PROGRAMLAMA PROBLEMLERİ

SORU 1: Metin web sitesine gidin ve kelime sözlüğünü indirin. Bir trie veri türü oluşturun

Bu sözcük listesini bulun ve bağımsızlık bildirgesinin yazım denetimi için kullanın,

Yanlış yazılan tüm kelimeleri ekrana yazdırır.

CEVAP:

Web sitesinden kelime sözlüğü indirip, bir \*Trie\* veri yapısı oluşturmak ve yanlış yazılmış kelimeleri ekrana yazdırmak istiyorsunuz. Bu tür bir yazım denetimi yapabilmek için aşağıdaki adımları takip edebilirsiniz:

\*Kelime Sözlüğünü İndirme\*

Web sitesinden kelime sözlüğü indirmeniz gerekiyor. Web tarayıcınızda \*Metin Web Sitesi\* (belirtilen site) üzerinden kelime listesine erişebilir ve bu listeyi bir dosya olarak indirebilirsiniz. Bu liste genellikle `.txt` veya `.csv` formatında olacaktır. Kelimeleri satır satır listeleyen bir dosya olduğunu varsayalım.

\*Trie Veri Yapısını Oluşturma\*

Trie, özellikle kelimeleri verimli bir şekilde depolamak ve aramak için kullanılan bir veri yapısıdır. Kelimelerin her harfi bir düğümde temsil edilir ve kelimeler bu harflerin birleşiminden oluşan bir yol olarak depolanır.

\*Yazım Denetimi\*

Trie veri yapısında, metni tarayarak her kelimeyi kontrol edebilir ve yazım hatalarını bulabilirsiniz. Yanlış yazılmış kelimeleri bulmak için kelimenin Trie yapısında olup olmadığını kontrol edersiniz. Eğer kelime Trie’da yoksa, o kelime yanlış yazılmıştır.

Aşağıda, Python dilinde bir örnek kodu adım adım yazalım:

\*Kodun Adımları:\*

\*Adım 1: Kelime Sözlüğünü Trie Veri Yapısına Aktarma\*

```python

Class TrieNode:

Def \_init\_(self):

Self.children = {}

Self.is\_end\_of\_word = False

Class Trie:

Def \_init\_(self):

Self.root = TrieNode()

# Kelimeyi trie’ya ekleme fonksiyonu

Def insert(self, word):

Node = self.root

For char in word:

İf char not in node.children:

Node.children[char] = TrieNode()

Node = node.children[char]

Node.is\_end\_of\_word = True

# Kelimeyi trie’da arama fonksiyonu

Def search(self, word):

Node = self.root

For char in word:

İf char not in node.children:

Return False

Node = node.children[char]

Return node.is\_end\_of\_word

Kelime sözlüğünü yükle ve trie’ya ekle

Def load\_dictionary(file\_path):

Trie = Trie()

With open(file\_path, ‘r’) as f:

For line in f:

Word = line.strip().lower() # Kelimenin küçük harfe çevrilmesi

Trie.insert(word)

Return trie

```

\*Adım 2: Bağımsızlık Bildirgesini Okuma ve Yazım Denetimi Yapma\*

```python

Bağımsızlık Bildirgesini okuma ve yazım denetimi yapma

Def check\_spelling(trie, text):

Words = text.split()

Misspelled\_words = []

For word in words:

Word = word.strip(“,.?!;:”).lower() # Noktalama işaretlerini temizle ve küçük harfe çevir

İf not trie.search(word):

Misspelled\_words.append(word)

Return misspelled\_words

Bağımsızlık Bildirgesi (örnek metin)

Declaration\_text = “””

When in the Course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the Laws of Nature and of Nature’s God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

“””

Sözlük dosyasının yolunu belirtin (örneğin: ‘words.txt’)

Dictionary\_path = ‘words.txt’

Kelime sözlüğünü yükle

Trie = load\_dictionary(dictionary\_path)

Yazım denetimi yap

Misspelled\_words = check\_spelling(trie, declaration\_text)

Yanlış yazılan kelimeleri ekrana yazdır

İf misspelled\_words:

Print(“Yanlış yazılmış kelimeler:”)

For word in misspelled\_words:

Print(word)

Else:

Print(“Tüm kelimeler doğru yazılmış.”)

```

\*Açıklamalar:\*

- \*TrieNode ve Trie Sınıfı:\* `TrieNode`, her harfi ve çocuk düğümlerini tutan bir sınıftır. `Trie`, kelimeleri eklemek (`insert`) ve aramak (`search`) için gerekli fonksiyonları içerir.

- \*Dictionary Load:\* `load\_dictionary` fonksiyonu, bir dosyadan kelimeleri alır ve her kelimeyi Trie yapısına ekler.

