YÖNTEM GELİŞTİRME

İÇİN

TÜRKÇEDE KONUŞMA ETİKETLEMENİN BİR BÖLÜMÜ

ile

Berna Arslan

&

Özlem Patan

Bilgisayar Mühendisliği Bölümüne Gönderildi

şartların kısmen yerine getirilmesi

derecesi için

Fen Fakültesi mezunu

içinde

Bilgisayar Mühendisliği

Boğaziçi Üniversitesi

Haziran 2009

SOYUT

Proje Adı: Türkçe Konuşma Etiketlemenin Bir Kısmına Yönelik Yöntemlerin Geliştirilmesi

Proje Ekibi: Berna Arslan ve Özlem Patan

Terim: 2008/09, II. Dönem

Anahtar Kelimeler: Türkçe için Konuşma Etiketleyicinin bir parçası, Destek Vektör Makineleri, Bir Parçası Konuşma Etiketleme

Özet: Projenin amacı, konuşma etiketlemenin bir kısmı için yöntemler geliştirmektir. Türkçe. Bu çalışma destek vektör makinesi yaklaşımı kullanılarak gerçekleştirilmiştir. Daha önce Türkçe için kullanılmıştı. Ana odak noktası, uygun özellikleri bulmaktır. farklı gramer sınıflarını ayırt etmek ve bu özellikleri bir araçla test etmek Destek vektör makineleri yaklaşımı için geliştirilmiştir, yani SVMLight. Sonuçlar gösteriyor nispeten düşük bir özellik sayısıyla yüzde 75'lik bir başarı seviyesine ulaşıldığını söyledi. Başarı oranı, daha fazla sayıda eğitim kelimesi ve özelliği kullanılarak artırılabilir.

İÇİNDEKİLER

ŞEKİL LİSTESİ

# 1. GİRİŞ

## 1.1 Konuşma Kısmında Etiketleme Nedir?

Konuşma bölümü etiketlemesi, bir metindeki sözcüklerin belirli bir şeye karşılık gelecek şekilde etiketlenmesidir.
Konuşmanın belirli bir kısmı. Kelime kategorisi belirsizliğini giderme olarak da adlandırılan POS etiketleme veya gramer etiketlemesi, hem tanıma hem de bağlama, yani ilişkiye dayanır
Kelimenin cümle, cümle veya paragraftaki bitişik ve ilişkili kelimelerle birlikte kullanılması
POS etiketlemenin basitleştirilmiş bir versiyonu, kelimelerin isim olarak tanımlanmasıdır,
Okul çağındaki çocuklara öğretilen fiiller, sıfatlar vb. cümle içinde kullanılır.[1]

POS Etiketleme, morfolojik analizin basitleştirilmiş bir biçimi olarak kabul edilebilir.
Ancak yalnızca sözcüğe uygun bir POS etiketi atanmasıyla ilgilenir, oysa
Morfolojik analiz, kelimenin iç yapısının bulunmasıyla ilgilenir.

## 1.2 Neden Konuşma Kısmı Etiketlemeye ihtiyacımız var?

Otomatik metin etiketleme, doğal dil işlemede önemli bir kavramdır.
Doğal dil işlemenin birkaç farklı aşaması vardır; önceki aşamalar
aşama bir sonrakini besler. Fonem ve morfemlere dayalı morfolojik analiz,
genel olarak ilk aşama. Bir sonraki aşama anlambilimin analizidir. Konuşmanın bir kısmını etiketleme bu dizideki ilk adımlardan biridir.

## 1.3 Konuşma Bölümü Etiketlemesine Kısa Bir Genel Bakış

### 1.3.1 Yaygın Olarak Kullanılan Konuşma Bölümleri

Dilbilimciler konuşmanın üç ana bölümü olduğunu iddia ederler: isim, fiil ve
sıfat. Sözcüksel sınıflandırma durumunda, ek konuşma bölümleri önerilir
Adpozisyon ve belirleyici gibi ikincil öneme sahip olanlardır. Bunlar ek
kategoriler, aşağıdaki gibi morfo-sözdizimsel özellikleri yansıtan alt kategorilere sahip olabilir:
zaman ve sayı veya isimler, sayım ve kütle isimleri gibi anlamsal özellikler.

Etiket kümesini oluştururken en önemli husus, ayrı bir parça sağlamaktır.
boyutuna rağmen, farklı gramer davranışına sahip her kelime sınıfı için konuşma dili
ve etiket kümesinin içeriği dilsel olarak yönlendirilir.

### 1.3.2 Konuşma Bölümü Etiketleme Sorunları

POS etiketlemede iki büyük sorun vardır: Belirsiz kelimeler ve bilinmeyen
kelimeler. İlk sorun, birden fazla etiketin mümkün olduğu kelimelerin varlığıdır.
Sorunun çözümü tek tek değil bağlamın dikkate alınmasıdır.
Bu insanlar için önemsiz bir görevdir ancak otomatik metin etiketleyiciler için o kadar kolay değildir. Aşağıda bir kelime için farklı olası etiketlere sahip örnek bir cümle:

Konserve kutusunu konserve yapabiliriz. [2]

Yukarıdaki örnekte 'can' sırasıyla yardımcı fiil, fiil ve isme karşılık gelmektedir.

İkinci sorun ise bilinmeyen kelimelerin, yani var olmayan kelimelerin ortaya çıkmasıdır.
korpusta. Bu nedenle, elleçleme mekanizmalarının sağlanması önemli bir tasarım meselesidir.
bilinmeyen kelimeler.

### 1.3.3 Konuşma Bölümü Etiketleme Yaklaşımları

#### 1.3.3.1 Kural Tabanlı Yaklaşımlar

En eski konuşma bölümü etiketleme sistemleri, kurala dayalı yaklaşımı kullanır.
elle oluşturulmuş bir dizi kurala dayanır. Bu kurallar daha sonra verilen metne uygulanır.
Dilsel bir altyapının gerekliliği ve kuralların manuel olarak oluşturulması temel unsurlardır.
Kural tabanlı sistemlerin dezavantajları.

#### 1.3.3.2 Dönüşüm Temelli Öğrenme (TBL)

Brill tarafından bir dizi düzeltme kuralını öğrenen bir sistem açıklanmıştır.
Dil kurallarından manuel olarak kaçının.[3] Sistemin dayandığı fikir bir atama yapmaktır.
Derlemdeki her kelimenin başlangıç etiketi. Başlatma işleminden sonra, bir kullanarak
önceden belirlenmiş kural şablonu, her kural şablonunun başlatılmasıyla bir dizi kural elde edilir

derlemden gelen verilerle. Her kural, etiketlenen kelimelere geçici olarak uygulanır.
Yanlış belirlenir ve en fazla hata sayısını azaltan en iyi kural belirlenir.

Kural öğrenilen kurallara eklenir ve süreç yeni derlemde yinelenir
yeni eklenen kuralın hata oranı 0.000'in altına düşene kadar uygulanmasıyla oluşturulur.
Geri kalan kuralların uygulanmasıyla önceden belirlenmiş bir eşik mümkün değildir.

1.3.3.3 Markov Modeli Yaklaşımları

Sözlük gibi veri kaynaklarının zengin çeşitliliğine sahip olmanın bir sonucu olarak
sıklık verilerini, büyük derlemleri ve iki dilli paralel derlemleri içerebilir, faydalanabiliriz
İstatistiksel yöntemlerden, yeni etiketlemede kullanılacak etiket dizilerinin kalıplarını öğrenin
cümleler. Gizli Markov Modeli (HMM), popüler bir istatistiksel etiketleme yöntemidir.
Markov Modelinden ve basit bir yumuşatma tekniğinden yararlanır.

1.3.3.4 Maksimum Entropi Yaklaşımları

Maksimum entropi yaklaşımı bize bağlam konusunda daha fazla esneklik sağlar , hangisi
HMM çerçevesinde kötü kullanılmış. Bağlamın kullanımı TBL yaklaşımına benzer.
özellik şablonları kümesinin kural şablonları kümesine benzer şekilde toplanmasının yolu
TBL.Bu özellik şablonları önceden tanımlanmıştır ve özellik şablonlarının
Derlemden alınan veriler, farklılaştırıcı özellikler sistem tarafından öğrenilir. Esneklik
yararlı olduğu düşünülen herhangi bir özellik şablonunun eklenmesinin sonucudur.

