Klasik Kelime Vektoru Cikarim Yontemleri ve Kosinus Benzerliklerine Gore Yetenekten Meslek Grubunu Bulma

Havvanur Dervişoğlu,drvshavva@gmail.com

September 12, 2019

1 Çalışmanın Tanımı ve Kullanılan Araçlar

Bu calismadaki adimlar:

- Veri setinde olan genel nitelikler ve is tanimi kismindan istenilen yetenklerin el ile cikarilmasi.
- Olusturulan veri setine preprocessing uygulanmasi.
- On islem asamasindan gecmis veri setindeki skills ozelligine feature extraction uygulanmasi.
- Oznitelik cikarimi asamasindan sonra kosinus benzerliklerine gore meslek gruplarindan yetenek gruplarinin cikarilmasi.
- Test veri setindeki yeteneklerin olusturulan yetenek gruplarindaki cosinus benzerliklerine gore skorlama yapilmasi ve ve bu skor degerine gore meslek grubuna atilmasi.

2 Veri Seti

Calismada kullanilan veri setinin yetenekler kismi genel nitelikler ve is tanimindan cikarilmistir. Bu veri setinde ilanlara ait kullanilan ozellikler; label, skill seklindedir. Siniflardaki ornek sayilari asagida gosterildigi gibidir:

1. is-analizi-raporlama : 424 yetenek
2. sap: 672 yetenek
3. web-tasarim: 487 yetenek

Veri seti 0.9 train ve 0.1 test olarak ayrilmistir. Mesleklerin yetenek gruplarini cikarirken train yetenekler kullanilistir. Test ornekleri uzerinde yeteneklerden meslek grubu bulma yapilmistir.

3 Preprocessing

Yetenekleri cikarmak icin kullanilan veri seti oncesinde asagida belirtilen islemlerden gecmistir. O yuzden bu calismada preprocess asamasinda tekrar yapilmasina ihtiyac duyulmamistir(Figure 1).

1. Harfleri kucuk harfe cevirme
2. Turkce karakter harfleri ingilizce karaktere cevirme
3. Noktalama isaretleri vb isaretleri atma
4. Rakamlari silme
5. Cumleyi kelimelerine ayirip stop words olup olmadigini kontrol etme ve eger stop word ise almama
6. Kelime uzunlugu 3 den kucuk olanlari alma(amac kesme isareti ile ayrilmis kelimelerden kurtulma nin nin gibi)
7. Kelimler arasi bir bosluk olacak sekilde birlestirme.

	genel_nit_is_tanimi	ilan_baslik	label	skills
0	universitelerin bilgisayar muhendisligi bilgis...	front developer	web_tasarim	html, javascript, jquery, github, photoshop il...
1	hedefini dunyanin dort tarafinda hayata gecirm...	yazilim uzmani	web_tasarim	wordpress, mysql, html, javascript
2	universitelerin endustri muhendisligi sletme m...	anali	web_tasarim	raporlama, veri analizi,office programlari,eki...
3	hedefini dunyanin dort tarafinda hayata gecirm...	junior yazilim uzmani	web_tasarim	wordpress, mysql, html, digital pazarlama
4	tanimi html taslaklari gereksinimleri gercekle...	gelistirme uzmani	web_tasarim	analitik dusunen, html, angular, jquery, front...

Figure 1: Veri setinin gorunumu

Bu calismada on islem olarak yan yana olan yetenekleri alt alta olacak sekilde yazilmasi oldu. Bunun icin yapilan islem Figure 2'de gosterilmistir. Figure 2'de ki program parcasinda yetenekler virgule gore ayrilmaktadir.

```
[ ]  skills=[]
     s_label = []
     ilan_no = []
```

```
[ ]  for i in range(0,len(data)):
       label = data['label'][i]
       text = data['skills'][i]
       x = text.split(',')
       for xe in x:
         skills.append(xe)
         s_label.append(label)
         ilan_no.append(i)
```

Figure 2: Parse skills

Veri setinin son gorunumu Figure 3'deki gibidir.

skill	label
html	web_tasarim
javascript	web_tasarim
jquery	web_tasarim
github	web_tasarim
photoshop illustrator	web_tasarim

