23.01.2022

Barış

SUBAŞİ

NLP (DOĞAL DİL İŞLEME)

İçindekiler

A. Dil Nedir?

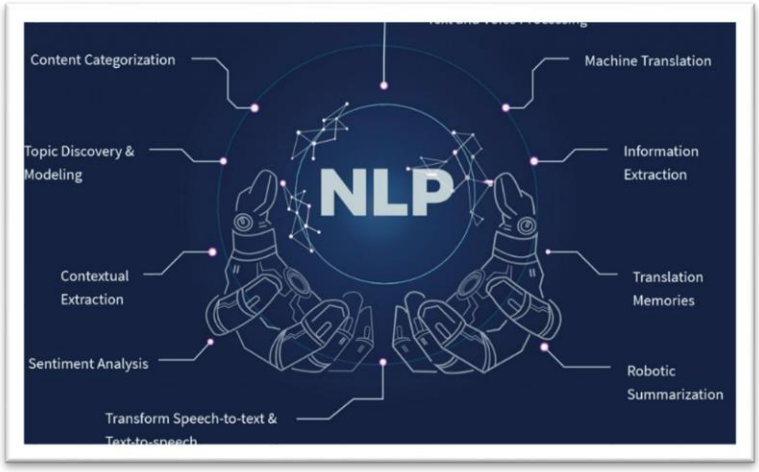
“Sözcük ve cümle birimleri aracılığıyla, düşünceyi konuşmayla ilişkilendiren çok seviyeli bir sistemdir”

N.Chomsky

Dil, insanoğlunun uygarlaşmasını sağlamakla birlikte zekasının giderek parlamasını da sağlamıştır. Kültür dediğimiz kavram dil kullanan ve iletişim kuran insanın sosyalleşme ürünüdür. Dilin bilgisayar ortamında modeli oluşturulursa iletişim için önemli bir araç elde edilmiş olur.

B. Doğal Dil İşleme Nedir?

Doğal Dil İşleme (NLP), makinelerin insan diliyle nasıl etkileşime girdiğini inceleyen yapay zekanın bir parçasıdır. NLP, sohbet robotları, yazım denetleyicileri veya dil çevirmenleri gibi her gün kullandığımız birçok aracı geliştirmek için perde arkasında çalışır. NLP, makine öğrenimi algoritmalarıyla birleştirildiğinde, görevleri kendi başına gerçekleştirmeyi öğrenen ve deneyim yoluyla daha iyi hale gelen sistemler oluşturur. Bilgisayarların, insan duygularını ölçmesine ve insan dilinin hangi bölümlerinin önemli olduğunu belirlemesine yardımcı olur.



Doğal dil işlemenin (NLP) amacı, metni anlamlandırabilen ve çeviri, dilbilgisi denetimi veya konu sınıflandırması gibi görevleri gerçekleştirebilen sistemler oluşturmaktır.

C. Doğal Dil İşlemenin Önemi Nedir?

Doğal dil işlemenin avantajını anlamak için önce onun önemini anlamak gerekir. Örneğin, bulut bilişimden söz ettikten sonra sonraki cümlelerde buluttan diye bahsetmek doğal dil işlemenin örneğidir. Bulut derken kastedilen şey bulut bilişimdir.

Arama için doğal dil işlemeyi kullanmanız durumunda program bulut bilişimin varlık olduğunu, bulutun bulut bilişimin kısaltılmış bir biçimi olduğunu hemen anlayacaktır. Bunun gibi örnekler insan dilinde sıklıkla görünen şeylerdir.

Bunun, analiz edilebilecek veri türleri için çıkarımları vardır. Her gün çevrimiçi ortamda daha fazla bilgi oluşturulmaktadır. Bunların çoğu insan diliyle oluşturulmaktadır. Yakın zamana kadar işletmeler bu verileri analiz edemezdi. Ancak doğal dil işleme ile bunlar mümkün hale gelmiştir.

