Basic Text Processing

Words, Corpora and Word Tokenization

Text / Metin Normalizasyonu

Aşağıdakiler her DDİ sürecinde mutlaka gerçekleştirilmelidir.

1. Tokenizing (segmenting) words
2. Normalizing word formats
3. Segmenting sentences

Boşluk Tabanlı Belirteçleme (Tokenizasyon) /Space-based Tokenization

- Belirteçlere ayırmanın en basit yolu kelimeler arasındaki boşluklara göre ayırım yapmaktır.
- Unix tools for space-based tokenization (boşluk tabanlı belirteçleme
- "tr" bir komut olarak Ken Church's tarafından UNIX platformunda şiirler için kullanıldı.
  - ✓ Bir metin dosyası verildiğinde, sözcüklerin etiketlenmesi ve frekanslarının çıktısı alındı.
  - ✓ *Örnekleri:* metni küçük harfe çevirme, sık geçen (stop words) kelimeleri filtreleme, basit noktalama ayrıştırma, sayı ve kelimeyi ayrı tokenlaştırma gibi …
  - ✓ *Örnek araçlar*: **tr** ile noktalama ayırma, **awk** ile token sayma, **sed** ile basit normalizasyon işlemleri yapan araçlar gibi.

tr -sc ’A-Za-z’ ’\n’ < shakes.txt | sort | uniq –c

tr: Karakter dönüştürme ve silme (translate) aracı

-s: Sıkıştır (squeeze). Aynı türden ardışık birden çok karakteri tek bir karaktere indirger.

-c: Complement (tamamlama). Belirtilen karakter sınıfının tersi, “bunların dışında kalan tüm karakterlerdir”

İlk argüman: 'A-Za-z’

-c kullanıldığı için “harf olmayan tüm karakterler” hedeflenmiştir.

İkinci argüman: '\n’

Karakteri yeni satırdır. Harf olmayan her karakter yeni satıra dönüştürülür.

Girdi: < shakes.txt

uniq –c: Ardışık yinelenen satırları (kelime bazındaki) tek satıra düşürerek tekrar sayısını verir.

Nasıl dönüşür?

-c : A–Z ve a–z dışındaki tüm karakterler yeni satıra çevrilir.

-s : ardışık birden fazla yeni satır tek bir yeni satıra sıkıştırılır.

Sonuç olarak

Harflerden oluşan dizilişler “kelime” olarak mevcut satırda kalır.

Boşluklar, noktalama işaretleri, sayılar, diğer semboller satır sonuna çevrilir ve tek tek ayraç haline gelir.

Büyük-küçük harf korunur (bu komut normalize edilmez).

İlk adım: Belirteçleme /tokenization

```
tr -sc 'A-Za-z' '\n' < shakes.txt | head
```

```
THE
SONNETS
by
William
Shakespeare
From
fairest
creatures
We
...
```

İkinci Adım : Sıralama / sorting

```
tr -sc 'A-Za-z' '\n' < shakes.txt | sort | head

A
A
A
A
A
A
A
A
A
...
```

Sonraki İşlemler

Küçük ve büyük harflerin uyumlandırılması / birleştirilmesi

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n' | sort
          | uniq -c
```

```
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
.... ...
```

Frekansa göre Sıralama yapılması

```
tr 'A-Z' 'a-z' < shakes.txt | tr -sc 'A-Za-z' '\n'
      | sort   |  uniq -c | sort -n -r
```

```
23243 the
22225 i
18618 and
16339 to
15687 of
12780 a
12163 you
10839 my
10005 in
8954  d
```

Burada ne oldu?

tr -sc ’A-Za-z’ ’\n’ < shakes.txt | sort | uniq –c

- Dosyada bulunan metin, satır başında birer kelimeye dönüştürülür. Örneğin “Gunlukler, neredesin? 2562” komutunun işlenmesi sonunda:

Gunlukler

neredesin

çıktısı cevaptır.

- Kelimeleri küçük harfe dönüştürmek için

tr 'A-Z' 'a-z' | tr -sc 'a-z' '\n’ yazılabilir.

- ASCII kod olmayan Unicode harfler (Türkçede “ç, ğ, ş”) bu komutla dönüşmeyecektir.
  - -c nedeniyle yeni satıra dönüşecektir.
  - Dönüşüm için Unicode destekli tokenizasyon yapan araçlar kullanılmalıdır.

Belirteçlemede Sorunlar

- Aşağıdaki gibi noktalama işaretleri kontrol edilmeden kaldırılamaz.
  - m.p.h., Ph.D., AT&T, cap'n
  - fiyatlar ($45.55)
  - Tarihler (01/02/06)
  - URLs (http://www.stanford.edu)
  - hashtags (#nlproc)
  - email addresleri (someone@cs.colorado.edu)
- Kendi başına anlamı olmayan sözcükler nasıl işenecektir?
  - "are" in we're, French "je" in j'ai, "le" in l'honneur
- Çoklu kelime ifadeleri (multiword expressions) ne olacaktır?
- New York, rock 'n' roll

Tokenizasyon: NLTK (Natural Language Toolkit

Bird, Loper and Klein (2009), *Natural Language Processing with Python*. O'Reilly