1.3.3.5 Destek Vektör Makineleri

Destek vektör makinelerinin diğer modellere göre iki avantajı vardır:
yüksek boyutlu uzayları, yani çok sayıda özelliği ele alır ve bunlar genellikle daha fazladır
aşırı takılmaya karşı dayanıklıdır. Destek Vektör Makineleri ile çalışmamızın nedeni şu olacaktır:
ilerleyen bölümlerde netleşecek.

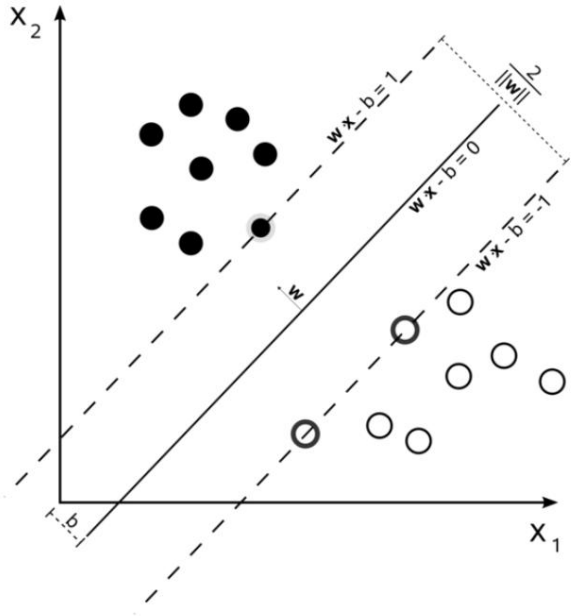
## 1.4 Türkçe Konuşma Etiketleme Kısmına İlişkin Önceki Çalışmalar

Morfolojik belirsizliğin giderilmesi, aşağıdaki gibi eklemeli dillerde çok önemlidir:
Türkçe. Türkçe için iki seviyeli bir spesifikasyona dayalı bir POS etiketleyici geliştirilmiştir.
24.000 kelimelik bir sözlüğün kullanıldığı Türkçe morfolojisi. Etiketleyici
Çok kelimeli ve deyimsel yapı tanıyıcının kullanımıyla geliştirilmiş ve
komşuluk kısıtlamalarından yararlanan morfolojik belirsizliği giderici,
sezgisel ve istatistiksel veriler. Etiketleyici yüzde 98 - 99'luk bir doğruluk seviyesine ulaşır.[4]

Türkçe metni etiketlemek için geliştirilen bir diğer etiketleyici ise bileşik bir etikete dayanmaktadır.
Kural tabanlı ve istatistiksel yaklaşımları birleştiren ve bazı yöntemlerden yararlanan yaklaşım
buluşsal yöntemlere ilişkin dilin özellikleri. Hem kelime frekansları hem de n-gram
(unigram, bigram ve trigram) olasılıkları kullanılır. Doğruluğunu arttırmak için
Sistem, morfolojik analiz cihazının çıktılarını stokastik yöntemlerle birleştiriyor.
Etiketleyici yüzde 80 doğruluk seviyesine ulaştı. [5]

## 1.5 Destek Vektör Makinesi Nedir?

Destek vektör makineleri (SVM'ler)
bir dizi ilgili denetlenen
kullanılan öğrenme yöntemleri
sınıflandırma ve regresyon için. Giriş verileri
bir n- içindeki iki vektör kümesi olarak görülür
siyah olarak gösterildiği gibi boyutlu uzay ve
Şekil 1'deki beyaz noktalar. Bir SVM
ayırıcı bir hiperdüzlem oluşturur
maksimuma çıkaran bu alan
iki veri seti arasındaki marj. İle
marjı hesapla,
iki paralel hiperdüzlem inşa edilir,
ayırmanın her iki tarafında birer tane




Şekil 1. Maksimum marj hiperdüzlemi

iki veri kümesine "yukarı doğru itilen" hiperdüzlem. Bir hiperdüzlem istenmektedir,
her iki sınıfın komşu veri noktalarına en büyük mesafeye sahip olanıdır, çünkü
genel olarak marj ne kadar büyük olursa, sınıflandırıcının genelleme hatası o kadar düşük olur. [6]

### 1.6 Neden DVM Yaklaşımını tercih ettik?

Yukarıda da belirttiğimiz gibi konuşma parçaları etiketleme konusunda çeşitli çalışmalar yapılmaktadır.
Türkçe ama hiçbiri Destek Vektör Makinelerine dayalı değildi. Ayrıca destek
vektör makinesi yaklaşımı, geniş bir özellik seti sağlama açısından çok esnektir.
hem kurala dayalı hem de istatistiksel verileri içerir. Çünkü eklemeli dillerde biçimbirimler
Bir kelimenin sınıfını değiştirebilme yeteneğine sahip olmaları dikkate alınmalıdır. Böylece,
Morfemler için bazı kurallar ve ayrıca biçimbirimlerle ilgilenen başka kurallar kullanmayı tercih ettik.
Türkçe kelimelerin özellikleri. SVM'lerin bir diğer avantajı ise
bilinmeyen kelimeler oluşabilir. Anlamlı olmadığı için tüm kelimeler bir anlamda bilinmiyor
Bir kelimenin bütüncede belirli bir etiketle kaç kez geçtiği. İkisi de elle yazılmış
kurallar ve istatistiksel veriler özellik setimizin oluşturulmasına katkıda bulundu.

## 2. Türkçe POS Etiketlemeye Yeni Bir Yaklaşım

### 2.1 Genel Bakış

#### 2.1.1 Açıklama

Projenin amacı, Türkçede Konuşmanın Kısımlarını Etiketleme için yöntemler geliştirmektir.
hem istatistiksel verilerden hem de manuel olarak yazılan kurallardan yararlanılır.

#### 2.1.2 Giriş

Derlem 753.248 kelime içermektedir. Yüzde 90'ı eğitim için, yüzde 10'u ise
test amaçlı kullanılmaktadır. Derlemenin formatı Kaynaklar bölümünde açıklanmıştır.

2.1.2.1 Kısıtlamalar

POS etiketlemeyle ilgili incelediğimiz diğer çalışmalarda daha geniş bir derlem kullanılıyor;
4,5 milyondan fazla kelimeden oluşan İngilizce için Penn Treebank külliyatı. Daha büyük korpus
eğitim için daha iyi.

Derlem büyüklüğüne ek olarak bir diğer kısıtlama da harici bir araca bağımlılıktır.
SVMlight Multiclass.Dolayısıyla çalışmanın doğruluğu doğrudan etkilenir.
SVM aracının performansı.

2.1.3 Çıkış

için etiket tahminlerini ve değerlerini gösteren çıktı.txt adlı bir metin dosyası oluşturulur.
her kelime için her sınıf. Bu dosyanın formatı aşağıda gösterilmiştir:

```
1 0,046902 -0,052224 0,004939 0,000384....
2 -0,029545 0,027581 0,000934 0,001031....
1 0,092526 -0,095487 0,002418 0,000543....
4 0,005377 -0,044700 0,001667 0,037656....
```

Şekil 2: Çıktı dosyasının formatı

Burada ilk sayı tahmin edilen etiketi, geri kalan sayılar ise tahmin edilen etiketi belirtir.
Her etiket için olasılıklar. Olasılık değeri en yüksek olan etiket, etikete atanır.
kelime.

SVMLight'ın komut penceresinde başarı oranını ve
Doğru ve yanlış etiketlenen sözcük sayısı.

2.1.4 Kaynaklar

Biçimi Şekil 3'te gösterilen önceden etiketlenmiş bir derlem kullanılır.

