyonları:

/scibertt-dataa ve /fasttextt-dataa fonksiyonları, ilgili koleksiyonlardan veri alır ve sorgu vektörüyle en yakın belgeleri bulmak için yardımcı fonksiyonları çağırır. Bu fonksiyonlar, belgelerin vektörlerini ve benzerlik skorlarını JSON formatında döndürür. 4. Makale Detaylarını Alma ve Güncelleme Fonksiyonları:

/makalee-detaylarii fonksiyonu, belirli bir makalenin detaylarını ve anahtar kelimelerini alır. Ayrıca, SciBERT ve FastText

vektörlerini günceller. Bu fonksiyon, veritabanından makale bilgilerini çeker, vektörleri günceller ve HTML şablonunu kullanarak makale detaylarını görüntüler. 5. Geri Bildirim ve Doğruluk Hesaplama Fonksiyonları:

/guncellee fonksiyonu, kullanıcı geri bildirimlerini alır ve SciBERT ile FastText modellerinin doğruluğunu hesaplar. Bu fonksiyon, geri bildirimleri kullanarak vektör ortalamalarını günceller ve doğruluk değerlerini hesaplar. 6. Arama ve Vektör Güncelleme Fonksiyonları:

/search fonksiyonu, kullanıcı tarafından girilen bir anahtar kelimeyi alır, bu anahtar kelimeyi arar ve ilgili vektörleri veritabanına ekler veya günceller. Bu fonksiyon, arama işlemlerini gerçekleştirir ve vektörleri günceller. 7. Ana Sayfa Görünümü:

/ yolunda bulunan index fonksiyonu, kullanıcıya ana sayfayı gösterir. Bu sayfa, kullanıcıya web uygulamasının temel görünümünü sunar. Bu Flask uygulaması, belirli bir bilimsel makalenin içeriğini temsil eden vektörlerin elde edilmesini, kullanıcı geri bildirimlerinin alınmasını ve vektörlerin güncellenmesini sağlar. Ayrıca, belirli bir anahtar kelimenin aranması ve ilgili vektörlerin veritabanına eklenmesi veya güncellenmesi için bir arama işlevselliği sunar.

benzerlik.html : bu belgede yapılan işlemlerden bahsedeyim: Sayfa Yapısı ve Stilleri:

HTML'de ¡head¿ bölümünde sayfanın başlık bilgisi, stil bağlantıları ve ¡style¿ etiketi içindeki özel CSS tanımları yer alır. Bootstrap ve Font Awesome gibi harici kütüphanelerden stiller çağrılır. ¡body¿ bölümü, sayfanın içeriğini ve kullanıcı arayüzünü oluşturur. Ana Sayfa ve Makale Arama:

Kullanıcıya makale arama imkanı sunan bir form vardır. Anahtar kelime girilerek bir POST isteği gönderilir. Bu istek, sunucuya makale arama isteğini iletecek olan /aramaa yoluna gönderilir. Makale arama sonuçları, sonuçları gösterecek bir bölüm içinde dinamik olarak güncellenir. SciBERT ve FastText Model Sonuçları:

SciBERT ve FastText modellerinin önerdiği makaleler tablolar halinde sunulur. Her makale için bir dizi işlem butonu (Uygun, Uygun Değil, Detaylar) eklenir. Bu butonlar, kullanıcıların her makale için bir karar vermelerini sağlar. Her buton, kullanıcının seçimine bağlı olarak sunucuya bir POST isteği gönderir. Bu istek, makale hakkında kullanıcının kararını (Uygun, Uygun Değil) sunucuya iletir. Yenileme ve Precision Bilgisi:

Sayfanın altında, sonuçları yenilemek ve modelin doğruluk bilgisini göstermek için bir kart bulunur. Yenileme butonu, sayfayı yenilemek için kullanılır. Precision bilgileri, sunucudan gelen verilere dayanarak güncellenir. JavaScript Kodları:

JavaScript, sayfanın dinamik davranışını sağlar. Örneğin, makale arama formunun gönderilmesi, makalelerin tablolara eklenmesi, karar butonlarının işlevselliği gibi. Bu kodlar, kullanıcı etkileşimlerini ele alır ve sunucu ile iletişim kurar. Yönlendirme Kontrolü:

