Yazılım Laboratuvarı 1

Proje 2

.

Umut SÜTCÜ 200202038

Bilgisayar Mühendisliği 3.sınıf

Özet

Bu doküman yazılım laboratuvarı 1 dersinin 2.projesi olan Müşteri şikayetleri kayıtlarının tutulduğu bir veri seti içerisindeki benzer kayıtlar tespit edilecek ve tespit edilen kayıtlar masaüstü uygulamasında gösterilecektir. Multithreading kullanarak benzerlik arama süresini düşürmek amaçlanmaktadır. Bu sayede bilgisayarın bütün gücünden faydanlanarak işlemlerimizi daha kısa bir sürede biterebileceğiz

Projenin Tanımı

Multithreading (Çok İş Parçacıklı Çalışma): Multithreading (çok iş parçacıklı çalışma), bir merkezi işlem biriminin (CPU) (veya çok çekirdekli bir işlemcideki tek bir çekirdeğin) aynı anda işletim sistemi tarafından desteklenen birden çok yürütme iş parçacığı sağlama yeteneğidir. Bu tür programlamada birden çok iş parçacığı aynı anda çalışır. Çok iş parçacıklı model, sorgulamalı olay döngüsü kullanmaz. CPU zamanı boşa harcanmaz. Boşta kalma süresi minimumdur. Daha verimli programlarla sonuçlanır. Herhangi bir nedenle bir iş parçacığı duraklatıldığında, diğer iş parçacıkları normal şekilde çalışır..

Veri seti: Kullanılacak veri setine aşağıdaki linkten ulaşabilirsiniz: https://www.kaggle.com/datasets/selener/consumer-complaint-database Bu veri seti; finansal ürünler ve hizmetler hakkında alınan gerçek dünya şikayetlerini içermektedir. Veri seti, müşterilerin Kredi Raporları, Öğrenci Kredileri, Para Transferi vb. gibi finans sektöründeki birden fazla ürün ve hizmet hakkında yaptığı şikayetlerin farklı bilgilerini içermektedir. Veri seti aşağıdaki kurallara uygun olacak şekilde yeniden düzenlenmelidir: Elde edilen tabloda 6 farklı sütun bulunmalıdır: Product (Ürün), Issue (Konu), Company (Şirket), State, Complaint ID, Zip Code. Null değer içeren kayıtlar bulunmamalıdır. Kayıtlardaki noktalama işaretleri kaldırılmalıdır. Kayıtlardaki stop word'ler kaldırılmalıdır (nltk kütüphanesi kullanılabilir).

Benzerlik Tespiti: Geliştirilecek projede tüm kayıtlar arasındaki benzerlik ilişkisinin incelenmesi beklenmektedir. Bu nedenle her bir kaydın diğer bir kayıtla karşılaştırılması gerekmektedir. Karşılaştırmanın mümkün olduğunca hızlı olması için multithread kullanılmalıdır. Benzerlik, kayıtların içerdikleri ortak kelime sayısına göre olmalıdır. Örneğin; ilk kayıt 5, ikinci kayıt 4 kelimeden oluşuyorsa ve ortak kelime sayısı 2 ise benzerlik oranı; 2/5 den yüzde(%) 40'dır Kayıtlar arasında olabilecek benzerlik oranları için örnek tablo aşağıda verilmiştir.

Debt collection Debt collection %100

Debt collection Mortgage %0

Managing loan lease Problems end loan lease %25

Araştırma ve Yöntemler

İlk önceliğim olarak kaggele'daki veriyi en iyi hangi dilde işleyebileceğimi araştırmak oldu. Bu araştırmamdan sonra python'ın büyük verilerle işlemler ile ilgili bir çok kütüphanesi olduğunu fark ettim ve dil olarak python dilini seçtim. Python dilini seçtikten sonra pythonda tecrübem olmadığından python öğrenmeye başladım ve öğrendiklerimi deneyebilmek için küçük çaplı projeler yaptım

