Proje 4 modülden oluşmaktadır. Bunlar DictionaryBase, EducationData, PMI ve WebSearch modülleridir.

DictionaryBase modülü uygulamanın çalışma mekanizmasını sağlayan modüldür. Bu modül, sürekli sözlük kelime sayısına ulaşıncaya kadar EducationData ile WebSearch modüllerini çağırır.

**DictionaryBase Modülü:**

Tüm akış DictionaryBase modülü içerisindeki RunProjectCountOfEnglishDictionary() metodu ile sağlanır.

def RunProjectCountOfEnglishDictionary():

CheckParameters()

EducationData.TruncateEnglishDictionaryTable()

while(GetWordCountOfEnglishDictionary()<thresholdEnglishDictionaryCountForStopProject):

EducationData.RunEducationDataProject()

WebSearch.WebSearch()

EducationData.findTheSuccessOfTheProject()

def CheckParameters():

EducationData.MeaningfulWordsSelectionType.type=0

EducationData.UsePMI.value=False

EducationData.TruncateEnglishDictionaryTable() metodu ile öncelikle sözlükteki tüm kelimeler silinir.

DictionaryBase modülü içerisindeki CheckParameters() metodu uygulamanın akışı için oldukça önemlidir. Çünkü bu metod ile Anlamlı kelime tespit yönteminin ne olacağına, PMI kullanılıp kullanılmayacağına karar verilir.

EducationData.MeaningfulWordsSelectionType.type=0 set edildiğinde anlamlı kelime tespiti amacıyla TFIDF kullanılacak demektir.

EducationData.MeaningfulWordsSelectionType.type=1 set edildiğinde anlamlı kelime tespiti amacıyla Helmholtz Prensibi kullanılacak demektir.

EducationData.UsePMI.value=False set edildiğinde anlamlı kelime tespiti sonrası PMI kullanılmayacak demektir.

EducationData.UsePMI.value=True set edildiğinde anlamlı kelime tespiti sonrası PMI kullanılacak demektir.

**EducationData Modülü:**

Bu modül başlangıç olarak sisteme beslenen ve Web araması sonucu elde edilen dokümanların ön işleme adımlarının, Anlamlı kelimelerinin tespiti, hangi kelimelerin sözlüğe ekleneceğine karar verildiği modüldür.

## Bağlantı Bilgileri

Programın çalışacağı bilgisayardaki mssql server bağlantı bilgileri ve database ismi bağlantı bilgilerinde düzeltilir.

connection = pypyodbc.connect('Driver={SQL Server};'

'Server=SHORTCUT;'

'Database= SmallWordsEducation;'

'uid= PhytonThesisUser;pwd=1)

cursor = connection.cursor()

## GetContentFreq

Metin de ön işlemelerin yapıldığı fonksiyon.

def GetContentFreq(content):

1-Noktalama işaretlerinin silinmesi için tanımlama yapılır.

translator = str.maketrans('', '', string.punctuation)

2-Metin kelimelerine ayrılır.

words = nltk.word\_tokenize(content)

3-Metin içinden noktalama işaretleri temizlenir.

words = [word.translate(translator) for word in words]

4-Tek karakterler temizlenir.

words = [word for word in words if len(word) > 1]

5-Metin içinden sayısal ifadeler temizlenir.

words = [word for word in words if not word.isnumeric()]

6-Kelimeler küçük harflere çevrilir.

words = [word.lower() for word in words]

7-Metin içinden İngilizce diline ait etkisiz kelimeler temizlenir.

words = [word for word in words if word not in stopwords.words('english')]

8-Birden fazla tekrar eden karakterler yanlış yazılma ihtimaline karşı kaldırılır:

tempWords = []

for word in words:

word = re.sub(r'(\w)\1+', r'\1', word)

tempWords.append((word))

words=tempWords

tempWords = []

for word in words:

if word!="" and len(word)>1:

tempWords.append((word))

words=tempWords

return words

## ClearDatabase

def ClearDatabase():

SQLCommand = ("EXEC sp\_MSForEachTable 'TRUNCATE TABLE ?'")

cursor.execute(SQLCommand)

connection.commit()

DB bağlantı işlemlerini gerçekleştirdikten sonra EducationData modülü içerisinde tüm süreç aşağıdaki metod ile gerçekleştirilir.