Son olarak, JavaScript ile bir kontrol yapılır. Eğer sayfa "http://127.0.0.1:5000/" adresine değilse, kullanıcı otomatik olarak bu adrese yönlendirilir. Bu kontrol, sayfanın sadece belirli bir adreste çalışmasını sağlar. Bu HTML belgesi, bir web uygulamasının ön uç (front-end) kodunu oluşturur ve kullanıcının etkileşimde bulunabileceği bir arayüz sunar. Sunucu tarafındaki işlevselliği ise belirli bir sunucu platformu ve bu platformdaki uygulamanın kodu sağlar.

ilgiAlaniKaydet.py : Bu kod, bir Flask web uygulaması aracılığıyla HTML formundan alınan ilgi alanı bilgilerini MongoDB veritabanına kaydetmek için kullanılır. Öncelikle, Flask uygulaması ve MongoDB istemcisi oluşturulur ve yazlab adlı veritabanındaki ilgialani koleksiyonuna erişim sağlanır. /sec-ilgi-alani endpoint'i POST isteklerini dinler. Bu endpoint'e bir POST isteği geldiğinde, HTML formundan alınan ilgi alanları request.form.getlist('ilgi-alani') ile alınır. Bu ilgi alanları bir belge olarak MongoDB'deki ilgialani koleksiyonuna insert-one metodu kullanılarak eklenir. Son olarak, işlem başarılı olduğunda bir JSON yanıtı döndürülür, bu yanıt ilgi alanlarının başarıyla kaydedildiğini ve kaydedilen ilgi alanlarını içerir. Flask uygulaması, debug modunda çalıştırılarak yerel sunucuda çalıştırılır.

III. SONUÇLAR

Bu proje kapsamında, kullanıcıların ilgi alanlarına ve okuma geçmişlerine göre kişiselleştirilmiş makale önerileri sunan bir sistem geliştirilmiştir. İlk olarak, Inspec hazır makale veri kümesi kullanılarak veri kaynağı oluşturulmuştur. Makaleler, doğal dil işleme (NLP) teknikleri kullanılarak ön işleme tabi tutulmuş ve metinler üzerinde stopwords temizlenmesi, noktalama işaretlerinin çıkarılması ve kelime köklerinin bulunması gibi işlemler gerçekleştirilmiştir. Bu süreçte Python dilinde NLTK kütüphanesi kullanılmıştır.

Makalelerin ve kullanıcı profillerinin vektör temsillerini oluşturmak için FastText ve SCIBERT modelleri kullanılmıştır. Kullanıcı profilleri, ilgi alanlarının vektör temsillerinin ortalaması alınarak oluşturulmuş ve makale-kullanıcı benzerliği Cosine Similarity metriği ile hesaplanmıştır. Bu sayede, her iki model için ayrı ayrı 5 makale önerisi sunulmuştur.

Sistemin performansı, Precision değeri hesaplanarak değerlendirilmiş ve sonuçlar kullanıcı arayüzünde gösterilmiştir. Kullanıcılar için üyelik ve profil oluşturma işlemleri sağlanmış, demografik bilgiler ve akademik ilgi alanları toplanarak profiller oluşturulmuştur. Bu profiller, kullanıcıların ilgi alanları ve okuma geçmişlerine göre dinamik olarak güncellenmiş ve öneriler sunulmuştur.

Kullanıcı geri bildirimleri toplanarak, öneri motoru bu geri bildirimler doğrultusunda sürekli olarak güncellenmiş ve kişiselleştirilmiş öneriler daha da iyileştirilmiştir. Kullanıcı dostu bir web arayüzü geliştirilmiş, kullanıcıların aradıkları makaleleri kolayca bulabilmeleri için arama çubuğu ve filtreleme seçenekleri eklenmiştir. Verilerin senkronizasyonu ve algoritmaların çalışabilmesi için MongoDB veri tabanı kullanılmıştır ve yapay zeka ile makine öğrenmesi işlemleri Python ile gerçekleştirilmiştir.

