Kocaeli Üniversitesi

Bilgisayar Mühendisliği Bölümü

Yazılım Laboratuvarı II

METİN BİRLEŞTİRME İŞLEMİ

Taha Uz

190202013@kocaeli.edu.tr

ÖZET

Yazılım Laboratuvarı II Projesi olarak bizden "Metin Birleştirme" adlı uygulama geliştirmemiz beklenmektedir.

Proje için Java ve React dili tercih edildi. Intelij geliştirme ortamını kullanıldı Java ve Spring kullanımı açısından. Web arayüz için ise React.js kullanıldı.

Proje dökümanında, kullanıcıdan bir web arayüzünde dinamik bir şekilde metin girdisi yapabileceği bir arayüz yapmamız beklenmekteydi. Daha sonra kullanıcının girdiği metinleri Java kısmında gerçekleştirilen ortak kelime gruplarını ortaya çıkartılarak tüm metini birleştirip birleştiremeyeceğine karar veren algoritmaya "localhost" üzerindeki server üzerinden taşınan verileri teslim eder. Daha sonra bu veriler algoritma tarafından işlenip "MongoDB" ye kaydedilir ve bulunan sonuç server üzerinden web arayüzüne yollayarak kullanıcıya geri bildirim sağlamaktadır.

1.GİRİŞ

Proje için Java ve React dili tercih edildi. Intelij geliştirme ortamını kullanıldı Java ve Spring kullanımı açısından. Web arayüz için ise React.js kullanıldı. Veri tabanı olarak ise .JSON verilerin düzgün bir yapıda ve kullanışlı Compass uygulaması sayesinde MongoDB tercih edildi.

Spring, kurumsal uygulamalar geliştirmek için kullanılan Java tabanlı bir web çerçevesidir. Minimum kodlama ile web uygulamaları geliştirmek için esnek bir platform sağlar.

ReactJs, geliştiriciler tarafından web uygulamaları için kullanıcı arayüzleri oluşturmak amacıyla kullanılan güçlü bir JavaScript kütüphanesidir.

2.TEMEL BİLGİLER

Projeyi geliştirme aşamasına başlamadan önce belirli alt başlıklar altına ayırıp geliştirmeye başlandı. Bu başlıklar; web arayüzü, web ile java

arasında bağlantı, metin birleştirme algoritması olmak üzere üç başlık altında bu dökümanda incelenecektir.

2.1.Web Arayüz

Genel olarak bakılırsa burada kullanıcıdan metin girdileri alınır ve 'axios' kütüphanesi ile 'localhost' üzerine giriler veriler işlenmek için yollanır. Kullanıcı burada istediği kadar metin girme yetkisine sahip olup bu girilen metinler işlendikten sonra sonucu görmek için 'Show Result' butonuna tıklayarak sonucunu görebilir.

Yazılan algoritmayı daha detaylı açıklamak gerekirsek; handleChange() fonksiyonu, her bir metin girdisi için bir değişiklik yapıldığında çağrılır ve inputValues array'indeki değerleri günceller. Fonksiyon, girdi değiştiğinde önce yeni bir array oluşturarak bu yeni array'in index'ine kullanıcının girdiği değeri yerleştirir. Ardından setInputValues() fonksiyonunu kullanarak yeni array'i state'e aktarır.

handleAddInput() ve handleRemoveInput() fonksiyonları, kullanıcının metin girdilerini eklemesi veya kaldırması için kullanılır. handleAddInput() fonksiyonu, kullanıcının bir yeni metin girdisi eklemesine olanak tanır ve inputValues array'ine bir boş string ekler. handleRemoveInput() fonksiyonu ise kullanıcının mevcut bir metin girdisini kaldırmasına izin verir ve inputValues array'inden belirtilen index'e sahip elemanı çıkarır.

Kullanıcıya bilgilendirmeyi Java da dönen sonuç kısmı ile bilgilendirme yapılır. Ancak buradaki işlemi bir sonraki alt başlıkta daha detaylı açıklanacaktır.

2.2.Web ve Java Arasındaki Bağlantı

handleClick() fonksiyonu, kullanıcının metin girdilerini birleştirmesini ve bir POST isteğiyle sunucuya göndermesini sağlar. Bu fonksiyon, inputValues array'indeki tüm metinleri birleştirerek '-' karakteri ile ayırır ve bu işlenmiş metin değerini sunucuya gönderir. İsteği gerçekleştirmek için, axios kütüphanesi kullanılır. Axios, JavaScript'te HTTP istekleri yapmak için kullanılan bir kütüphanedir. İstek başarılı bir şekilde tamamlandığında, handleResultData() fonksiyonu, sunucudan işlenmiş metin sonucunu almak için bir GET isteği gönderir ve result state'ini günceller. Bu fonksiyon, sunucudan bir cevap alındığında, cevaptaki result değerini setResult() fonksiyonu kullanarak result state'ine aktarır ve böylelikle kullanıcı bilgilendirilir.