Figure 3: Veri setinin son hali

4 Feature Extraction

Calismada iki farkli oznitelik cikarimi yontemi kullanilmistir; tf-idf ve count vectorizer. Bu iki yontemi de kullanmadan once Figure 4'teki gibi her satirdaki yetenekler tokenlarina ayrilmistir.

```
%pyspark
#OZNITELIK CILARIMI UYGULANACAK SUTUN KELIMELERE BOLUNUYOR
tokenizer = Tokenizer(inputCol="skill", outputCol="words")
wordsData = tokenizer.transform(df)
wordsData.show()
```

```
+--------------------+-----------+--------------------+
|               skill|      label|               words|
+--------------------+-----------+--------------------+
|                html|web_tasarim|              [html]|
|          javascript|web_tasarim|        [javascript]|
|              jquery|web_tasarim|            [jquery]|
|              github|web_tasarim|            [github]|
|photoshop illustr...|web_tasarim|[photoshop, illus...|
|           wordpress|web_tasarim|         [wordpress]|
|               mysql|web_tasarim|             [mysql]|
```

Figure 4: Tokenlarina ayirma

Figure 5'de tf-idf oznitelik cikarimi yonteminin kullanimi gosterilmistir.

```
%pyspark
#OZNITELIK CIKARIMI OLARAK TF-IDF YONTEMI KULLANILDI
hashingTF = HashingTF(inputCol="words", outputCol="rawFeatures", numFeatures=1000)
featurizedData = hashingTF.transform(wordsData)

idf = IDF(inputCol="rawFeatures", outputCol="features")
idfModel = idf.fit(featurizedData)
result = idfModel.transform(featurizedData)
result.show()
```

```
+--------------------+-----------+--------------------+--------------------+--------------------+
|               skill|      label|               words|         rawFeatures|            features|
+--------------------+-----------+--------------------+--------------------+--------------------+
|                html|web_tasarim|              [html]|  (1000,[652],[1.0])|(1000,[652],[3.30...|
|          javascript|web_tasarim|        [javascript]|  (1000,[281],[1.0])|(1000,[281],[3.41...|
|              jquery|web_tasarim|            [jquery]|  (1000,[880],[1.0])|(1000,[880],[4.07...|
|              github|web_tasarim|            [github]|  (1000,[510],[1.0])|(1000,[510],[6.26...|
|photoshop illustr...|web_tasarim|[photoshop, illus...|(1000,[791,910],[...|(1000,[791,910],[...|
|           wordpress|web_tasarim|         [wordpress]|  (1000,[717],[1.0])|(1000,[717],[4.88...|
```

Figure 5: Tf-idf oznitelik cikarim yontemi

Figure 6'da cv oznitelik cikarimi yonteminin kullanimi gosterilmistir.

```
%pyspark
#OZNITELIK CIKARIMI OLARAK COUNT_VECTORIZER YONTEMI KULLANILDI
cv = CountVectorizer(inputCol="words", outputCol="features", vocabSize=1000)
model = cv.fit(wordsData)

result = model.transform(wordsData)
result.show()

+--------------------+-----------+--------------------+--------------------+
|               skill|      label|               words|            features|
+--------------------+-----------+--------------------+--------------------+
|                html|web_tasarim|              [html]|     (778,[7],[1.0])|
|          javascript|web_tasarim|        [javascript]|    (778,[12],[1.0])|
|              jquery|web_tasarim|            [jquery]|    (778,[20],[1.0])|
|              github|web_tasarim|            [github]|   (778,[255],[1.0])|
|photoshop illustr...|web_tasarim|[photoshop, illus...|(778,[26,60],[1.0...|
|           wordpress|web_tasarim|         [wordpress]|    (778,[56],[1.0])|
|               mysql|web_tasarim|             [mysql]|    (778,[45],[1.0])|
|                html|web_tasarim|              [html]|     (778,[7],[1.0])|
|          javascript|web_tasarim|        [javascript]|    (778,[12],[1.0])|
|           raporlama|web_tasarim|         [raporlama]|     (778,[2],[1.0])|
```

Figure 6: CV oznitelik cikarim yontemi

5 Kosinus Benzerligine Gore Yetenekleri Gruplandirma

Bu calismada amacimiz yetenek ozellikleri cikarilan meslek ilanlarindan, yeteneklerine bagli olarak hangi meslek grubunda oldugunu bulabilmektir. Bunun icin:(Figure 8)

- Oncelikle meslekleri filtreledik,olusturulacak yetenek gruplarinin hangi meslek grubuna ait oldugunu belirlemek icin.
- Dataframe de kullanilacak olan ozellikleri belirledik.
- Dataframeler satirlara donusuturuldu.