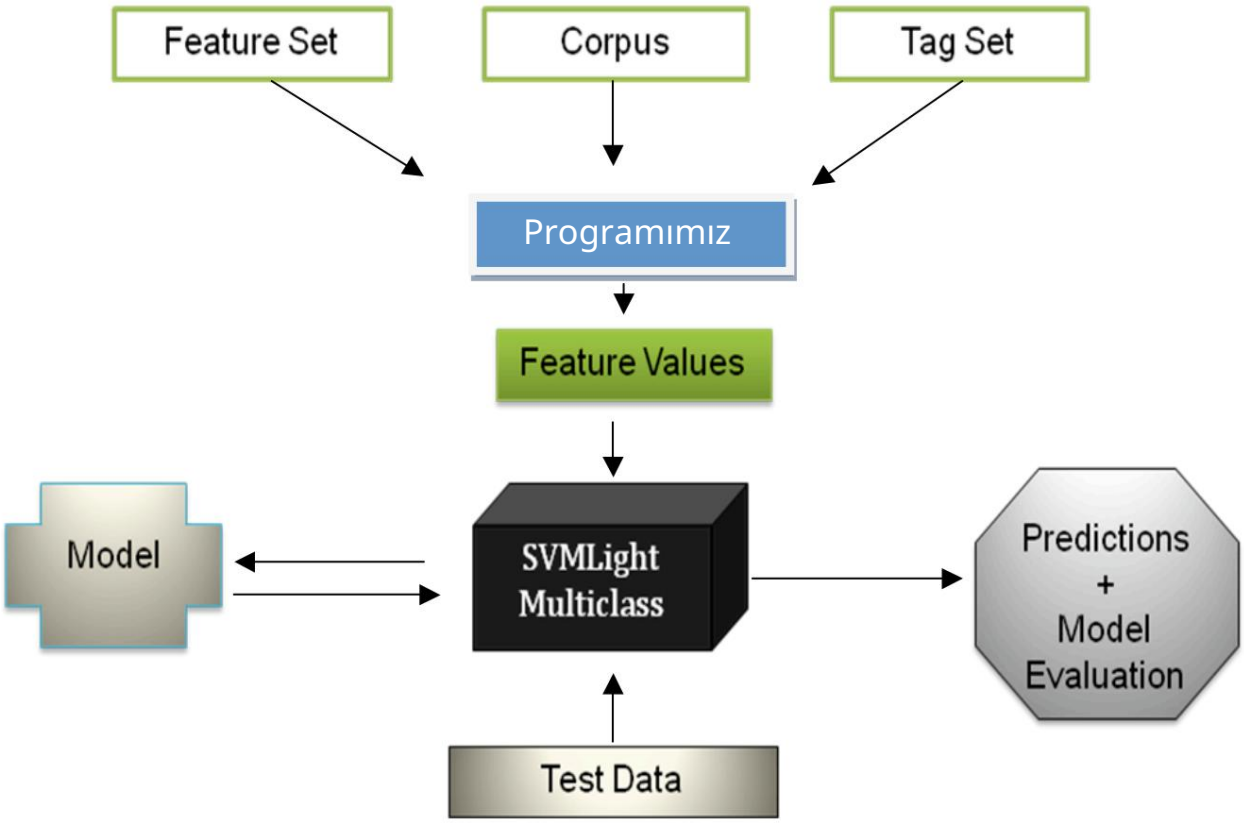
```
<S>+BSTag

İki iki[Adj] iki[İsim]+[A3sg]+[Pnon]+[Nom]
senaryonun senaryosu[İsim]+[A3sg]+[Pnon]+NHn[Gen] senaryo[Noun]+[A3sg]+Hn[P2sg]+NHn[Gen]
da da[Bağlaç]
, ,[Delgi]
beyin beyin[İsim]+[A3sg]+[Pnon]+[Nom] Bey[İsim]+[Prop]+[A3sg]+Hn[P2sg]+[Nom]
bey[İsim]+[A3sg]+[Pnon]+NHn[Gen] bey[İsim]+[A3sg]+Hn[P2sg]+[Nom]
cimnastiği cimnastiği[Unknown]
uğruna uğur(II)[Noun]+[A3sg]+SH[P3sg]+NA[Dat] uğur(II)[Noun]+[A3sg]+Hn[P2sg]+NA[Dat]
uğru[İsim]+[A3sg]+Hn[P2sg]+NA[Dat]
diğer diğer[Adj]
koşullar şart[İsim]+lAr[A3pl]+[Pnon]+[Nom] şartla[Fiil]+[Pos]+Hr[Aor]+[A3sg]
eşit eşit[Adj] Eşit[İsim]+[Prop]+[A3sg]+[Pnon]+[Nom]
zarfları hazırlayacak[Noun]+[A3sg]+SH[P3sg]+[Nom] hazırlanan[Noun]+[A3sg]+[Pnon]+YH[Acc]
altında alt[Noun]+[A3sg]+SH[P3sg]+NDA[Loc] alt[Noun]+[A3sg]+Hn[P2sg]+NDA[Loc] alt[Adj]-
[İsim]+[A3sg]+SH[P3sg]+NDA[Loc] alt[Adj]-[İsim]+[A3sg]+Hn[P2sg]+NDA[Loc]
altı[İsim]+[A3sg]+Hn[P2sg]+NDA[Loc] altın[Adj]-[İsim]+[A3sg]+[Pnon]+DA[Loc] altı[Adj]-
[İsim]+[A3sg]+Hn[P2sg]+NDA[Loc] altın[İsim]+[A3sg]+[Pnon]+DA[Loc]
çizilen çiz[Verb]-Hl[Verb+Pass]+[Pos]-YAn[Adj+PresPart] çizi[İsim]+[A3sg]+[Pnon]+[Nom]-
lAn[Fiil+Edinme]+[Pos]+[Imp]+[A2sg]
kataloğu ak[Verb]+[Pos]-YHş[Noun+Inf3]+[A3sg]+[Pnon]+YH[Acc] ak[Verb]+[Pos]-
YHş[Noun+Inf3]+[A3sg]+SH[P3sg]+[Nom] gelişim[Noun]+[A3sg]+[Pnon]+YH[Acc]
...

</S> </S>+ESTag
```

Şekil 3: Bütüncenin formatı – bütüncedeki bir cümle

Şekil 3: Önceden etiketlenmiş derlemden örnek bir cümle

BSTag ve ESTag sırasıyla cümlenin başlangıcını ve bitişini belirtir. Cesur sözler
Şekilde gösterilenler cümledeki kelimelerdir. Birden fazla ayrıştırma olabilir
Kelimenin sağına yazılan tek kelime. Bir kelimenin etiketi şu adresten edinilebilir:
ilk etiketten veya bir çekim ekinin etiketinden. Etiketler 2.2.1.1 bölümünde açıklanmıştır.

2.1.5 Süreç




Şekil 4: Sürecin genel resmi

## 2.2 Metodoloji

### 2.2.1 Yüksek Düzey Açıklama

POS etiketleyici, her biri farklı amaçlara hizmet eden farklı modüllerden oluşur. Altında, Bu modüllerin detaylarına buradan ulaşabilirsiniz. Ancak modüllerin yararlandığı ilk kaynaklar açıklanacak.

#### 2.2.1.1 Etiket Seti

Derlemden aşağıdaki 14 etiketi içeren bir etiket seti toplanır:
İsim, Bağlaç, Postp, Fiil, Adj, Punc, Det, Adv, Zamir, Num, Bilinmeyen, Interj, Dup,Ques.
Bu kısaltmalar İsim, Bağlaç, Edat, Fiil, Sıfat anlamına gelir.
Noktalama İşareti, Belirleyici, Zarf, Zamir, Sayı, Bilinmeyen, Ünlem,
Sırasıyla Çoğalt ve Soru.

Etiketleri toplamak için, derlemdeki her etiketi alacak bir program yazılır. Bizden sonra Bu etiketleri eleyerek 13'e çıkardık. Daha sonra farkettik ki adında bir etiket var. Derlemde "Bilinmeyen". Bu etiketi olduğu gibi bıraktık ve aynı anlama sahip yeni bir etiket ekledik. etiket kümesinin adı. Daha sonra değerlendirme kısmında, söz konusu olan kelimeleri almadık. 'bilinmeyen' etiketini dikkate alın.

#### 2.2.1.2 Özellik Seti

Bir özellik seti, derlem ve farklı özelliklerin incelenmesiyle oluşturulur.
sözcük kategorilerini dikkatlice inceleyin.

| | |
|---|---|
| Kelime maksimum kelimelerden biridir | w |
| Kelime bigramları | (w-1, w) |
| Kelime trigramları | (w-2, w-1, w) |
| Sonraki kelime w+1 olan maksimum kelimelerden biridir | |
| Sonraki kelime cümlenin sonunu belirtir | Son söz olmak |
| Önceki kelime cümlenin sonunu belirtir | İlk kelime olmak |
| Önceki kelime mevcut kelimeye eşittir Kopyalar için | |
| Önceki kelimenin POS etiketi | s-1 |
| İkili sözcük özellikleri | Başlangıç sermayesi<br>Kesme işareti içerir<br>Kısa çizgi içerir<br>Numara içerir<br>Uzunluk 15'ten büyük<br>Uzunluk bire eşittir ve kelime a değildir sayı<br>Noktalama işaretine eşit kelime<br>Bağlaçlara eşit kelime (“da”)<br>Soruya eşit kelime (“mu ile başlar, mü vb.”) |
| Son ekler | Ek A'da tanımlanmıştır. |

Tablo 1: Çalışmada Kullanılan Özellik Seti

2.2.1.3 Modüller

İstatistiksel Veri Toplama

Bu bölümün rolü, derlemden istatistiksel veri toplamaktır. İlk olarak, korpus
Kelimelerin ve etiketlerinin bulunduğu bir dosyaya dönüştürülür, böylece işlem daha kolay olur. Bölme
Bu dosyanın iki dosyaya dönüştürülmesi, eğitim ve test programları için girdi dosyaları oluşturmak amacıyla yapılır.
Daha sonra yeni oluşturulan dosyadan istatistikler toplanır. Bu istatistikler arasında şunları yapabiliriz:
maksimum kelime, çift, üçlü isim bulma; maksimum oluşan son iki veya üç harf
başarılı olmanıza yardımcı olacak bazı gramer dersleri ve diğer faydalı bilgiler
kelimelerin etiketlenmesi.

