
# BİTİRME PROJESİ

Haftalık Rapor – 06.05.2022

06 MAYIS 2022

KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ

AHMET MUNGAN – 160255081

# İÇİNDEKİLER

# ÖZET

Kişisel doğal dil işleme araçları oluşturulmaya çalışılmıştır. Bu araçlar belirli örüntüler keşfedilerek ve dil bilgisinin izin verdiği ölçüde anlamı değiştirmeden uygulanmaya çalışılmıştır. Bu kapsamda, neredeyse her araç için program kodu paylaşılmıştır. İstisnalar ve aracın dayanıklılığı yorumlanmıştır.

## KİŞİSEL TÜRKÇE DOĞAL DİL İŞLEME KÜTÜPHANESİ UYGULAMASI

Türkçe diline doğal dil işleme uygun bir kütüphane oluşturmak mümkündür fakat zorludur. Ayrıca tam olarak doğal dil işleme konusunun hakkını vermek gerekir. Sadece yazılarla sınırlı kalmayıp, anlamsal çıkarımlarda bulunabilecek bir yapay bilinç oluşturmak gerekir. Bu proje kapsamında konu metin madenciliğinden çıkmaması için, konuya sadık kalınıp asıl kısımlara odaklanılması için doğal dil işleme ayrıntılı ve hakkını vererek yapılmayacaktır.

Doğal dil işleme yerine kullanılabilecek bir yöntem şudur; doğal dilde örüntüler çıkarmak. Nasıl ki veri madenciliğinde bizlerin asıl amacı anlamsız bir bütünden anlamlı ve yararlı örüntüler elde etmek ise, aynı yaklaşımla dökümanlar bu örüntülerin ışığında incelenebilir. Türkçe diline ait, sıkça kullanılan ve yaygın eklerin neler olduğu tespit edilmiştir. Bu eklerden sonra gelen ekler de tespit edilmiştir. Fakat karıştırılmaması gereken bir durum söz konusudur. Bu ekler yapım ekleri olmamakla birlikte, anlam taşımayan çekim ekleri ile sınırlıdır. Dolayısıyla tam manasıyla bir stemming veya lemmatization'dan söz etmek mümkün değildir.

Program kodlarında örnek olarak gösterdiğim cümleler ya da kelimeler, o an yapılan işlemi en zorlayacak kelime öbekleri olarak seçilmiştir. Özenli bir araştırma ve seçim durumu söz konusudur.

*Code 1*

```
#-miş, -mişlik vb. eklerin silinmesi/değiştirilmesi
>>> cumle = "görebilmişlik varmış kanımdaymışsın amaymış bir yaranmuşluk
olabilitesimüşlük olabilirmüş"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding = 'u
tf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 6:
>>>         if cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "miş":
>>>             if cumle[i][0:-3] in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]
>>>             elif cumle[i][0:-3]+"me" in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]+"me"
>>>     if len(cumle[i]) > 9:
>>>         if cumle[i][-6]+cumle[i][-5]+cumle[i][-4]+cumle[i][-3]+cumle
[i][-2]+cumle[i][-1] == "mişlik":
>>>             if cumle[i][0:-6] in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]
>>>             elif cumle[i][0:-6]+"me" in sozluk:
```

```
>>>                 cumle[i] = cumle[i][0:-6]+"me"
    #----Duyulan Geçmiş Zaman Eki + Yapım Eki----
>>>     if len(cumle[i]) > 6:
>>>         if cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "mış":
>>>             if cumle[i][0:-3] in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]
>>>             elif cumle[i][0:-3]+"ma" in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]+"ma"
>>>     if len(cumle[i]) > 9:
>>>         if cumle[i][-6]+cumle[i][-5]+cumle[i][-4]+cumle[i][-3]+cumle
[i][-2]+cumle[i][-1] == "mışlık":
>>>             if cumle[i][0:-6] in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]
>>>             elif cumle[i][0:-6]+"ma" in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]+"ma"
    #----Duyulan Geçmiş Zaman Eki + Yapım Eki----
>>>     if len(cumle[i]) > 6:
>>>         if cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "muş":
>>>             if cumle[i][0:-3] in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]
>>>             elif cumle[i][0:-3]+"ma" in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]+"ma"
>>>     if len(cumle[i]) > 9:
>>>         if cumle[i][-6]+cumle[i][-5]+cumle[i][-4]+cumle[i][-3]+cumle
[i][-2]+cumle[i][-1] == "muşluk":
>>>             if cumle[i][0:-6] in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]
>>>             elif cumle[i][0:-6]+"ma" in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]+"ma"
    #----Duyulan Geçmiş Zaman Eki + Yapım Eki----
>>>     if len(cumle[i]) > 6:
>>>         if cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "müş":
>>>             if cumle[i][0:-3] in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]
>>>             elif cumle[i][0:-3]+"me" in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]+"me"
>>>     if len(cumle[i]) > 9:
>>>         if cumle[i][-6]+cumle[i][-5]+cumle[i][-4]+cumle[i][-3]+cumle
[i][-2]+cumle[i][-1] == "müşlük":
>>>             if cumle[i][0:-6] in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]
>>>             elif cumle[i][0:-6]+"me" in sozluk:
>>>                 cumle[i] = cumle[i][0:-6]+"me"
>>> cumle = " ".join(cumle)
>>> cumle
'görebilme varmış kanımdaymışsın amaymış bir yaranma olabilitesimüşlük o
labilir'
```