Calismada kullanilan kosinus benzerlik fonksiyonu Figure 7'dedir.

```
%pyspark
#COSINUS BENZERLIGINI BULMAK ICIN KULLANILAN FONKSIYON
import numpy as np
def cos_sim(a,b):
    dot_product = np.dot(np.array(a),np.array(b))
    norms_product= np.linalg.norm(np.array(a))*np.linalg.norm(np.array(b))
    return dot_product/norms_product
```

Took 0 sec. Last updated by admin at September 04 2019, 10:01:22 AM.

Figure 7: Kosinus benzerligi fonksiyonu

MESLEK FILTRELEME

```
%pyspark
```

```
%pyspark
#HER GRUP ICIN DF'DE FILTRELEME YAPILIYOR
web =train.filter(result.label.like('web_%'))
is_analiz = train.filter(result.label.like('is_analizi%'))
sap = train.filter(result.label.like('sap'))
```

Took 0 sec. Last updated by admin at September 06 2019, 2:55:36 PM.

OZELLIK SECME

```
%pyspark
```

```
%pyspark
#GRUPLANDIRMAK ICIN GEREKLI OLAN OZELLIKLERI SECIYORUZ
featured_web = web.select('skill','features')
featured_sap = sap.select('skill','features')
featured_is_analiz = is_analiz.select('skill','features')
```

Took 0 sec. Last updated by admin at September 06 2019, 2:55:36 PM.

SATIRLARI LISTEYE DONUSTURME

```
%pyspark
```

```
%pyspark
#DF'I SATIR SATIR OLACAK SEKILDE LISTEYE CEVIRIYORUZ
web_rows =featured_web.collect()
sap_rows = featured_sap.collect()
is_rows = featured_is_analiz.collect()
```

Figure 8: DF'den satirlara donusturme

Her meslek grubunun icinde benzer yetenekleri iceren yetenek gruplari olusturulmustur(Figure 9). Figure 9'daki kod parcasinda meslek grubundaki butun satirlari tek tek diger satirlarla benzerligine bakip eger belirlenen benzerlik oranindan fazla ise benzerligi o yetenek grubuna atiliyor ve yetenek grubu belirlenen yetenegi tum satiirlari iceren listeden cikariyoruz.

```
%pyspark
#LABEL BILGISI VE O LABELA AIT ROWLAR YOLLANILDIGINDA O LABELA AIT OLUSAN GRUPLARI DONDURUR
def gruping_skills(rows,label):
    i = 0
    grup_names = []
    print(str(label)+' icin gruplandirma baslatildi...')
    while rows: #TUM ROWLARI BITENE KADAR DOLASIR
        skill = rows[0][1] #DIGER OZELLIKLER ILE KARSILASTIRILACAK OLAN YETENEK
        grup_names.append(str(rows[0][0])+'_grup')
        grup_names[i] = [] #ITH GRUBA EKLENECEK OLAN SATIRLAR BURADA TUTULACAK
        silinecekler = []   #ITH GRUBA EKLENEN SATIRLARI TUM SATIRLARDAN SILIYORUZ
        print('gruplandirilmayan '+str(len(rows))+' skill kaldi...')
        for j in range(0,len(rows)): #YETENEK TUM DIGER YETENEKLER ILE KARSILASTIRILIR
            sim = cos_sim(skill,rows[j][1])
            if sim >= 0.6: #BENZERLIK 0.6 DAN FAZLA ISE
                grup_names[i].append(rows[j]) #O SATIRI GRUBA EKLIYORUZ
                silinecekler.append(rows[j])  #O SATIRI TUM SATIRLARDAN SILIYORUZ
        for item in silinecekler:
            rows.remove(item) #O SATIRI TUM SATIRLARDAN SILIYORUZ
        i += 1
    print('Gruplandirma tamamlandi...')
    return grup_names
```