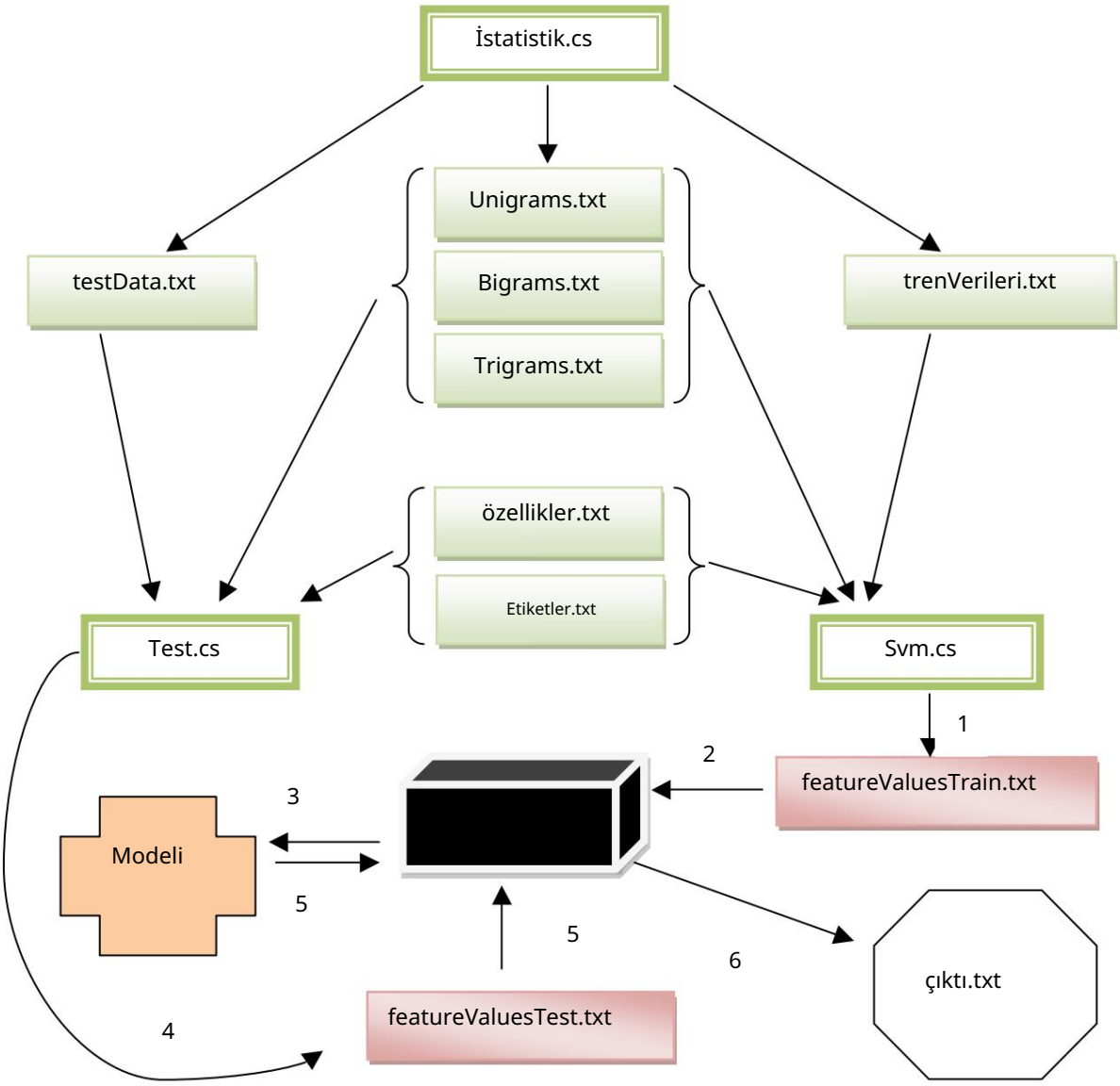
SVMLight için giriş dosyası oluşturma

Derlem iki bölüme ayrılmıştır; %90'ı eğitim verileri için ve geri kalan %10'u eğitim verileri için.
test verileri. Hem eğitim hem de test programları SVMLight için girdi dosyaları oluşturur. Kodlar
bu dosyalar birbirine çok benzer.

Öncelikle derlemdeki her bir kelime için özellik değerleri incelenerek hesaplanır.
geçerli özellik geçerli kelime için tutuluyorsa. Özellik kelime için geçerliyse, değer
çıktı dosyasına yazılır. Özellik kelime için geçerli değilse bu nedenle özellik
değer sıfırdır, dosya boyutunu büyütmemek için çıktı dosyasına eklenmez
aşırı derecede.

Daha sonra aynı işlem test verileri için tekrarlanır. , özellik değerleri hesaplanır
ve bir çıktı dosyasına yazılır.

2.2.2 Orta Seviye Açıklaması




Şekil 5: Uygulamanın ayrıntılı resmi

1: Tren verilerine ilişkin özellik değerleri, SVM.cs çalıştırılarak hesaplanır ve
featureValuesTrain.txt dosyası çıktı olarak elde edilir.

2: featureValuesTrain.txt dosyası SVMlight Multiclass'a giriş ve 'öğrenme' olarak verilir
Aracın model oluşturmasını sağlayan komut komut satırından girilir.

3: SVM aracı ile okunabilir olmasına gerek olmayan bir model oluşturulur ve kullanıcıya verilir.
Test verileri için tahminler yaparken aracı girdi olarak kullanın.

4:featureValuesTest.txt dosyası, Test.cs dosyasının ve özellik değerlerinin çalıştırılmasıyla elde edilir.
test verileri hesaplanır. Bu adımda Modelin kullanılması için Test.cs'ye girdi olarak Model verilir.
Geçerli kelimenin bir özelliği olarak önceki kelime için tahmin edilen POS etiketi.

5: featureValuesTest.txt ve Model, SVM aracına ve 'sınıflandırmaya' girdi olarak verilir
Test için etiket sınıflarını tahmin etmek amacıyla komut satırından komut girilir.
veri .

Şekil 6: Test verilerinin etiket sınıfları, SVM aracı tarafından tahmin edilir ve çıktı.txt'ye yazılır.
daha önce oluşturulan model. Kelimeler için verilen sınıflar ve tahmin edilenler
karşılaştırıldığında başarı oranı sonuç olarak döndürülür.

2.2.3 Düşük Seviye Açıklaması

POS etiketleyicinin kodu C# programlama dilinde aşağıdaki dil kullanılarak yazılmıştır:
Microsoft .NET ortamı. Derslerin detayları aşağıda açıklanmıştır. Söylendiği gibi
öncesinde asıl odak noktası SVMLight için bir giriş dosyası oluşturmaktı. Öncelikle şunu inceleyelim
Bu giriş dosyasının biçimi.

2.2.3.1 SVMlight Çoklu Sınıf için giriş dosyası

Giriş dosyasının formatı aşağıdaki gibidir:

<sınıf kimliği> <özellik kimliği>:<özellik değeri> <özellik kimliği>:<özellik değeri>...

<sınıf kimliği> <özellik kimliği>:<özellik değeri> <özellik kimliği>:<özellik değeri>...

Şekil 6: SVMLight Çoklu Sınıf aracı için giriş dosyasının formatı

Burada sınıf kimliği, eğitim verilerindeki etiketlerden birine karşılık gelir. (Bu konuda bilgi edinmek için etiketleri için lütfen Bölüm 2.2.1.1'e bakın.)