Bunlardan sonra veriyi düzenlemek için neler yapacağıma bakmak oldu. Veriyi aldığımız yer olan kaggle da bu konular hakkında birçok kaynak buldum. Bu kaynakları kendi projeme göre uyarlamakla başladım. Bu sayede 800 mb olan veriyi 53 mb a kadar düşürebildim.

```
data = data.drop('Sub-product',axis=1)
data = data.drop('Sub-issue',axis=1)
data = data.drop('Consumer complaint narrative',axis=1)
data = data.drop('Company public response',axis=1)
data = data.drop('Tags',axis=1)
data = data.drop('Consumer consent provided?',axis=1)
data = data.drop('Submitted via',axis=1)
data = data.drop('Consumer disputed?',axis=1)
data = data.drop('Company response to consumer',axis=1)
data = data.drop('Date sent to company',axis=1)
data = data.drop('Timely response?',axis=1)
data = data.drop('Date received',axis=1)

data1 = data.dropna()

data1['Product']=[re.sub('[^\w\s]+', '', s) for s in data1['Product'].tolist()]
data1['Issue']=[re.sub('[^\w\s]+', '', s) for s in data1['Issue'].tolist()]
data1['Company']=[re.sub('[^\w\s]+', '', s) for s in data1['Company'].tolist()]
data1['State']=[re.sub('[^\w\s]+', '', s) for s in data1['State'].tolist()]
data1['ZIP code']=[re.sub('[^\w\s]+', '', s) for s in data1['ZIP code'].tolist()]
```

Verileri düzenlemek için kullandığım kodun bir parçası

Verileri düzenlerken birçok hatayla karşılaştım. Bu hataların en başında da boyut sorunu geliyordu. Csv dosyasının hepsini okumuyordu bu yüzden verilerin hepsini değil çok daha küçük bir kısmını alıyordu.

```
(1282355, 18)
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1282355 entries, 0 to 1282354
Data columns (total 18 columns):
 #   Column                        Non-Null Count    Dtype
---  ------                        --------------    -----
 0   Date received                 1282355 non-null  object
 1   Product                       1282355 non-null  object
 2   Sub-product                   1047189 non-null  object
 3   Issue                         1282355 non-null  object
 4   Sub-issue                     751169 non-null   object
 5   Consumer complaint narrative  383564 non-null   object
 6   Company public response       449082 non-null   object
 7   Company                       1282355 non-null  object
 8   State                         1262955 non-null  object
 9   ZIP code                      1167057 non-null  object
 10  Tags                          175643 non-null   object
 11  Consumer consent provided?    690654 non-null   object
 12  Submitted via                 1282355 non-null  object
 13  Date sent to company          1282355 non-null  object
 14  Company response to consumer  1282348 non-null  object
 15  Timely response?              1282355 non-null  object
 16  Consumer disputed?            768501 non-null   object
 17  Complaint ID                  1282355 non-null  int64
dtypes: int64(1), object(17)
memory usage: 176.1+ MB
```

Verilerin düzenlemeden önceki hali

Verileri düzenlemeden önce toplamda 18 sutun ve 1282355 tane de satır vardı. Bunların boyutu da 180 mb civarındaydı.

```
(1167023, 6)
<class 'pandas.core.frame.DataFrame'>
Int64Index: 1167023 entries, 0 to 1282354
Data columns (total 6 columns):
 #   Column        Non-Null Count    Dtype
---  ------        --------------    -----
 0   Product       1167023 non-null  object
 1   Issue         1167023 non-null  object
 2   Company       1167023 non-null  object
 3   State         1167023 non-null  object
 4   ZIP code      1167023 non-null  object
 5   Complaint ID  1167023 non-null  int64
dtypes: int64(1), object(5)
memory usage: 62.3+ MB
```

Verilerin düzenledinkten sonraki hali

Verileri düzenledikten sonra 18 sutun sayısını istenen sutun sayısına yani 6 sutuna düşürdüm ve stopword ve noktalama işaretlerini de kaldırdıktan sonra total verinin boyutu 60 mb civarına kadar düştü.

```
def dongu(a, b, isim, Benzerlik):
    with time_it('thread'):
        count = 0
        for i in range(a, b):
            ilist = data[isim][i].split(" ")
            for k in range(a, b):
                count += 1
                klist = data[isim][k].split(" ")
                if (len(ilist) > len(klist)):
                    if(len(list(set(ilist) & set(klist)))/len(ilist)*100 > Benzerlik ):
                        oran_list.append(len(list(set(ilist) & set(klist)))/len(ilist)*100)
                        birinci_list.append(ilist)
                        ikinci_list.append(klist)
                elif (len(ilist) < len(klist)):
                    if(len(list(set(ilist) & set(klist)))/len(klist)*100 > Benzerlik ):
                        oran_list.append(len(list(set(ilist) & set(klist)))/len(klist)*100)
                        birinci_list.append(ilist)
                        ikinci_list.append(klist)
                else:
                    if(len(list(set(ilist) & set(klist)))/len(ilist)*100 > Benzerlik ):
                        oran_list.append(len(list(set(ilist) & set(klist)))/len(ilist)*100)
                        birinci_list.append(ilist)
                        ikinci_list.append(klist)
```

