

# İNTİHAL KONTROL UYGULAMASI

Furkan BARAN

11-2023

# İçerik Tablosu

|   |   |
|---|---|
| 1. Giriş .....  | 3 |
| 2. İntihal Tespitinde Kullanılacak Algoritma.....                           | 3 |
| 2.1. “İntihal”ın Tanımı .....   | 3 |
| 2.2. “Kelime”nin Tanımı.....  | 3 |
| 2.3. “N sayısı”nın Tanımı .....   | 3 |
| 2.4. Kelimelerin Karşılaştırılması .....                                    | 3 |
| 2.5. Algoritma .....  | 4 |
| 2.6. Kontrol Algoritmasının Python Koduyla Fonksiyon Olarak Yazılması ..... | 4 |
| 3. Dosyaların Okunması ve İşlenmesi.....                                    | 5 |
| 3.1. Asıl Dosyayı Okuma.....  | 5 |
| 3.2. Kaynak Dosyaları Okuma .....   | 5 |
| 4. Flask Web Çerçevesinin Kurulumu .....                                    | 6 |
| 4.1 Web Uygulamasının Anasayfa (index.html) Dosyasını Oluşturma .....       | 6 |
| 4.2 Route Methodu ile Sayfa Yönlendirmeleri .....                           | 6 |
| 4.3 Yüklenen Dosyaların İşlenmesi .....                                     | 7 |
| 4.4 Kontrol Edilmiş Metinleri HTML Formatına Uygun Hale Getirme .....       | 7 |
| 4.5 Web Uygulamasının İlk Kurulum Çıktısı .....                             | 8 |

## 1. Giriş

Bu uygulamada **Python** programlama dilini kullanarak bir metni, verilecek diğer metinler ile karşılaştırarak aralarındaki **benzerliklerin tespit edilmesi ve işaretlenmesi** hedeflenmiştir. Aynı zamanda kullanıcıların dosya seçimlerini rahatlıkla yapabilmeleri, intihal olarak işaretlenen kısımları görsel olarak ayırt edici şekilde görebilmeleri, intihallerin hangi dosyanın hangi cümlesinden yapıldığına kadar ayrıntılı gösterimlere erişebilmesi için bir **web arayüzü** tasarlanması ihtiyaç olarak görülmüştür.

İlgili ihtiyaçlar göz önünde bulundurulduğunda, uygulama için **Django** veya **Flask** Python web çerçevelerinden birinin kullanılması uygun görülmüştür. Daha sonra sahip olduğu basitlik, esneklik ve özgürlük özellikleri nedeniyle **Flask** çerçevesinin kullanılmasına karar verilmiştir.

## 2. İntihal Tespitinde Kullanılacak Algoritma

İntihal tespiti için çeşitli kütüphaneler, yapay zeka modelleri araştırılmış ve test edilmiştir. Yapılan araştırma ve testler sonucu, bu kütüphaneler ve modellerin hem dil konusunda esneklik sağlamaması, hem de istenen seviyede hızlı sonuçlar verememesi nedeniyle uygulamaya özel algoritmaya sahip bir fonksiyon ile intihallerin tespit ve işaretlenmesinin yapılmasına karar verildi.

### 2.1. “İntihal”ın Tanımı

Uygulamanın algoritmasında intihal olarak işaretlenme koşulları belirlenmiştir. Buna göre; Verilen metinde ardışık N adet kelimenin, diğer metinler içinde de aralarında başka kelime olmaksızın ardışık şekilde aynı sırayla bulunması, intihal sayılacaktır.

### 2.2. “Kelime”nin Tanımı

Uygulamanın algoritmasında intihal olarak işaretlenme koşullarında, bir karakter dizisinin “kelime” olarak sayılması koşulları belirlenmiştir. Buna göre; Metindeki karakter dizilerinden, aralarında en az bir boşluk veya yeni satır karakteri olanlar birer kelime sayılacaktır. Kelimeler

### 2.3. “N sayısı”nın Tanımı

N sayısı varsayılan ve en az 3 olmak üzere bir tam sayı olmalı ve kullanıcıdan alınabilmelidir. N sayısının büyüklüğü, intihal sayımına toleransın artmasına sebep olur, düşüklüğü ise toleransın çok azalmasına ve intihal oranının yüksek bulunmasına sebep olabilir.