Aşağıda örnek bir cümle ve cümlenin özellik değerleri yer almaktadır.
giriş dosyası.


Şekil 7: Özellik değerlerinin hesaplandığı örnek bir cümle

Kullandığımız derlemden alınan bu örnek iki bloğu, yani özellik değerini gösteriyor
Altı kelime için değerlendirmeler. Bu örnekte, yalnızca sınırlı sayıda özellik vardır.
her kelime gösteri amaçlıdır. Bizim durumumuzda yaklaşık 56000 özellik vardır.
n-gram özelliklerinden dolayı korpusun boyutuna bağlıdır.

2.2.3.2 Kod sınıfları

Etiket.cs

Etiket numaralarını (idlerini) ve etiket adlarını saklayan bir etiket yapısı oluşturulur. Etiketler okunur dosyadan alınır ve daha sonra kullanılmak üzere bu yapıda saklanır. Yöntemler Svm.cs tarafından çağrılır.

İstatistik.cs

Bu kod dosyasında asıl iş, derlemden istatistiksel veri toplamaktır. Önce bu veriler toplanarak derlem okunur ve wordsWithTags.txt adı verilen yeni bir dosyaya yazılır. tüm kelimelerin etiketleriyle yazıldığı, her satıra bir kelime etiketi çifti. Bu dosya şu şekilde şöyle:

```
Refah İsim
da Conj
Türkçe İsim
için Mesaj
görüş İsim
istedi Fiil
Öztürkçe İsim
çalışmalarıyla
...
```

Şekil 8: wordsWithTags.txt dosyasının örnek bir kısmı

Daha sonra bu dosya biri eğitim, diğeri test için olmak üzere iki parçaya bölünür. amaçlar için,trainData.txt ve testData.txt sırasıyla. Bu wordsWithTags.txt dosyasının oluşturulması için dosya, kelimeler ve etiketleri derlemden alınmıştır. Derlemde bir etiket olduğunda, Cümlenin sonunu işaretleyen bu dosyaya sonunu işaretlemek için bir nokta konur.

Derlemdeki bir kelime aşağıdaki yapıya sahiptir:

```
kelime kökü[etiket ] + sonek[etiket1] + sonek[etiket2+etiket3] + ··
```

Şekil 9: Bir kelimenin formatı ve derlemdeki ayrıştırılmış yapısı

Bir kelime için genellikle birden fazla etiket bulunduğundan doğru olanın seçilmesi gerekir.
alınmış. Bu şu şekilde yapılır: Bir kelimenin çekim eki yani değişen bir eki varsa
kelimenin gramer kategorisi, ardından ortaya çıkan etiket alınır. Eğer sahip değilse
böyle bir ek, ilk etiket alınır.

Test amaçlı dosya testData.txt olarak adlandırılır ve Test.cs'ye girdi olarak verilir.
Formatı Svm.cs'nin giriş dosyasıyla aynıdır.

Bu kod dosyasının girdi dosyaları oluşturmanın yanı sıra toplama konusunda da önemli bir görevi vardır.
derlemden istatistiksel veriler. Bu istatistikler aşağıda sıralanmıştır:

1. En çok tekrarlanan kelimeleri, kelime çiftlerini ve kelime üçlülerini bulma

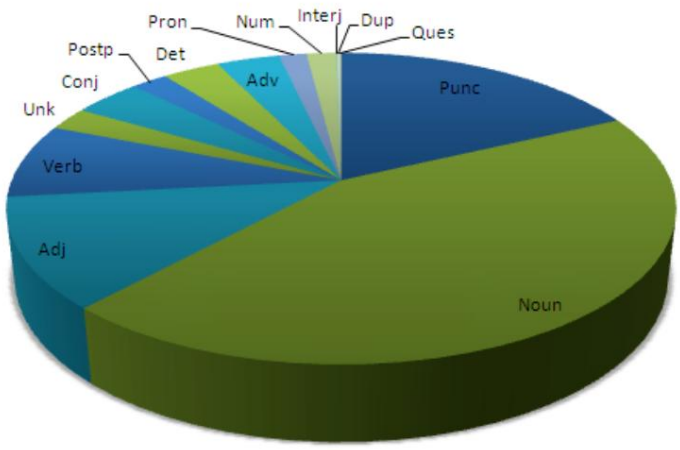
Bu veriler kelime unigram, bigram ve trigram amaçları için kullanılır. Eşik değerleri
Çeşitlidir ve deneysel olarak test edilmiştir. Bu değerler sırasıyla 300, 100 ve 2'dir.

2.Kelimelerin sınıflarına göre en çok geçen son 2 veya 3 harfini bulma (fiil, fiil,
sıfat vb.)

Bu harflerin toplanmasındaki amaç, kullanılacak yeni eklerin belirlenmesidir.
bazı sınıfları etiketlemek. Bu eklerin oluş durumuna göre ele alınırlar.
hesap.

3. Derlemdeki kelime sınıflarının yüzdelerini alma

Bu, külliyat hakkında fikir edinmek için yapılır. Dağılım şu şekilde görülebilir:
Aşağıdaki grafik.




Şekil 10: Derlemdeki gramer sınıflarının dağılımı

4. Bazı eklerle ilgili veri toplamak yani etiketlemede ne kadar başarılı oldukları belirli bir gramer dersi

Bu, son eklerin performansını ölçmek için yaygın olarak kullanılır.

5. Bazı sınıflara ait kelimeleri dosyalara yazmak

Bir sınıfa ait kelimelerin ek ve tekli kelime bakımından benzerliğini gözlemlemek özellikler.

Svm.cs

Bu, SVMLight yazılımı için giriş dosyası oluşturmanın ana kısmıdır. Bu dosyada, İstatistik.cs tarafından oluşturulan trainData.txt kullanılır.

Yürütmenin her adımında bu dosyadan bir kelime okunur ve özellik işlemleri üzerinde yürütülür. Bazı işlemler istatistiksel veriler içeren dosyalardan yararlanırken, diğerleri ise ekler ve diğer tekli özellikler için tek kelime üzerinde yapılan işlemlerdir. verilen ekleri içeren her kelime, 3'ten az içeren daha kısa ekler için Harflerde kelimenin sadece sonu dikkate alınır, daha uzun ekler için ise 2/3 dikkate alınır. Kelimenin rd'si geçerli eki içerme açısından incelenir.

İşlem sırasında, o anda işlenmekte olan kelimenin sayısı çıktı olarak verilir, böylece infazın ne zaman sona ereceği yaklaşık olarak tahmin edilebilir.
Sonuç olarak modelin giriş dosyası olan featureValuesTrain.txt isimli bir dosya oluşturulur.
SVMLight'ın eğitimi.

Test.cs

Bu kod dosyası, test amacıyla SVMLight Multiclass için bir giriş dosyası oluşturmak içindir amaçlar. Svm.cs'ye çok benzer ancak dosya adları ve bazı özellikleri bakımından ondan farklıdır. özellik işlemleri. Önemli bir fark önceki kelimenin POS etiketidir.

Svm.cs'de bu işlem bir önceki kelimenin POS etiketi alınarak kolaylıkla yapılır.
Ancak test için bu etiketin öncelikle SVMLight tarafından bulunması gerekir. Yani ilk başta svm_multiclass_learn komutu, Svm.cs tarafından oluşturulan giriş dosyasıyla yürütülür. Daha sonra, model dosyası kullanılarak Test.cs kodu içinden svm_multiclass_classify çağrılır.
kelimeye bir etiket atayın. Bu etiket SVMlight yazılımı tarafından bir dosyaya yazılır. Şimdi bu etiketi bir sonraki kelime için önceki POS etiketi olarak kullanılabilir.

Kodun bir diğer önemli kısmı başarının değerlendirilmesidir. Her ne kadar başarı SVMLight tarafından değerlendirilen bilinmeyen etiketli kelimeler de dikkate alınır. Ayrıca, Hangi gramer dersi için başarı oranına sahip olmak iyidir, böylece hangisinin hangisi olduğunu bilebiliriz. geliştirilecek yanlar.

## 3. SONUÇ VE SONUÇ

Proje, Türkçe bir metindeki kelimelere konuşma etiketlerinin doğru kısımlarının atanmasını amaçlamıştır. SVMLight aracını kullanarak destek vektör makineleri yaklaşımını kullanarak.
Farklı denemeler sonunda ulaşılan başarı oranı yüzde 75'tir.