Verileri karşılaştırmak için kullandığım arama metodu

Verileri karşılaştırmak için dataframe olan veri tipini list haline çevirmem gerekiyordu. Bu sorun beni en çok zorlayan sorunlardan biriydi. Aldığım hataları sürekli internete arayarak çözümlerini bulmaya çalışıyordum. Bu şekilde araştırmam bana çok fazla bilgi ve tecrübe kazandırmıştı. Bu şekilde devam ederek burada yaşadığım hataların bir çoğunu çözebildim.

```
def com_Multithread(thread_sayisi,isim,Benzerlik,id):
    with time_it('main Thread'):
        a = 100
        b = 0
        with ThreadPoolExecutor(thread_sayisi) as ex:
            for _ in range(10):
                b += a
                x = b-a
                if(id==0):
                    tanim = [x, b, isim, Benzerlik]
                    ex.submit(dongu, *tanim)
                else:
                    tanim = [x, b, isim, Benzerlik,id]
                    ex.submit(com_dongu, *tanim)

    raise_frame(window)
    BilgiEkrani()
```

Multithreading için kullandığım kod

Multithreading benim için çok fazla sorun oldu çünkü Multithreading pythonda düzgün bir şekilde çalışmıyordu. thread sayısının işlem hızına hiçbir katkısı olmuyordu. Pythonda bulunan GIL yüzüden threadler düzgün bir şekilde çalışıyordu. Bu hata için uzun süre araştırdıktan sonra Multiprocessing buldum. Bu işlem hızını gerçekten arttırıyordu. Ama hocamız Multithreading istediği için o şekilde bıraktım.

```python
def BilgiEkrani():
    y = 1
    for i in range(len(zaman_list)):
        lb1 = Label(window, text=zaman_list[i])
        lb1.grid(column=3, row=y)
        y += 1


    lb1 = Label(window, text="Benzerlik Oranı")
    lb1.grid(column=0, row=0)
    lb1 = Label(window, text="Kayıt 1")
    lb1.grid(column=1, row=0)
    lb1 = Label(window, text="Kayıt 2")
    lb1.grid(column=2, row=0)
    lb1 = Label(window, text="Thread süreleri")
    lb1.grid(column=3, row=0)

    y = 2
    sayı= len(oran_list)
    for i in range(150):
        lb1 = Label(window, text=oran_list[i])
        lb2 = Label(window, text=birinci_list[i])
        lb3 = Label(window, text=ikinci_list[i])
        lb1.grid(column=0, row=y)
        lb2.grid(column=1, row=y)
        lb3.grid(column=2, row=y)
        y += 1
```

Arayüzün bilgi ekranını tasarlamak için kullandığım kod

```python
def arayuz():
    lb1 = Label(window2, text="Kayıt: ")
    lb1.grid(column=0, row=0)
    kayit = Entry(window2)
    kayit.grid(column=1,row=0)
    lb2 = Label(window2, text="Thread Sayisi: ")
    lb2.grid(column=0, row=1)
    j = Entry(window2)
    j.grid(column=1,row=1)
    lb2 = Label(window2, text="Benzerlik Oranı: ")
    lb2.grid(column=0, row=2)
    k = Entry(window2)
    k.grid(column=1,row=2)
    lb3 = Label(window2, text="Complaint Id: ")
    lb3.grid(column=0, row=3)
    z = Entry(window2)
    z.grid(column=1,row=3)

    Button(window2, text='Onayla', command=lambda:com_Multithread(int(j.get()),kayit.get(),int(k.get()),int(z.get()))).grid(column=0, row=4)
```

Arayüzün giriş ekranını tasarlamak için kullandığım kod

Arayüz yapmak ile çok bilgim olmadığı için bütün projede almadığım kadar hatayı burada aldım. Ama bunları çok hızlı bir şekilde atlarak bir arayüz tasarlayabildim.

