YAZILIM LABORATUVARI II

PROJE III :

GRAF TABANLI METİN ÖZETLEME PROJESİ

SENA ÖKTEM

Kocaeli Üniversitesi

Mühendislik Fakültesi(İÖ)

190202054@kocaeli.edu.tr

ÖZET:

Projede bir masaüstü uygulaması geliştirilmesi istenmektedir. Masaüstü uygulamada ilk olarak doküman yükleme işlemi gerçekleştirilecektir. Ardından yüklenen dokümandaki cümlelerin graf yapısı haline getirilmesi ve bu graf yapısının görselleştirilmesi istenmiştir. Bu grafta her bir cümle bir düğümü temsil edecektir. Cümleler arasındaki anlamsal ilişkinin kurulması ve cümlelerin skorlanması gerekmektedir. Belirli parametreleri kullanarak cümle skorunun hesaplama algoritmasını ve cümle skorlarına göre metin özeti çıkarma algoritmaları kendi kurduğumuz algoritmalardan oluşturulmuştur. Kurulan algoritmalar sonucu özet metin kullanıcıya arayüzde sunulmaktadır. Sonuç olarak ilgili dökümandaki bir metnin özetinin bu yöntem ile çıkartılması gerekmektedir ve kullanıcı arayüzünden girilen gerçek özet metin ile benzerliğinin "ROUGE" skorlaması ile ölçülmesi gerekmektedir.

1. GİRİŞ:

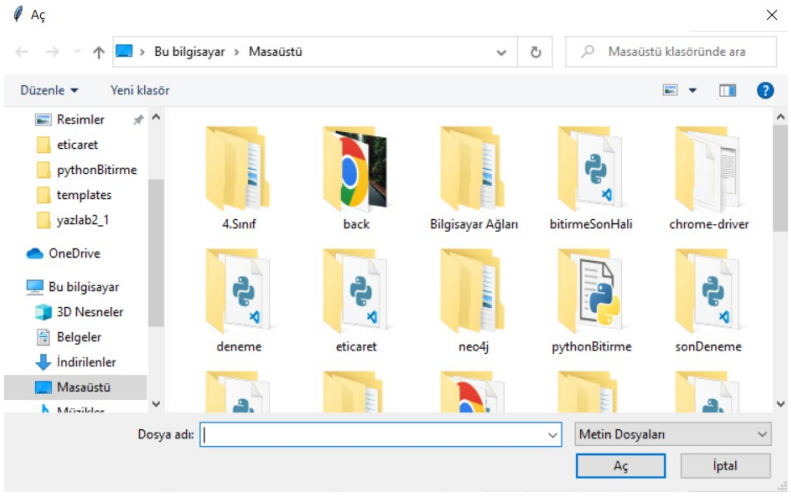
Projenin algoritması Visual Studio kullanarak Python dilinde yazılmıştır. Veri tabanı olarak neo4j graf veri tabanı kullanılmıştır. Veri tabanı, arayüz ve algoritma ile ilgili detaylar ilerleyen sayfalarda detaylandırılmıştır.

2. YÖNTEM, ARAYÜZ VE ALGORİTMA

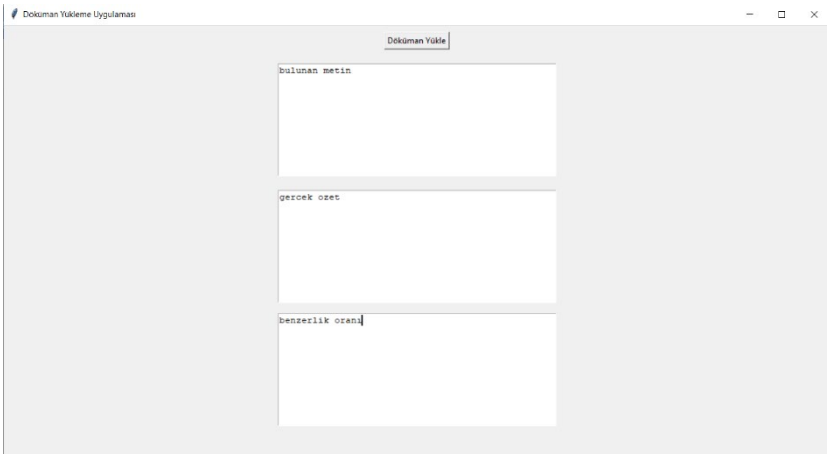
Projede bir masaüstü arayüzü kullanıcıya sunulmuştur. Sunulan arayüzden özetin çıkartılması istenen metnin seçilebilmesi için bir dosya yükleme alanı konmuştur. Yükleme yapıldıktan sonra gerçek özetle benzerliğin hesaplanabilmesi için gerçek özetin yazılacağı textbox oluşturulmuştur. Hesaplama sonucunda ortaya çıkartılan özet ve bulunan özet ile gerçek özetin benzerliğini veren textbox'lar oluşturulmuştur.