**EducationData.RunEducationDataProject()**

Bu metod aşağıdaki işlemleri gerçekleştirir.

1-Çalışmaya başlamadan önce, eski veriler veritabanından temizlenir.

ClearDatabase()

2-Kullanıcıdan eğitim ve sınama verilerinin anahtar kelimelerinin tespitinin yapılabilmesi için dokümanların bulundukları dosya yolları girilerek dokümanlar işlem yapılabilmesi için isimleri bir listeye kayıt edilir.

path = input("Enter the Directory location to list:")

sortlist = sorted(os.listdir(path))

3-Listede bulunan tüm doküman isimleri tek tek dönülerek içeriklerinin tek tek okunarak işlemlerin yapılması sağlanır.

while(i < len(sortlist)):

4-Dokümanlar girilen dosya yolundan okunarak metinler elde edilir.

dna = open(path+"\\"+sortlist[i],encoding='utf8',errors='ignore')

5-Metinler paragraflarına ayrılır.

soup = BeautifulSoup(dna)

paragraphs = soup.find\_all("p")

paraghraphId = 1

6-Paragraflar ön işlemeden geçirilir. Daha sonrasında kalan kelimeler etiketlenerek bu etiketlere göre isim olanlar seçilir.

tokens = GetContentFreq(element.text)

tagged = pos\_tag(tokens)

nouns = [word for word,pos in tagged \

if (pos == 'NN' or pos == 'NNP' or pos == 'NNS' or pos == 'NNPS')]

7-“nouns” listesinde bulunan kelimelerin kelime kökleri bulunarak paragrafta aynı kelime kökünden kaç kez geçtiği bilgisi hesaplanarak veri tabanına kayıt edilir.

for word in nouns:

stems = stemmer.stem(word)

if stems in counts.keys():

shortest,count = counts[stems]

counts[stems] = (shortest,count + 1)

else:

counts[stems] = (word,1)

for kok in counts:

shortest,count = counts[kok]

SQLCommand = ("INSERT INTO Words (DocumentId,Word, Count,StemWord,Paragraph) VALUES (?,?,?,?,?)")

Values = [i,shortest,count,kok,paraghraphId] cursor.execute(SQLCommand)

connection.commit()

9-İşlem yapılan dokümanın konusu ve alt konu bilgisi dokümanın isminden alınarak veritabanına kayıt edilir. Doküman isimlendirme “Dokümanİsmi\_Tema\_AltKavram” şeklinde yapılmalıdır.

SQLCommand = ("INSERT INTO Documents (Id,DocumentName,Topic,SubTopic) VALUES (?,?,?,?)")

Values = [i,sortlist[i],sortlist[i],sortlist[i]]

cursor.execute(SQLCommand,Values)

connection.commit()

10- Kelimeler tespit edildikten sonra bu kelimeler arasından anlamlı kelimelerin tespit edilmesi gerekmektedir. Anlamlı kelime tespiti amacıyla 2 farklı yöntem kullanılmıştır.

Helmholtz Prensibi ile anlamlı kelime tespiti DB tarafında saklı yordam (stored procedure) ile yapılmaktadır.

**Helmholtz Prensibi ile anlamlı kelime tespiti DB tarafında saklı yordam**

SQLCommand = ("EXEC InsertMeaningValue")

cursor.execute(SQLCommand)

Diğer bir anlamlı kelime tespit yöntemimiz TF-IDF metrikleridir. TF-IDF metrikleri ile anmalı kelime tespiti EducationData modülü içerisindeki CalculateTFIDF() metodu ile yapılmaktadır.