- \*Yazım Denetimi:\* `check\_spelling` fonksiyonu, verilen metni alır, her kelimeyi kontrol eder ve doğru yazılmayanları bulur.

- \*Bağımsızlık Bildirgesi Metni:\* Bu metin örnek olarak verilmiştir ve herhangi bir metinle değiştirilebilir.

\*Dosya Yapısı:\*

- \*Kelime Sözlüğü (words.txt):\*

```txt

When

İn

The

Course

Of

Human

Events

İt

Becomes

Necessary

For

One

People

...

```

- \*Bağımsızlık Bildirgesi Metni (declaration\_text):\*

Bu metin, yukarıda verdiğimiz gibi örnek bir metin olabilir veya kendi metninizi kullanabilirsiniz.

. \*Çalıştırma:\*

Bu kodu çalıştırarak, belirtilen kelime sözlüğüyle, bağımsızlık bildirgesindeki yazım hatalarını tespit edebilirsiniz. Eğer metindeki kelime, trie’da yer almıyorsa, yazım hatası olarak kabul edilir ve ekrana yazdırılır.

. \*Geliştirme ve Optimizasyon:\*

- \*Trie’nin Hızlı Arama:\* Trie, kelime aramaları için çok hızlıdır. Bu, özellikle büyük veri kümelerinde yazım denetimi yaparken faydalıdır.

- \*Daha Karmaşık Yazım Denetimi:\* Bu temel yazım denetimi, yalnızca kelimenin var olup olmadığını kontrol eder. Gerçek dünyada, daha gelişmiş yazım denetimi, önerilen doğru yazımlar ve yakın eşleşmeleri sunabilir.

Eğer daha fazla geliştirme yapmak isterseniz, benzer şekilde kelime önerileri ve yakın eşleşme algoritmalarını ekleyebilirsiniz.

SORU 2 : Önceki alıştırmada olduğu gibi bir trie oluşturun, ancak aşağıdakiler için önerilen değişBu zor görevi gerçekleştirmek için, yanlış yazılmış kelimeleri bulduktan sonra, doğru yazım önerilerini sunmak üzere daha gelişmiş bir algoritma kullanmamız gerekecek. Bu tür bir yazım denetimi ve öneri işlevselliği için, \*Levenshtein mesafesi\* (düzenleme mesafesi) gibi algoritmalar kullanılabilir. Bu algoritma, iki kelime arasındaki farkları (ekleme, silme, değiştirme) hesaplar ve yanlış yazılmış kelimeye en yakın doğru kelimeleri önerir.

CEVAP:

\*Adımlar:\*

\*Trie Veri Yapısı\*: Kelimeleri verimli bir şekilde saklamak için Trie veri yapısını kullanacağız.

\*Yanlış Yazılmış Kelimeler\*: İlk olarak, verilen metindeki yanlış yazılmış kelimeleri bulacağız.

\*Levenshtein Mesafesi ile Öneriler\*: Yanlış yazılmış kelimeler için doğru yazım önerileri oluşturacağız. Bu, yanlış yazılmış kelimenin Trie’deki doğru kelimelerle karşılaştırılmasıyla yapılacak.

Şimdi, Python dilinde bu işlemi gerçekleştirecek olan kodu adım adım yazalım.

\*Trie Veri Yapısını Oluşturma:\*

```python

Class TrieNode:

Def \_init\_(self):

Self.children = {}

Self.is\_end\_of\_word = False

Class Trie:

Def \_init\_(self):

Self.root = TrieNode()

# Kelimeyi trie’ya ekleme fonksiyonu

Def insert(self, word):

Node = self.root

For char in word:tirmeleri de yazdırın

Tüm yanlış yazılmış kelimeler. Bu zor bir görev. Önerilen değişiklikler

Bölümde önerilen yollardan birden fazlasında orijinalinden farklı değildir.

İf char not in node.children:

Node.children[char] = TrieNode()

Node = node.children[char]

Node.is\_end\_of\_word = True

# Kelimeyi trie’da arama fonksiyonu

Def search(self, word):

Node = self.root

For char in word:

İf char not in node.children:

Return False

Node = node.children[char]

Return node.is\_end\_of\_word

# Levenshtein mesafesi ile yakın eşleşen kelimeleri önerme

Def suggest(self, word):

Suggestions = []

Self.\_find\_suggestions(self.root, word, “”, suggestions)

Return suggestions

# Kelimeyi trie’da ararken yakın eşleşmeleri bulma fonksiyonu

Def \_find\_suggestions(self, node, word, current\_word, suggestions):

İf node.is\_end\_of\_word:

Suggestions.append(current\_word)

İf not word:

For char, child\_node in node.children.items():

Self.\_find\_suggestions(child\_node, “”, current\_word + char, suggestions)

Return

First\_char = word[0]

Remaining\_word = word[1:]

İf first\_char in node.children:

Self.\_find\_suggestions(node.children[first\_char], remaining\_word, current\_word + first\_char, suggestions)

For char, child\_node in node.children.items():

İf char != first\_char:

Self.\_find\_suggestions(child\_node, remaining\_word, current\_word + char, suggestions)