- Dokümantasyonun iyileştirilmiş doğruluğu ve verimliliği artış göstermiştir.
- Otomatik olarak okunabilir bir özet metni oluşturma yeteneği ortaya çıkmıştır.
- Alexa gibi kişisel asistanlar için kullanışlı bir çözüm olmuştur.
- Bir kuruluşun müşteri desteği için sohbet botlarını kullanmasına izin vermiştir.
- Duygu analizi yapmak çok daha kolay hale gelmiştir.

NLP henüz tam olarak mükemmel durumda değildir. Örneğin anlamsal analiz NLP için hala bir zorluktur. Diğer zorluklar arasında, dilin soyut kullanımının programlar tarafından anlaşılmasının zor olmasıdır. Örneğin, NLP alaycılık durumunu kolayca algılayamaz.

Bu konular genellikle kullanılan kelimelerin ve kullanıldıkları bağlamın anlaşılmasını gerektirir. Bir cümle, konuşmacının vurguladığı bir kelimeye bağlı olarak anlamını değiştirebilir. NLP, dilin ve insanların onu kullanma şeklinin sürekli olarak gelişmesiyle baş etmeye çalışır.

D. NLP Nasıl Çalışır?

NLP'de yapılandırılmamış dil verilerinin makine tarafından bir forma dönüştürülecek şekilde doğal dil kurallarına göre tanımlamak ve çözümlemek amacıyla algoritmaların uygulanması adımlarını içerir.

Bir yazılıma metin girdisi verildiğinde, bilgisayar her cümle ile ilişkili anlam çıkarmak ve bunlardan gerekli verileri toplamak için algoritmalar kullanır.

E. Doğal Dil İşleme Algoritması Nasıl Çalışır?

Doğal dil işleme algoritmasında bilgisayar öncelikle kelimenin kökü üzerine gelen ekler ile dönüşüme bakar, buna kelime bilimi (lexical) denir.

Daha sonra ise cümledeki kelimelerin dizilimine göre ne anlama geldiğini kavramaya çalışır, buna da sözdizimi (syntactic) denir.

Ardından cümlenin özünde ne anlatmaya çalıştığına bakar, buna semantic (dilimize tam olarak çevrilmemesine rağmen anlamsal ağ diyebiliriz) denir.

Son olarak cümlelerin bir araya gelerek ne ifade etmek istediğine bakar, buna da söylev (pragmatics) denir.

Kısaca bilgisayar kelime kökünü ayrı, kelimelerin dizilmesini ayrı, cümlenin ve söylevin anlamını ayrı ayrı inceleyerek konuşmanın bağlamını öğrenir ve bu konuşmadan bir anlam çıkarır.

Tabii ki bu işlemler esnasında bilgisayar zekâsı başarısız olabilir. Başarısız bir sonuç alındığında ise genellikle ardışık düzen oluşturmak gerekir.

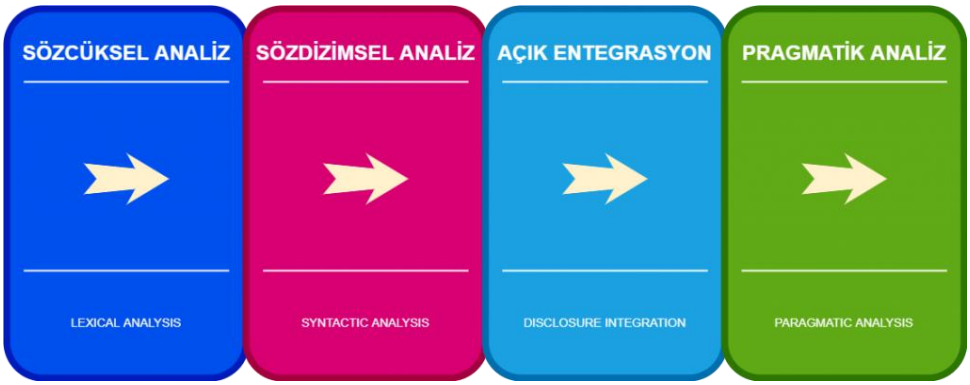
F. Doğal Dil İşleme Alanında Kullanılan Teknikler

- Metin (Text)
- Ses (Speech)

Metin verilerini doğrudan işleyip analiz yapılırken ses verileri üzerinde doğrudan işlem yapılamıyor, öncelikle ses verileri yazılı metinlere çevriliyor ve bu yazılı metinler doğal dil işleme modüllerine uygun hale getirilip makine tarafından işleniyor.