Türkçe eklemeli bir dil olduğundan POS etiketleyicilerin başarı düzeyleri Türkçe ile İngilizce gibi eklemeli olmayan diller karşılaştırılmamalıdır. Ama biz bu dillerin başarı düzeylerine de örnekler vereceğiz. Bir kısmıyla ilgili çeşitli çalışmalar İngilizce üzerinde gerçekleştirilen destek vektör makinesi yaklaşımını kullanarak konuşma etiketleme yüzde 90'ın üzerinde bir doğruluk seviyesi. Örnek olarak Jesus Gimenez ve Lluis'in çalışması Marquez yüzde 97,16'lık bir doğruluğa dikkat çekti. [7]

Tetsuji Nakagawa, Taku Kudoh ve Yuji Matsumoto tarafından yürütülen bir başka çalışma
doğruluk oranı yüzde 97,1'dir.[8]

Türkçede konuşma etiketlemenin bir kısmı ile ilgili daha önce tamamlanmış çalışmalar bulunmaktadır.
Ancak bunların hiçbiri destek vektör makinelerinin kullanımına dayanmamaktadır. Bir çalışma
Levent Altınyurt, Zihni Orhan ve Tunga Göngör tarafından tamamlanmıştır.
yaklaşık yüzde 80 düzeyinde. [9]

Tablo 2'ye bakarak her birini etiketlemenin başarı oranını yorumlayabiliriz.
gramer sınıfı:

| Sınıf | Başarıyı Etiketleme |
|---|---|
| İsim | 96,91 |
| Bağlaç | 74,13 |
| Sonra | 1,37 |
| Fiil | 66,37 |
| Ayar | 5,65 |
| yumruk | 100 |
| Det | 71,57 |
| İlan | 12,11 |
| zamir | 4,3 |
| Sayı | 67,74 |
| Bilinmeyen | - |
| Interj | 0 |
| Çift | 0 |
| Sorular | 21,95 |

Tablo 2: Farklı sınıfları etiketlemenin başarı oranları

Bu tablodan bazı sınıfların başarıyla etiketlendiğini, bazılarının ise başarıyla etiketlendiğini görebiliriz.
değiller. Etiketleme başarısının olmaması, bunun için başarılı özellik seçiminin eksikliğini gösterir
sınıf. Örneğin zamirlerin genellikle ayırt edici ekleri yoktur. zor
Zamirleri ayırt edecek özellikleri bulun. Sıfatlar için kullanılması istenen ekler şunlardır:
diğer sınıflarda ortak olması nedeniyle bu özelliklerin seçilmemesine neden olacaklardır.
hatalı sonuçlar.

Aşağıdaki hususların dikkate alınmasıyla başarının artırılabileceğine inanıyoruz:

- Daha büyük derlem boyutu, eğitim ve model oluşturmada daha iyi olacaktır,

- Yeni ayırt edici özellikler içermesi,

- Kullanmak için sağdan sola okuyarak derlemin özelliklerle birlikte analiz edilmesi sonraki kelimelerin posta etiketi.

## 4.REFERANSLAR


1. Vikipedi – http://en.wikipedia.org/wiki/Part-of-speech_tagging

2. Tunga Güngör, "Konuşma Bölümlerini Etiketleme", 2009

3. Eric Brill "Dönüşüm Tabanlı Hata Odaklı Öğrenme ve Doğal Dil İşleme: Konuşma Etiketlemenin Kısmında Bir Vaka Çalışması.", Hesaplamalı Dilbilim,1995

4. Kemal Oflazer ,İlker Kuruöz, "Etiketleme ve Morfolojik Belirsizliğin Giderilmesi Türkçe Metin"

5. Levent Altınyurt, Zihni Orhan,Tunga Güngör, " Kural tabanlı birleştirmeye doğru ve eklemeli dillerde konuşma etiketlemenin istatistiksel kısmı",2007

6. Destek Vektör Makinesi - http://en.wikipedia.org/wiki/Support_vector_machine

7. Gimenez, D. ve L. Marquez, "SVMTool: A general POS etiketleme oluşturucusu Vektör makineleri desteklemek"

8. Tetsuji Nakagawa, Taku Kudoh ve Yuji Matsumoto , "Bilinmeyen Kelime Tahmini ve Destek Vektör Makinelerini Kullanarak Konuşma Bölümü Etiketlemesi"

9. Levent Altınyurt, Zihni Orhan,Tunga Güngör, "Bir Kısmına Bileşik Bir Yaklaşım Türkçede Konuşma Etiketleme" , 2007

ALINTILANMAYAN KAYNAKLAR

Gimenez J, Marquez L. Konuşmanın Hızlı ve Doğru Kısmında Etiketleme: SVM Yaklaşımı Tekrar ziyaret edildi , 2003

Poel M, Stegeman L, Akker R. Hollanda Parçasına Destek Vektör Makinesi Yaklaşımı Konuşma Etiketleme,2007

Pla, Molina. Sözcükselleştirilmiş HMM ile Konuşma Kısmı Etiketleme, 2001

Brill, Konuşma Etiketleyicinin Basit Kural Tabanlı Bir Parçası, 1992

James Mayfield, Paul McNamee, Christine Piatko, Claudia Pearce , Kafes Tabanlı Etiketleme Destek Vektör Makinelerinin kullanılması, Bilgi ve Bilgi Yönetimi Konferansı, 2003

Jason Weston, Chris Watkins, Çok Sınıflı Destek Vektör Makineleri, 1998

Ethem Alpaydın, Destek Vektör Makineleri, Makine Öğrenimine Giriş, 2004

Taku Kudo, Yuji Matsumoto , Destek vektör makineleriyle parçalama, Kuzey Amerika Hesaplamalı Dilbilim Derneği Bölümü, 2001

Vapnik, V., "İstatistiksel Öğrenme Teorisinin Doğası", Springer, 1995.

SVMLight Destek Vektör Makinesi - http://svmlight.joachims.org/

SVMLight Çok Sınıflı Destek Vektör Makinesi - http://svmlight.joachims.org/svm_multiclass.html

Kemal Oflazer ,İlker Kuruöz, "Türkçenin Etiketlenmesi ve Morfolojik Olarak Belirsizliğin Giderilmesi Metin"

5. EK: Kodun önemli kısımları ve son ekler

Son ekler: (Aynı satırda yer alan son ekler ikamedir)

ağı eği

ben öyleyim

ama eme

amak emek

arak,erek,yip,yıp

esi esi

av ev

bilir

deki teki

baraj dem tem

diz diz düz

dik dık duk

dir dır tir dur tur tır

dı du dü ti tü di tı tu

yapmamış olacaktı

gı gi gü kı ku

gıç giç

IR

ince,ince,

inci,üncü, inci

eni

ıp

silah

ki

li li

lak lek

lan len

lendi lendi

lama leme

daha ler

uluslararası

luk lük

mış miş muş muş olmuştu olmuştu

olmuştu olmuştu

medi yoktu yorlar

düler dular dilerdiler

mişler mışlar müşler muşlar

sek

se

sız siz suz süz

cü çı cu çi cı ci çu

daş

inci üçüncü ncü

cil cil

günah

sal sel

BT

kır

acak olacak

ge

ıcı gücü

içinde ün içinde

ünç inç

inti üntü

im üm ım um günah sün sın güneş

siz sizsiniz siz siz dirler dırlar durlar

yormalısın

mek mak

İngiltere'deyim

siyon siyon

Ben henüz

kar

dan

katran

madan meden

ne olursa olsun

mala mele

Mali

mar mer

mazlık mezlik

bizim miz muz

nak nek

nin nin nun nün

ntı nti ntü

rak rek

rga rge

rsa rse zsa zse

rlar

yani

sil sul

sın güneş sün

şar

kal

ntı nti ntü

van ven

gan gen kan

Ken

gün kın

kün

ıcı içi ucu gücü

vi

adamlar adam

olduğu

le la

sini

luk gibi

lanmalanma

ns

yan yen

Nan

nesel zel

ruz

ica

Ren

rek rak

'li'li'lu'lü

Şan

İngiltere

Kod (Yalnızca önemli parçalar)

