
# BİTİRME PROJESİ

Haftalık Rapor – 20.05.2022

20 MAYIS 2022

KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ

AHMET MUNGAN – 160255081

# İÇİNDEKİLER

# ÖZET

Metin madenciliğinde önemli bir dönüm noktası olan terim ağırlıklandırma üzerinde durulmuştur. Ağırlıklardan yola çıkarak ortalamaya bağlı ayırt edicilik skorları elde edilmiştir. Doküman tipinden, sayısından bağımsız ve aynı şekilde benzersiz kelime sayısından bağımsız ölçekli, normalize edilmiş, hesaplanabilir ve maliyeti düşük bir algoritma ortaya koyulmuştur. Bu algoritmanın matematiksel formları ve matris çıktıları verilmiştir.

## AĞIRLIKLANDIRMA İÇİN BENZERSİZ KELİMELERİN BULUNMASI

Geçen hafta itibariyle ağırlıklandırmanın alt parametreleri bulunmuş ve benzersiz (unique) kelimelerin çıkartılması işlemi yapılmalıdır. Bu sayede tekrar eden veriler yerine tekrar ettiği sayı kadar frekansların hesaplanması sağlanabilecektir. Frekans direkt bir biçimde kullanılmasa da ağırlıkların hesaplanması için logaritma ile yerel ağırlığa ve entropi ile global ağırlığa hesaplama açısından yardımcı olacaktır. Bu sayede hem algoritmanın maliyeti düşecektir hem de matematiksel hesap karmaşasının önüne geçilecektir.

*Code 1*

```
>>> kelimeler_unique = []
>>> for i in range(len(kelimeler)):
>>>     for j in range(len(kelimeler[i])):
>>>         if kelimeler[i][j] in kelimeler_unique:
>>>             pass
>>>         else:
>>>             kelimeler_unique.append(kelimeler[i][j])
>>> len(kelimeler_unique)
869
```

Code 1'de[1] unique kelimeler bulunmuştur. Ardından matris oluşumu için dökümanların saydırılması ve herhangi bir isim verilmesi gerekmektedir.

*Code 2*

```
>>> dokumanlar = []
>>> for i in range(len(kelimeler)):
>>>     dokumanlar.append("D" + str(i))
>>> dokumanlar.append("gfi")
>>> dokumanlar
['D0',
 'D1',
 ...
 'D169',]
```

Dökümanlar ismen ayırt edici olmadığından, tüm dökümanların aynı çerçevede değerlendirilmesi gerektiğinden doküman adları Code 2'deki şekilde oluşturulmuştur.

[1] Python'da kodun denenmesi açısından küçük çaplı bir corpus örneği alınmıştır. Bağlaçlar, durak kelimeler gerçek projenin kodlarında atıldığı unutulmamalıdır.

Ayrıca dökümanlar için; unutulmamalıdır ki her biri birer restoran yorumudur. Bu yorumlardan yemeksepeti veya getiryemek uygulamasından yapılmış yorumlar olması bir öncelik veya ayırt edicilik barındırmaz.

Frekans için sıfır olasılık durumlarından kurtulmak için matriste türlü hilelere başvurulabilir. Sıfır matrisi gelmemesi için ortalamalar yöntemi, tekil ondalıklı ekleme yöntemleri kullanılabilir. Fakat ortalamalar yöntemi için tüm matrisin değerleri hesaplanmalı ve hazır vektörizerlerin kullanılmadığı göz önüne alınırsa; algoritma maliyetini ciddi derecede arttıran bir yöntem olacaktır. Tekil ondalıklı ekleme yönteminde ise 0 olan değerler yerine çok küçük ondalıklı sayılar eklemektir. Bu yöntemin sorunu ise üst seviye programlama dillerinde zero division hatalarına sebep olması yüksek ihtimaldir. Çok küçük ondalıklı sayıları Python gibi üst seviye diller daha farklı durumlara odaklandığı için sıfıra yuvarlamaya çalışır. Bu sebeple hata alınır.

Bu iki yöntemin kullanılması maliyet ve hatalar açısından mantıklı olmadığından, toplamı 1 edecek şekilde ondalıklı değerler verilecektir. Buradan hareketle frekansın en az +1 kadar değişeceği söylenebilir fakat; hem local hem global ağırlıklandırmada logaritma ile normalleştirme kullanıldığı için +1 değeri çok daha ufak bir ondalıklı sayıya eş değer olacaktır. Ayrıca tüm terimlerin frekansının o kadar ufak bir değerin toplanmasıyla minimum sayıda değişim gözlenir.