Doğal dil işleme ile ilgilenirken üzerinde çalışılan dilin, dil morfolojisi seviyelerini bilmek oldukça önemlidir, bunlar;

- Kelime bilimi (Morphological-Lexical)
- Sözdizimi (Syntactic)
- Anlamsal (Semantic)
- Söylevdir (Sragmatic-discourse)



F.1. Kelime bilimi: En küçük parçalar olarak ifade edilebilir. Yapım ekleri, çekim ekleri bunlara örnek verilebilir. Örneğin -cı,-ci gibi gelen yapım ekleri bir kelime parçalanırken bölünen en küçük parçadır.

F.2. Sözdizimi: Anlamlı birimlerin tümce oluşturacak biçimde bir araya gelme, tümcelerin, birbirine bağlanma, üretilme, dönüştürülme, vb. kurallarını inceleyen dilbilim dalıdır.

F.3. Anlamsal: Kelime gruplarının bilgisayarın anlayabileceği, kelimeleri birbiriyle karşılaştırabileceği gruplayabileceği eklemeler yaparak yeni kelime türetebileceği ,üretirken kullanacağı formüller halinde ele alınabilecek bir gösterim durumudur. Kısacası mantıksal gösterine ihtiyaç duyulmaktadır.

F.4. Söylev: Bir kelimenin hangi alanda ne anlama geldiğinin bilinmesi gerekir. Cümlede kullanılan bir kelime bir alan için farklı bir anlama gelebilecekken bir başka terminolojide bambaşka anlama gelebilir.

F.5. Uygulama alanlarına 10 farklı proje örnek verilebilir, bunlar:

1. Metin Sınıflandırma ve Kategorizasyon (Text Classification and Categorization)
2. Adlandırılmış Varlık Tanıma (Named Entity Recognition (NER))
3. Konuşma Bölümü Etiketleme (Part-of-Speech Tagging)
4. Anlamsal Ayrıştırma ve Soru Cevaplama (Semantic Parsing and Question Answering)
5. Yorum Bulma (Paraphrase Detection)
6. Dil Üretimi ve Çok Belgeli Özetleme (Language Generation and Multi-document Summarization)
7. Dil Çeviri (Machine Translation)
8. Ses Tanıma (Speech Recognition)
9. Karakter Tanıma (Character Recognition)
10. Yazım Denetimi (Spell Checking)

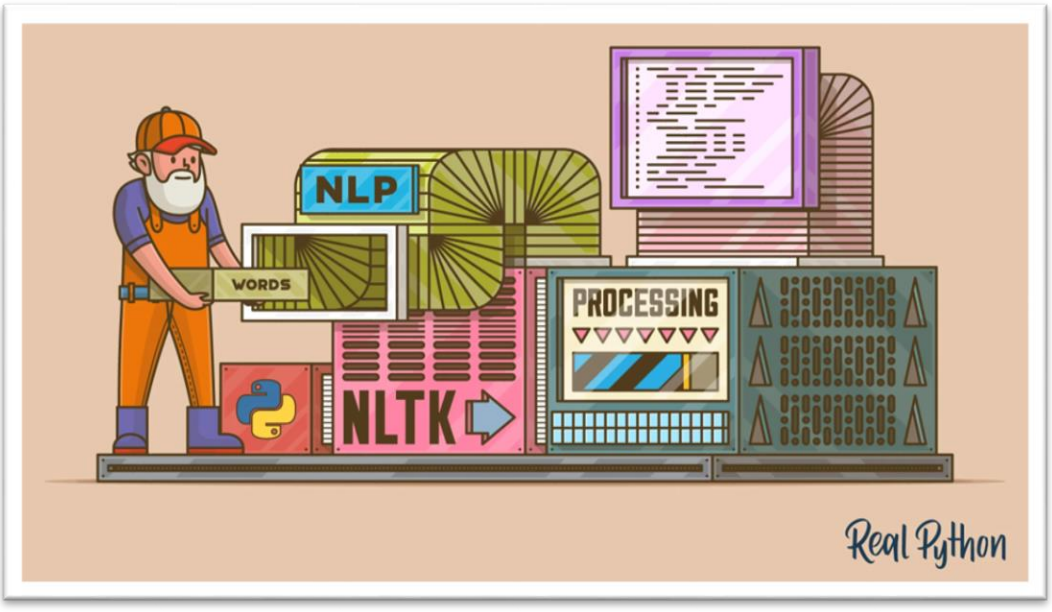
G. NLP (Doğal Dil İşleme) Nerelerde Kullanılır?