İstatistik.cs

```
kamu yapısı bigramı
    {
        genel dize sözcüğü;
        genel dize öncekiWord }
    kamu yapı trigramı
    {
        genel dize sözcüğü;
        genel dize öncekiWord;
        genel dize ikiPrevWord;}
    genel yapı Word
    {
        genel dize sözcüğü;
        genel dize etiketi;
    };
```

```
genel geçersiz writeWordsToFile()
{
    dictUnigram = new Dictionary<string, int>();
    dictUnigramThreshold = new Dictionary<string, int>();
    dictBigram = new Dictionary<bigram, int>();
    dictBigramThreshold = new Dictionary<bigram, int>();
    dictTrigram = new Dictionary<trigram, int>();
    dictTrigramThreshold = new Dictionary<trigram, int>();

    bigram yeniBigram = yeni bigram();
    trigram yeniTrigram = yeni trigram();

    StreamReader sr = File.OpenText(corpus);
    StreamWriter sw = File.CreateText("wordsWithTags10.txt");
    dize satırı;
    string[] tagları = yeni string[20];
    string word = String.Empty;
    string etiketi = String.Empty;

    string[] WordsInLine = yeni dize[20];
    string[] splitMatch = yeni dize[2];
    dize[] ayrıştırma = yeni dize[13];
    dize[] bölünmüş = yeni dize[2];

    for (int i = 1; i <= 3; i++)
    {
    sr = File.OpenText(corpus);
    satır = sr.ReadLine();
    kelimeSayısı = 1;
    while (!String.IsNullOrEmpty(line))
    {

        if (line.StartsWith("<") && line.Contains("ESTag"))
```

```
{
      sw.WriteLine("." +          "" + "Delgi");
}
else if (!line.StartsWith("<"))
{
    WordsInLine = line.Split(' ');
    kelime = Satır İçi Kelimeler[0];

    if (!WordsInLine[1].StartsWith("+"))
    {
        string input = WordsInLine[1];
        dize modeli = "]";
        string[] substrings = Regex.Split(giriş, model);

        int dizini = 0;

        for (int m = 0; m < substrings.Length; m++)
        {
            if (!substrings[m].Equals(String.Empty))
            {
                string[] splittedNew = substrings[m].Split('[');
                etiketler[dizin] = bölünmüşYeni[1];
                indeks++;
            }
        }

        int kontrol = -1;
        int j = 0;
        int tagIndex = 0;

        for (j = 0; j < dizin; j++)
        {
            if (!tags[j].Equals(String.Empty))
```

```
    {
        if (etiketler[j].İçerir("+"))
        {
            tagIndex = j;
            kontrol = tagIndex;
        }
    }
}
eğer (kontrol == -1)
    etiket = etiketler[0];
başka
    etiket = etiketler[tagIndex].Split('+')[0];

   sw.WriteLine(kelime +          " "     + etiketi);
eğer (i == 1)
{
    if (dictUnigram.ContainsKey(word.ToLower()))
    {
        dictUnigram[word.ToLower()]++;

        if (dictUnigram[word.ToLower()] == wordThreshold)
        {
            dictUnigramThreshold.Add(word.ToLower(), wordThreshold);
            dictUnigram.Remove(word.ToLower());
        }

    }
    else if (dictUnigramThreshold.ContainsKey(word.ToLower()))
    {
        dictUnigramThreshold[word.ToLower()]++;
    }
    başka
    {
```

```
                dictUnigram.Add(word.ToLower(), 1);
            } } } }
        eğer (i == 2)
        {
            newBigram.prevWord = newBigram.word;
            newBigram.word = word.ToLower();

            if (dictBigram.ContainsKey(newBigram))
            {
                dictBigram[yeniBigram]++;
                if (dictBigram[newBigram] == çiftEşik)
                {
                    dictBigramThreshold.Add(newBigram,pairThreshold);
                    dictBigram.Remove(newBigram);
                }

            }
            else if (dictBigramThreshold.ContainsKey(newBigram))
            {
                dictBigramThreshold[newBigram]++;
            }
            else if (!String.IsNullOrEmpty(newBigram.prevWord))
            {
                dictBigram.Add(newBigram, 1);
            }
        }

        satır = sr.ReadLine();
        kelime Sayısı++;
        if (!String.IsNullOrEmpty(line))
        {
            while (!String.IsNullOrEmpty(line) && line.StartsWith("<"))
            {
```

```
                if (line.Contains("ESTag"))
                {
                    eğer (i == 1)
                     sw.WriteLine("." +  "  "  + "Delgi");
                }
                satır = sr.ReadLine();
            }
            if (i == 3 && !String.IsNullOrEmpty(line))
            {

                WordsInLine = line.Split(' ');
                kelime = Satır İçi Kelimeler[0];
                newTrigram.twoPrevWord = newTrigram.prevWord;
                newTrigram.prevWord = yeniTrigram.word;
                newTrigram.word = word.ToLower();

                if (dictTrigram.ContainsKey(newTrigram))
                {
                    dictTrigram[yeniTrigram]++;
                    if (dictTrigram[newTrigram] == tripleThreshold)
                    {
                        dictTrigramThreshold.Add(newTrigram, tripleThreshold);
                    }

                }
                else if (dictTrigramThreshold.ContainsKey(newTrigram))
                {
                    dictTrigramThreshold[yeniTrigram]++;
                }
                else if (!String.IsNullOrEmpty(newTrigram.prevWord) &&
!String.IsNullOrEmpty(newTrigram.twoPrevWord))
                {
                    dictTrigram.Add(newTrigram, 1);
```

```
                }
            }
        }
        Console.WriteLine(wordCount);
    }

    eğer (i == 1)
    {
        getUnigrams();
        dictUnigram.Clear();
        dictUnigramThreshold.Clear();
    }

    else if (i == 2)
    {
        getBigrams();
        dictBigram.Clear();
        dictBigramThreshold.Clear();
    }
    else if (i == 3)
    {
        getTrigrams();
        dictTrigram.Clear();
        dictTrigramThreshold.Clear();
    }
    sr.Close();
    }
    sw.Close();

}

genel geçersiz splitCorpus()
{
```

```
        StreamReader sr = File.OpenText("corpus.txt");
        StreamReader sr2 = File.OpenText("wordsWithTags.txt");
        dize satırı = sr.ReadLine();
        dize satır2 = sr2.ReadLine();
        int sayısı = 1;
        StreamWriter sw = File.CreateText("corpusSplitted.txt");
        StreamWriter sw2 = File.CreateText("corpusRemaining.txt");
        StreamWriter sw3 = File.CreateText("trainData.txt");
        StreamWriter swTest = File.CreateText("testData.txt");

        while ((satır != null) && (satır.Uzunluk > 0) && (satır2 != null) && (satır2.Uzunluk > 0)
&& (sayı <= 753248))
        {
            if (sayı <= 75000)
            {
                if (!line.StartsWith("<") || (!line2.StartsWith("<")))
                {
                    swTest.WriteLine(satır2);
                    sw.WriteLine(satır);
                }
            }
            başka
            {
                sw2.WriteLine(satır);
                sw3.WriteLine(satır2);
            }

            satır = sr.ReadLine();
            satır2 = sr2.ReadLine();
            sayım++;
        }
        sw.Close();
        sw2.Close();
```

```
        sr.Close();
        sw3.Close();
        sr2.Close();
    }

    genel geçersiz getUnigrams()
        {

            StreamWriter sw = File.CreateText("Unigrams.txt");
            foreach ( dictUnigramThreshold'da KeyValuePair<string, int> kvp )
            {
                sw.WriteLine(kvp.Key +                 " "  + kvp.Değeri);
            }
            sw.Close();
        }
```