*Code 3*

```
>>> data_frekans = []
>>> for k in kelimeler_unique:
>>>     dizi = []
>>>     for i in range(len(kelimeler)):
>>>         sayi = 1 / (len(kelimeler) + 1)
>>>         for j in range(len(kelimeler[i])):
>>>             if k == kelimeler[i][j]:
>>>                 sayi += 1
>>>         dizi.append(sayi)
>>>     toplam = 1 / (len(kelimeler) + 1)
>>>     for l in range(len(dizi)):
>>>         toplam += dizi[l]
>>>     dizi.append(toplam)
>>>     data_frekans.append(dizi)
>>> frekans = pd.DataFrame(data = data_frekans, index = kelimeler_unique
, columns = dokumanlar)
```

Code 3'te kullanılan ve yukarıda bahsedilen yöntem matematiksel forma dönüştürülürse:

$$frequency = \begin{cases} frequency += 1 & if \quad tf_{ij} > 0 \\ frequency += \dfrac{1}{length(unique_{words})} & if \quad tf_{ij} = 0 \end{cases}$$

## AYIRT EDİCİLİK MATRİSİ

Global ve local ağırlıkların çarpımıyla bir terimin ayırt edicilik katsayısı bulunabilmektedir. Fakat çarpıldığı zaman bu iki değer, ortaya her bir benzersiz kelimenin ağırlığı çıkmaktadır. Dolayısıyla elde edilen matris $documents * item$ büyüklüğünde olmaktadır. İyi bir ön işleme süreci olduğu takdirde bu matrisin boyutu çok büyük olmayacaktır fakat Türkçe dili üzerinde çalışıldığı düşünülürse ve doğal dil işleme çok aktif bir şekilde kullanılmadığı düşünülürse boyutun kontrol edilemeyeceği göz önünde bulundurulmalıdır. Dolayısıyla terimlerin dökümanlarda elde edilen her bir local ağırlığınının dökümanın uzunluğunca aritmetik ortalaması alındığında, dökümana ait bir local ağırlık ortalamasından söz etmek mümkündür. Bu sayede her doküman için local ve global ağırlıklar ise aritmetik ortalamadan geçirilerek hem dinamik hem de ölçekli bir ortalama ağırlık elde edilmiş olur. Matematiksel formu ise aşağıdaki gibidir:

$$documents_{local\ or\ global\ score} = \frac{document\ items\ local\ or\ global\ score}{length(document)}$$

Bu sayede her bir dökümanın 2 paremetreli matrisi elde edilmiş olur. Elde edilen matrisin matematiksel formu ve matris çıktısı aşağıdaki gibidir:

$$documents_n * \left(L(i,j)_{avarage}, G(i)_{avarage}\right) = \begin{bmatrix} a_{ij} & L(i,j)_{avg} & G(i)_{avg} \\ D_0 & 0{,}34784 & 0{,}74422 \\ D_1 & 0{,}09442 & 0{,}45577 \\ \vdots & \vdots & \vdots \\ D_n & \cdots & 0{,}14487 \end{bmatrix}$$

Elde edilen matrisle birlikte her bir dökümanın temsilcisi (ayırt ediciliği) bulunmuştur. Ayrıca logaritma tabanlı ağırlıklandırma yöntemleri kullanıldığı için 0 ile 1 arasında normalize değerler alındığı gözlemlenebilmektedir. Bu skorların da çarpımı aslında dökümanın, doküman yığını içindeki ortalama ağırlıklandırmaya bağlı ayırt edicilik değerini verir diyebiliriz.

*Code 4*

```
>>> lij = []
>>> gi = []
>>> for i in range(len(kelimeler)):
>>>     ortalama_lij = 0
>>>     ortalama_gi = 0
>>>     for j in range(len(kelimeler[i])):
```

```
>>>         ortalama_lij += agirlik["Lij"][kelimeler[i][j]]
>>>         ortalama_gi += agirlik["Gi"][kelimeler[i][j]]
>>>     ortalama_lij = ortalama_lij / len(kelimeler[i])
>>>     ortalama_gi = ortalama_gi / len(kelimeler[i])
>>>     lij.append(ortalama_lij)
>>>     gi.append(ortalama_gi)
>>> data = zip(lij, gi)
>>> data_agirlik = list(data)
>>> agirlik = pd.DataFrame(data = data_agirlik, index = dokumanlar[0:-1]
, columns = ["Lij","Gi"])
>>> agirlik
```

| | Lij | Gi |
|---|---|---|
| **D0** | 0.017211 | 0.421588 |
| **D1** | 0.019134 | 0.409237 |
| **D2** | 0.004382 | 0.380573 |
| **D3** | 0.015387 | 0.438197 |
| **D4** | 0.004382 | 0.380573 |
| **...** | ... | ... |
| **D272** | 0.045087 | 0.387748 |
| **D273** | 0.027325 | 0.458976 |
| **D274** | 0.029077 | 0.438558 |
| **D275** | 0.004382 | 0.380573 |
| **D276** | 0.033765 | 0.364382 |

277 rows × 2 columns

Code 4'te ise yukarıda matematiksel formu verilen matris işleminin program kodu ve çıktısı mevcuttur. Matrisin boyutu $2 * len(documents)$ olmuştur ve makine öğrenmesi yöntemlerine hazır bir haldedir. Hazır olmasının yanı sıra bir de boyutu sebebiyle makineyi yormayacak düzeyde bir veri seti elde edilmiştir denebilir.

## REFERANS VE KAYNAKÇA

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

[2] BeautifulSoap kütüphanesi dökümanları. Link için tıklayınız.

[3] Pandas Documentation. Link için tıklayınız.

# EKLER

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