### 2.4. Kelimelerin Karşılaştırılması

- Kelimeler karşılaştırılırken, içerdikleri tüm harfler küçük harfe çevrilir. Böylece büyük-küçük harfe karşı duyarsızlaşılır.
- Kelime içinde çeşitli noktalama işaretlerinin olması durumunda noktalama işaretlerinin kaldırılarak birleştirilmiş hali karşılaştırma için ele alınır.

Örnek metin: **Pendikspor’un 2023-2024 sezonunda şampiyon olma ihtimali kalmadı.**

Karşılaştırma için kullanılacak çıktı: **pendiksporun 20232024 sezonunda şampiyon olma ihtimali kalmadı**

## 2.5. Algoritma

Bir kelimenin veya kelime grubunun intihal olarak işaretlenmesi ile ilgili yukarda verilen koşullara göre, Python kullanılarak bu algoritmanın gerçekleştirilmesi için şu şekilde bir sözde kod (psödokod) hazırlanmıştır:

- Kontrole başlamadan önce kelimeleri uygun formata dönüştürüp sakla.
- Kontrol edilecek metnin ilk kelimesinden başlayarak döngü şeklinde ardışık N adet kelimeyi ele al,
  - Bu kelimeleri verilen kaynak metinler arasında aramak için aynı şekilde her kaynak dosyası için;
    - İlk kelimedenden başlayarak döngü şeklinde N ardışık kelimeyi ele al
      - eğer her iki kelime grubu da aynı ise, her iki gruptaki kelimeleri de intihal olarak işaretle
    - kaynak dosyasından ele alınan kelime grubundan ilk kelimeyi çıkar, metindeki sonraki kelimeyi grubun sonuna ekle ve karşılaştırmalara devam et.
  - Kontrol edilecek metnin ele alınan kelime grubundan ilk kelimeyi çıkar, metindeki sonraki kelimeyi grubun sonuna ekle ve karşılaştırmalara devam et.

## 2.6. Kontrol Algoritmasının Python Koduyla Fonksiyon Olarak Yazılması

```
main_text = []      # asıl metin listesi      [kelime]
main_gui = []       # asıl metnin gui'de kullanılmak üzere saklanan ve işaretlerini tutan listesi
                    [kelime, intihalMi?, yeni satır var mı?]
source_texts = []   # kaynak metinlerin listesi  [dosya] [kelime]
source_gui = []     # kaynak metinlerin gui'de kullanılmak üzere saklanan ve işaretlerini tutan listesi
                    [dosya] [kelime, intihalMi?, yeni satır var mı?]

def kontrol():
    N = 3 # varsayılan değer
    n = N - 1 # N sayısının bir eksiği, eklenecek kelime ilk kelimedenden N-1 kelime sonraki kelime olacaktır
    main_N= main_text[:N-1].copy() # ilk N-1 kelimeyi main_N listesine ekliyoruz, N tane eklememe sebebimiz
                                   # sonraki kelimenin döngüde eklenecek olmasıdır
    main_len = len(main_text)
    for id in range(main_len - n):
        main_N.append(main_text[id + n]) # main_N listesine sonraki kelimeyi ekliyoruz
        for file_id in range(len(source_texts)): # dosya sayısı kadar döngü
            for j in range(len(source_texts[file_id]) - n): # dosyadaki kelime sayısı-1 kadar döngü
                if main_N == source_texts[file_id][j:j+N]: # eğer main_N listesi, kaynak kelime listesinin
                                                            # ardışık N kelimesine eşitse
                    for k in range(N): # N kadar döngü şeklinde her kelimeyi işaretliyoruz
                        main_gui[id + k][1] = True # main_text'in arayüz için kullanılacak dönüştürülmemiş
                                                    # halini içeren listede, kelimenin 1. indisi True
                                                    # değerini(intihal) alıyor (0. indisi kelimeyi tutuyor)
                        source_gui[file_id][j + k][1]=1 # kaynak kelime listesinin arayüz için kullanılacak
                                                         # dönüştürülmemiş halini içeren listede, kelimenin 1.
                                                         # indisi 1 değerini (intihal) alıyor (0. indisi kelimeyi tutuyor)
    main_N.pop(0) # main_N listesinin ilk elemanını çıkarıyoruz,
                 # böylece yeni kelime eklenebilir hale geliyor
```

## 3. Dosyaların Okunması ve İşlenmesi

Asıl dosya ile kaynak dosyaların işlenişi ve sayıları farklı olduğundan her iki grup için ayrı fonksiyonlar kullanılması uygun görüldü. Ayrıca dosyalar kullanıcıdan web arayüzle alınacağından dosya nesneleri string listeleri olarak varsayıldı.