11- Anlamlı kelime tespiti yapıldıktan sonra kelimeler İngilizce kelimelerimizi tuttuğumuz tabloya kaydedilir.(Sözlük kelimelerinin kaydedildiği tablo) Bu amaçla EducationData modülündeki SaveMeaningWords() metodu çalıştırılır.

**WebSearch Modülü:**

Bu modülde sözlüğe eklenen kelimeler ile Web araması yapılır. Web araması için Azure Cognitive Web Search API’si kullanılır.

subscription\_key='806fdf38921049c29a2e0d808a1b202c'

search\_url = <https://api.cognitive.microsoft.com/bing/v7.0/search>

headers = {"Ocp-Apim-Subscription-Key" : subscription\_key,"mkt":"en-US"}

Bu modülde tüm süreç WebSearch() metodu üzerinden işlemektedir.

def WebSearch():

a=datetime.datetime.now()

print(Fore.LIGHTBLUE\_EX +"Web aramasına başlanıyor.", a)

RemoveAllItemsFromFolder()

result\_set=getMeaningWordForWebSearching()

number\_of\_rows\_MeaningWords = len(result\_set)

if number\_of\_rows\_MeaningWords > 0:

searchTerm=""

for row in result\_set:

searchTerm+=row[0]+" "

if GetCountOfDocumentsOnPath()<thresholdCountOfDocumentsOnPath and GetWordCountOfEnglishDictionary()<thresholdEnglishDictionaryCountForStopProject:

searchAndSaveToFile(searchTerm,True)

else:

searchAndSaveToFile(prefix\_Search\_term,True)

b=datetime.datetime.now()

print(Fore.MAGENTA +"Web araması süresi " ,b-a)

1. Öncelikle işlenen tüm dokümanlar path’den silinir.

RemoveAllItemsFromFolder()

1. Daha sonra Web aramasında kullanılacak kelimeler db’den getirilir.

getMeaningWordForWebSearching()

1. Daha sonra aşağıdaki metod çağrılarak Web araması yapılır ve Web’ten elde edilen dokümanlar ilgili path’e kaydedilir.

searchAndSaveToFile()

**PMI Modülü:**

Bu modül sonradan sisteme entegre edilmiştir. Bu modülün amacı, TF-IDF ya da Helmholtz Prensibi ile anlamlı kelime tespiti yapıldıktan sonra bu anlamlı kelimelerin PMI benzerliğinin tespit edilmesi sonrasında PMI değeri belli bir değerin üzerinde olan kelimerin sözlüğe eklenmesinin sağlanmasıdır.

Bu modül kullanılacak ise EducationData modülünde anlamlı kelime tespiti yapıldıktan sonra çalıştırılmalıdır. Modülde tüm süreç RunPMI() metodu ile sağlanmaktadır.

def RunPMI():

document\_list = GetDocumentList()

if len(document\_list)<2:

print("PMI hesaplaması için en az 2 doküman sisteme yüklenmelidir.")

return

VectorizeDocument(document\_list)

for document in document\_list:

CalculateSingleWordProbabilityForDocuments(document[0])

CalculateTwoWordAssociation

CalculatePMI()

connection.commit()

1. Öncelikle işlem yapılan dokümanlar DB’den getirilir.

document\_list = GetDocumentList()

1. PMI hesaplaması yapılması için doküman sayısının en az 2 olması gerektiğinden bu kontrol gerçekleştirilir.

if len(document\_list)<2:

print("PMI hesaplaması için en az 2 doküman sisteme yüklenmelidir.")

return

1. Dokümanlar vektörlerine ayrılır.

VectorizeDocument(document\_list)

1. Daha sonra kelimelerin tekli ve birliktelik PMI değerleri hesaplanır.

CalculateSingleWordProbabilityForDocuments(document[0])

CalculateTwoWordAssociation

1. Daha sonra tüm bu süreçlerin sonunda genel PMI hesaplaması yapılır.

CalculatePMI()