Sonuç olarak, geliştirdiğimiz sistem, kullanıcıların akademik ilgi alanlarına uygun makaleleri hızlı ve etkili bir şekilde bulmalarını sağlamış, kullanıcı deneyimini ve memnuniyetini artırmıştır. Bu projenin çıktıları, akademik araştırmaların daha verimli bir şekilde yürütülmesine katkı

sağlamış ve gelecekteki çalışmalar için sağlam bir temel oluşturmuştur.

REFERENCES

[1] https://github.com/LIAAD/KeywordExtractor-Datasets
[2] https://github.com/allenai/scibert
[3] https://github.com/facebookresearch/fastText
[4] https://www.kaggle.com/datasets/sanyatargrenkin/cc-en-300-bin?resource=download
[5] https://medium.com/machine-learning-t
[6] https://www.geeksforgeeks.org/how-to-calculate-cosine-similarity-in-python/

IV. ER DİYAGRAMI VE AKIŞ ŞEMALARI

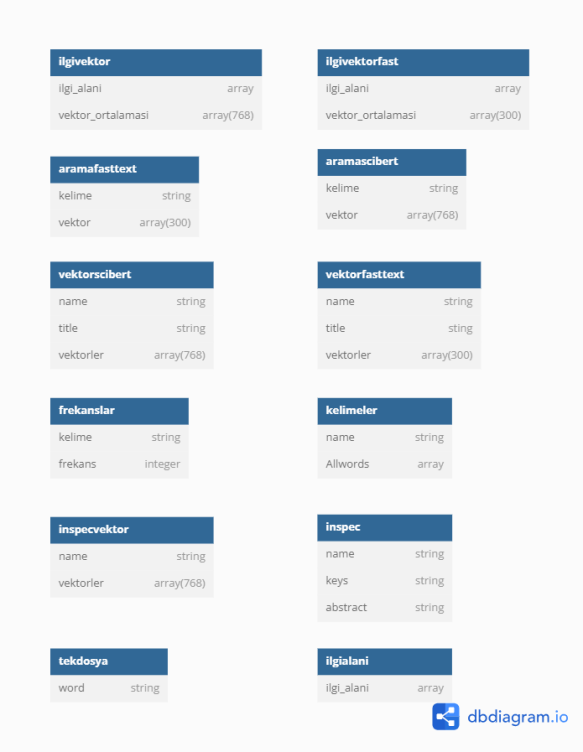

Fig. 1: ERD