```
>>> text = 'That U.S.A. poster-print costs $12.40...'
>>> pattern = r'''(?x)    # set flag to allow verbose regexps
...     ([A-Z]\.)+        # abbreviations, e.g. U.S.A.
...   | \w+(-\w+)*        # words with optional internal hyphens
...   | \$?\d+(\.\d+)?%?  # currency and percentages, e.g. $12.40, 82%
...   | \.\.\.            # ellipsis
...   | [][.,;"'?():-_`]  # these are separate tokens; includes ], [
... '''
>>> nltk.regexp_tokenize(text, pattern)
['That', 'U.S.A.', 'poster-print', 'costs', '$12.40', '...']
```

Boşluk içermeyen Dillerde Tokenizasyon

- Pek çok dil (Çince, Japonca, Tai gibi) sözcükleri ayırmada boşlukları kullanmaz.
- Token sınırlarının nerede olacağına nasıl karar verilir?
- Çince kelimeler "hanzi" (veya bazen sadece "zi") adı verilen karakterlerden oluşur.
  - Her biri morfem (sözcüğün en küçük birimi) adı verilen bir anlam birimini temsil eder.
  - Her kelimede ortalama 2,4 tane vardır.
  - Ancak neyin kelime olarak sayılacağına karar vermek karmaşıktır ve üzerinde uzlaşılmış bir konu değildir.
- Tai ve Japonca gibi daha karmaşık diller için sözcük parçalama gerekir.
- Standart algoritmalar, gözetimli (supervised) makine öğrenmesi ile sinir dizisi modelleri eğitilir.

Basic Text Processing

Word Tokenization / Sözcük Belirteçlemesi

- Metin normalizasyonunda önemli bir adımdır.
  - Boşluk- odaklı (space –based) ve
  - Karakter odaklı (character–based) tokenizasyon

  işlemleri ile gerçekleştirilir.

Basic Text Processing

Byte Çifti Kodlaması / Byte Pair Encoding

- Bir metnin belirteçlere/tokenlarına nasıl bölüneceğine karar vermek gerekir.
- Sözcüklerin gövde istatistiklerini kullanan *Bayt Çifti Kodlaması /BPE* algoritması örnek bir metottur.

Metin Belirteçlemede Farklı Yaklaşım

- White-Space Segmentation (boşluklara göre bölütleme)
  - Tek tek karakter bölütlemesi (single-character segmentation) gerçekleştirildikten sonra tüm veri kullanılır ve belirteçlemenin nasıl yapılacağı tanımlanır.
    - ✓ **Altsözcük belirteçlemesi /tokenizasyonu :** Tokenler (belirteçler), kelimenin parçaları (sıralı) olabileceği gibi kelimenin kendisi de olabilir.

Altsözcük Belirteçlemesi / Subword Tokenization

Bu işlemleri gerçekleştiren 3 algoritma vardır:

- **Byte-Pair Encoding (BPE)** (Sennrich et al., 2016)
- **Unigram language modeling tokenization** (Kudo, 2018)
- **WordPiece** (Schuster and Nakajima, 2012 Google tarafından konuşma tanımada kullanıldı ve makine çevirisine uygulandı.BERT (2018)için kullandı. BPE'den farkı olasılıksal kriterlerle işleyişidir)

Bu algoritmaların tümünde süreç iki farklı aşamadan oluşur.

- ***Token (belirteç) öğrenici*** /***Token learner***: Ham eğitim derlemini alır ve bir sözlük oluşturur. Sözlük, *belirteçlerin bir dizilişi* olarak tanımlanabilir.

- **Token (belirteç) bölücü** /***Token segmenter***: Ham test cümlesini alır ve cümleyi sözlüğe göre belirteçlerine ayırır.

Belirteçleme /Tokenizasyon niçin önemlidir?

- Eğitim verisi *low, new, newer* kelimelerini içersin; ama *lower* kelimesin içermesin.
- *lower* kelimesi test veri sitesinde olsun.
- Sistem bu sözcükle ilgili olarak nasıl bir süreç izleyecektir?
  - Çünkü eğitim veri setinde istenilen sözcük bulunmamaktadır.
- Bilinmeyen sözcük sorunu nasıl çözülür?
  - Modern *belirteçleyicilerle* otomatik olarak alt sözcükler oluşturulur
- Bu alt sözcüklerden daha küçük belirteçler içeren *belirteç kümeleri* üretilir.

Belirteçleme Nasıl Yapılır?

- Alt sözcükler keyfi alt dizgiler olabilir ya da *-est* veya *-er* gibi anlam taşıyan birimler olabilir. Bunlar morfemdir.

