WEB INDEXLEME PROJESI

Kocaeli Üniversitesi Bilgisayar Mühendisliği Yazılım Labaratuvarı 2 – 1. Proje

Emre Yelbey
Kocaeli Üniversitesi
Bilgisayar Mühendisliği
180202043
emre_yelbey@hotmail.com

Ege Özeren
Kocaeli Üniversitesi
Bilgisayar Mühendisliği
180202047
ege_ozeren@gmail.com

Hazırlanan program verilen bir veya daha fazla URL üzerinde bir takım hesaplamalar ve analizler yapan bir web aplikasyonudur.

I. GİRİŞ

Uygulama başlatıldığında kullanıcıyı farklı analiz ve hesaplamalar için ayrılmış sekmelerden, varsayılan olarak anasayfa sekmesi karşılar. Bu kısım başta kullanıcıyı yapabileceği işlemler hakkında bilgilendirmek ve analiz yapmak istediği konu hakkında yönlendirmek amacıyla tasarlanmıştır.

Bu sekmeler kısaca şunları içerir:

- **Anasayfa:** Bu sekme yukarıda da bahsedildiği şekilde kullanıcı karşılayan ve bilgilendirme amaçlı varsayılan sekmedir.

- **Kelime Frekanslarını Hesaplama:** Bu sekmede kullanıcıdan URL girdisi almaya yarayan bir text alanı ve submit etmeye yarayan bir buton bulunur. Bu kısımda kullanıcın girdiği URL'den çıkarılan metinde her kelimenin kaçar kez yer aldığı hesaplanır.

- **Anahtar Kelime Çıkarma**: Bu sekmede kullanıcıdan URL girdisi almaya yarayan bir text alanı ve submit etmeye yarayan bir buton bulunur. Bu kısımda kullanıcının girdiği URL'den çıkarılan metindeki 10 anahtar kelime tespit edilir.

- **Benzerlik Skorlaması:** Bu sekmede kullanıcıdan URL girdisi almaya yarayan iki adet text alanı ve submit etmeye yarayan bir buton bulunur. Bu kısımda kullanıcın girdiği iki URL'den çıkarılan metinler arasındaki benzerlik oranı hesaplanır.

- **Site İndexleme ve Sıralama:** Bu sekmede kullanıcıdan URL girdisi almaya yarayan iki adet text alanı ve submit etmeye yarayan bir buton bulunur. Benzerlik skorlama kısmından farklı olarak bu kısımda URL girmek için ayrılmış ikinci kısıma birden fazla URL girilebilir. Bu kısımda yapılan işlem ilk text alanına girilen URL ile ikinci kısma girilen URL kümesi ve bu kümedeki URL'lerin alt URL'leri arasında benzerlik hesaplaması yapmak ve buna dayalı olarak elde edilen tüm URL'leri benzerlik skoruna göre indexlemektir. Ayrıca bu sekmede girilen URL kümesi ve alt URL'leri ağaç yapısı şeklinde gösterilir.

- **Semantik Analiz:** Bu sekmede kullanıcıdan URL girdisi almaya yarayan bir text alanı ve submit etmeye yarayan bir buton bulunur. Bu kısımda kullanıcının girdiği URL'den çıkarılan metindeki 10 anahtar kelime ile aynı metinde bu kelimelerle anlamsal olarak yakınlık veya benzerlik içeren kelimerin (varsa) tespiti yapılır.

II. TEMEL BİLGİLER

Program **Python** programlama dilinde geliştirilmiş olup geliştirme ortamı olarak **PyCharm Community 2020.3.4** kullanılmıştır.

III. TASARIM

Web indexleme projesinin geliştirme aşamaları belirtilen başlıklar altında açıklanmıştır.

A. *Yöntem*

Hazırlanan program Python üzerinde Flask microframeworkü kullanılarak oluşturulmuş bir web aplikasyonudur. Kısaca microframeworkten bahsetmek gerekirse, Flask dökümantasyonunda belirtildiği şekilde micro kelimesinden kasıt tüm web aplikasyonunun bir web sayfasına sığabilmesi (aslında mümkündür) ya da işlevsellikten yoksun olması demek değildir. Bahsedilen aslında Flask'in çekirdeğinde basit ama ihtiyaca göre genişletilip şekillendirilebilen esnek bir framework olmasıdır. Bu avantajı nedeniyle projeyi Flask üzerine inşa ettik.