Figure 9: Yetenekleri cos benzerliklerine gore gruplandirma

Her meslek grubu icin yeteneklere gore gruplandirma islemi yapiyoruz(Figure 10).

```
%pyspark
#SAP MESLEK GRUBU ICIN YETENEK GRUPLARI OLUSTURULDU
sap_grups = gruping_skills(sap_rows,'sap')
```

```
sap icin gruplandirma baslatildi...
gruplandirilmayan 672 skill kaldi...
gruplandirilmayan 662 skill kaldi...
gruplandirilmayan 661 skill kaldi...
gruplandirilmayan 660 skill kaldi...
gruplandirilmayan 659 skill kaldi...
gruplandirilmayan 658 skill kaldi...
gruplandirilmayan 657 skill kaldi...
gruplandirilmayan 655 skill kaldi...
gruplandirilmayan 652 skill kaldi...
gruplandirilmayan 617 skill kaldi...
gruplandirilmayan 596 skill kaldi...
gruplandirilmayan 577 skill kaldi...
gruplandirilmayan 576 skill kaldi...
gruplandirilmayan 575 skill kaldi...
gruplandirilmayan 574 skill kaldi...
gruplandirilmayan 571 skill kaldi...
gruplandirilmayan 570 skill kaldi
```

Took 34 min 32 sec. Last updated by admin at September 04 2019, 10:45:18 AM.

Figure 10: Meslek icin yetenek gruplari olusturma

Figure 11'de olusan yetenek gruplarindan birinin ornegi bulunmaktadir.

```
[Row(skill=u'adobe', features=SparseVector(643, {17: 1.0})), Row(skill=u'adobe ', features=SparseVector(643, {17: 1.0})), Row(skill=u'adobe after effects', features=SparseVector(643, {17: 1.0, 456: 1.0, 459: 1.0})), Row(skill
0})), Row(skill=u'adobe indesign', features=SparseVector(643, {17: 1.0, 109: 1.0})), Row(skill=u'adobe photoshop', features=SparseVector(643, {13: 1.0, 17: 1.0})), Row(skill=u'adobe photoshop', features=SparseVector(643, {13:
(643, {13: 1.0, 17: 1.0})), Row(skill=u'adobe premiere dreamweaver', features=SparseVector(643, {17: 1.0, 512: 1.0, 575: 1.0}))]
```

Figure 11: Ornek yetenek grubu

6 Test Orneklerinin Yeteneklerine Gore Meslek Grubunu Bulma

Calismada yeteneklere gore meslek grubunu bulmak icin her yetenek grubundaki yetenek tekrarlarindan kurtulunmustur(Figure 12).

```
%pyspark
for i in range(0,len(web_grups)): #butun yetenek gruplarinda
    new = []
    for x in web_grups[i]: #ith yetenek grubuna giriyor
        if x not in new:
            new.append(x)
    web_grups[i] = new
```

Figure 12: Yetenek grubundaki yetenekleri tekillestirme

Figure 13'deki kod parcasinda meslek grubu aranan yetenek, arama yapilacak meslegin yetenek gruplarinin elemanlarindan ilkine olan benzerligini buluyor ve sonrasinda benzerlik orani kadar meslek grubunun puanini artiriyor(ilk meslek grubuna bakilmasindaki sebep o yetenek grubundaki her yetenegin benzerliklerinin ilk yetenege olan benzerliginin yuksek olmasidir.).

```
%pyspark
#BELIRTILEN OZELLIK ICIN GRUPTAKI SCORERUNU KONTROL  EDIYOR
def get_grup_score(grup,ilan_feature):
    score = 0
    #MESLEK ICIN OLUSTURULAN TUM GRUPLARI DOLASIYOR
    for i in range(0,len(grup)):
        g_score = 0 #HER GRUP ICIN ILK DEGER 0
        sim = cos_sim(grup[i][0][1], ilan_feature) #GIRILEN HER GRUBUN ILK ELEMANINI KONTROL EDIYOR
        if sim >= 0.5:
            print('grup:',grup[i][0][0])
            g_score += sim #EGER KOSULDAKI BENZERLIK ORANI TUTUYORSA SCORE benzerlik orani kadar art
        score += g_score #KAC YETENEK GRUBUNA BENZEDIYSE O KADAR ARTIYOR
    return score
```