Code 1'de görüldüğü şekliyle anlamı olmayan –mış, -miş, -mışlık, -mişlik vb. hem tek ek alan hem de –mış'tan sonra –lık eki alanların kontrolü yapılmıştır. Code 1'de kontrol edilen sözlük geçen hafta raporlarda bahsedilen ve indisli bir şekilde oluşturulan sözlüktür. Belirli büyüklükteki kelimelerin bu işleme dahil edilmesi ise,

anlamlı olan –mış, -muş gibi eklerin kaybedilmemesi içindir. Bu sebeple en optimum kelime uzunluğu el ile ve deneyimler ile belirlenmiştir.

*Code 2*

```
#-ki ekinin anlamsız olanlarının silinmesi
>>> cumle = "olabilmişki benimki olaki benki seninki araki olaki
olabilirki oluruki olurki"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding =
'utf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 4:
>>>         if cumle[i][-2] + cumle[i][-1] == "ki":
>>>             if cumle[i][0:-1] in sozluk:
>>>                 cumle[i] = cumle[i][0:-1]
>>>             elif cumle[i][0:-2] in sozluk:
>>>                 cumle[i] = cumle[i][0:-2]
>>> cumle = " ".join(cumle)
>>> cumle
'olabilmişki benim ola ben seninki arak ola olabilir oluruki olur'
```

Code 2'de Türkçe dilinde insanların bile çoğu zaman yanlış kullandığı –ki eki ele alınmıştır. Fakat Code 2'de yazılan örnek cümlenin çıktısından görüleceği üzere hatalı yazımlara karşı hassasiyeti düşük bir araç denebilir.

*Code 3*

```
#-sal ekinin anlamsız olanlarının silinmesi
>>> cumle = "yapısal bir reformsal anasal değişiklik anayasal hukuksal
bir haktır"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding =
'utf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "sal":
>>>             if cumle[i][0:-3] in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]
>>> cumle = " ".join(cumle)
>>> cumle
'yapı bir reform ana değişiklik anayasal hukuk bir haktır'
```

Code 3'te alışıla gelmişin dışında bir ek olan –sal eki değerlendirilmiştir. Bu ek çoğu zaman anlamsız bir ek olabildiği gibi, anlamlı olduğu yerlerin varlığından söz etmek de mümkündür. Örneğin "anayasal" kelimesinin "anaya" şeklinde anlamsız olacağına program karar verip asıl anlamını korumaya devam etmiştir. "yapısal" kelimesinin "yapı" kelimesine dönüştürülmesi mantıken hatalı gibi gözükse de, her iki

kelimenin kökü aynıdır. Dolayısıyla hata gibi gözükmesinden ziyade bu araç daha doğru sonuçlar üretebilir.