1. ARAYÜZ

Sunulan arayüzden özetin çıkartılması istenen metnin seçilebilmesi için bir dosya yükleme alanı konmuştur. Doküman yükleme butonuna tıklandığında aşağıdaki gibi bir dosya seçim ekranı açılmaktadır.

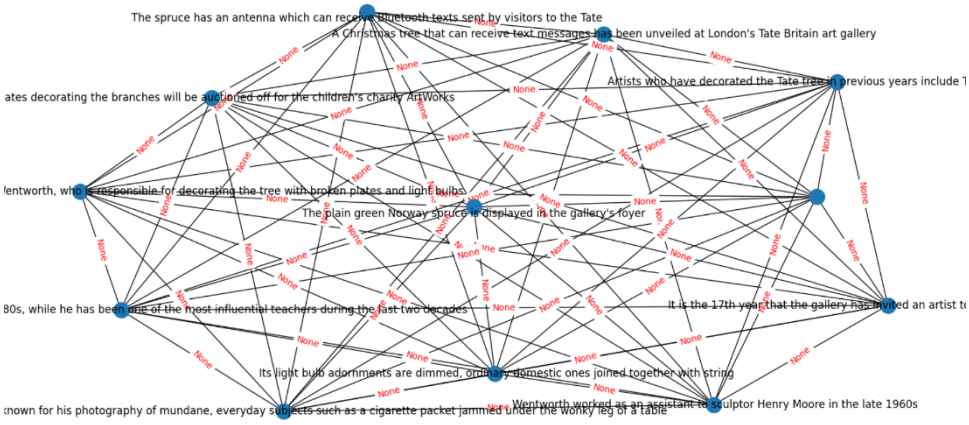


Yükleme yapıldıktan sonra gerçek özetle benzerliğin hesaplanabilmesi için gerçek özetin yazılacağı ikinci textbox oluşturulmuştur. Benzerlik oranı textboxı ve bulunan metin textboxına giriş yapılmamalıdır. Hesaplamalar sonucunda bu iki textboxın doldurulması sağlanacaktır.



Hesaplamalar sonucunda bulunan metin textboxı ve benzerlik oranı kutucuğu algoritmalar tarafından hesaplanarak kullanıcıya sunulmuştur.

Bulunan sonuçlar bir grafta görselleştirilmiştir.



2. VERİ TABANI

Projede veritabanı olarak neo4j graf veritabanı kullanılmıştır.

NEO4J :

Neo4j, bir graf veritabanı yönetim sistemidir. Graf veritabanları, verileri düğüm (node) ve kenar (edge) olarak adlandırılan yapılarla temsil ederler. Neo4j, graf veritabanı yönetim sistemi olarak kullanılarak ilişkisel veritabanlarına göre daha karmaşık ve ilişkili verileri depolamak ve sorgulamak için kullanılır.
Neo4j, çeşitli alanlarda kullanılmaktadır:

1. Sosyal Ağ Analizi: Neo4j, sosyal ağlar gibi karmaşık ilişkisel veri yapılarıyla çalışmak için idealdir. Kullanıcılar, ilişkiler ve etkileşimler arasındaki ilişkileri kolayca modelleyebilir ve sosyal ağ analizleri yapabilir.
2. Öneri Sistemleri: Neo4j, kullanıcı tercihlerini ve ilişkileri temsil etmek için kullanılabilir. Bu, öneri sistemlerinin geliştirilmesi ve kişiye özelleştirilmiş önerilerin sunulması için etkili bir yöntem sağlar.
3. Kimlik ve Erişim Yönetimi: Neo4j, kullanıcıların rolleri, izinleri ve erişim haklarını temsil ederek kimlik ve erişim yönetimi sistemlerinin geliştirilmesini sağlar. Kullanıcıların yetkilendirme süreçlerini ve roller arasındaki ilişkileri kolayca yönetebilirsiniz.
4. Ağ ve Altyapı Yönetimi: Neo4j, ağ yapıları, sistemler ve altyapı bileşenleri gibi karmaşık ağ yapılarının yönetimi için kullanılabilir. İlişkileri temsil etmek ve ağ topolojilerini analiz etmek için kullanılarak sorun tespiti, kapasite planlaması ve ağ yönetimi gibi işlemler gerçekleştirilebilir.