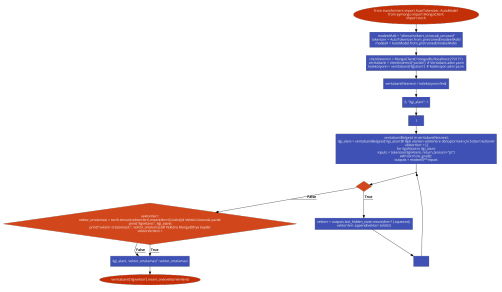

Fig. 2: ilgivektor

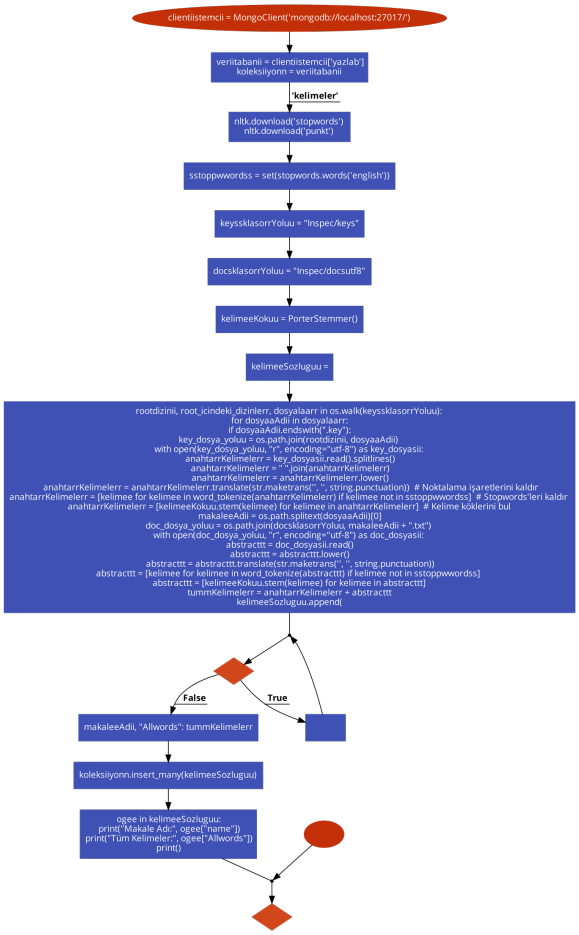

Fig. 3: kelimeler

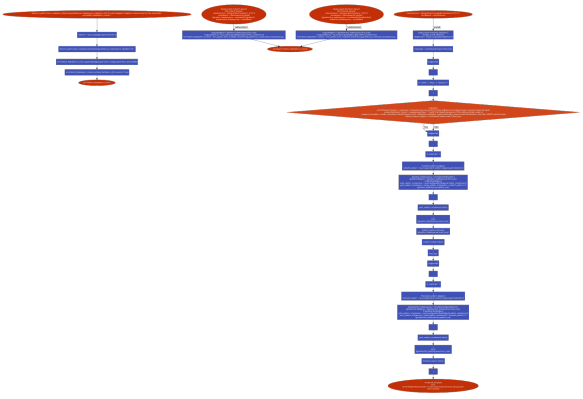

Fig. 4: benzerlik

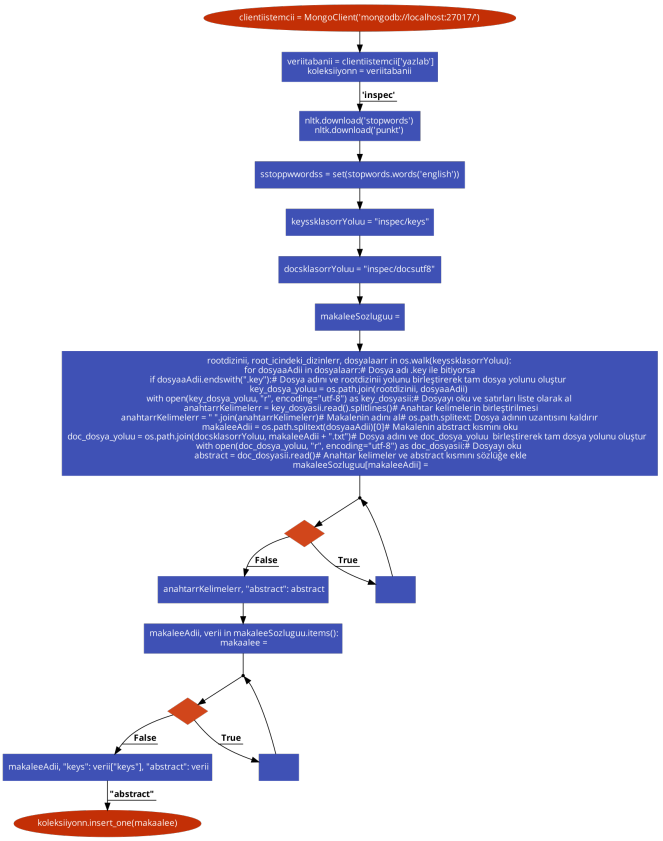

Fig. 5: inspec

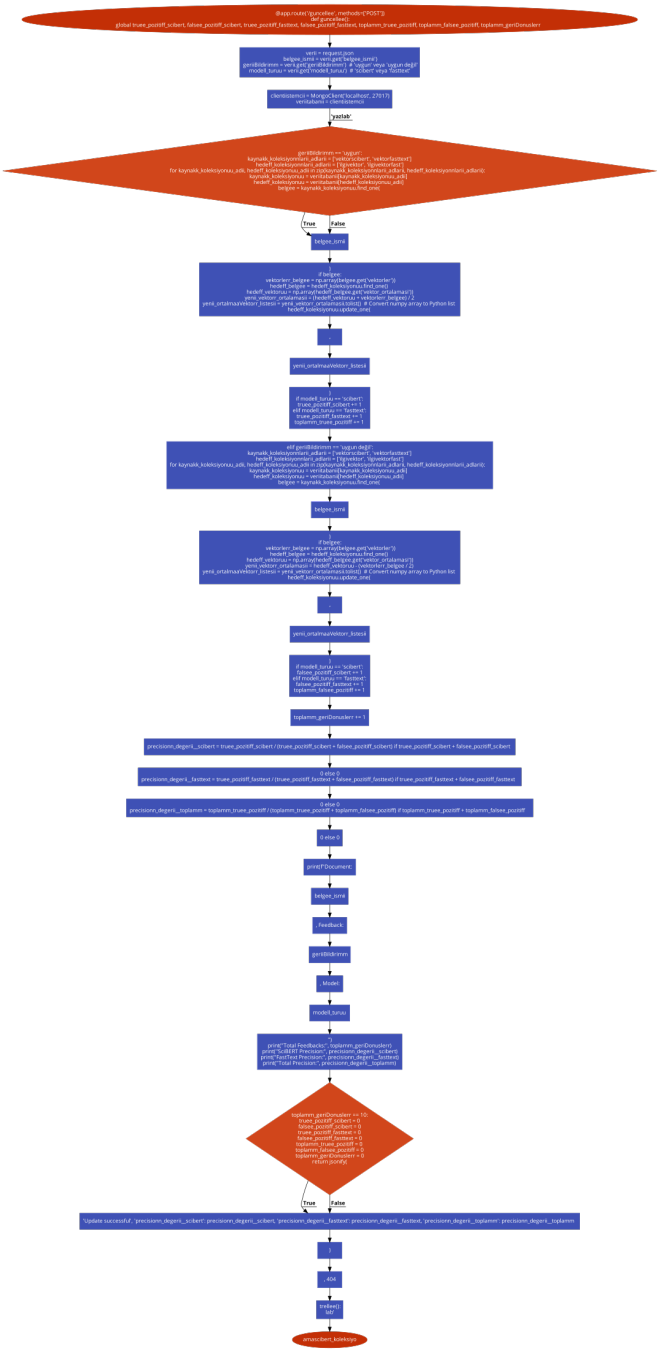

