Yazılım Laboratuvarı II
Proje I
Bilgisayar Mühendisliği Bölümü
Kocaeli Üniversitesi

Furkan Kasap
210201107

I. ÖZET

Bu rapor Yazılım Laboratuvarı 2 dersinin 1. projesi için oluşturulmuştur. Bu proje Java, HTML, CSS, JavaScript programlama dilleri ile geliştirilmiştir. Algoritmayı geliştirme ortamı olarak IntelliJ IDEA ve web arayüzünü geliştirme ortamı için ise Visual Studio Code kullanılmıştır.

Belgede akış diyagramı, özet, giriş, yöntem, deneysel sonuçlar gibi projeyi açıklayan başlıklara yer verilmiştir. Belge sonunda projenin sonucu ve projeyi hazırlarken kullanılan kaynaklar bulunmaktadır.

II. GİRİŞ

Bu raporda, iki ya da daha fazla metnin birleştirilmesi problemi için yeni algoritmalar geliştirilmesi konusu ele alınmaktadır. Projenin amacı, bu problemin çözümü için daha iyi ve etkili yöntemler tasarlamak ve bu algoritmaları bir görsel arayüz ile sunmaktır. Geliştirilen algoritmaların performans özellikleri ve çalışma süreleri farklı girdilerde karşılaştırılarak değerlendirilecektir.

Metin birleştirme işlemleri için JAVA dili, veri tabanı olarak MongoDB (NoSQL), Web arayüzü için ise HTML,CSS,JS programlama dilleri kullanılmıştır.

III. YÖNTEM

A. Algoritma

mergeTexts fonksiyonu, cümleleri birleştirmek için ana fonksiyondur. Parametre olarak bir List alır ve String olarak birleştirilmiş cümle döndürür. İlk olarak, wordsList adında bir List oluşturulur, her bir elemanı bir metnin kelimelerini içerir. wordSet adında bir Set de oluşturulur ve bu, tüm metinlerin benzersiz kelimelerinin bir koleksiyonunu tutar. freqMaps adında bir başka List oluşturulur ve her bir elemanı, kelimelerin frekanslarının bir Map'ini içerir. wordsListlen adında bir tamsayı tanımlanır ve tüm metinlerin toplam kelime sayısını saklar.

Daha sonra, for döngüsü aracılığıyla her bir metnin kelimeleri alınır ve wordsList koleksiyonuna eklenir. Ayrıca, her bir kelime wordSet koleksiyonuna eklenir ve freqMaps koleksiyonunda kelimenin frekanslarını saklayan bir Map oluşturulur.

Eğer tüm metinlerin benzersiz kelimeleri toplam kelime sayısına eşitse, yani hiçbir kelime tekrar etmiyorsa, mergeTexts fonksiyonu birleştirilemeyeceğini belirtmek için "Bu cümleler birleştirilemez" şeklinde bir dize döndürür.

Eğer cümleler birleştirilebiliyorsa, ortak kelimelerin bir listesini elde etmek için getCommonWords fonksiyonu çağrılır. Bu fonksiyon, wordSet koleksiyonundaki her kelime için, her bir metinde kelimenin frekansını freqMaps koleksiyonundan alır ve bu frekansları kullanarak kozinüs benzerliğini hesaplar. En yüksek benzerliğe sahip ortak kelime veya kelimeleri seçer ve bu kelimeleri commonWords listesine ekler.

StringBuilder nesnesi kullanılarak, tüm metinlerdeki kelimeler gezilir ve eğer kelime ortak kelimeler listesinde yer alıyorsa, sadece bir kez eklenir. Eğer kelime ortak kelimeler listesinde yoksa, metnin orijinal sırasında StringBuilder nesnesine eklenir. Son olarak, birleştirilmiş cümle toString() metodu kullanılarak döndürülür.

getWordFrequency fonksiyonu, bir kelimenin her bir metinde kaç kez göründüğünü saymak için kullanılan bir Map döndürür. İlk adım olarak boş bir HashMap oluşturulur. Daha sonra, metnin her bir kelimesi için HashMap içinde bir sayacı arttırarak o kelimenin kaç kez görüldüğünü kaydedilir. Böylece, HashMap sonunda, metindeki her bir kelimenin kaç kez görüldüğünü saklanmış olur. getCommonWords fonksiyonu, tüm metinlerde ortak olan kelimeleri bulmak için kullanılır. İlk olarak, tüm metinlerdeki kelime listesi ve frekansları bir liste olarak toplanır. Ardından, tüm kelime listesi üzerinde döngü oluşturulur ve her kelimenin frekansı, tüm metinlerdeki frekansları içeren bir diziye eklenir. Bu, kelimenin tüm metinlerdeki frekanslarını temsil eder. Sonra, bu frekans dizisi üzerinden kosinüs benzerliği hesaplanır. kosinüs benzerliği kelimenin tüm metinlerdeki frekanslarının ne kadar benzer olduğunu ölçer. Benzerlik ne kadar yüksekse, kelimenin tüm metinlerde ortak olduğu o kadar yüksektir.