Neo4j'nin avantajları şunlardır:

1. Yüksek Performans: Neo4j, graf veri modeline özel olarak tasarlanmıştır ve graf verilerin depolanması ve sorgulanması için optimize edilmiştir. Bu sayede, karmaşık ilişkileri hızlı bir şekilde analiz etmek ve sorgulamak mümkündür.
2. Esnek Veri Modeli: Graf veri modeli, ilişkileri ve bağlantıları doğrudan temsil ettiği için verileri daha doğru ve anlamlı bir şekilde modellemeyi sağlar. İlişkiler ve bağlantılar üzerinde kolayca gezinmek ve sorgulamalar yapmak mümkündür.
3. Kolay Entegrasyon: Neo4j, yaygın programlama dilleri ve teknolojilerle kolayca entegre olabilir. API'leri ve sürücüleri sayesinde farklı uygulamalarla veri paylaşımı ve entegrasyonu kolayca gerçekleştirilebilir.

4. Gerçek Zamanlı Analiz: Neo4j, graf veritabanı yapısıyla gerçek zamanlı analizleri destekler. Verilerin hızlı bir şekilde güncellenmesi ve sorgulanması sayesinde anlık bilgilere erişim sağlanabilir.
5. Veri Yoğun İşlemler: Neo4j, büyük miktardaki veriyi etkili bir şekilde işlemek için tasarlanmıştır. Graf veritabanları, çok sayıda düğüm ve kenar içeren büyük veri kümeleme yapılarını yönetebilir.

3. ALGORİTMA

CÜMLELER ARASI ANLAMSAL İLİŞKİNİN KURULMASI

Metin Ön İşleme Adımları:

NLTK kütüphanesi kullanılarak aşağıdaki ön işleme adımları uygulanmaktadır.

Tokenization: Bir metnin küçük parçalara ayrılmasıdır.

Stemming: Kelimelerin kökünün bulunması işlemidir.

Stop-word Elimination: Bir metindeki gereksiz sözcükleri çıkarma işlemidir. Stop word'ler, genellikle yaygın olarak kullanılan, ancak metnin anlamını belirlemekte önemli bir rol oynamayan kelime ve ifadelerdir.

Punctuation: Cümledeki noktalama işaretlerinin kaldırılmasıdır.

Punctuation işlemi aşağıdaki fonksyon ile gerçekleştirilmiştir.

```
def NOKTALAMA_KALDIR(text):
    # Metindeki noktalama işaretlerini kaldır
    no_punct = "".join([char for char in text if char not in string.punctuation])
    return no_punct
```

Diğer işlemler tek bir fonksyon altında denetlenmiştir. Aşağıdaki kod ile stopword kütüphanesi yüklenerek işlemlere devam edilmiştir.

```
# Türkçe stopwords listesini indirin
nltk.download("stopwords")
stopwords = nltk.corpus.stopwords.words("turkish")

# Türkçe özel isimleri bulmak için bir dilbilgisi kaynak dosyası indirin
nltk.download("averaged_perceptron_tagger")

def KOK_BUL_STOPWORD_KALDIR_NOKTALAMA_KALDIR(metin2):
    metin =NOKTALAMA_KALDIR(metin2)
    # Sastrawi StemmerFactory nesnesini oluştur
    factory = StemmerFactory()
    stemmer = factory.create_stemmer()


    # Metni tokenlere ayır
    tokens = word_tokenize(metin, language='turkish')

    # Her kelimenin kökünü bul ve stopwords'leri çıkar
    stemmed_words = [stemmer.stem(word) for word in tokens if word.lower() not in stopwords]

    # Köklendirilmiş ve stopwords'leri çıkarılmış kelimeleri birleştir
    processed_text = ' '.join(stemmed_words)
    return processed_text
```

Bert Algoritması:

BERT (Bidirectional Encoder Representations from Transformers), doğal dil işleme (NLP) alanında kullanılan derin öğrenme tabanlı bir dil modelidir. BERT, Google tarafından geliştirilen ve 2018 yılında yayımlanan bir modeldir. Bu model, dil anlama ve doğal dil işleme görevlerinde yüksek performans sergilemek için büyük bir önceden eğitim sürecinden geçirilmiştir.
BERT, Transformer mimarisini temel alır. Transformer, dil modellerinde başarılı bir şekilde kullanılan bir sinir ağı mimarisidir ve dilin bağlamını daha iyi anlamak için dikkat mekanizmasını kullanır.
BERT, bir önceki kelimeye veya sonraki kelimelere bağımlı kalmadan bir kelimenin anlamını çıkarabilmek için biçimlendirilmiş giriş metnini kullanır. Bu özelliği sayesinde BERT, önceki dil modellerine kıyasla daha iyi bir bağlam anlayışı sergileyebilir.
BERT modeli, iki aşamalı bir önceden eğitim sürecinden geçer. İlk aşamada, büyük bir dil veri kümesi üzerinde maskeleme (masking) ve ardışık cümle tahmini (next sentence prediction) gibi görevlerle önceden eğitilir. Maskeleme görevi, cümle içindeki bazı kelimeleri maskeleyerek, modelin boşlukları doldurma yeteneğini geliştirmesini sağlar. Ardışık cümle tahmini görevi ise iki cümle arasındaki ilişkiyi anlamasını sağlar.

```python
def bert_anlamsal_iliski_ve_benzerlik(cumle1, cumle2):

    # BERT modeli ve tokenizer'ı yükleyin
    model = BertModel.from_pretrained('bert-base-uncased')
    tokenizer = BertTokenizer.from_pretrained('bert-base-uncased')

    # Cümleleri BERT için giriş formatına dönüştürün
    girdi1 = tokenizer.encode_plus(text=cumle1, add_special_tokens=True, return_tensors='pt')
    girdi2 = tokenizer.encode_plus(text=cumle2, add_special_tokens=True, return_tensors='pt')

    # Giriş verisini alın
    input_ids1 = girdi1['input_ids']
    token_type_ids1 = girdi1['token_type_ids']
    input_ids2 = girdi2['input_ids']
    token_type_ids2 = girdi2['token_type_ids']

    # Cümleleri BERT modeline göndererek özellik vektörlerini alın
    with torch.no_grad():
        outputs1 = model(input_ids1, token_type_ids=token_type_ids1)
        outputs2 = model(input_ids2, token_type_ids=token_type_ids2)

    # Özellik vektörlerini alın
    ozellik_vektorleri1 = outputs1.last_hidden_state[:, 0, :].numpy()
    ozellik_vektorleri2 = outputs2.last_hidden_state[:, 0, :].numpy()

    # İki cümle arasındaki anlamsal ilişkiyi belirleyin
    if np.argmax(ozellik_vektorleri1) == 0:
        iliski = "Cumle2, Cumle1'e göre devamıdır."
    else:
        iliski = "Cumle2, Cumle1'e göre devam etmez."

    # Kosinüs benzerliğini hesaplayın
    benzerlik = cosine_similarity(ozellik_vektorleri1, ozellik_vektorleri2)[0][0]

    return iliski, benzerlik
```

Benzerliği hesaplamak için kosinüs benzerliği kullanılmalıdır. from sklearn.metrics.pairwise import cosine_similarity kütüphanesi projeye dahil ederek yukarıdaki bert algoritması içinde kullanılmıştır.

Cümle Skoru Hesaplama Algoritmasının Geliştirilmesi:

Cümle Skoru Hesaplama sırasında aşağıdaki parametreleri oluşturulmuştur. Oluşturulan parametreler ile her bir cümle için bir cümle skoru hesaplanmıştır. Bunun için bir fonksyon yazılmıştır. Cümle skoru algoritması için belirli bir denklem dökümanda verilmediği için aşağıda belirtilen tüm parametrelerin toplanması ile bir cümle skoru hesaplama fonksyonu yazılmıştır.