took 0 sec. Last updated by admin at September 11 2019, 8:36:34 AM

Figure 13: Yetenek grubundan benzerlik puanini bulma

Figure 14'de, yeteneklerden olusan bir listedeki her yetenek meslek gruplarindaki benzerliklerine gore puanlama yapiliyor.

```
%pyspark
scores = []
print('Scorelama baslatildi.....')
print('[web_tasarim, sap, is_analizi]')
for i in range(100): #TUM YETENLERI DOLASIYOR
    print('YETENEK:',ilans[i][0]) #YETENEK NEDIR
    print('MESLEK:',ilans[i][2])#YETENEK HANGI MESLEK GRUBUNDA

    score = [0,0,0]  #HER MESLEK GRUBUNUN SCORLARI SIFIRLANIYOR
    try:
        score[0] = get_grup_score(web_grups,ilans[i][1]) #ILK MESLEK GRUBUNDAN ALINA SCORE
        score[1] = get_grup_score(sap_grups,ilans[i][1])
        score[2] = get_grup_score(is_grups,ilans[i][1])
        scores.append(score)  #ILK YETENEK ICIN OLUSAN SONUCLAR SCORES EKLENIYOR
        print(score)

    except:
        print('Scorelama sonlandirildi...')
        break
```

```
Scorelama baslatildi.....
[web_tasarim, sap, is_analizi]
('YETENEK:', u'abap')
('MESLEK:', u'sap')
('grup:', u'abap')
[0, 1, 0]
```

Figure 14: Yeteneklerin meslek gruplarindaki puanlarini bulma

Calismada iki farkli dogruluk orani bulunmustur; ilkinde test yetenegin meslek grubundaki yetenek gruplarindan herhangi birinde sadece 0.5 oraninda benzerlik gostermesi yeterliyken(Figure 15), ikincisinde test yetenegin mesleginin yetenek gruplarina olan benzerliginin diger mesleklerin yetenek gruplarina olan benzerligden fazla olmasi beklenmektedir(Figure 16).

```
%pyspark
#tahmin edilen orneklerden kac tanesinin dogru siniflandirildigini buluyoruz
#j=0-> web_tasarim, j=1 -> sap, j=2 -> is_analizi
true = 0 #toplam dogru tahmin sayisini tutuyor
for i in range(0,len(scores)):
    if str(ilans[i][2]) == 'web_tasarim':
        if scores[i][0] >= 0.5: #web tasarimin score varsa dogru tahmin
            true += 1
    if str(ilans[i][2]) == 'sap':
        if scores[i][1] >= 0.5: #sap score varsa dogru tahmin
            true += 1
    if str(ilans[i][2]) == 'is_analizi_raporlama':
        if scores[i][2] >= 0.5: #is_analizi_raporlama score varsa dogru tahmin
            true += 1
print(len(scores),' TANE YETENEKTEN :',true,' TANE ORNEK DOGRU SINIFLADIRILMISTIR')
```

```
100, ' TANE YETENEKTEN :', 77, ' TANE ORNEK DOGRU SINIFLADIRILMISTIR')
```

Figure 15: Dogru meslek grubunu tahmin edip etmedigini kontrol etme-1

```
%pyspark
#tahmin edilen orneklerden kac tanesinin dogru siniflandirildigini buluyoruz
#j=0-> web_tasarim, j=1 -> sap, j=2 -> is_analizi
maxi = 0
true = 0
for i in range(0,len(scores)):
    maxi = scores[i][0]
    loc = 0
    for j in range(1,len(scores[i])):
        if scores[i][j] > maxi:
            maxi = scores[i][j]
            loc = j
    if str(ilans[i][2]) == 'web_tasarim':
        if loc == 0: #web tasarimin score varsa dogru tahmin
            true += 1
    if str(ilans[i][2]) == 'sap':
        if loc == 1: #sap score varsa dogru tahmin
            true += 1
    if str(ilans[i][2]) == 'is_analizi_raporlama':
        if loc == 2: #is_analizi_raporlama score varsa dogru tahmin
            true += 1
print(len(scores),' TANE YETENEKTEN :',true,' TANE ORNEK DOGRU SINIFLADIRILMISTIR')


100, ' TANE YETENEKTEN :', 62, ' TANE ORNEK DOGRU SINIFLADIRILMISTIR')
```

Figure 16: Dogru meslek grubunu tahmin edip etmedigini kontrol etme-2

7 Sonuc

Calismada iki farkli oznitelik cikarimi yontemi kullanilmistir. Sonuclar karsilastirildiginda:

1. TF-IDF ve count vectorizer olusturula 'adobe' yetenegi gruplari Figure 17'de gosterilmistir. Karsilastildiginda count vectorizer ile olusturulmis olan yetenek grubunun daha basarili oldugunu soyleyebiliriz.