Java da ise Spring yardımı ile Controller sınıfı içerisinde getData() yöntemindeki @GetMapping açıklaması, bu yöntemin HTTP GET isteklerini ele aldığını belirtir. addData() yönteminde ise @PostMapping açıklaması, bu yöntemin HTTP POST isteklerini ele aldığını belirtir. Bu sınıf, bir veri hizmetini çağıran getData() yöntemi ve bir veri nesnesi alarak kaydeden ve işleyen addData() yöntemi ile birlikte çalışır.

2.3.Metin Birleştirme Algoritması

Genel anlamda bu algoritma iki boyutlu bir string dizisi alır. ilk boyutundaki öge cümle ikinci boyutunda o cümleye ait kelimeleri saklamaktadır. İlk olarak ortak kelime araması gerçekleştirmektedir. Ortak kelimeyi bulduktan sonra diğer cümleye geçer ve sonucu kaydedilen diziye bir şey kaydedilmez. İkinci cümlede o ortak kelime bulunduğu zaman tekrardan kaydetmeye başlar ve sonuna kadar bu şekilde ilerler. Bulunan ortak kelimelerde kökler benzer ve birisinde başka diğerinde de başka bir ek var ise hangi kelime en uzun ise onu öncelik yazar.

İlk olarak, "concatenateTexts" fonksiyonu, bir metin dizisini alır ve her metni kelimelere ayırır. Ardından, her metindeki her kelimeyi diğer metinlerle karşılaştırır ve ortak bir kelime bulursa, bu kelimeyi "commonWord" dizisine

ekler. Bu dizi, ortak kelimeyi ve bu kelimenin hangi iki metinde bulunduğunu içerir.

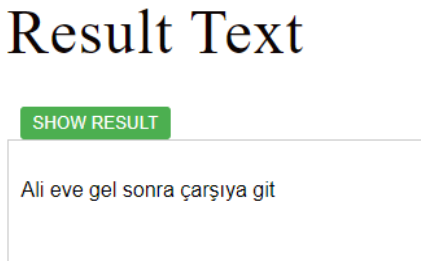
Daha sonra, "commonWord" dizisi, hangi iki metinde ortak kelime bulunduğunu gösteren bir diziye dönüştürülür. Dizideki tüm öğeleri gezinir ve ardışık öğelerin ilk harflerinin aynı olup olmadığını kontrol ederek, son ortak kelime öğesini bulur.

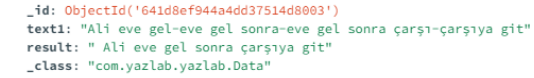
```
public String findCommonChr(String[] word1, String word2){
        int length = 0;
        boolean found ;
        String foundWord = "";
        for (int i = 0; i < word1.length; i++) {
            found = true;
            if(word1[i].length() > word2.length()){
                length = word2.length();
            }else{
                length = word1[i].length();
            }
            for (int j = 0; j < length; j++) {
                if(word2.toLowerCase().charAt(j) != word1[i].toLowerCase().charAt(j)){
                    found = false;
                }
            }
            if(found){
                if(word2.length() == length){
                    foundWord = word1[i];
                }else{
                    foundWord = word2;
                }

            }
        }

        if(foundWord.equals("")){
            return "";
        }

        return foundWord;
    }
```

Son olarak, "concatenateTexts" fonksiyonu, bulunan ortak kelimeyi kullanarak metin dizisini birleştirir. Metin dizileri, ortak kelimeyi içeren kelimelerle birleştirilir. Bu işlem, "StringBuilder" sınıfı kullanılarak gerçekleştirilir.

Son olarak, "concatenateTexts" fonksiyonu, sonuç metnini döndürür. Bu işlem, "toString" metodu kullanılarak gerçekleştirilir. Eğer ortak kelime bulunamazsa, fonksiyon, "ortak kelime bulunamadı" hatası verir.

3.DENEYSEL SONUÇLAR






```
Location(1-0) finalCommonWord: gel
Location(2-1) finalCommonWord: sonra
Location(3-2) finalCommonWord: çarşıya
```

4.SONUC

Bu proje Java ile React yani bir web uygulamasının iş mantığı ile yazılmış bir algoritma ile server üzerinden nasıl konuşturulucağını öğretmiş oldu.

5.REFERANSLAR

https://stackoverflow.com/questions/18521009/spring-mongodb-query-sorting

https://www.freecodecamp.org/news/how-to-use-axios-with-react/#how-to-make-a-get-request

https://www.youtube.com/watch?v=vtPkZShrvXQ

https://www.youtube.com/watch?v=6utzRKiBZt0

6.AKIŞ DİYAGRAMI