*Code 4*

```
#-deki -daki eklerinin anlama göre silinmesi/değiştirilmesi
>>> cumle = "buradaki yapılan bendeki eldeki büyük bir hatadaki derstir
daki deki dedidaki demişdaki"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding =
'utf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-4] + cumle[i][-3] + cumle[i][-2] + cumle[i][-1]
== "deki" or cumle[i][-4] + cumle[i][-3] + cumle[i][-2] + cumle[i][-1]
== "daki":
>>>             if cumle[i][0:-4] in sozluk:
>>>                 cumle[i] = cumle[i][0:-4]
>>> cumle = " ".join(cumle)
>>> cumle
'bura yapılan ben el büyük bir hata derstir daki deki dedi demişdaki'
```

Code 4'te, sahiplik belirten ve silindiği taktirde anlamı yüksek oranda bozmayan –deki, -daki ekinin ele alınması mevcuttur. Örnek cümlede bile bu eklerin anlamlı olduğu durumları bulmak çok zordur. Çünkü bu ekler genelde kök halinde kelimelerde bulunmadığından, bulunduğu takdirde sözlükte ek almış olarak bulunduğundan doğru bir örüntü olduğundan söz edilebilir.

*Code 5*

```
#-meli -malı eklerinin anlamı bozmadan değiştirilmesi/silinmesi
>>> cumle = "bunu böylemeli yapmalı gelmemeli etmemeli etmeli olmalı"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding =
'utf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-4] + cumle[i][-3] + cumle[i][-2] + cumle[i][-1]
== "meli" or cumle[i][-4] + cumle[i][-3] + cumle[i][-2] + cumle[i][-1]
== "malı":
>>>             if cumle[i][0:-2] + "k" in sozluk:
>>>                 cumle[i] = cumle[i][0:-2] + "k"
>>>             elif cumle[i][0:-2] in sozluk:
>>>                 cumle[i] = cumle[i][0:-2]
>>>             elif cumle[i][0:-4] in sozluk:
>>>                 cumle[i] = cumle[i][0:-4]
>>> cumle = " ".join(cumle)
>>> cumle
'bunu böyle yapmak gelme etme etmek olmak'
```

Code 5; –meli, -malı eklerinin anlamı bozmadan (olumsuz ise olumsuz çıktı, olumlu ise olumlu çıktı vermesi) silinmesini ele almıştır. Code 5'in, Code 4 ve Code 3'le neredeyse aynı program kodlarını içerdiği görülmektedir. Bu çerçevede bu örüntüler çıkarıldıkça farklılaşmayan program kodları görmek mümkündür.

Örüntünün türleri ve eklerin yapısı değişiyor olsa da program kodları çok büyük değişiklikler göstermediği için, yorumlarının yapılmaması ve sadece koda bakılarak anlaşılabileceği için diğer araçlarda aşağıda yazılmıştır. Birbirini takiben Code 6, 7 ve 8 bu bağlamda paylaşılmıştır.