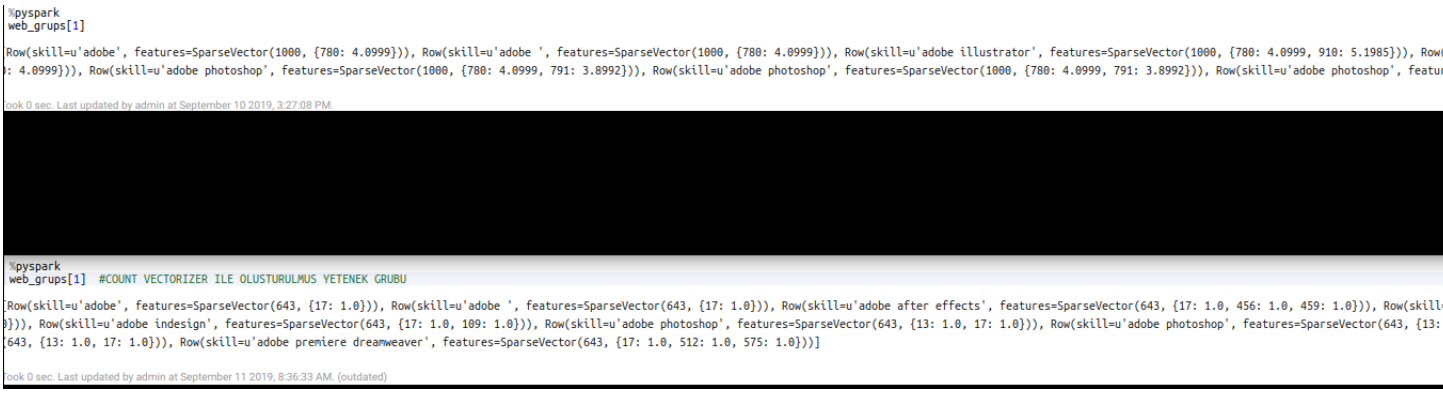

Figure 17: Farkli kelime vektorleri yetenek gruplari karsilastirma

2. Meslek gruplarini tahminlerini karsilastirdigimizda ise iki oznitelik cikarimi yontemi de ayni asariyi gostermistir; ilk basari olcumu yontemi ,yetenegin sinifina 0,5 oraninda benzerlik gostermesi yeterli, 100 test yeteneginden 77 tanesini ayni sinifa koymustur ve ikinci basarimi olcumunde ,o yetenek grubuna benzerlik orani digerler meslek gruplarini olan benzerliginden fazla, 100 ornekten 62 tanesini dogru sinifa atmistir.

3. Sonuclarin alinmasi Figure 18'deki gibidir.YETENEK ile belirtilen meslek grubu aranan yetenek, MESLEK ile belirtilen yetenegin gercek benzerligi ve gruplar ise benzerlik gorulen yetenek gruplari. Dizinin ilk puani web tasarimi meslek grubu yeteneklerine benzerligi, ikincisi sap meslek grubuna benzerligi ve ucuncusu ise is analizi raporlama meslek grubuna benzerlik oranini gostermektedir.

```
('YETENEK:', u'akis diyagramlari')
('MESLEK:', u'is_analizi_raporlama')
('grup:', u'iveri akis diyagramlari')
('grup:', u'akis aktivite diyagramlari')
[0, 0.8164965809277259, 0.8164965809277259]
('YETENEK:', u'analitik dusunen')
```

Figure 18: Yetenek hangi gruba hangi oranda benzedi

7.1 Neler Yapilabilir?

Bu calismaya ek olarak neler yapilarbilir?

- Yetenekleri elle ayiklarken harcanan zaman fazlaydi buna alternatif olarak elde edilen genel nitelikler ve is tanimi metnini tokenlara ayirip yetenek olarak kullanirsak belki daha kirli yetenek gruplari elde ederiz ama elle ayiklarken kaybedilen zamandan tasarruf etmis oluruz. Buna ek olarak butun meslek gruplarinda benzer olan yetenekleri tum meslek gruplarindan silerek belki daha iyi sonuc alinabilir.