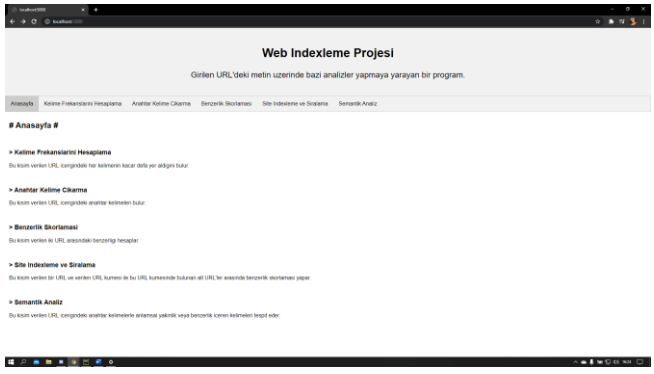
Projeye ilk olarak Flask web sunucusunu oluşturarak başladık. Aşağıdaki kod parçası oluşturulan index.html dosyasını render eden web sunucusunu oluşturmak için yeterli oldu. (*bkz. Screenshot 1*)

```
from flask import Flask,
render_template, request, redirect,
url_for

app = Flask(__name__)

@app.route("/")
def home():
    return render_template("index.html")
```

Flask aracılığıyla web server oluşturma (app.py)



Screenshot 1: index.html

Yukarıdaki ekran görüntüsünde kullanıcıyı karşılayan varsayılan olarak anasayfa sekmesi görülmektedir. Kullanıcı sekmeler aracılığıyla işlemler arasında gezinebilir. Sekmelerin tasarımları ve web sunucusunda yapılan işlemler nerdeyse aynıdır ve şu şekildedir.

Sekmelerin genel yapısı aşağıdaki gibidir:

```
<form action="/kelimefrekansi"
method="POST">
  <input type="text" name="gelen_url"
size="50" value="Buraya bir URL
girin...">
  <button type="submit">Hesapla</button>
</form>
```

Kelime frekansları sekmesinde bulunan kod parçası (index.html)

Yukarıdaki html kod parçasında bulunan form yapısı sayesinde kullanıcıdan URL girdisi alınıp Flask web sunucusuna (app.py) gönderilir.

Serverdaki genel yapı ise aşağıdaki gibidir:

```
@app.route("/kelimefrekansi", methods =
["GET", "POST"])
def kelimefrekansi():
    if request.method == "POST":
        gelenUrl =
request.form.get("gelen_url")
        frekans =
kelimefrekanslari.hesapla(gelenUrl)
        toplam_ks =
kelimefrekanslari.ret_toplam_ks(frekans)
        farkli_ks =
kelimefrekanslari.ret_farkli_ks(frekans)
        return
render_template("kelimefrekansi.html",
frekans=frekans, toplam_ks=toplam_ks,
farkli_ks=farkli_ks)
    else:
        return redirect(url_for("home"))
```

Kelime frekansları işlemi için tasarlanmış yönlendirme kod parçası (app.py)

Yukarıdaki python kod parçası post request metodu ile gelen linki alıp gerekli hesaplamalar veya analizler yapıldıktan sonra sonuçların görüntülenecek olduğu kelimefrekansi.html sayfasını render eder.

Hesaplama veya analizlerden sonra render edilen sayfanın yapısı ise aşağıdaki gibidir.

```
<ul>
    {% for i in range(0, farkli_ks) %}
        <li>{{frekans[i][0]}} -
{{frekans[i][1]}}</li>
    {% endfor %}
</ul>
```

Sonuçların yazdırıldığı kod parçası (kelimefrekansi.html)

Flask ile yukarıda görüldüğü şekilde html dosyası içinde python kodu yazılabilir. Python dışında Flask'in ayrıca kendi metodları da bulunur.

B. *Algoritma*

1. Anahtar Kelimelerin Tespit Edilmesi

Öncelikle daha önceden oluşturulmuş farklı anahtar kelime çıkarma algoritmalarını taradık. Bulduğumuz bütün yöntemleri anahatlarıyla gözden geçirdik. Elemelerimiz sonucu elimizde BERT, RAKE ve TFIDF algoritmaları kaldı ve TFIDF'de karar kıldık.

TFIDF (term frequency-inverse document frequency) yönteminin çalışma yönteminden kısaca bahsedecek olursak:

TF: Bir doküman içerisinde geçen terim ağırlıklarını hesaplamak için kullanılan yöntemdir.

IDF: Birden fazla dokümanda kelimenin geçme sayısını bularak bu kelimenin terim olup olmadığını bağlaç vb (Stop Words) olduğu anlamaya çalışır. Bunun için Terimin Geçtiği Dokuman Sayısı / Dokuman Sayısı'nın logaritmasının mutlak değeri alınmaktadır.

Formülüzasyonu ise şu şekilde:

Variants of TF weight

weighting scheme	TF weight
binary	{0,1}
raw frequency	$f_{t,d}$
log normalization	$\log(1 + f_{t,d})$
double normalization 0.5	$0.5 + 0.5\frac{f_{t,d}}{\max f_{t,d}}$
double normalization K	$K + (1 - K)\frac{f_{t,d}}{\max f_{t,d}}$

Algoritmayı hayata geçirmek için kendi IDF sözlüğünüzü yaratmanız gerekiyor. Bunu Ekşi Sözlük'ten 120 mb'lık text verisi çekerek sağladık. Tekil herbir kelimeyi bulup, IDF formülü ile, kullanacağımız formata uygun bir sözlük yarattık.

