BİTİRME PROJESİ

Haftalık Rapor – 13.05.2022

13 MAYIS 2022

KIRIKKALE ÜNİVERSİTESİ – BİLGİSAYAR MÜHENDİSLİĞİ İÖ

AHMET MUNGAN – 160255081

İÇİNDEKİLER

ÖZET

Metin madenciliğinde önemli bir dönüm noktası olan terim ağırlıklandırma üzerinde durulmuştur. Uygulanan metodların çıktıları gösterilmiştir. Bu metodlar matematiksel alt yapılara dayandırılarak formüller ile açıklanmaya çalışılmıştır. Vektörel dönüşümler yapılmadan, ağırlıklar makine öğrenmesi modellerine uygun hale getirilmeye çalışılmıştır. Alt parametreler program kodu yardımıyla hesaplanıp, nasıl hesaplandığına dair otomatize edilmeden sunulmuştur.

TERİM AĞIRLIKLANDIRMAYA GİRİŞ

Metin madenciliğinde önemli bir süreç olarak metin dönüşümü, projenin ilerleyişinde önem arz etmektedir. Makine dilinin metinsel ifadeleri anlamadığı düşünüldüğünde, string yapısında veriler tek başına işlenebilir değildir. Dolayısıyla bu metinlerin sayısal ve makinenin anlayacağı dilden veri yapılarına döndürmek gerekir. Yapılan dönüşümlerde metin madenciliğinin temellerinden olan ağırlıklandırma başta olmak üzere, matematiksel ifadeler ile kanıtlanabilir ve desteklenebilir olması önemlidir.

Kullanılacak ağırlıklandırmalar normal metin madenciliği uygulamalarında TF IDF çarpımlarıyla ya da tek başına frekans tutucu yapılar üzerinden ilerler. Fakat bu projede çıktıların direkt bir şekilde makine öğrenmesi algoritmalarına da uygun olması sebebiyle, çıktılar işlevsel ve anlamsal olmalıdır. Ağırlıklandırmayı incelediğimizde aslında iki farklı ağırlıklandırma literatürde karşımıza çıkmaktadır. Yerel ağırlıklandırma ve global ağırlıklandırma olmak üzere iki farklı kriterin çarpımıyla, metin madenciliğinin sonraki adımı olan özellik seçimi için fikir verir. Burada ağırlıkların nasıl hesaplanacağı ise projenin bel kemiğini oluşturmaktadır.

Yerel ağırlıklandırma için logaritmadan yardım alınacaktır. Logaritma ile yerel ağırlıklandırma aynı zamanda gelen sayıların normalize edilmiş bir hale çevireceğinden, daha standart bir yapı kurulması için ön ayak olacaktır. Bu sayede normalize edilmiş sayıların maliyeti gözle görünür oranda düşüreceğinden bahsedilebilir. Aşağıda bu logaritmik ifadenin parçalı fonksiyonu verilmiştir. Bu fonksiyonda ikinci parça, yani terimin dökümanda hiç geçmemesi durumu olmayacaktır. Çünkü hazır bir kelime çantası ya da terim doküman matrisi mevcut değildir. Kelime çantası gibi vektörel bir yapıya geçmeden ağırlıkların hesaplanması ise maliyeti düşüren etkenlerden olacaktır.

$$L(i,j) = \begin{cases} \log(tf_{ij} + 1) & if \quad tf_{ij} > 0 \\ 0 & if \quad tf_{ij} = 0 \end{cases}$$

Global ağırlıklandırma için ise kimyada karşımıza çıkan ve lineer sistemlerin etkisiyle biraz olsun değişmiş olan entropi formülünden yararlanılacaktır. Entropi

formülü aynı zamanda olasılıksal bir ifade barındırdığından, bayes teoremi gibi uygulamalarda da kullanılabilir. Fakat bir öğrenme olmayacağı için, sadece kümeleme algoritmaları kullanılacağı için, metin özetinde bayes teoremi kullanmak neredeyse mümkün değildir. Kullanıldığı durumların literatürde yerleri mevcuttur fakat veri seti bu kullanımı engelleyebilir. Veri seti etiketsiz ve özetin belirli bir kalıpta olmadığı düşünülürse, doğrulayıcı mekanizmaların olmaması bu tercihi zorunlu kılmaktadır. Aşağıda entropi metodunun nasıl hesaplandığına dair de bir formül bulunmaktadır.