Morfem: bir dilin anlam taşıyan en küçük birimidir. Örneğin *unlikeliest* sözcüğü *un-, likely ve -est* morfemlerine sahiptir.

- Belirteçleme sürecinde çoğu belirteç sözcük oluşturur
  - Bazı belirteçler sıklıkla görülen morfemler veya *-er* gibi diğer alt sözcüklerdir.
- *Böylece lower* gibi görünmeyen bir sözcük, *low* ve *er* gibi bilinen alt sözcük birimlerinin bir dizilişi olarak veya tek tek harflerin art arda eklemesi olarak temsil edilebilir.

Byte Pair Encoding (BPE) /Token Öğrenici

Sözlük, uygulamanın oluştuğu alfabenin tek tek karakterlerinden oluşur.

sözlük= {A, B, C, D,..., a, b, c, d....}

Sözlüğü tamamlamak amacıyla algoritma aşağıdaki adımları tekrarlar:

- Eğitim derleminden (training corpus) iki sembol seçilir ve bu derlemde sıklığı birbirine en yakın olanlar seçilir (örneğin 'A', 'B')
- 'AB' olarak birleştirilmiş yeni sembol sözlüğe eklenir. 'A' ve 'B' sembolleri derlemdeki ardışık iki semboldür.
- k defa birleştirme tamamlanana kadar aynı süreçler tekrarlanır.

BPE Token Öğrenici Algoritması

function BYTE-PAIR ENCODING(strings C, number of merges k) **returns** vocab V

$V \leftarrow$ all unique characters in C # initial set of tokens is characters
for $i = 1$ **to** k **do** # merge tokens til k times
$t_L, t_R \leftarrow$ Most frequent pair of adjacent tokens in C
$t_{NEW} \leftarrow t_L + t_R$ # make new token by concatenating
$V \leftarrow V + t_{NEW}$ # update the vocabulary
Replace each occurrence of t_L, t_R in C with t_{NEW} # and update the corpus
return V

Byte Pair Encoding (BPE) Algoritması

- Alt sözcük algoritmalarının çoğu *boşlukla ayrılmış belirteçler olarak* işleme girer.
- Bu nedenle eğitim datasında (training data) boşluktan önce sözcüğün sonu olarak özel '__' sembolü eklenir (Bu satırın sonun belirleme yapısı ***derleyici tasarımının*** da ilkesidir).
- Harfleri ayırma işlemi başlar.

BPE Belirteç Öğrenici /Token Learner

derlem

low low low low low lowest lowest newer newer newer newer newer newer wider wider wider new new

Belirteçlerin (tokens) sonunu gösteren im sözlüğe eklenir

vocabulary

_, d, e, i, l, n, o, r, s, t, w

BPE Belirteç Öğrenici /Token Learner

corpus

5 l o w _
2 l o w e s t _
6 n e w e r _
3 w i d e r _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w

BPE algoritması bitişik tüm sembol çiftlerini sayar *er* toplamda 9 kere geçmiştir ve bu semboller birleştirilmelidir.

Merge e r to er

corpus

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

Birleştirilen er tek sembol olarak sözlüğe eklenir.

BPE Belirteç Öğrenici /Token Learner

corpus

5 l o w _
2 l o w e s t _
6 n e w er _
3 w i d er _
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er

er _ yine fazla frekansa sahiptir.
Sistem *er son eki* için belirteçleme yapar ve *er_* olarak birleştirilir.

Merge er _ to er_

corpus

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

BPE belirteç öğrenici /token learner

corpus

5 l o w _
2 l o w e s t _
6 n e w er_
3 w i d er_
2 n e w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_

Merge n e to ne

corpus

5 l o w _
2 l o w e s t _
6 ne w er_
3 w i d er_
2 ne w _

vocabulary

_, d, e, i, l, n, o, r, s, t, w, er, er_, ne

BPE belirteç öğrenici /token learner

Sonraki birleştirmeler:

Merge	Current Vocabulary
(ne, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new
(l, o)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo
(lo, w)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low
(new, er_)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_