### 3.1. Asıl Dosyayı Okuma

Önceki listeler yeni okuma için temizlenir, dosya satır satır okunur, her satır kelimelere ayrılır, her kelime uygun formatlı haliyle uygun listeye eklenir, satır sonlarındaki kelimeler satır sonu olarak işaretlenir.

```
def dosya_oku(file_content):
    main_text.clear()
    main_gui.clear()
    text_lines = file_content.splitlines()
    word_id = 0
    for line in text_lines:
        for word in line.split():
            main_text.append(word.translate(str.maketrans(' ', ' ', r'""!"#$%&'()*+,-
./:;<=>?@[\\]^_`{|}~^"'''"))).lower())
            main_gui.append([word, False, False])
            word_id += 1
        main_gui[word_id - 1][2] = True
```

### 3.2. Kaynak Dosyaları Okuma

Listeler temizlenir, listelerde yeni dosyalar için yer açılır, döngü halinde her dosya ayrı ayrı, tek tek satırlar ve kelimelere ayrılarak okunur ve uygun listelere eklenir. Her dosyaya sırasına göre bir id verilmiş olur.

```
def dosyalar_oku(file_contents):
    source_gui.clear()
    source_texts.clear()
    file_id = 0
    file_dict.clear()
    for file_content in file_contents:
        source_texts.append([])
        source_gui.append([])
        text_lines = file_content.splitlines()
        word_id = 0
        for line in text_lines:
            for word in line.split():
                source_gui[file_id].append([word, 0, False])
                source_texts[file_id].append(word.translate(str.maketrans(' ', ' ', r'""!"#$%&'()*+,-
./:;<=>?@[\\]^_`{|}~^"'''"))).lower())
                word_id += 1
            source_gui[file_id][word_id - 1][2] = True
        file_id += 1
```

## 4. Flask Web Çerçevesinin Kurulumu

Uygulamanın çıktılarını görüp test edebilmek için Flask ile web ortamına entegresine başlandı. Öncelikle bilgisayarda Flask'in kurulu olduğundan emin olmalıyız.

```
>> pip install Flask
```

Daha sonra açacağımız proje klasörü içinde “app.py” dosyası oluşturuyoruz:

```
from flask import Flask
```

```
app = Flask(__name__)
if __name__ == '__main__':
    app.run(debug=True)
```

### 4.1 Web Uygulamasının Anasayfa (index.html) Dosyasını Oluşturma

Kullanıcıları karşılayan, gerekli girdileri almak üzere kullanışlı bir arayüze sahip html sayfası oluşturulmalıdır. Ayrıca bu sayfa flask arkaplanıyla iletişimde olacaktır. Girilen dosyaları, arkaplanda python fonksiyonları ile işlenmek üzere sunucuya almalıdır. Bu html dosyası, ‘templates’ klasörü içinde bulunmalıdır.

Şimdilik işlevsel olarak test amaçlı basit bir arayüz ile kullanıcıdan metin dosyaları almak üzere bir sayfa hazırlandı.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>İntihal Kontrol</title>
</head>
<body>
  <h1>İntihal Kontrol</h1>

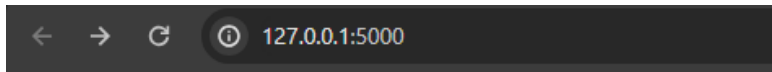
  <form action="/upload" method="POST" enctype="multipart/form-data">
    <label for="mainfile">Tek Dosya Seç:</label>
    <input type="file" id="mainfile" name="mainfile" accept=".txt" onchange="updateSubmitButton()">
    <br><br>

    <label for="otherfiles">Birden Fazla Dosya Seç:</label>
    <input type="file" id="otherfiles" name="otherfiles" accept=".txt" multiple onchange="updateSubmitButton()">
    <br><br>