- Benzerlik bulurken tum yeteneklere ikili olarak bakildigi icin cok zaman alan bir islem. Benzerlikleri bu sekilde bulmak yerine tekrar eden yetenekleri ilk dataframeden silip(df.dropDuplicates()komutu ile) ilerlenebilir ama bu sekilde hangi meslek grubunda hangi yeteneklerin on plana ciktigi konusunda bir fikir sahibi olamayiz.

- Figure 19'da de goruldugu gibi test yeteneklerinden bazilari herhangi bir yetenek grubuyla benzerlik gostermemistir, buna cozum olarak mesleklerin yetenek gruplarini olusturmak icin kullanilan veri setindeki yetenek ornekleri artirmayi dusunebiliriz.

```
('YETENEK:', u'devops')
('MESLEK:', u'web_tasarim')
[0, 0, 0]
('YETENEK:', u'diagram')
('MESLEK:', u'is_analizi_raporlama')
[0, 0, 0]
('YETENEK:', u'disiplinli calisma')
('MESLEK:', u'is_analizi_raporlama')
[0, 0, 0]
('YETENEK:', u'dokumantasyon')
('MESLEK:', u'web_tasarim')
[0, 0, 0]
```

Figure 19: Yetenek gruplariyla eslesmeyen yetenekler

- Yetenekler etiket haline getirilip klasik makine ogrenmesi yontemleri kullanilarak siniflandirma yapilabilir, test ornek tanimlari tokenlarina ayrilip her meslek sinifi icin egitilmis modellerde testini yaptigimizda en yuksek olasilik degerini veren sinifa atilmalidir.(her meslek grubu kendi icerisinde yetenek gruplari etiket olacak sekilde egitilmelidir. Bunun icin her sinif icinde her yetenekten yeterli ve esit miktarda olamlidir.)

8 TEST ILANININ SINIFININ BULUNMASI

Bu baslikta web sitesinden cekilen bir ilanin ,onceden preprocess isleminden gecmis olmasi gerekli, el ile yeteneklerini ayirmak yerine ilani kelimelere bolup ilanin kelimelerini yetenek olarak kullanilmistir. Figure 20'deki ilan icin yapilan testte meslek sinifini dogru bulmustur.

```
'tanimi pazarlama ekibinin sorumlu oldugu alanlar dahilinde olusan projeler iyilestirme talepleri icin gereken tasarim analizleri yapmak birimlerden g
stemlerin performansini takip etmek verim artirici iyilestirmeleri planlamak birimleri analistlerle araya gelerek iyilestirici faaliyetler gelistirmek
rini etkileyen anahtar kriterleri belirlemek analiz etmek ilgili konularda anketler duzenlemek onlari analiz etmek hedeflerine ulasmak icin yapisal de
etkili reklam kampanyalari gelistirmek icin pazarlama departmanlari birligi icerisinde calismak gereksinimlerini tanimlamak ilgili birimlere raporlama
yde olan seyahat engeli olmayan konusunda bilgili laninda senelik deneyim sahibi ekip calismasina uyumlu maliyet fayda analizi yapabilen letisim anali
sahip coklu tanimlarini onceliklendirebilmek icin zaman yonetimini dogru planlayabilen'
```

Figure 20: Ornek is ilani

Figure 21'de, verilen ilandan yeteneklerin nasil cikarildigini gostern bir kod parcasi vardir.

```
%pyspark
#BU FONKSIYONDA VERILEN TEST ILANINDAN YETENEKLERINI CIKARIYOR
import numpy as np
def ilan_to_skill(test_ilan,ilan_no):
    skiller = []
    s_label = []
    label = str(test_ilan[ilan_no][0])
    skills= str(test_ilan[ilan_no][1]).split()
    for skill in skills:
        skiller.append(skill)
        s_label.append(label)
    return sqlContext.createDataFrame(zip(skiller, s_label), schema=[ 'skill','label'])
```

Figure 21: Ilandan yeteneklerin cikarimi

Figure 22'de, Figure 20'deki test ilan icin elde edilmis dataframe gosterilmektedir.
Figure 23'de, yetenkleri cikarilmis test ilaninindan ozellikleri cikarilmis satirlar elde etmek icin kullanilan fonksiyon vardir. Burada dikkat edilmesi gereken nokta yeteneklerin oznitelliklerini cikarirken yetenek gruplarini olusturuken egittigimiz count vectorizer modelini kullandik.