(low, _)	_, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_

Test Cümlesinin Belirteçlenmesi

- Sözlük (kelime dağarcığı) öğrenimi tamamlandıktan sonra, belirteç bölücü (**token segmentation**) bir test cümlesi belirteçlemek üzere kullanılır.
- Belirteç bölücü (token segmenter), eğitim verilerinden öğrenilen birleştirmeleri öğrenilen sırada, *test verileri* üzerinde çalıştırır.
  - Bu süreçte test verilerinin frekanslarının herhangi bir etkisi olmaz, **sadece *eğitim verilerindeki frekanslar*** algoritmanın işleyişinde önemlidir.

<u>Örneğin işleyişi:</u>

- ***İlk adım***: her test cümlesi kelimesi karakterlere bölünür.
- ***İkinci adım***: ilk kural uygulanır. Test gövdesindeki her e r örneği er ile değiştirilir.
- ***Üçüncü kural***: ikinci kural uygulanır. Test gövdesindeki her er _ örneği er_ ile değiştirilir.

BPE Andaç Ayrıştırıcı /**segmenter** Algoritma

Test datasında, eğitim datasından öğrenilmiş olan her birleştirme çalıştırılır, yani sonraki işlemlere girecektir.

- Öğrenildiği sırada (testteki frekansların bir etkisi olmayacaktır.

Bu nedenle de her e r yerine er yazılacaktır; daha sonra er _ gösterimi şeklinde birleşecektir (merge).

Sonuç:

"n e w e r _" kümesi bütünüyle tek bir sözcük olarak tokenize edilir.

"l o w e r _" kümesinin iki token (belirteci) olacaktır: . "low er_"

Basic Text Processing

Sözcük Normalizasyonu / Word Normalization

Word Normalization

Sözcüklerin /belirteçlerin standart bir formata koyulması

- U.S.A. or USA
- uhhuh or uh-huh
- Fed or fed
- am, is, be, are

Lemmatization

Represent all words as their lemma, their shared root
= dictionary headword form:

- *am, are, is* → *be*
- *car, cars, car's, cars'* → *car*
- Spanish quiero (‘I want’), quieres (‘you want’)
  → querer ‘want'
- *He is reading detective stories*
  → *He be read detective story*

Lemmatization is done by Morphological Parsing

Morphemes:

- The small meaningful units that make up words
  - **Stems**: The core meaning-bearing units
  - **Affixes**: Parts that adhere to stems, often with grammatical functions

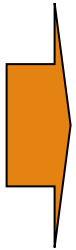
Morphological Parsers:

- Parse *cats* into two morphemes *cat* and *s*

Stemming

Reduce terms to stems, chopping off affixes crudely

This was not the map we found in Billy Bones's chest, but an accurate copy, complete in all things-names and heights and soundings-with the single exception of the red crosses and the written notes.



Thi wa not the map we found in Billi Bone s chest but an accur copi complet in all thing name and height and sound with the singl except of the red cross and the written note.

https://tartarus.org/martin/PorterStemmer/

Porter Stemmer

Based on a series of rewrite rules run in series

A cascade, in which output of each pass fed to next pass

Some sample rules:

$$
\begin{aligned}
\text{ATIONAL} &\rightarrow \text{ATE} \quad \text{(e.g., relational} \rightarrow \text{relate)} \\
\text{ING} &\rightarrow \epsilon \quad \text{if stem contains vowel (e.g., motoring} \rightarrow \text{motor)} \\
\text{SSES} &\rightarrow \text{SS} \quad \text{(e.g., grasses} \rightarrow \text{grass)}
\end{aligned}
$$

Karmaşık Morfolojik Yapıya sahip olan diller

- e.g., the Turkish word:
- Uygarlastiramadiklarimizdanmissinizcasina
- `(behaving) as if you are among those whom we could not civilize’
- Uygar `civilized’ + las `become’

+ tir `cause’ + ama `not able’

+ dik `past’ + lar ‘plural’

+ imiz ‘p1pl’ + dan ‘abl’

+ mis ‘past’ + siniz ‘2pl’ + casina ‘as if’