Fig. 6: güncelle

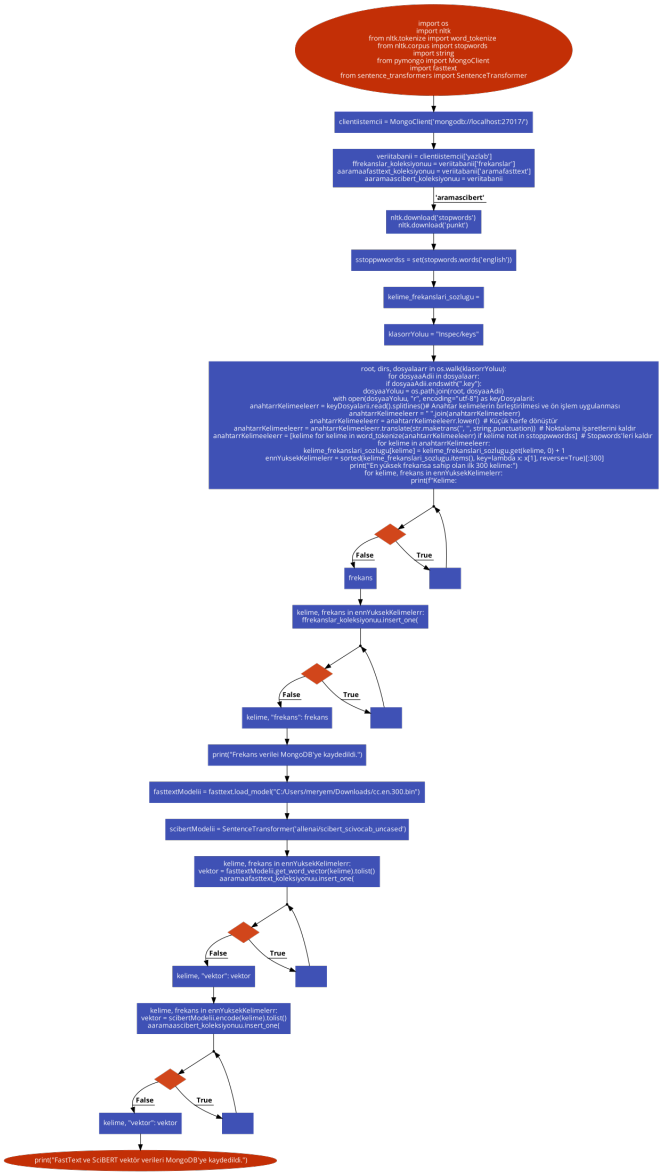

Fig. 7: frekanslar

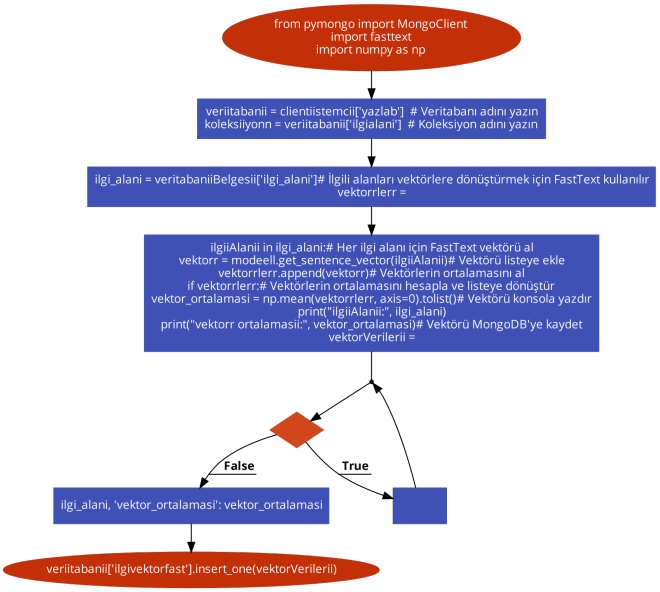

Fig. 8: ilgivektorfast

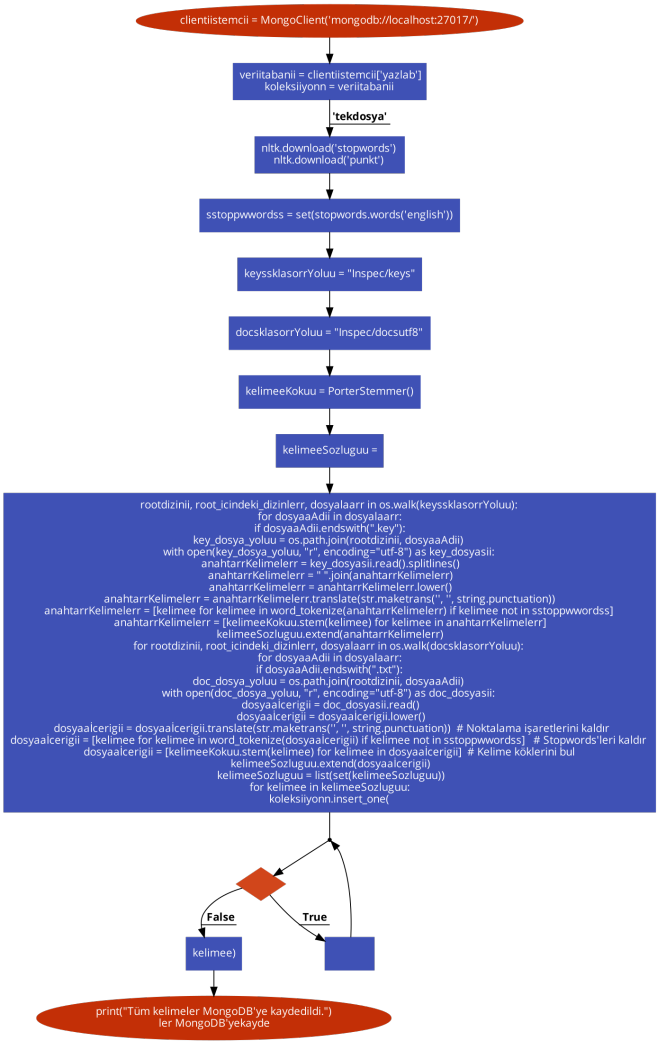

Fig. 9: tekListe

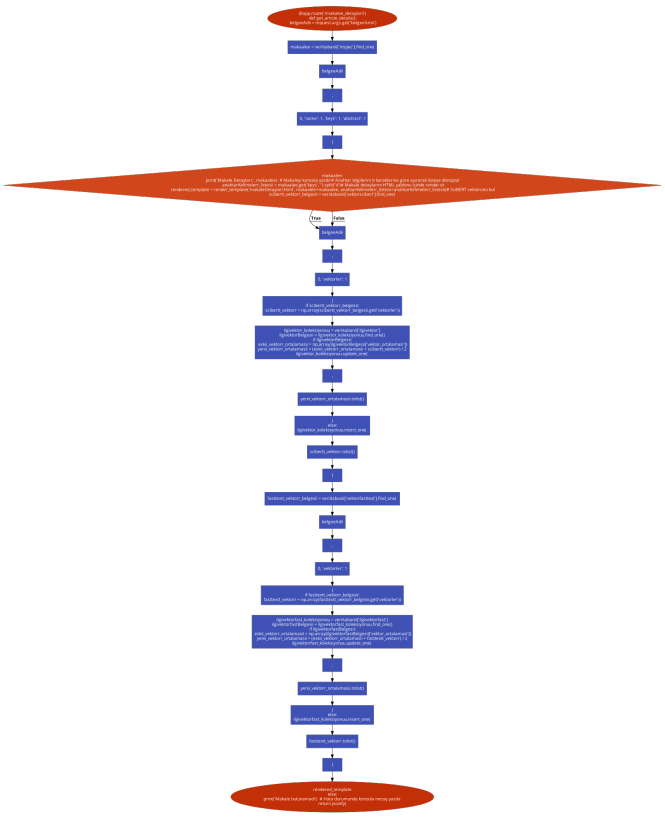