Aslında NLP ile geliştirilmiş birçok teknolojik uygulama hayatımızın tam da içinde, bizimle gelişmeye devam etmekte. Bunlar;

- Google, Yandex Çeviri gibi dil çeviri uygulamaları
- Metinlerin gramer doğruluğunu kontrol etmek için Microsoft Word ve Grammarly gibi uygulamalar
- Paraphrasing, Word spinner gibi cümlelerdeki anlamsal bütünlüğü bozmadan değişiklik yapan kelime işlemcileri
- Çağrı merkezlerinde kullanılan Sesli Yanıt (IVR) uygulamaları
- OK Google, Siri, Cortana ve Alexa gibi kişisel asistan uygulamaları olarak örneklendirilebilir.

H. Python İle Doğal Dil İşleme (NLP) Nasıl Yapılır?

Python basit söz dizilimi, şeffaf semantiği, açık kaynak kodlu olması ve topluluk desteği gibi birçok özelliği sayesinde doğal dil işlemede kullanılmaktadır. Python, makine öğrenimi ve derin öğrenme gibi doğal dil işleme üzerinde uygulanacak işlemler ve kullanılabilecek araçlar için adeta biçilmiş bir kaftandır. Çok sayıda teknoloji ile uyum içinde çalışabilmesi ve öğrenim kolaylığı ile Python diğer dillerden bir adım öne çıkmaktadır.



I. Python NLP Kütüphaneleri

NLP tekniklerini oluşturmak ve problemleri çözmek için oluşturulmuş birçok araç ve kütüphane vardır.

Python programlama dilinin basit sözdizimi ve şeffaf semantiği, NLP görevlerini içeren projeler için mükemmel bir seçimdir. Ayrıca, geliştiriciler, makine öğrenimi gibi teknikler için kullanışlı olan diğer diller ve araçlarla entegrasyon için fazlaca destek bulabilirler.

I.1. Python dilinde kullanabileceğiniz birkaç NLP kütüphanesi:

- Natural Language Toolkit (NLTK)
- TextBlob
- CoreNLP
- Gensim
- spaCy
- polyglot
- scikit–learn
- PyText
- Pattern

J. Matbu Dokümanların Python Dili ile Sınıflandırılması

Bu projede, bir NLP problemi olan matbu dokümanların sınıflandırılması probleminin nesneye dayalı özellikler kullanılarak Python dili ile gerçekleştirdik.

K. Python Kodumuz

K.1. Terminalimiz de öncelikle pdfminer ve nltk komutlarını girdik.

- PDFMiner, PDF belgeleri için bir metin çıkarma aracıdır.
- Kodumuz da pdfminer kütüphanesini pdf'den stringe çevirebilmek için kullandık.
- NLTK (Natural Language Toolkıt): NLTK'yı dilimize Doğal Dil Araç Seti olarak çevirebiliriz. Doğal dil işleme alanında en çok bilinen kütüphanedir. İngilizce için geliştirilmiş sembolik ve istatistiksel bir doğal dil işleme kütüphanesidir. Doğal dil işlemeye girişte kullanılan ve oldukça pratik olan bir kütüphanedir. NLTK ile cümle tespiti, tokenleştirme, kök bulma gibi doğal dil işleminin temel işlevlerini çok kolay bir şekikde kullanıma sunulur. SemCor, WordNet ve Web Text Corpus gibi 50'den fazla corpora'ya sahiptir.
- Kodumuz da nltk kütüphanesini stopwords ve lemmatize işlemini yapabilmek için kullandık .

```
>pip install pdfminer
```

```
>pip install nltk
```

K.2. pdfminer kütüphanesini rahatça kullanabilmek için import işlemleri gerçekleştirdik.

```
import io
from pdfminer.pdfinterp import PDFResourceManager,PDFPageInterpreter
from pdfminer.pdfpage import PDFPage
from pdfminer.converter import TextConverter
from pdfminer.layout import LAParams
```

K.3. stopwords ve lemmatize için gereken import işlemlerini gerçekleştirdik.