Kod içerisinde çalışırken, herbir kelime için ayrı bir vektör oluşturuyoruz. Bu vektörde sözlükteki tüm kelimeler mevcut oluyor. Anahtar kelime çıkaracağımız websiteden kelimeler geldikçe, kelimelerin değerini hesaplayıp, 0 olan değerlerini yenisiyle değiştiriyoruz. En yüksek puana sahip 10 kelime de anahtar kelimelerimiz oluyor.

2. Benzerlik ve Semantik Analiz

Türkçe sondan eklemeli bir dil olduğu için, ne yaparsak yapalım, benzerlik sonuçlarında ve alakalı kelimelerde sapmalar oluyordu. Daha doğru sonuç almak için kullandığımız ilkel yöntemlerden vazgeçip işin yapay zeka kısmına yöneldik.

Word2Vec 2013'te geliştirilmiş, semantik analiz için kullanılan yaygın bir yöntem.

Amaç, birbirine benzeyen kelimeleri matematik model ile tespit etmek. Bunu da kelimeler arasında 0 ile 1 arasında kosinüs benzerliği ile birbirine benzeyen kelimelere ağırlık vererek yapıyoruz.

Bir adet giriş, bir adet çıkış ve bir adet de hesaplamaların yapıldığı gizli katman var. Eğer bir tamlama veya kelime birlikte sık sık kullanılıyorsa ağırlığı, yani benzerlik oranı artıyor. Sonuçların daha doğru çıkması için modeli bolca veri ile eğitmek gerekiyor. Modeli eğitmeyi Vikipedi'nin veri işleme için açık olarak paylaştığı 500 mb'lık "dump" verisini ve ek olarak kendi hazırladığımız text verilerini kullanarak sağladık.

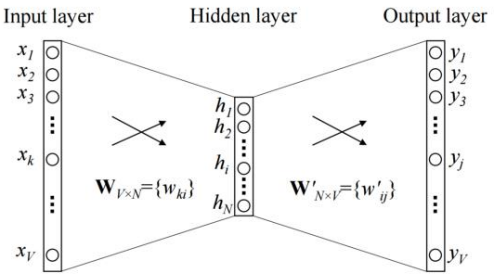


Figure 1: A simple CBOW model with only one word in the context

C. *Kullanılan Bazı Metodlar ve Classlar*

- ***def kelimefrekansi( ):***

Bu metod web sayfasından gelen linki kelime frekanslarının hesaplandığı kod kısmına gönderir ve gelen return değerleri ile sonuçların görüntülendiği web sayfasını render eder.

- ***def anahtarkelime( ):***

Bu metod web sayfasından gelen linki anahtar kelimelerin tespit edildiği kod kısmına gönderir ve gelen return değerleri ile sonuçların görüntülendiği web sayfasını render eder.

- ***kelimefrekanslari.py***

Bu sınıf verilen URL içinde her kelimenin kaçar kez geçtiğini hesaplar.

- ***benzerlikskorlama.py***

Bu sınıf verilen iki URL arasındaki benzerlik oranını hesaplar.

- ***semantik.py***

Bu sınıf verilen bir URL içindeki anahtar kelimelerle semantik olarak ilişkili kelimeleri tespit eder.

D. *Karşılaşılan Bazı Sorunlar*

- Karşılaştığımız ilk problem web sayfasından veri çekme ve bu veriyi temizleme kısmıydı. Hesaplamalardan olabildiğince doğru sonuç alabilmek için hangi yöntem ve metodu kullanacağımız konusunda epey düşündük ve farklı yollar denedik.

- İkinci ve daha büyük bir sorun ise topladığımız verileri temizlemek ve modeli eğitmek oldu. Daha önce NLP konusunda yeterli bilgi ve deneyimimiz olmadığı için başta zorlandık.

E. *Kazanımlar*

- Flask frameworkü kullanarak web aplikasyonu geliştirmeyi öğrendik.

- NLP hakkında bilgi sahibi olduk.

- Veri toplamayı ve işlemeyi öğrendik.

KAYNAKÇA

[1] https://medium.com/fedeveloper/regex-hakk%C4%B1nda-2c4e80501802

[2] https://regexr.com/5p2hk

[3] https://towardsdatascience.com/keyword-extraction-python-tf-idf-textrank-topicrank-yake-bert-7405d51cd839

[4] https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

[5] https://programminghistorian.org/en/lessons/analyzing-documents-with-tfidf

[6] https://www.analyticsvidhya.com/blog/2018/11/introduction-text-summarization-textrank-python/

[7] https://docs.python.org/3/library

[8] https://flask.palletsprojects.com/en/1.1.x/

[9] https://medium.com/@mikejschoi/multi-threading-with-python-3383de01b95a

[10] https://www.youtube.com/watch?v=pNqtDpcPJm0&list=PLIHume2cwmHfDUQ2t6f-Q7VCBvNSEAMAS