    <button type="submit" id="submit-button" >Gönder</button>
  </form>
</body>
</html>
```

### 4.2 Route Methodu ile Sayfa Yönlendirmeleri

Hazırladığımız **index.html** sayfasını, web uygulamasının **anasayfası** olarak ayarlamak için app.py dosyamız içinde **@app.route('/')** dekoratörü ile tanımlayacağımız index fonksiyonu içinde Flask'in **render\_template** fonksiyonuna html sayfamızı göndererek, sayfamızı uygun şekilde işleyip isteği işlenmiş html sayfası ile yanıtlamasını sağladık.



## İntihal Kontrol

Tek Dosya Seç:  Dosya seçilmedi

Birden Fazla Dosya Seç:  Dosya seçilmedi

### 4.3 Yüklenen Dosyaların İşlenmesi

Index sayfasındaki form üzerinden POST methoduyla gönderilen dosyaları /upload'ta alıp gerekli kontrolleri yaptıktan sonra anlamlı verilere çevirecek şekilde okuyan fonksiyonlara gönderdik. Daha sonra kontrol fonksiyonunu çalıştırarak sonucu main ve other değişkenlerine atadık. Ve sonucu göstermek üzere render\_template fonksiyonuna result.html dosyasını main-other (asıl dosya ve kaynak metinler) değişkenleriyle çalıştırıyoruz.

```
@app.route('/upload', methods=['GET', 'POST'])
def upload():
    if request.method == 'POST':
        mainfile = request.files.get('mainfile')
        otherfiles = request.files.getlist('otherfiles')

        if not mainfile and not otherfiles:
            return render_template('index.html', error='Dosya seçilmedi.')

        if mainfile and '.' in mainfile.filename and mainfile.filename.rsplit('.', 1)[1].lower() in ["txt"]:
            main_content = mainfile.read().decode('utf-8')
            dosya_oku(main_content)

            others_contents = []

            for file in otherfiles:
                print(file.filename)
                others_contents.append(file.read().decode('utf-8'))

            dosyalar_oku(others_contents)
            main, other = kontrol()
            return render_template('result.html', main=main, other=other)

    return render_template('index.html')
```

### 4.4 Kontrol Edilmiş Metinleri HTML Formatına Uygun Hale Getirme

Kontrol fonksiyonu ile işlenmiş ve gerekli işaretlemelerin yapıldığı metinleri result.html sayfasında uygun ve ayırt edici şekilde görüntülemek üzere **html\_format** isiminde bir fonksiyon yazıldı. Dosya numarasını alıp ona göre sonuç döndüren bu fonksiyon, -1 numaralı dosyayı (asıl metin), intihal tespit edilmiş kelimeleri **kırmızı** renkle işaretlenmiş şekilde, yeni satırları <br> etiketiyle göstererek tüm metni html formatında döndürür. Eğer dosya numarası -1 değilse, kaynak dosyaları listesinden ilgili indisteki metni alır ve benzer işaretlemeleri yaparak html metnini döndürür.

```
def html_format(fileid):
    if fileid == -1:
        main_html = []
        text_len = len(main_text)
        plag_count = 0
        for word in range(text_len):
            if main_gui[word][1] == False:
                main_html.append(main_gui[word][0] + " ")
            else:
                main_html.append('<span style="color:red;">{}</span> '.format(main_gui[word][0]))
                plag_count += 1
            if main_gui[word][2] == True:
                main_html.append("<br>")
        return "".join(main_html)
    else:
        comp_text = []
        lenn = len(source_gui[fileid])
        print(source_gui[0])
        for word in range(lenn):
            if source_gui[fileid][word][1] == False:
                comp_text.append(source_gui[fileid][word][0] + " ")
            elif source_gui[fileid][word][1] == True:
                comp_text.append('<span style="color:red;">{}</span> '.format(source_gui[fileid][word][0]))

