BLM307 YAZILIM LABORATUVARI – I

PROJE 2

Barış Kakilli

Kocaeli Üniversitesi

Bilgisayar Mühendisliği

200201012@kocaeli.edu.tr

Muhammed Sina Çimen

Kocaeli Üniversitesi

Bilgisayar Mühendisliği

200201032@kocaeli.edu.tr

I.ÖZET

Bu projede amaç elimizde bulunan müşteri şikayetleri kayıtlarının tutulduğu bir veri seti(18 sütundan oluşan bu veri seti için sadece belirli 6 sütun için işlem yapılmıştır.) içerisindeki benzer kayıtlar tespit etmek ve tespit edilen kayıtlar masaüstü arayüz uygulamasında gösterilmesidir. Multithreading kullanarak benzerlik arama süresini düşürmek amaçlanmaktadır.

II. GİRİŞ

Bilgisayar mimarisinde, MultiThreading (çok iş parçacıklı), 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 yaklaşım, çoklu işlemden farklıdır. MultiThreading (çok iş parçacıklı) bir uygulamada, iş parçacıkları, hesaplama birimlerini, CPU önbelleklerini ve çeviri ön tampon tamponunu (TLB) içeren tek veya çok çekirdeğin kaynaklarını paylaşır. Çok işlemcili sistemler, bir veya daha fazla çekirdekte birden çok tam işlem birimi içerdiğinde, çok iş parçacıklı, iş parçacığı düzeyinde paralellik ve aynı zamanda talimat düzeyinde paralellik kullanarak tek bir çekirdeğin kullanımını artırmayı amaçlar. Bu iki teknik birbirini tamamlayıcı nitelikte olduğundan, bazen çoklu çok iş parçacıklı CPU'lara ve birden fazla çok iş parçacıklı çekirdeğe sahip CPU'lara sahip sistemlerde birleştirilirler.

Multithreading aynı anda birden fazla iş parçacığı çalıştırmanıza izin verir. Çok çekirdekli bir makinede bu, iki iş parçacığının gerçekten paralel olarak çalışabileceği ve her seferinde bir tane çalıştırdıklarının iki katı işi yapabileceği anlamına gelir. İdeal olarak, 4 çekirdekli bir makinede, 4 iş parçacığı ile, tek bir iş parçacığına kıyasla neredeyse 4 kat daha fazla iş yaparsınız. Bunun işe yaraması için, birbirinden bağımsız çalışan birden çok iş parçacığı ile çözülebilecek bir soruna ihtiyacınız var. Programı nasıl iş parçacığına

böleceğinizi bulmak için oldukça zeki olmanız gerekir. Ve çoğu zaman, bu iş parçacıklarının birbirlerinin verilerini çöpe atmasını (veya daha kötüsü, kurnazca sabote etmesini) engellemek için gerçekten zeki olmanız gerekir. Bazen konuları aynı programın farklı oturumları gibi neredeyse bağımsız olarak çalıştırmak mümkün olsa da, bunu yapabildiğiniz zaman güzel bir şeydir.

III. YÖNTEM

Program "Python" programlama dilinde yazılmış olup IDE olarak "Visual Studio Code" ve "Pycharm" kullanılmıştır. Veri manipülasyonu için "panda" kütüphanesi ve görsel arayüz gösterimi için "PyQt5" kütüphanesi, bunun yanında pek çok küçük işlevi gerçekleyebilmek için pek çok farklı kütüphaneden yararlanılmıştır. Programın çalışabilirliği "Windows ˇ 10 64 bit" ve "Windows 11 64 bit" işletim sistemlerinde test edilmiştir. Proje bir ekip tarafından geliştirildiğinden dolayı işlevlerin birleşimi, kodun takibi gibi amaçlarla "Github" versiyon kontrol sistemi kullanılmıştır.

Öncelikle biz bu projede Python'da olan bir kilitten dolayı multithread gerçekleyemediğimiz için hocalarımızın onayıyla multiprocess kütüphanesini kullandık. Projeyi tekrardan tanımlamamız gerekirse bu projedeki amaçlar:

1. Veri seti içerisindeki arama işlem süresini multithreading kullanılarak azaltmak.

2. Belirtilen sütun/sütunlar için her bir satırdaki kayıtların birbiriyle kelime bazlı

karşılaştırılması ve aralarındaki benzerliğin tespit edilmesi.

3. Uygulama içerisinde istenen özelliklere göre kayıtları filtrelemek ve kullanıcıya göstermek.

4. Masaüstü uygulama geliştirme hakkında bilgi ve beceriye sahip olmak.

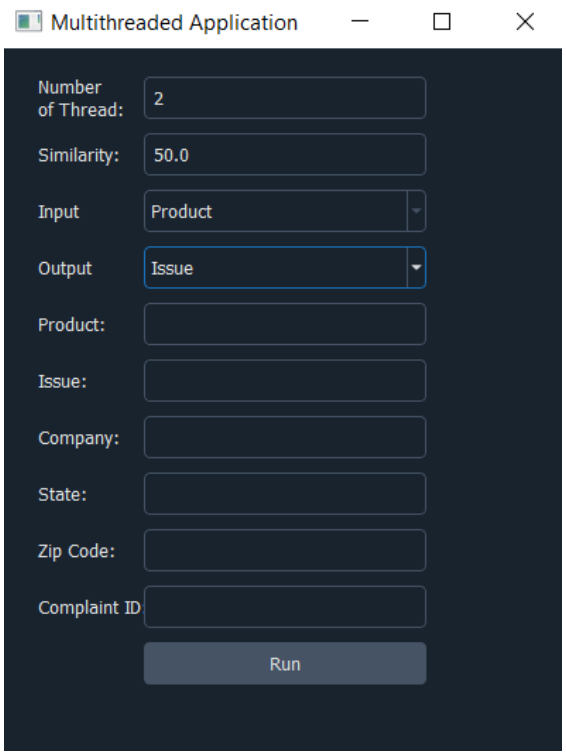
Projede ilk gerçekleştirdiğimiz eylem bu büyük veri setimizi organize etmek.