Svm.cs

```
    public void includeSuffix(string word, string etiketi)
    {
        StreamWriter sww=null;
        if (tag.Equals("Punc"))
            geri dönmek;
        StreamReader SR;
        StreamWriter Yazılımı;
        dize satırı;
        dize[] sonekler = yeni dize[20];
        int sayısı = 1;

        SW = File.CreateText("suffixOut.txt");
        SR = Dosya.OpenText(dosyaAdı);
        satır = SR.ReadLine();
        while ((satır != null) && (satır.Uzunluk > 0))
```

```
{ int değer = 0;
    sonekler = line.Split(' ');
    int ben = 0;
    //int sıcaklık = 0;
    for (i = 0; i < sonekler.Uzunluk; i++)
    {
        if (!sonekler[i].Eşittir(""))
        {
            if (word.Contains(sonekler[i]))
            {
                sww = File.AppendText("suffixesAndTags.txt");
                sww.WriteLine(sonekler[i] +          " "     + etiketi);
                sww.Close();
                değer = 1;
               if (sonek.ContainsKey(sonekler[i]))
                    sonek[sonekler[i]]++;
                başka
                    sonek.Add(sonekler[i],1);
            }
            başka
            {
                değer = 0;
                if (!suffix.ContainsKey(suffixes[i]))
                {
                    sonek.Add(sonekler[i], 0);
                } } } }
    sayım++;
    SW.WriteLine(değer);
    satır = SR.ReadLine();
}
SR.Close();
SW.Close();
}
```

```
public bool startCapital(string word)
{
    if (char.IsUpper(word[0]))
                        doğruyu döndür;
    yanlış döndür;
}

public bool içerirKesme işareti(dize sözcüğü)
{
    if (word.Contains("\'") && !word.Equals("\'"))
        doğruyu döndür;
    yanlış döndür;
}
statik void Main(string[] args)
{

    #bölge İSTATİSTİKLERİ
    İstatistik istatistiği = yeni İstatistikler();
    stat.splitCorpus();
    stat.writeWordsToFile();

    #sonbölge



    SVMMain pr = new SVMMain("");
    pr.setSuffixes("features.txt");
    int sayılanKelimeler = 0;

    string dosya adı = "tags.txt";
    Tag.setTags(dosya adı);

    // özellik seti:
```

```
            string[] featureSet = new string[8] { "startsCapital",
"containsKesme İşareti", "containsNumber", "longerThan15",
"lengthOneAndNotNumber", "equalToPunc", "equalToConj", "equalToQues" };
            //yöntemleri ismine göre çağırmak için
            SVMMain p = new SVMMain("");
            nesne[] parametreleri = yeni nesne[1];
            bool dönüşDeğeri = false;
            //girdi dosyasını yazmak için
            StreamWriter swFeatureValues = File.CreateText("featureValuesTrain.txt");
            //girdi dosyasını okumak için
            StreamReader srTrain = File.OpenText("trainData.txt");
            dize satırı = srTrain.ReadLine();
            dize[] kelimeTag = yeni dize[2];
            İkinci KelimeÖnceki Kelime ;
            Kelime öncekiKelime;
            Sonraki Kelime ;
            twoPrevWord.word = String.Empty;
            twoPrevWord.tag = String.Empty;
            öncekiWord.word = String.Empty;
            öncekiWord.tag = String.Empty;
            nextWord.word = String.Empty;
            nextWord.tag = String.Empty;
            int numFeatures = 0;
            Kelime yeniKelime = yeni Kelime();
            countWords++;
            while (!String.IsNullOrEmpty(line))
            {
                twoPrevWord.word = öncekiWord.word;
                öncekiWord.word = yeniWord.word;
                twoPrevWord.tag = prevWord.tag;
                öncekiWord.tag = yeniWord.tag;
                wordTag = line.Split(' ');
                yeniWord.word = kelimeTag[0];
```

```
yeniWord.tag = kelimeTag[1];


//operasyonlar:
for (int j = 0; j < 8; j++)
{
    // İstenilen yöntemi ada göre alın:
    MethodInfo methodInfo = typeof(SVMMain).GetMethod(featureSet[j]);
    // Yöntemi argümanlar olmadan çağırmak için örneği kullanın
    parametreler[0] = yeniWord.word;
    denemek
    {
        //bir özellik yöntemini çağır
        returnValue = Convert.ToBoolean(methodInfo.Invoke(p, parametreler));
        //özellik değerleri dosyasına yaz:
      //dosya formatı:
        //tagID featureID:featureValue featureID:featureValue...
        dize değeri = "0";
       if (dönüş değeri)
            değer = "1";
       int etiket kimliği = 0;
        eğer (j == 0)
        {
            for (int k = 0; k < Tag.tagArr.Length; k++)
                if (Tag.tagArr[k].tagName.Equals(newWord.tag))
                {
                    tagID = Tag.tagArr[k].tagID;
                    kırmak;
                }
            if (değer == "1")
                swFeatureValues.Write(tagID +          " "          + (j + 1) + ":" + değer + " ");
            başka
                swFeatureValues.Write(tagID + " ");
        }
```

```
            başka
            {
                if (değer == "1")
                    swFeatureValues.Write((j + 1) + ":" + değer + " ");
            }
        }
        yakalama (Örneğin istisna )
        {
            Console.WriteLine(ex.Message);
        }
    }//sonlandır
    denemek
    {
      // kelimenin sonek içerip içermediğini kontrol et
        pr.containsSuffix(newWord.word);
        StreamReader srNew = File.OpenText("suffixOut.txt");
        string yeniLine = srNew.ReadLine();
        while (yeniSatır != null)
        {
            int değerF = Convert.ToInt32(newLine);
            if (değerF == 1)
                swFeatureValues.Write(numFeatures + ":" + değerF + " ");
            sayıÖzellikler++;
            yeniLine = srNew.ReadLine();
        }
        srNew.Close();
        sayıÖzellikler = 4;
    }
    yakalamak { }
    //kelime unigramı
    StreamReader srUnigram = File.OpenText("Unigrams.txt");
    string line2 = srUnigram.ReadLine();
    while (!String.IsNullOrEmpty(line2))
```

```
{
    if (line2.Split(' ')[0].ToLower().Equals(newWord.word.ToLower()))
    {
        swFeatureValues.Write(numFeatures + ":" + 1 + " ");
        sayıÖzellikler++;
    }
    satır2 = srUnigram.ReadLine();
}
srUnigram.Close();


if (!String.IsNullOrEmpty(prevWord.word))
//kelime Bigramı:
{
    {
        StreamReader srBigram = File.OpenText("Bigrams.txt");
        satır2 = srBigram.ReadLine();
        dize[] çifti = yeni dize[2];

        while (!String.IsNullOrEmpty(line2))
        {
            çift = satır2.Split(' ');

            if ((pair[0].Equals(prevWord.word.ToLower())) &&
(pair[1].Equals(newWord.word.ToLower())))
            {
                swFeatureValues.Write(numFeatures + ":" + 1 + " ");
                sayıÖzellikler++;
            }

            satır2 = srBigram.ReadLine();
        }
        srBigram.Close();
```

```
                }
            }
            if (!String.IsNullOrEmpty(prevWord.word) &&
!String.IsNullOrEmpty(twoPrevWord.word))
            //kelime Trigramı:
            {
                {
                    StreamReader srTrigram = File.OpenText("Trigrams.txt");
                    satır2 = srTrigram.ReadLine();
                    dize[] üçlü = yeni dize[3];
                    while (!String.IsNullOrEmpty(line2))
                    {
                        üçlü = satır2.Split(' ');
                        if ((triple[0].Equals(twoPrevWord.word.ToLower())) &&
(triple[1].Equals(prevWord.word.ToLower())) &&
(üçlü[2].Equals(newWord.word.ToLower())))
                        {
                            swFeatureValues.Write(numFeatures + ":" + 1 + " ");
                            sayıÖzellikler++;
                        }
                        satır2 = srTrigram.ReadLine();
                    }
                    srTrigram.Close();
                }
            }
            //kopyalamak
            if (!String.IsNullOrEmpty(prevWord.word))
            {
                if (newWord.word.ToLower().Equals(prevWord.word.ToLower()))
                {
                    swFeatureValues.Write(numFeatures + ":" + 1 + " ");
                }
                sayıÖzellikler++;
```

```
                }
                //İlk kelime
                if (Convert.ToBoolean(p.firstWord(prevWord.word)))
                    swFeatureValues.Write(numFeatures + ":" + 1 + " ");
                sayıÖzellikler++;
                // İlk kelime
                string ilk = (Convert.ToInt32(p.firstWord(prevWord.word))).ToString();
                swFeatureValues.Write(numFeatures + ":" + ilk + " ");
                sayıÖzellikler++;
                satır = srTrain.ReadLine();
                if (!String.IsNullOrEmpty(line))
                {
                    //sonraki kelime
                    nextWord.word = line.Split(' ')[0];
                    if (Convert.ToBoolean(p.nextWord(nextWord.word)))
                        swFeatureValues.Write(numFeatures + ":" + 1 + " ");
                    sayıÖzellikler++;
                    if (Convert.ToBoolean(p.lastWord(nextWord.word)))
                        swFeatureValues.Write(numFeatures + ":" + 1 + " ");
                    sayıÖzellikler++;
                }
                countWords++;
                Console.WriteLine(countWords);
                swFeatureValues.Write(Environment.NewLine);
            }
            swFeatureValues.Close();
            srTrain.Close();
        }
    }
}
```