            if source_gui[fileid][word][2] == True:
                comp_text.append("<br>")
        return "".join(comp_text)
```

kontrol fonksiyonunun dönüş değerleri olarak html\_format fonksiyonuna iki dosya indisi gönderilerek işlenmiş çıktılar atandı. Test amaçlı ilk indis asıl dosya, ikinci indis kaynak dosyalarından ilki olarak verildi. Basit bir html sayfası test için hazırlandı.

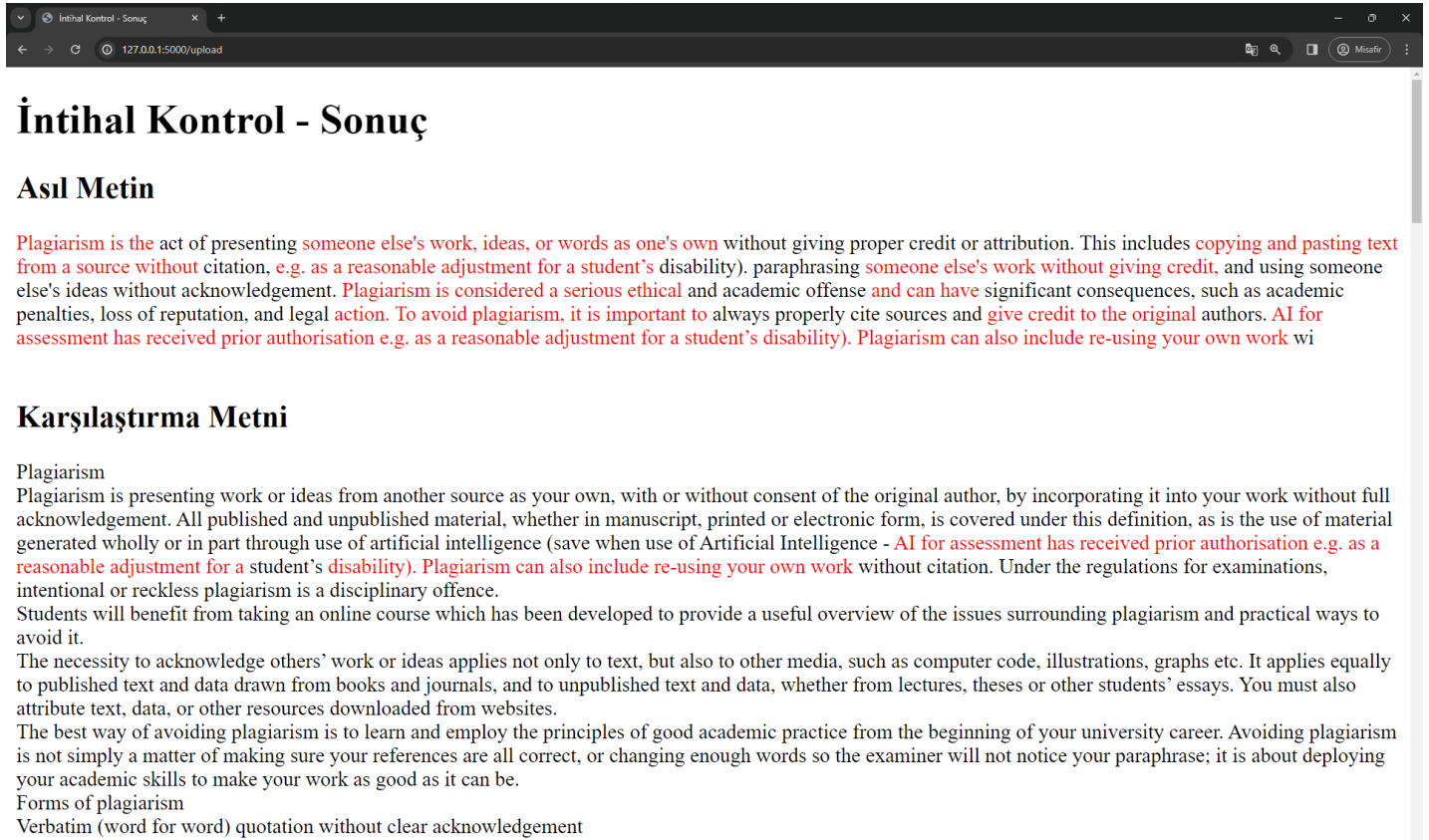
```
def kontrol():
    N = 3 # varsayılan değer, daha sonra kullan
    n = N - 1 # N sayısının bir eksiği, eklenece
    main_N= main_text[:N-1].copy() # ilk N-1 k
    main_len = len(main_text)
    for id in range(main_len - n):
        main_N.append(main_text[id + n]) # mai
        for file_id in range(len(source_texts)):
            for j in range(len(source_texts[file
                if main_N == source_texts[file
                    for k in range(N): # N kad
                        main_gui[id + k][1] =
                        source_gui[file_id][j
            main_N.pop(0) # main_N listesinin ilk
    return html_format(-1), html_format(0)

<!DOCTYPE html>
<html lang="tr">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>İntihal Kontrol - Sonuç</title>
</head>
<body>
    <h1>İntihal Kontrol - Sonuç</h1>
    <h2>Asıl Metin</h2>
    {{ main | safe }}
    <br>
    <h2>Karşılaştırma Metni</h2>