- Stopwords:(Etkisiz Kelimeler), arama motorları üzerinde yapılacak sorgular sonucunda kullanıcıya sonucu daha hızlı gösterebilmek için yok sayılan kelimelerdir.
- Lemmatizasyon (Lemmatization): Bir kelimenin çeşitli şekillerde değiştirilmiş formlarını kolay analiz için tek bir forma indirgemeyi kapsar. Lemmatizasyon, önemine bağlı bir kelimenin lemmasını bulmanın algoritmik döngüsüdür.
- NLP'de her kelimenin en yalın halini bulma işlemine lemmatizasyon denir.

```
from nltk.corpus import stopwords
from nltk.stem import WordNetLemmatizer
```

Kullanım rahatlığı için lemmatize fonksiyon ismini kısaltma işlemini yaptık.

```
wnl=WordNetLemmatizer()
```

PDF'den string'e dönüştürmek için gereken fonksiyonu gerçekleştirdik.

```
def pdf2str(inPDFfile):
    infile = open(inPDFfile,'rb')
    resMgr = PDFResourceManager()
    retData = io.StringIO()
    TxtConverter = TextConverter(resMgr,retData,laparams=LAParams())
    interpreter = PDFPageInterpreter(resMgr,TxtConverter)
```

Her bir sayfadaki yazıları topladık.

"getvalue()" fonksiyonu ile tek bir değerde toplamış olduk.

```
for page in PDFPage.get_pages(infile):
    interpreter.process_page(page)
txt=retData.getvalue()
```

K.4. Text Sınıfı

Veri kümeleri ve sınıflandırılacak pdf dosyaları string olarak sınıfta tuttuk.

```
class Text():
    pdf_text=""      #Sınıflandırılacak pdf
    mat_data=""      #Matematik eğitim kümesi (pdf)
    med_data=""      #Tıp veri kümesi (pdf)
    hist_data=""     #Tarih veri kümesi (pdf)
```

K.5. Structure

Obje oluşturulurken verilen değerler sınıftaki değişkenlere tanımlattırdık ve main kısmında yapılan seçime göre sınıf fonksiyonlarını çağırdık.

```
def __init__(self,pdf,matdata,meddata,histdata,choice):
    self.pdf_text=pdf
    self.mat_data=matdata
    self.hist_data=histdata
    self.med_data=meddata


    self.stripPdfText()  # ---------->Sınıflandırma yapılacak pdf dosyasını temizler.
    if choice==1:
        self.stripDataText()#------->Veri kümesindeki pdf dosyalarını temizler.
    elif choice==2:
        self.striptext()
    self.searchAndClassify()#------->Temizlenen dosyalar içerisinde arama ve sınıflandırma yapar.
```

K.6. Sınıflandırılacak pdf string'ini Temizleyen Fonksiyon

Pdf'deki sayıları ve noktalama işaretlerini temizlendikten sonra string halindeki her bir kelimeyi bir liste olarak ayırdık.

Listedeki en az 3 harfli kelimeleri teker teker "lemmatize" edip yeni bir listeye atadık.

Sonrasında ise yeni listeyi sınıflandırılacak pdf stringin yerine geçirdik.

```
def stripPdfText(self):
    self.pdf_text = self.pdf_text.lower()#---------------------->Tüm harfleri küçük harfe çevirir.

    symbols = "1234567890!\"#$%&()*+-./:;<=>?@[\]^_`{|}~\n"#
    for i in symbols:                                       #
        self.pdf_text = self.pdf_text.replace(i, ' ')       #       Şekiller,işaretler,unicode'lar,
                                                            #------>sayılar ve noktalama işaretlerini
    x = ["\u2002", "\x0c", "–", "—", "'", "”"]              #       temizler.
    for i in x:                                             #
        self.pdf_text = self.pdf_text.replace(i, ' ')       #

    a=self.pdf_text.split(" ")



    new_text =[]
    for w in a:
        if len(w) > 2:
            new_text.append(wnl.lemmatize(w))



    self.pdf_text=new_text
```

K.7. Veri Kümesindeki pdf'leri Temizleyen Fonksiyon

stopwords kelimeleri erişim kolaylığı açısından farklı bir değişkene atadık.