Cümle özel isim kontrolü (P1) : Cümledeki özel isim sayısı / Cümlenin uzunluğu

Cümlede numerik veri olup olmadığının kontrolü (P2) :Cümledeki numerik veri sayısı / Cümlenin uzunluğu

Cümlede başlıktaki kelimelerin olup olmadığının kontrolü (P4): Cümledeki başlıkta geçen kelime sayısı / Cümlenin uzunluğu

```
def SKOR_HESAPLA(p1,p2,p3,p4,p5):
    skor = p1+p2+p3+p4+p5
    return skor
```

Skorlara Göre Metin Özetleme Algoritmasının Geliştirilmesi:

Önemli cümleler üzerinden cümle seçerek özetleme yapılmıştır. Var olan cümle yapısı bozulmadan cümleler seçilerek çıkarılıp özet elde edilecektir. Oluşan node skorlarına göre node seçip bunlar ile özet oluşturacak bir metin özetleme algoritması geliştirilmiştir. Algoritmada metin özetlenirken hangi cümlelerin hangi sıra ile seçileceği, cümle skorları kullanarak oluşturulmuştur.

```
def OZET_BUL(lst_metin, lst_skor):
    # İlk liste üzerinde sıralama yapılırken indekslerini kaydetmek için enumerate kullanılır
    skorListe_indexleri = sorted(range(len(lst_skor)), key=lambda i: lst_skor[i], reverse=True)

    # İlk liste sıralanır
    lst_skor_siralanmis = [lst_skor[i] for i in skorListe_indexleri]

    # İkinci liste de aynı indekslere göre yeniden düzenlenir
    lst_metin_siralanmis = [lst_metin[i] for i in skorListe_indexleri]

    # Alt listeleri düz metin dizilerine dönüştürme
    lst_metin_siralanmis = [' '.join(item) for item in lst_metin_siralanmis]

    en_onemli_cumleler = lst_metin_siralanmis[:int(len(lst_metin_siralanmis) / 2)]
    ozet_metin = ' '.join(en_onemli_cumleler)

    return ozet_metin
```

Özetleme Başarısının ROUGE Skoru ile Hesaplanması:

Algoritma sonucu oluşan Özet ile metnin gerçek özeti arasındaki benzerliği ROUGE skoru ile hesaplanmaktadır.ROUGE skoru, iki metnin benzerliğini ölçmek için kullanılan bir yöntemdir. Bu benzerlik genellikle referans metinde bulunan kelimelerin özetlenmiş metinde de bulunup bulunmadığına dayanır.

```
def ROUGE_BENZERLIK_HESAPLA(bulunanOzet):
    rouge = Rouge()
    gercekOzet=get_textbox_value()
    scores = rouge.get_scores(bulunanOzet, gercekOzet)
    rouge_n_score = scores[0]['rouge-1']['f']  # ROUGE-1 skorunu alabilirsiniz (diğer değerler de mevcuttur)
    return rouge_n_score
```

4. DENEYSEL SONUÇLAR

Python kullanarak vs code ile özet çıkarma projesi oluşturulmuştur.

Neo4j graf veritabanı kullanarak verilerin veritabanına kaydedilmesi sağlanmıştır.

Python tkinter kullanarak veritabanından çekilen verinin arayüzden girilen veri ile kıyaslanması sağlanmıştır. Bu işlem sonucu bir özet çıkartılıp kullanıcıya sunulmuştur.

Doküman yükleme butonu filedialog.askopenfilename(filetypes=[("Metin Dosyaları", "*.txt")]) kullanılarak dosya seçimine imkan tanımıştır.

Bert algoritması kullanılarak bir cümleyi tamamen anlamak ve cümleyi oluşturan kelimelerin birbirleriyle olan ilişkilerini anlamak için kullanılmıştır.

NLTK kütüphanesi kullanılarak ön işleme adımları uygulanmıştır.

Algoritma sonucu oluşan Özet ile metnin gerçek özeti arasındaki benzerliği ROUGE skoru ile hesaplanmıştır.