    {{ other | safe }}
</body>
</html>
```

## 4.5 Web Uygulamasının İlk Kurulum Çıktısı

Flask ile basit bir web sunucusu üzerinde daha önce yazdığımız fonksiyonların işlevselliğini, çalıştırarak sadece bir tasarımla birleştirdik. Sonuç istediğimiz gibi oldu. İntihal tespit edilen kelimeler kırmızı olarak işaretlendi. Yaklaşık 1000 karakterden oluşan bir metin dosyası ile toplamda yaklaşık 55 bin karakter içeren 4 farklı metin dosyasını neredeyse anında kontrol ederek sonucu yazdırdık.





## 5. Web Uygulamasının Kullanıcı Arayüzünü Geliştirme

### 5.1 Anasayfa

Uygulamanın arayüzünü daha kullanışlı ve estetik yapmak için Bootstrap kütüphanesi ve özel JavaScript kodları kullanıldı. Kontrol edilecek dosyayı yüklemek için bir alan, kaynak olarak kullanılacak dosyaları seçmek için başka bir alan oluşturuldu. Dosyaların sürükle-bırak şeklinde eklenebilmesi için JavaScript kullanıldı. Ayrıca kaç ardaşık kelime benzerliğinin intihal sayılacağını kullanıcının seçebilmesi için bir seçici eklendi.

The screenshot shows a web browser window with the title 'Plagiarism Checker'. The address bar shows 'localhost:5000'. The main content area has a dark header with the title 'Plagiarism Checker'. Below the header, there are two dashed boxes for file uploads. The first box is labeled 'Dosya Yükleme 1:' and the second 'Dosya Yükleme 2:'. Both boxes contain the text 'Dosyanızı buraya bırakın veya tıklayın.' Below these boxes, there is a slider for 'Hassasiyet aralığı:' (Sensitivity range) with a value of 3. At the bottom, there is a blue button labeled 'Gönder'.

Bu sayfa hem görsel olarak yeniden tasarlandı, hem de önemli birkaç işlevsel özellik eklendi. İntihal oranını gösteren bir bar, farklı bir tekrar sayısına göre yeniden kontrol çalıştırmak için bir seçici ve bir buton, asıl metin ve karşılaştırılan metinleri içeren metin kutuları, karşılaştırılan metni seçebilme için bir açılır menü eklendi. Ayrıca kontrol ve html\_format fonksiyonlarında birkaç değişiklik yapılarak, her kelimenin intihal olarak işaretlenmesine sebep olan karşılaştırma metnindeki karşılığının kolayca bulunabilmesine yardımcı olacak bir özellik eklendi. Asıl metinde kırmızı ile yazılı olan metinlere eklenen özel linkler (bağlantılar) sayesinde, tıklandığında ilgili metnin ilgili kelimesine yeşil ile işaretlenip ekranda gösterilmesi sağlandı.

The screenshot shows the result page of the 'İntihal Kontrol Uygulaması'. The header is 'İntihal Kontrol Uygulaması - Sonuç'. Below the header, there is a progress bar showing '39.0%' and a button labeled 'Tekrar Kontrol Et'. The main content area is divided into two columns. The left column is titled 'Asıl Metin' and contains the text '1-whatsplag.txt'. The right column is titled 'Karşılaştırma Metinleri' and contains a dropdown menu with 'oxford.edu.txt' selected. Below the dropdown, there is a section titled '#1 - oxford.edu.txt' with the text 'Plagiarism'. The text in the right column is highlighted in red, indicating the detected plagiarism.

### 5.3 İntihal Oranını Gösteren Bar

html\_format fonksiyonunda intihal olarak görülen kelimelerin toplam kelime sayısına bölümüyle elde edilen plag\_ratio değişkeni, toplam intihal oranını göstermesi için daha sonra result.html'e gönderildi. result.html içinde is basit bir html kodu ile gösterildi.

## İntihal Kontrol Uygulaması - Sonuç

65.0%