cosineSimilarity fonksiyonu, iki vektörün benzerliğini ölçmek için kullanılan bir formüldür. Bu algoritmada, metinlerin kelime sıklıklarını içeren vektörlerin benzerliği ölçülür. İki kelime sıklık vektörünün benzerliği, bu vektörlerin nokta çarpımının, vektörlerin normlarının çarpımına bölünmesiyle hesaplanır.

Nokta çarpımı, iki vektörün aynı pozisyondaki eleman-

larının çarpımlarının toplamıdır. Örneğin, iki vektör a ve b olsun. Bu durumda, a'nın i. elemanı ile b'nin i. elemanının çarpımı, a[i] * b[i] olarak hesaplanır. Bu işlem, tüm elemanlar için tekrarlanır ve sonuç olarak tüm çarpımların toplamı bulunur.

Norm, bir vektörün büyüklüğünü ifade eder. Bu algoritmada, normu hesaplamak için sıklık vektörünün elemanlarının kareleri toplanır ve kökü alınır.

Sonuç olarak, cosineSimilarity fonksiyonu, iki vektör arasındaki açının kosinüsüne göre benzerlik skoru hesaplar. Bu skor, 1'e yaklaştıkça vektörlerin benzerliği artar, 0'a yaklaştıkça benzerlik azalır.

Daha sonra ise tüm kelime listesi üzerinde döngü oluşturulur ve her kelimenin benzerliği hesaplanır. En yüksek benzerliğe sahip kelime ortak kelime olarak kabul edilir ve bir liste halinde döndürülür.

mergeTexts fonksiyonu, tüm metinleri birleştirmek için kullanılır. İlk olarak, tüm metinlerdeki kelimeler bir listeye eklenir ve tüm kelime seti oluşturulur. Eğer tüm metinlerdeki kelime sayısı kelime seti boyutuna eşitse, metinler birleştirilemez ve hata mesajı döndürülür.

Aksi takdirde, tüm metinlerdeki ortak kelimeler bulunur ve bir StringBuilder kullanılarak metinler birleştirilir. Her kelime, eğer ortak bir kelime ise, yalnızca bir kez eklenir. Aksi takdirde, her kelime ayrı olarak eklenir. Sonunda, birleştirilmiş metinler döndürülür.

B. Web Arayüz/Frontend

Bu kısımdaki kodlar ise, bir metin birleştirme uygulamasının web arayüzünü içermektedir. HTML, CSS ve JavaScript kullanarak tasarlanmıştır.

Index.html dosyası, kullanıcı arayüzünü ve bileşenlerini (input alanları, butonlar, tablo) tanımlar. Başlığı, birleştirme kutularını, yeni metin ekle butonunu, birleştir butonunu, kaydet butonunu, listele butonunu ve sonuçlar için paragrafları içermektedir.

Style.css, web sayfasında yer alan bazı elementlerin stil özelliklerini belirlemektedir. Örneğin, sayfadaki yazıların fontu Arial ve sans-serif olarak belirlenmiştir. Sayfa içeriği ortalanmıştır. Metin kutuları, dikey yönde bir sütun olarak hizalanmıştır.

JavaScript, sayfadaki dinamik işlevleri yürütmekten sorumludur. Bu kod, metin kutularının dinamik olarak eklenebilmesini sağlayan bir fonksiyon, metin kutularındaki metinlerin birleştirilmesini sağlayan bir fonksiyon ve birleştirilmiş metinlerin kaydedilmesini ve listelenmesini sağlayan iki fonksiyon içerir.

Web sayfasına girilen verilerin java algoritmasına gönderilmesi için ise Axios adlı bir JavaScript kütüphanesi kullanarak HTTP istekleri yapılır. Örneğin, metin kutularındaki metinleri sunucuya göndermek için bir POST isteği yapar ve sunucudan gelen yanıtı işler. Listeleme işlemi, sunucudan metinleri almak için bir GET isteği kullanır.

C. REST API/Backend

Bu REST API, "TextService" adı verilen bir hizmetin kontrolörünü içerir ve metinleri birleştirir ve veritabanına kaydeder.

Controller, "/texts" yolunda bir GET isteği aldığında, TextService'deki tüm metinleri getirir ve bunları bir ResponseEntity olarak döndürür.