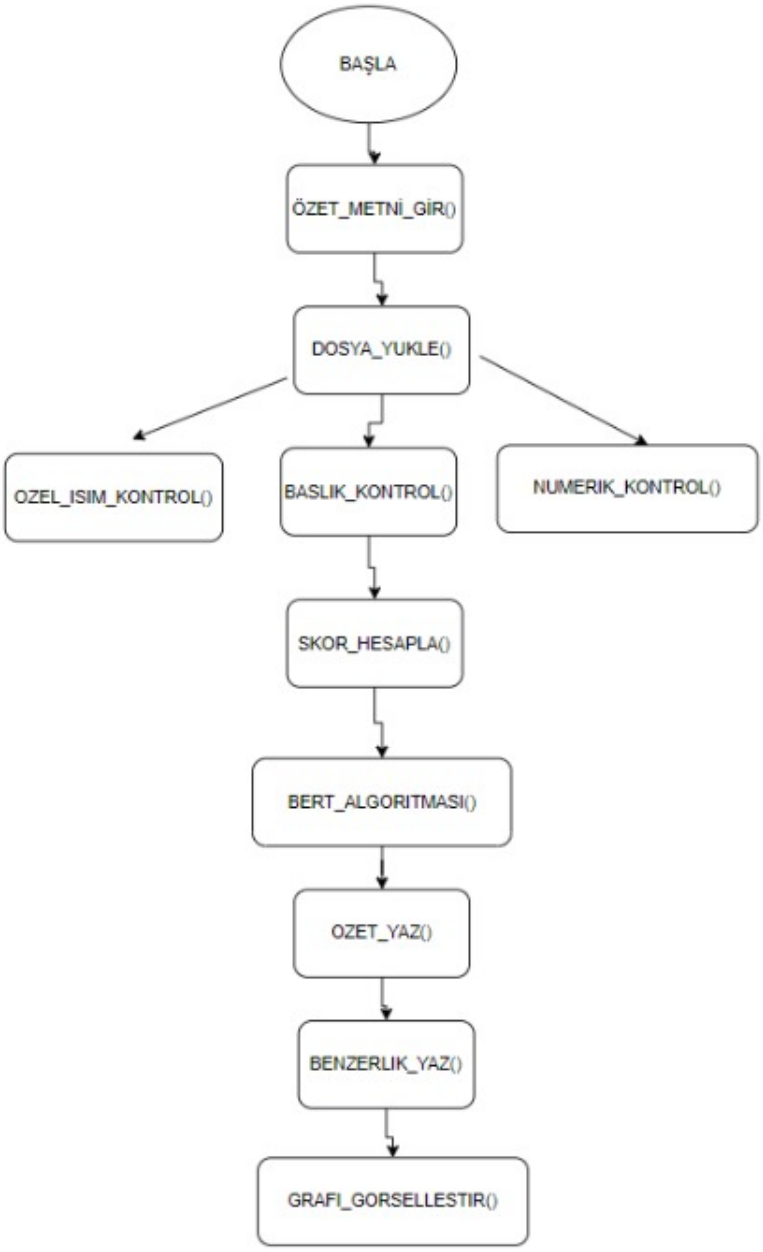
5. SONUÇ

Proje sayesinde python dili kullanılarak vscode ortamında bir masaüstü uygulamasının nasıl gerçekleştirilebileceği deneyimlenmiştir.
Neo4j graf veritabanı kullanılarak graf veritabanı mantığı anlaşılmıştır.
NLTK kütüphanesi kullanılarak metin ön işleme adımlarının nasıl olacağı hakkında bilgi edinilmiştir.
Bert algoritması kullanılarak bir cümleyi tamamen anlamak ve cümleyi oluşturan kelimelerin birbirleriyle olan ilişkilerin nasıl olacağı anlaşılmıştır.

6. KAYNAKÇA

https://www.veribilimiokulu.com/natural-language-toolkitnltk/
https://www.geeksforgeeks.org/removing-stop-words-nltk-python/
https://pypi.org/project/nltk/
https://stackoverflow.com/questions/53570495/object-has-no-attribute-when-removing-stop-words-with-nltk
Neo4j Dersleri 1. ; PC ye İndirip kurma, Çalıştırma, Docker Neo4j, Neo4j Desktop/
https://www.hosting.com.tr/blog/bert/
https://medium.com/@toprakucar/bert-modeli-ile-t%C3%BCrk%C3%A7e-metinlerde-s%C4%B1n%C4%B1fland%C4%B1rma-yapmak-260f15a65611
Tkinter Dersleri -0 - Neden GUI bilmeliyim, Neden Tkinter
PYTHON MASAÜSTÜ UYGULAMASI // PYQT5 İLE KULLANICI ARAYÜZÜ TASARIMI // QtDesigner
Displaying file browser to upload read file path in Tkinter window using filedialog askopenfilename

AKIŞ ŞEMASI


BAŞLA
ÖZET_METNİ_GİR()
DOSYA_YUKLE()
OZEL_ISIM_KONTROL()
BASLIK_KONTROL()
NUMERIK_KONTROL()
SKOR_HESAPLA()
BERT_ALGORITMASI()
OZET_YAZ()
BENZERLIK_YAZ()
GRAFI_GORSELLESTIR()