```
</header>
<div class="container">
  <h1 class="mt-4">İntihal Kontrol Uygulaması - Sonuç</h1>
  <div class="progress">
    <div class="progress-bar" role="progressbar" style="width: {{plag_ratio}}%; aria-valuenow="{{plag_ratio}}" aria-valuemin="0" aria-valuemax="100">
      {{plag_ratio}}%
    </div>
  </div>
</div>
<form action="/tekrar" method="POST" enctype="multipart/form-data" >
```

### 5.4 Yeniden Kontrol Formu

Kullanıcının aynı metinleri sadece farklı bir N değeri ile tekrar kontrol edebilmesini sağlamak amacıyla tekrarKontrol Fonksiyonu yazıldı. Bu fonksiyon verilen N değeriyle kontrol fonksiyonunu tekrar çalıştırmayı sağlar. Ayrıca önceki intihal işaretlerini sıfırlamak için deleteGui fonksiyonu ile tüm kelimeler tekrar 'intihal yok'(0, False) olarak işaretlendi.

20.0%

5

Tekrar Kontrol Et

Açıl Metin

Karşılaştırma Metinleri

ayrıld edüt

```
def deleteGUI():
    for word in main_gui:
        word[1]=False
    for file in (source_gui):
        for word in file:
            word[1]=0

@app.route('/tekrar', methods=['GET', 'POST'])
def tekrarKontrol():
    print(main_text)
    if request.method == 'POST':
        deleteGUI()
        N = int(request.form.get('N'))
        result = kontrol(N)
        return render_template('result.html', result=result, files=file_dict, mainfile=mainfilename, plag_ratio=plag_ratio//1, N=N)
```

## 5.5 Metin Kutuları

Kullanıcının yüklediği asıl ve kaynak metinlerini görebilmesi, intihalleri metin içinde görebilmesi için yan yana iki metin kutusu hazırlandı. Karşılaştırma metinleri kutusuna, metinler arasında geçiş yapabilmek için bir açılır menü eklendi. İlk sorgu sonucunda tüm metinler gösterilirken, daha sonraki kullanıcı seçimlerine bağlı olarak sadece seçilen metin kalacak şekilde yeniden düzenlenir. Bu özellik için CSS ve Javascript kullanıldı.

Asıl Metin

### 1-whatsplag.txt

Plagiarism is the act of presenting someone else's work, ideas, or words as one's own without giving proper credit or attribution. This includes copying and pasting text from a source without citation, e.g. as a reasonable adjustment for a student's disability). paraphrasing someone else's work without giving credit, and using someone else's ideas without acknowledgement. Plagiarism is considered a serious ethical and academic offense and can have significant consequences, such as academic penalties, loss of reputation, and legal action. To avoid plagiarism, it is important to always properly cite sources and give credit to the original authors. AI for assessment has received prior authorisation e.g. as a reasonable adjustment for a student's disability). Plagiarism can also include re-using your own work wi

Karşılaştırma Metinleri

oxford.edu.txt

### #1 - oxford.edu.txt

Plagiarism

Plagiarism is presenting work or ideas from another source as your own, with or without consent of the original author, by incorporating it into your work without full acknowledgement. All published and unpublished material, whether in manuscript, printed or electronic form, is covered under this definition, as is the use of material generated wholly or in part through use of artificial intelligence (save when use of Artificial Intelligence - AI for assessment has received prior authorisation e.g. as a reasonable adjustment for a student's disability). Plagiarism can also include re-using your own work without citation. Under the regulations for examinations, intentional or reckless plagiarism is a disciplinary offence.

Students will benefit from taking an online course which has been developed to provide a useful overview of the issues surrounding plagiarism and practical ways to avoid it.

The necessity to acknowledge others' work or ideas applies not only to text, but also to other media, such as computer code, illustrations, graphs etc. It applies equally to published text and data drawn from books and journals, and to unpublished text and data, whether from lectures, theses or other students' essays. You must also attribute text, data, or other resources downloaded from websites.

The best way of avoiding plagiarism is to learn and employ the principles of good academic practice from the beginning of your university career. Avoiding plagiarism is not simply a matter of making

## 5.6 İntihal Kaynağını Bulma

Bu eklenen özellik sayesinde intihal işaretlenmesine sebep olan karşılaştırma metinlerindeki kelimelere, dosya id ve kelime id olarak birer kimlik atandı. Ayrıca asıl metindeki kelimelerde ise intihal tespitleri sırasında ilgili karşılaştırma metnindeki kelimenin kimliği saklandı. Daha sonra bu kimlikler html hash linklerde kullanılarak javascript yardımıyla ilgili metnin sayfaya getirilmesi, ilgili kelimenin sayfada gösterilecek şekilde kaydırma yapılması, ilgili kelimenin yeşil ile işaretlenmesi sağlandı.

Asıl Metin

### 1-whatsplag.txt

Plagiarism is the act of presenting someone else's work, ideas, or words as one's own without giving proper credit or attribution. This includes copying and pasting text from a source without citation, e.g. as a reasonable adjustment for a student's disability). paraphrasing someone else's work without giving credit, and using someone else's ideas without acknowledgement. Plagiarism is considered a serious ethical and academic offense and can have significant consequences, such as academic penalties, loss of reputation, and legal action. To avoid plagiarism, it is important to always properly cite sources and give credit to the original authors. AI for assessment has received prior authorisation e.g. as a reasonable adjustment for a student's

Karşılaştırma Metinleri

oxford.edu.txt

### #3 - wikipedia.txt

Plagiarism is the fraudulent representation of another person's language, thoughts, ideas, or expressions as one's own original work. Although precise definitions vary, depending on the institution, such representations are generally considered to violate academic integrity and journalistic ethics as well as social norms of learning, teaching, research, fairness, respect, and responsibility in many cultures. It is subject to sanctions such as penalties, suspension, expulsion from school or work, substantial fines, and even imprisonment.

Plagiarism is typically not in itself a crime, but like counterfeiting, fraud can be punished in a court for prejudices caused by copyright