Proje ilk çalıştığında bizi kullanıcının yapmak istediği işlemi seçmesi gereken bir arayüz karşılıyor. Bu arayüz 18 sütun arasından seçtiğimiz 6 sütun için belli başlı istekleri seçmemize imkân tanıyor. Bu seçilen sütunları biz önceden başka bir kod kısmında bazı kıstaslara göre(NULL değer içeren sütunları kaldırma, noktalama işaretlerini kaldırma, stop wordleri[and, or vb..] kaldırma) kaldırdık.

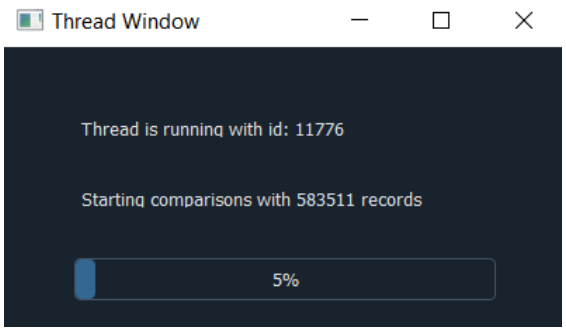
Ardından bu verilerin sütunlarını distinct bir şekilde elde ettik. Bu elde ettiğimiz verileri farklı bir csv dosyasına taşıdık.

Yukarı kısımda arayüzümüzü tanımlamıştık. Bu arayüzden kullanıcının girdiği verilere göre ilk olarak process sayısını belirliyoruz. Ve arayüze girilen senaryo isteğine göre seçim yapıyoruz. Bu önişlemden sonra oluşan dataframe'i ürünleri birbiriyle karşılaştırdığımız algoritmaya(döngü) sokarak gerekli verileri elde ediyoruz. Bu elde ettiğimiz verileri bir listeye aktardıktan sonra arayüzde kullanıcıya sunuyoruz.

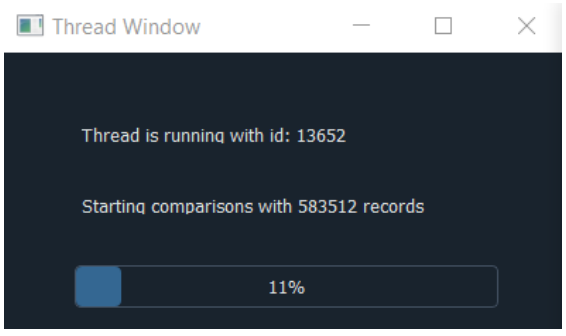
IV. DENEYSEL SONUÇLAR



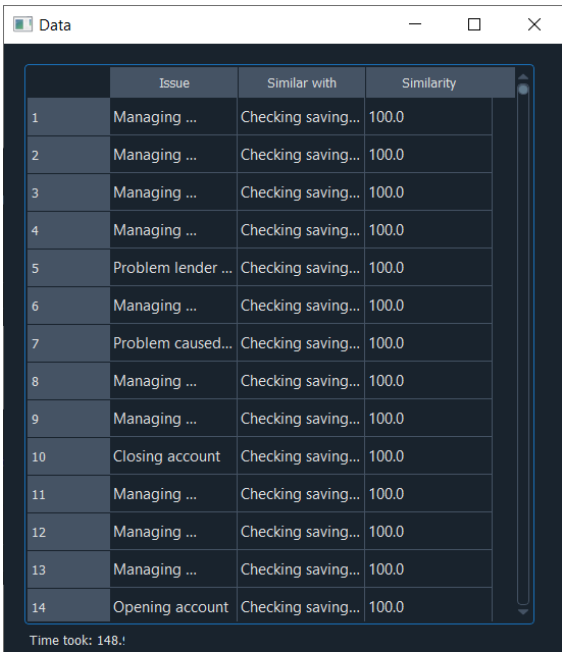
Şekil 1 Örnek Arama



Şekil 2 1.thread



Şekil 3 2.thread



Şekil 4 Sonuç Ekranı 1.örnek

V. SONUÇ

Proje kapsamında bizden istenen tüm amaçları yerine getirdik ve projemizi pek çok farklı senaryo testinden geçirip başarılı bir sonuca ulaştık.

VI. YALANCI KOD

0. BAŞLA

1. Kütüphaneleri import et

2. Belirli sütunları elde et

3. Null değerleri kaldır

4. Noktalama işaretlerini kaldır

5. Stop word'leri kaldır

6. Tüm sütunların distinct verisini al

7. Başka bir csv dosyasına aktar

8. Arayüzü göster ve processleri içeren programı başlat

9. Girilen process sayısına göre processi oluştur

10. Senaryoya göre isterleri düzenle

11. Oluşan tabloyu karşılaştırma döngüsüne koy

12. Gelen veriyi bir listede kaydet

13. Listeyi arayüzde görüntüle

VII. KAYNAKÇA

1) https://docs.python.org/3/library/index.html
2) https://stackoverflow.com
3) https://www.geeksforgeeks.org
4) https://mertmekatronik.com/thread-ve-multithread-nedir
5) https://www.pythonguis.com/pyqt5/
6) https://www.pythonguis.com/tutorials/creating-multiple-windows/
7)