*Code 6*

```
#-nden -ndan ikinci veya üçüncü tekil şahıs eki + ayrılma eki anlama gör
e silinmesi
>>> cumle = "onundan burnundan oradan kumandan burnudan aldığından aldığ
ıdan alacağından olmadan"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding = 'u
tf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 6:
>>>         if cumle[i][-4] + cumle[i][-3] + cumle[i][-2] + cumle[i][-1]
== "nden" or cumle[i][-4] + cumle[i][-3] + cumle[i][-2] + cumle[i][-1] =
= "ndan":
>>>             if cumle[i][0:-4] in sozluk:
>>>                 cumle[i] = cumle[i][0:-4]
>>>         elif cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "den" or
cumle[i][-3] + cumle[i][-2] + cumle[i][-1] == "dan":
>>>             if cumle[i][0:-3] in sozluk:
>>>                 cumle[i] = cumle[i][0:-3]
>>>             elif cumle[i][0:-1] in sozluk:
>>>                 cumle[i] = cumle[i][0:-1]
>>> cumle = " ".join(cumle)
>>> cumle
'onu burnu oradan kuma burnu aldığı aldığı alacağı olma'
#-çe -ça sert eşitlik eki silinmesi (anlam değişimi düşük)
>>> cumle = "poğaça dilekçe gerekçe kelepçe poliçe tarihçe "
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding = 'u
tf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-2] + cumle[i][-1] == "ça" or cumle[i][-2] + cum
le[i][-1] == "çe":
>>>             if cumle[i][0:-2] in sozluk:
>>>                 cumle[i] = cumle[i][0:-2]
>>> cumle = " ".join(cumle)
>>> cumle
'poğaça dilek gerek kelep poliçe tarih'
```

*Code 7*

```
#-te -ta sertleşmiş bulunma halinin anlamına göre silinmesi
>>> cumle = "balata altta buracıkta cıvata hayata hayatta icraata kanata
kanatta kızarta kızartta "
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding = 'u
tf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-2] + cumle[i][-1] == "ta" or cumle[i][-2] + cum
le[i][-1] == "te":
>>>             if cumle[i][0:-1] in sozluk:
>>>                 cumle[i] = cumle[i][0:-1]
>>>             elif cumle[i][0:-2] in sozluk:
>>>                 cumle[i] = cumle[i][0:-2]
>>> cumle = " ".join(cumle)
>>> cumle
'balat altta buracıkta cıva hayat hayat icraat kanat kanat kızarta kızar
tta'
```

*Code 8*

```
#-ce -ca eşitlik eki silinmesi (anlam değişimi yüksek)
>>> cumle = "almanca atmaca bulmaca abaca çatalca çukurca darıca haraca
kanlıca kaplıca karınca kokarca sapanca"
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding = 'u
tf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-2] + cumle[i][-1] == "ca" or cumle[i][-2] + cum
le[i][-1] == "ce":
>>>             if cumle[i][0:-2] in sozluk:
>>>                 cumle[i] = cumle[i][0:-2]
>>> cumle = " ".join(cumle)
>>> cumle
'alman atma bulma abaca çatal çukur darı hara kanlı kaplı karın kokar sa
pan'


#-çe -ça sert eşitlik eki silinmesi (anlam değişimi düşük)
>>> cumle = "poğaça dilekçe gerekçe kelepçe poliçe tarihçe "
>>> cumle = cumle.lower().split()
>>> for i in range(len(cumle)):
>>>     sozluk = open("sozluk/{}.txt".format(cumle[i][0]), encoding = 'u
tf-8', mode = 'r').read().split('\n')
>>>     if len(cumle[i]) > 5:
>>>         if cumle[i][-2] + cumle[i][-1] == "ça" or cumle[i][-2] + cum
le[i][-1] == "çe":
>>>             if cumle[i][0:-2] in sozluk:
>>>                 cumle[i] = cumle[i][0:-2]
>>> cumle = " ".join(cumle)
>>> cumle
'poğaça dilek gerek kelep poliçe tarih'
```

## REFERANS VE KAYNAKÇA


[1] Dil bilgisi için klavuz TDK. Link için tıklayınız. (Güvenlidir.)

[2] Yemeksepeti gizlilik politikası. Link için tıklayınız. (Güvenlidir.)

[3] Yemeksepeti kişisel verilerin korunumu. Link için tıklayınız. (Güvenlidir.)

[4] BeautifulSoap kütüphanesi dökümanları. Link için tıklayınız.

# EKLER

Bitirme Projesi 2’ye ait doküman, program kodu, haftalık rapor ve ek bilgilerin paylaşıldığı github linki için tıklayınız. (Güvenlidir.)