Aynı şekilde, "/texts" yolunda bir POST isteği aldığında, gelen Text dizisini alır ve bunları birleştirmek için TextMerger adlı sınıfı kullanır. Birleştirme işlemi, metinlerin listesi üzerinde döngü kullanarak yapılır ve birleştirilmiş metin bir Response nesnesi içinde döndürülür.

Ayrıca, "/texts/save" yolunda bir POST isteği aldığında, kontrolör gelen DBtext nesnesini alır ve veritabanında mevcut olan diğer metinlerle karşılaştırır. Eğer gelen metinler zaten veritabanında varsa, başarısızlık mesajı döndürülür; aksi takdirde, yeni metinler veritabanına kaydedilir.

Bu REST API'da, Spring Boot ve Lombok gibi kütüphaneler kullanılmıştır. Ayrıca, WebConfig adlı bir yapılandırma sınıfını ve TextService adlı bir servisi kullanır. Cross-origin HTTP isteklerine izin vermek için @CrossOrigin(maxAge = 3600) kullanılır.

D. Veritabanı

Veritabanı için yazılan kodlar ise, Java Spring kütüphanesi ile MongoDB veritabanına bağlantı kurmak için yazılmıştır.

TextService sınıfı, metinlerin oluşturulması ve tüm metinlerin getirilmesi işlemlerini gerçekleştirir. Bu sınıfın createText metodunda, eğer metin boş değilse, TextRepository aracılığıyla veritabanına kaydedilir.

TextRepository sınıfı, MongoRepository sınıfından türetilir ve DBtext sınıfı ile ilişkilendirilir. Bu sınıf, MongoRepository tarafından sağlanan yöntemler aracılığıyla DBtext nesnelerinin oluşturulmasını, güncellenmesini, silinmesini ve getirilmesini sağlar.

Bu kodlarda, @Service ve @AllArgsConstructor anotasyonları kullanılarak, Spring Dependency Injection özelliği kullanılarak, TextService sınıfının yapıcı yönteminde TextRepository arayüzü enjekte edilir. Bu, TextService sınıfının textRepository değişkeninin otomatik olarak atanmasını sağlar.

Bağlantı ayarları ise application.properties dosyasında yapılandırılmıştır.

IV. SONUÇ

Bu projede bir metin birleştirme algoritması geliştirilerek algoritma bilgisi pekiştirilmiştir. Basit bir web arayüzü tasarımı öğrenilmiştir. Bu web arayüzü ile java algoritmasının birbiri ile haberleşmesi için Rest API'ler hakkında bilgi edinilip java spring kütüphanesi ile basit bir REST API oluşturulmuştur.

KAYNAKLAR

[1] https://bilgisayarkavramlari.com/2012/11/08/kosinus-benzerligi-cosine-similarity-2/
[2] https://www.veribilimiokulu.com/jaccard-benzerligi-ve-kosinus-benzerligi/
[3] https://spring.io/guides/tutorials/rest/
[4] https://www.baeldung.com/spring-data-mongodb-tutorial
[5] https://medium.com/crony-community/java-spring-restful-servisler-1-565602deeed5
[6] https://hankins.medium.com/how-to-make-ajax-requests-with-axios-e6992ebd0463
[7] https://www.javatpoint.com/spring-boot-rest-example
[8] https://developer.mozilla.org/en-US/docs/Learn

1. Tüm metinlerin kelimeleri alınır ve tekrar eden kelimeler çıkarılarak bir set oluşturulur.
2. Her bir metindeki kelimeler, ayrı ayrı listelere kaydedilir.
3. Her bir kelimenin her bir metinde kaç kez geçtiği hesaplanarak bir frekans haritası (map) oluşturulur.
4. Frekans haritaları kullanılarak, metinler arasında ortak olan kelimeler belirlenir.
5. Ortak kelimeler birleştirilerek yeni bir metin oluşturulur.
6. Yeni metin döndürülür.

Ek1: Sözde Kod


```
_id: ObjectId('6422f2162b2bee180c6daf48')
▾ metinList: Array
    0: "Ali eve gel"
    1: "eve gel sonra"
    2: "eve gel sonra çarşıya"
    3: "çarşı git"
  MergedText: "Ali eve gel sonra çarşıya git "
  ElapsedTime: 0.171
  _class: "com.example.TextService.model.DBtext"
```


Ek2: Veritabanı modeli

Metin Birleştirme Arayüzü

Metin 1:

Ali eve gel

Metin 2:

eve gel sonra

Metin 3:

eve gel sonra çarşıya

Metin 4:

çarşı git

Yeni Metin Ekle | Metinleri Birleştir | Birleşik Metni Kaydet | Veritabanını listele

Birleştirilmiş Metin: Ali eve gel sonra çarşıya git

Geçen süre: 0.171 ms

Kaydetme işlemi: Başarılı

Ek3: Web sayfası