Fig. 10: Makale Detayları

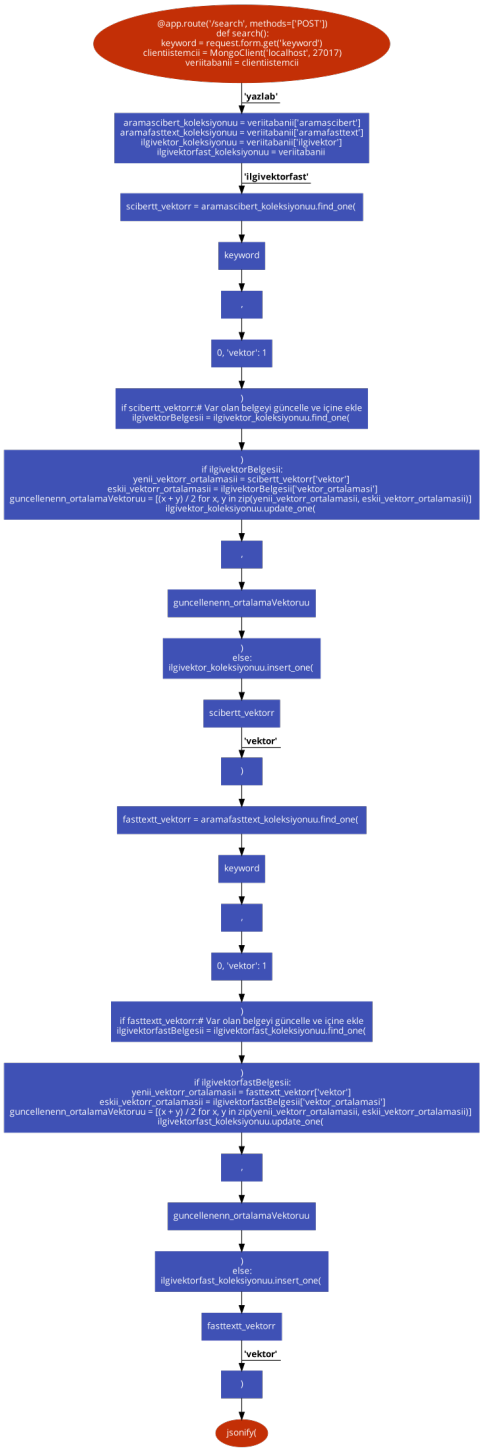

Fig. 12: arama işlemi

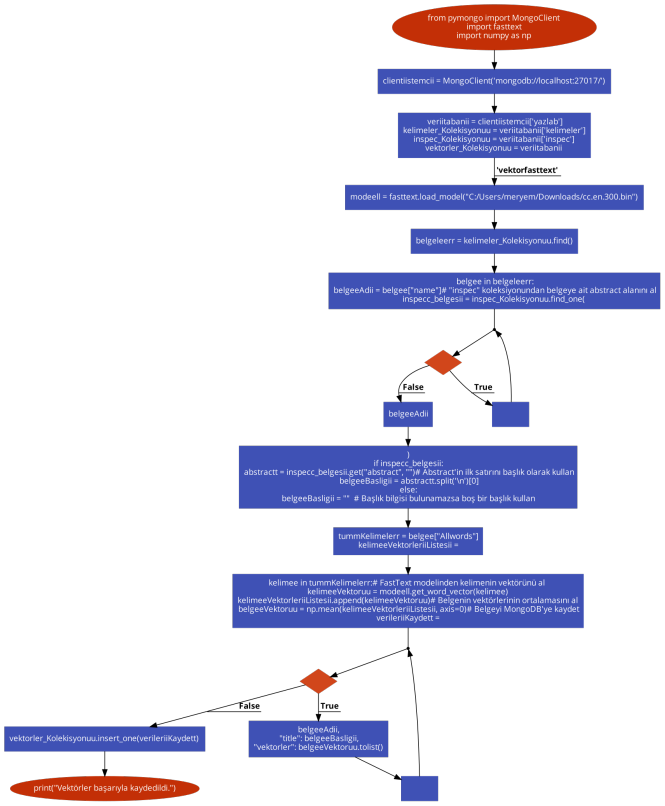

Fig. 11: vektorfasttext

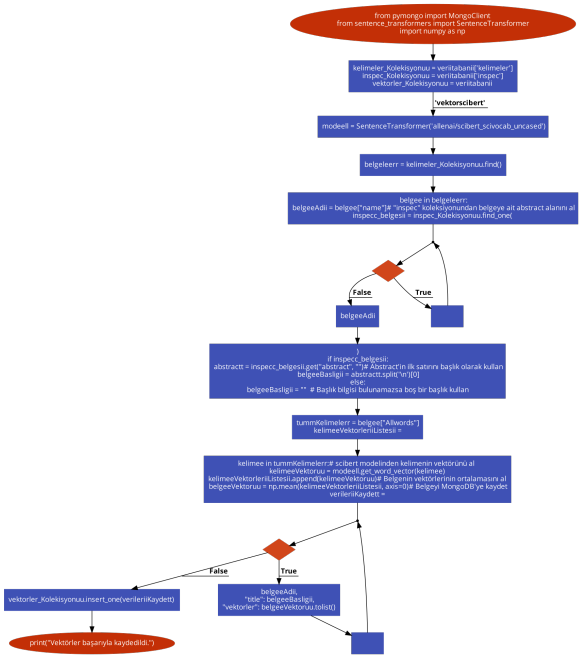

Fig. 13: vektorscibert

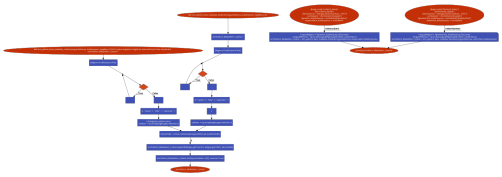

Fig. 14: vektorler