Figure 24'de ise her meslek grubunun puanlarini elde edip, dogru meslek grubunu bulup bulmadigini kontrol eden bir kod parcasi vardir. Bu ornek icin sinifalndirmayi dogru yapmistir. Ama burada siniflandirmanin basarisi tamamen mesleklerin yetenek gruplarinin sayisinin fazla olamasina baglidir, burada siniflandirma basarisini arttirmak icin yetenek gruplarini elde ettigimiz veri setinin boyutunu arttirmaliyiz ve her meslek grubunda bulunan, sinif seciminde belirleyici rol oynamayan yetenekleri(ingilizce, raporlama yapan gibi) veri setinden temizlemeliyiz.

```
%pyspark
test_df = ilan_to_skill(data,0) #ILK ILAN ICIN LABEL DENEMESI YAPILIYOR
test_df.show()
```

```
+---------------+-----------+
|          skill|      label|
+---------------+-----------+
|universitelerin|web_tasarim|
|     bilgisayar|web_tasarim|
|   muhendisligi|web_tasarim|
|     bilgisayar|web_tasarim|
|  programciligi|web_tasarim|
|         ilgili|web_tasarim|
|  bolumlerinden|web_tasarim|
|          mezun|web_tasarim|
|       derecede|web_tasarim|
|           html|web_tasarim|
|     javascript|web_tasarim|
|         jquery|web_tasarim|
```

Figure 22: Ornek ilanin yetenekleri

```
%pyspark
def ilan_df_to_featured_rows(test_df):
    test_df = test_df.dropna()
    #OZNITELIK CILARIMI UYGULANACAK SUTUN KELIMELERE BOLUNUYOR
    test_tokenizer = Tokenizer(inputCol="skill", outputCol="words")
    test_words = test_tokenizer.transform(test_df)
    #YETENEK GRUPLARINI OLUSTURURKEN EGITTIMIZ COUNT VECTORIZER MODELINE GORE OZELLIKLER CIKARILDI
    features_test = model.transform(test_words)
    features_test = features_test.select('label','skill','features')
    test_rows = features_test.collect()
    return test_rows
```

ook 0 sec. Last updated by admin at September 11 2019, 11:13:24 AM.

Figure 23: Yeteneklerden oznitelikleri cikarilmis satirlara

```
%pyspark
#burada her meslek grubu icin elde edilen scorelar toplaniyor ve eger aralarinda maksimum olanin yeri kendi meslek grubunda ise dogru siniflandirlmis
import numpy as np
#her meslek grubunun ilk scorelari 0
web_toplam_score = 0
sap_toplam_score = 0
is_toplam_score = 0
for i in range(0,len(scores)): #her scores scorelarinin
    web_toplam_score += scores[i][0] #ilk elemanin web scorea
    sap_toplam_score += scores[i][1] #ikinci elemani sap'e
    is_toplam_score += scores[i][2] #ucuncu elemani ise is_analiz score'a
all_scores = [web_toplam_score, sap_toplam_score, is_toplam_score]
print('[web_tasrim,sap,is_analizi]')
print(all_scores)
index = all_scores.index(np.max(all_scores)) #tum scorelardan hangisinin max olanin indeksi bulunur
if str(test_rows[i][0]) == 'web_tasarim':
    if index == 0: #web tasarimin score varsa dogru tahmin
        print('DOGRU SINIFLANDIRILMISTIR')
if str(test_rows[i][0]) == 'sap':
    if index == 1: #sap score varsa dogru tahmin
        print('DOGRU SINIFLANDIRILMISTIR')
if str(test_rows[i][0]) == 'is_analizi_raporlama':
    if index == 2: #is_analizi_raporlama score varsa dogru tahmin
        print('DOGRU SINIFLANDIRILMISTIR')
```

```
[web_tasrim,sap,is_analizi]
[14.966255326250886, 13.093493648205174, 10.899494936611665]
DOGRU SINIFLANDIRILMISTIR
```

Figure 24: Test ilaninin tahmin sonucu