$$P_{ij} = \frac{tf_{ij}}{gf_i} \quad , \quad G(i) = 1 + \sum_{j}^{N} \frac{P_{ij} * \log_2(P_{ij})}{\log_2(N)}$$

Program Ve en son bir kelimenin ayırt ediciliğini öğrenmek için, kelimeye ait yerel ağırlık ve global ağırlık çarpılarak nümerik bir ayırt edicilik değeri elde edilir. Bu değere çarpımdan gelme ifadeler de eklenince aşağıdaki gibi bir işlem formülü görülmektedir.

$$a_{ij} = L(i,j) * G(i)$$

Formüllerdeki ifadeler ele alınırsa;

a_{ij} → ayırt edicilik katsayısı (i. terimin j. dökümandaki ağırlığı),

tf_{ij} → i. terimin j. dökümanda toplam geçme sayısı,

gf_i → i. terimin doküman yığınında toplam geçme sayısı,

N → toplam doküman sayısı,

$L(i,j)$ → yerel ağırlık,

$G(i)$ → global ağırlıktır.

Formül ve bileşenlerinin açıklamalarından çıkarımla, bu tarzda bir ağırlıklandırma yapan hazır Python kütüphanesi bulmak neredeyse mümkün değildir. Fakat elde matematiksel hesabı olan bir durumun kütüphanesini yazmak da pek zor bir iş değildir.

Global Frekans İçin Alt Değerlerin Hesaplanması (gfi)

Öncelikle ağırlıkların hesaplanması için formüllerde geçen alt değerler program kodu yardımıyla otomatize hesaplanmalıdır. Burada öncelikle doküman yığının işlenmeye uygun yapıda olduğu unutulmamalıdır. Bu uygun yapıdan kasıt, cümlelerin kelimelere bölünmesidir.

Code 1

```
>>> for i in range(len(kelimeler_split)):
>>>     for j in range(len(kelimeler_split[i])):
>>>         if kelimeler_split[i][j] not in kelimeler_data:
>>>             kelimeler_data[(kelimeler_split[i][j])] = 1
>>>         elif kelimeler_split[i][j] in kelimeler_data:
>>>             kelimeler_data[(kelimeler_split[i][j])] += 1
>>> gfi = pd.Series(kelimeler_data)
>>> gfi
ömer         6
besim        5
usta         2
küçük        1
liseye       2
            ..
bittikten    1
sonra        1
ailesini     1
de           1
alarak       1
Length: 115, dtype: int64
```

Code 1'de[1] yazılan kod ile dökümanda bulunan kelimelerin sadece frekansı hesaplanmış ve pandas serisi haline getirilmiştir. Burada formüllerden hareketle gf_i değeri hesaplanıp, terimlerin doküman yığınında toplam geçme sayıları hesaplanmıştır. İstenirse bu sayılara göre seride sıralama yapılabilir. Bu sıralama maliyet açısından düşünüldüğünde quick sort veya maliyeti quick sort'a yakın bubble sort ile yapılabilir. Fakat bunun gerekli olup olmadığı ilerleyen süreçlerde kesinleşeceği için şimdilik bu yapı korunmuştur.

Yerel Frekans İçin Alt Değerlerin Hesaplanması (tfij)

Yerel frekans için de alt değerler tıpkı global frekans için hesaplandığı için birim bazında hesaplanmıştır.

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

Code 2

```
>>> ornek_kelime = "ömer"
>>> ornek_kelime_sayisi = []
>>> for i in range(len(kelimeler_split)):
>>>     ornek_kelime_sayisi.append(0)
>>>     if ornek_kelime not in kelimeler_split[i]:
>>>         pass
>>>     elif ornek_kelime in kelimeler_split[i]:
>>>         ornek_kelime_sayisi[-1] +=  1
>>> ornek_kelime_sayisi
[1, 0, 2, 1, 1, 0, 0, 3, 0, 1, 0, 0]
```

Code 2'de görüldüğü üzere "ömer" teriminin dökümanlarda sırasıyla, 1. dökümanda 1 kere, 2. dökümanda 0 kere, … 8. dökümanda 3 kere geçtiği saydırılmıştır. Örnek kelimeleri otomatize etmeden, sonuçların görünmesi için Code 2 önemli bir ölçüttür. Burada hesaplanması hedeflenen parametre tf_{ij} dir. Formülden terimlerin dökümanda geçme sayısı olarak bildiğimiz tf_{ij} parametresi de bu şekilde hesaplanmıştır.

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.)