Listedeki kelimelerden; stopwords içinde olmayan ve en az 3 harfli olan kelimeleri teker teker "lemmatize" edip yeni bir listeye atadık.

Liste, set'e dönüştürülüp listedeki aynı kelimeleri kaldırdık ve tekrar listeye çevirdik.

```
def stripDataText(self):
    for i in (self.hist_data, self.med_data, self.mat_data): # Veri kümesindeki her bir pdf için
        a = i
        a = a.lower()#---------------------->Tüm harfleri küçük harfe çevirir.

        symbols = "1234567890!\"#$%&()*+-./:;<=>?@[\]^_`{|}~\n"
        for i in symbols:
            a = a.replace(i, ' ')
        x = ["\u2002", "\x0c", "–", "—", '"', "”"]
        for i in x:
            a = a.replace(i, ' ')
        a = a.replace("’", "")

        new_text = []
        stop_words = set(stopwords.words('english'))

        a = a.split(" ")#---->Temizlenen string halindeki pdf'in her bir kelimesini listeye atar.
        for word in a:
            if word not in stop_words:
                if len(word) > 2:
                    new_text.append(wnl.lemmatize(word))
        a = new_text
        a = set(a)
        a = list(a)

        if i==self.mat_data:
            self.mat_data=a
        elif i==self.hist_data:             #Veri kümesindeki her bir pdf için
            self.hist_data=a                #gerekli sınıf değişkenine atama yapılır
        elif i==self.med_data:
            self.med_data=a
```

K.8. Temizlenmiş Veriyi Listeye Atama Fonksiyonu

```
def striptext(self):
    matlist=[]
    medlist=[]
    histlist=[]
    for i in self.med_data:
        a=i.replace("\n","")
        medlist.append(a)
    for i in self.mat_data:
        a = i.replace("\n", "")
        matlist.append(a)
    for i in self.hist_data:
        a = i.replace("\n", "")
        histlist.append(a)

    self.mat_data=matlist
    self.med_data=medlist
    self.hist_data=histlist
```

K.9. Arama ve Sınıflandırma Fonksiyonu

K.9.1. Arama Kısmı

```
def searchAndClassify(self):
    mat_count=0
    med_count=0     #   Sayaclar 0 olarak başlatılır.
    hist_count=0
#---------- Arama Kısmı  -------
    for i in self.pdf_text:
    #Sınıflandırılacak pdf'teki her bir kelime için
        if i in self.mat_data and i in self.med_data and i in self.hist_data:
            pass
        # Veri kümesindeki üç pdf'te de olan kelimeler hesaba katılmadan
        else:
            if i in self.mat_data:  #
                mat_count+=1        #
            if i in self.hist_data: #   kelimer bulunduğu dala göre sayaçları arttırır.
                hist_count+=1       #
            if i in self.med_data:  #
                med_count+=1        #
```

K.9.2. Sınıflandırma ve Output Kısmı

```
    print("\nOut of the", len(self.pdf_text), "words in selected PDF file")
    print("Medicine:",med_count)
    print("History:",hist_count)
    print("Mathematics:",mat_count)
#Her bir dal için kaç tane kelime bulunduğu ifade edilir.
    print("words were found from the subjects above.")
    if med_count>hist_count and med_count>mat_count:
        print("According to the information above, most of the words are related to medicine with",med_count,"words.")
        print("\nTherefore, the selected PDF is a medical document.")
    elif hist_count>med_count and hist_count>mat_count:
        print("According to the information above, most of the words are related to history with",hist_count,"words.")
        print("\nTherefore, the selected PDF is a historical document.")
    elif mat_count>med_count and mat_count>hist_count:
        print("According to the information above, most of the words are related to mathematics with",mat_count,"words.")
        print("\nTherefore, the selected PDF is a mathematical document.")
#En çok sayıda kelimesi geçen dal hangisi ise ona göre:
    #PDF'in ilgili olduğu dal ve o dalla ilgili bulunan kelime sayısı yazdırılır.
```