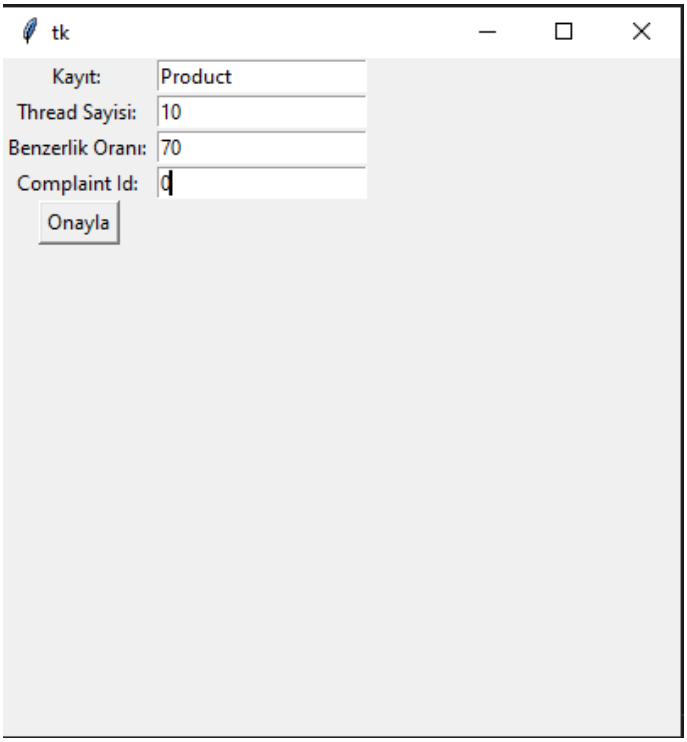
KOD BILGISI(YALANCI KOD)

Kodu txt dosyası olarak ek olarak gönderdim.

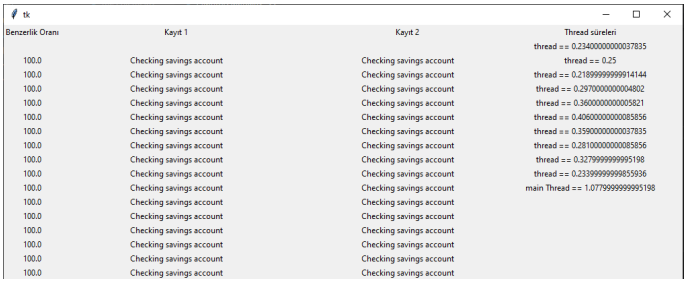
SONUÇ

Sonuç olarak python ile Multithreading yaparak büyük veri üzerinde işlem yapabilmeyi ve bu verilerin nasıl işleneceğini öğrenmiş oldum. Bu işlediğim veriyi kullanıcıya düzgün bir şekilde sunmak için de basit de olsa bir arayüz tasarlamayı bu arayüzde girilen değerlerin kodun içinde çalışma mantığını anlamış oldum.

Aşağıda da projeme ait ekran görüntüleri:

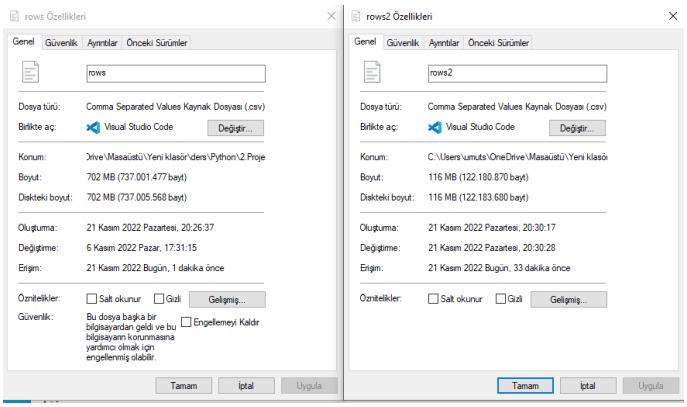


Projenin giriş ekranı



Pronin arama yaptıktan sonraki çıktı ekranı

Çıktı ekranında giriş ekranında istenen verilerin gösterilmesini sağlıyorum.



Verinin ilk ve son halinin bir ekran görüntüsü

KAYNAKÇA

https://www.kaggle.com/datasets/selener/consumer-complaint-database

https://mertmekatronik.com/thread-ve-multithread-nedir

https://www.w3schools.com/python/

https://www.geeksforgeeks.org/multithreading-python-set-1/

https://www.geeksforgeeks.org/python-programming-language/?ref=shm

https://docs.google.com/document/d/18CXhDb1ygxg-YXNBJNzfzZsDFosB5e6BfnXLlejd9l0/edit

https://github.com/tensorflow/swift/blob/master/docs/WhySwiftForTensorFlow.md

https://www.youtube.com/watch?v=gOtzf-Yhn-k

https://docs.python.org/3/library/tkinter.html