```

def kontrol(N):
    n = N - 1 # N sayısının bir eksiği, eklenecek kelime ilk kelimedenden N-1 kelime sonraki kelime olacaktır
    main_N = main_text[:N-1].copy() # ilk N-1 kelimeyi main_N listesine ekliyoruz, N tane eklememe sebebimiz sonraki kelimenin döngüde eklenecek olmasıdır
    main_len = len(main_text)
    for id in range(main_len - n):
        main_N.append(main_text[id + n]) # main_N listesine sonraki kelimeyi ekliyoruz
        for file_id in range(len(source_texts)): # dosya sayısı kadar döngü
            for j in range(len(source_texts[file_id]) - n): # dosyadaki kelime sayısı-1 kadar döngü
                if main_N == source_texts[file_id][j:j + N]: # eğer main_N listesi, kaynak kelime listesinin ardışık N kelimesine eşitse
                    for k in range(N): # N kadar döngü şeklinde her kelimeyi işaretliyoruz
                        main_gui[id + k][1] = True # main_text'in arayüz için kullanılacak dönüştürülmemiş halini içeren listede, kelimenin 1. indisi True değerini(intin
                        main_gui[id + k][2] = [file_id, j + k] # main_text'in arayüz için kullanılacak dönüştürülmemiş halini içeren listede, kelimenin 2. indisi, intih
                        source_gui[file_id][j + k][1] = True # kaynak kelime listesinin arayüz için kullanılacak dönüştürülmemiş halini içeren listede, kelimenin 1. ind
                    main_N.pop(0) # main_N listesinin ilk elemanını çıkarıyoruz, böylece yeni kelime eklenebilir hale geliyor
    return [html_format(i) for i in range(-1, len(source_gui))]

```

```

def html_format(fileid):
    if fileid == -1:
        main_html = []
        text_len = len(main_text)
        main_html.append("<p> ")
        plag_count = 0
        word_count = 0
        global plag_ratio
        for word in range(text_len):
            if main_gui[word][1] == False:
                main_html.append(main_gui[word][0] + " ")
                word_count += 1
            else:
                main_html.append('<a href="#w{}-{}" style="color:red;">{}'.format(main_gui[word][2][0]+1, main_gui[word][2][1], main_gui[word][0]))
                plag_count += 1
                word_count += 1
            if main_gui[word][2] == True:
                main_html.append("</p><p>")
        plag_ratio = (plag_count / word_count) * 100
        return "".join(main_html) # metnin başına ve sonuna <p> tagı ekler
    else:
        comp_text = []
        lenn = len(source_gui[fileid])
        comp_text.append("<h2>#{} - {}</h2> <p>".format(fileid + 1, file_dict[fileid]))
        for word in range(lenn):
            if source_gui[fileid][word][1] == 0:
                comp_text.append(source_gui[fileid][word][0] + " ")
            elif source_gui[fileid][word][1] == 1:
                comp_text.append('<span id="w{}-{}" style="color:red;">{}</span>'.format(fileid+1, word, source_gui[fileid][word][0]))
            if source_gui[fileid][word][2] == 1:
                comp_text.append("</p><p>")
        return "".join(comp_text)

```

Projenin canlı versiyonunu denemek için <https://plgtest.vercel.app/> adresini ziyaret edebilirsiniz.