K.10. Main

Kodumuzdaki main kısmında ilk olarak kullanıcıya 2 seçenek sunduk:

1) Öncesinde temizlenmiş veriyi kullanarak sınıflandırma yapmak

2) En baştan eğitim veri kümesindeki pdf'leri temizledikten sonra bu veriler ile sınıflandırma yapmak(Bu işlem uzun sürebilir).

```
secim=int(input("Type '1' to classify with cleaned data.\nType '2' to clean test data set.Then start classify(Note:This might increase process time)\n"))
```

Eğer 1. Seçenek Seçilirse:

Sınıflandırılacak pdf dosyasını string'e çevirdik.

Önceden temizlenmiş verileri open( ) fonksiyonu ile bir değere atadık.

Sonrasında sınıftan bir obje oluşturarak işlemi tamamladık.

```
elif secim==1:
    #Temizlenmis veri ile islem yapmak için
    selectedfile = pdf2str('example3.pdf')
    history=open("hist_data.txt","r")
    mathematic=open("math_data.txt","r")
    medical=open("med_data.txt","r")

    text=Text(selectedfile,mathematic,medical,history,2)
```

Eğer 2. Seçenek Seçilirse:

Sınıflandırılmak istenen pdf 'path'(dosya yolu) ile veya pdf, kod dosyasında ise direkt ismi yazılarak işlem yapılabilir.

Dosya yolu (path) ile selectedfile=pdf2str('path') yaptık.

```
selectedfile=pdf2str('example.pdf')
```

Veri Kümesi PDF'leri de değişkenlere atadık.

```
history=pdf2str('hist_data.pdf')
```

```
mathematic=pdf2str('math_data.pdf')
```

```
medical=pdf2str('med_data.pdf')
```

Text sınıfından obje oluşturup gerekli değerleri vermek yeterlidir

```
text=Text(selectedfile,mathematic,medical,history)
```

K.11. Çıktı

```
The classification process time may vary depending on the number of pages of the selected file.

---------- Please Wait ----------
Type '1' to classify with cleaned data.
Type '2' to clean test data set.Then start classify(Note:This might increase process time)
1

Out of the 3321 words in selected PDF file
Medicine: 811
History: 620
Mathematics: 272
words were found from the subjects above.
According to the information above, most of the words are related to medicine with 811 words.

Therefore, the selected PDF is a medical document.

Process finished with exit code 0
```

```
The classification process time may vary depending on the number of pages of the selected file.

---------- Please Wait ----------
Type '1' to classify with cleaned data.
Type '2' to clean test data set.Then start classify(Note:This might increase process time)
2

Out of the 3321 words in selected PDF file
Medicine: 799
History: 492
Mathematics: 276
words were found from the subjects above.
According to the information above, most of the words are related to medicine with 799 words.

Therefore, the selected PDF is a medical document.
```

```
The classification process time may vary depending on the number of pages of the selected file.

---------- Please Wait ----------
Type '1' to classify with cleaned data.
Type '2' to clean test data set.Then start classify(Note:This might increase process time)
2

Out of the 2730 words in selected PDF file
Medicine: 330
History: 761
Mathematics: 282
words were found from the subjects above.
According to the information above, most of the words are related to history with 761 words.

Therefore, the selected PDF is a historical document.

Process finished with exit code 0
```

```
The classification process time may vary depending on the number of pages of the selected file.

---------- Please Wait ----------
Type '1' to classify with cleaned data.
Type '2' to clean test data set.Then start classify(Note:This might increase process time)
2

Out of the 2712 words in selected PDF file
Medicine: 156
History: 237
Mathematics: 461
words were found from the subjects above.
According to the information above, most of the words are related to mathematics with 461 words.

Therefore, the selected PDF is a mathematical document.

Process finished with exit code 0
```