```

\*Kelime Sözlüğünü Yükleme:\*

Kelime sözlüğünü yüklemek ve Trie yapısına eklemek için bir fonksiyon yazalım:

```python

Def load\_dictionary(file\_path):

Trie = Trie()

With open(file\_path, ‘r’) as f:

For line in f:

Word = line.strip().lower() # Kelimenin küçük harfe çevrilmesi

Trie.insert(word)

Return trie

```

\*Yazım Denetimi ve Öneri Sistemi:\*

Burada, metindeki her kelimeyi kontrol ederek yanlış yazılmış kelimeleri bulacağız ve her bir yanlış yazılmış kelime için önerilerde bulunacağız:

```python

Bağımsızlık Bildirgesi Metni (örnek metin)

Declaration\_text = “””

When in the Course of human events, it becomes necessary for one people to dissolve the political bands which have connected them with another, and to assume among the powers of the earth, the separate and equal station to which the Laws of Nature and of Nature’s God entitle them, a decent respect to the opinions of mankind requires that they should declare the causes which impel them to the separation.

“””

Yazım denetimi yap ve önerilerde bulun

Def check\_spelling\_and\_suggest(trie, text):

Words = text.split()

Misspelled\_words = {}

For word in words:

Word\_cleaned = word.strip(“,.?!;:”).lower() # Noktalama işaretlerini temizle ve küçük harfe çevir

İf not trie.search(word\_cleaned):

Suggestions = trie.suggest(word\_cleaned)

İf suggestions:

Misspelled\_words[word] = suggestions

Else:

Misspelled\_words[word] = [“No suggestions found”]

Return misspelled\_words

Kelime sözlüğünü yükle

Dictionary\_path = ‘words.txt’

Trie = load\_dictionary(dictionary\_path)

Yazım denetimi yap ve önerileri al

Misspelled\_words = check\_spelling\_and\_suggest(trie, declaration\_text)

Yanlış yazılmış kelimeleri ve önerilerini ekrana yazdır

İf misspelled\_words:

Print(“Yanlış yazılmış kelimeler ve öneriler:”)

For word, suggestions in misspelled\_words.items():

Print(f”Kelime: {word}, Öneriler: {‘, ‘.join(suggestions)}”)

Else:

Print(“Tüm kelimeler doğru yazılmış.”)

```

\*Levenshtein Mesafesi ile Eşleşme:\*

Levenshtein mesafesi, iki kelime arasındaki farkları belirlemek için kullanılan bir algoritmadır. Bu algoritma, edit mesafesi (ekleme, silme, değiştirme) hesaplar. Python’da bu tür bir mesafe hesaplamak için \*`python-Levenshtein`\* kütüphanesini kullanabiliriz, ancak burada \*yakın eşleşmeler için basit bir karşılaştırma\* yapıyoruz.

İlgili fonksiyonlar şunlardır:

- `suggest`: Yanlış yazılmış kelimenin Trie’deki diğer kelimelerle karşılaştırılması sonucu önerilen doğru yazımlar.

- `\_find\_suggestions`: Kelimenin her karakteri ile karşılaştırılacak alternatifler önerir.

\*Çıktı:\*

Örnek bir çıktı şöyle olabilir: Yanlış yazılmış kelimeler ve öneriler:

Kelime: dissolvve, Öneriler: dissolve

Kelime: politicall, Öneriler: political

\*Geliştirme ve İyileştirme:\*

- \*Levenshtein Mesafesi\*: Bu örnek, kelimeler için en yakın eşleşmeleri öneriyor. Ancak, daha hassas ve doğru öneriler için Levenshtein mesafesi kullanılabilir ve öneriler bir eşik değeri ile sıralanabilir.

- \*Kelime Sözlüğü Güncellenmesi\*: Gerçek dünyada, kelime sözlüğü daha geniş olmalı ve daha fazla kelimeyi kapsamalıdır.

Sonuç:

Bu çözümde, Trie veri yapısını kullanarak, yanlış yazılmış kelimeleri bulduk ve her biri için doğru yazım önerileri sunduk. Levenshtein mesafesini veya daha gelişmiş öneri algoritmalarını eklemek, önerilerin doğruluğunu artırabilir.