# *Text Processing: Basics*

Pawan Goyal

CSE, IITKGP

Week 1: Lecture 5

# *Text processing: tokenization*

*What is Tokenization?*

Tokenization is the process of segmenting a string of characters into words.

Depending on the application in hand, you might have to perform *sentence segmentation* as well.

# Sentence Segmentation

The problem of deciding where the sentences begin and end.

## Challenges Involved

# Sentence Segmentation

The problem of deciding where the sentences begin and end.

## Challenges Involved

- While '!', '?' are quite unambiguous

# Sentence Segmentation

The problem of deciding where the sentences begin and end.

## Challenges Involved

- While '!', '?' are quite unambiguous
- Period "." is quite ambiguous and can be used additionally for
  - Abbreviations (Dr., Mr., m.p.h.)

# Sentence Segmentation

The problem of deciding where the sentences begin and end.

## Challenges Involved

- While '!', '?' are quite unambiguous
- Period "." is quite ambiguous and can be used additionally for
  - Abbreviations (Dr., Mr., m.p.h.)
  - Numbers (2.4%, 4.3)

# Sentence Segmentation

The problem of deciding where the sentences begin and end.

### Challenges Involved

- While ‘!’, ‘?’ are quite unambiguous
- Period “.” is quite ambiguous and can be used additionally for
  - Abbreviations (Dr., Mr., m.p.h.)
  - Numbers (2.4%, 4.3)

### Approach: build a binary classifier

For each “.”

- Decides EndOfSentence/NotEndOfSentence

# Sentence Segmentation

The problem of deciding where the sentences begin and end.

## Challenges Involved

- While '!', '?' are quite unambiguous
- Period "." is quite ambiguous and can be used additionally for
  - Abbreviations (Dr., Mr., m.p.h.)
  - Numbers (2.4%, 4.3)

## Approach: build a binary classifier

For each "."

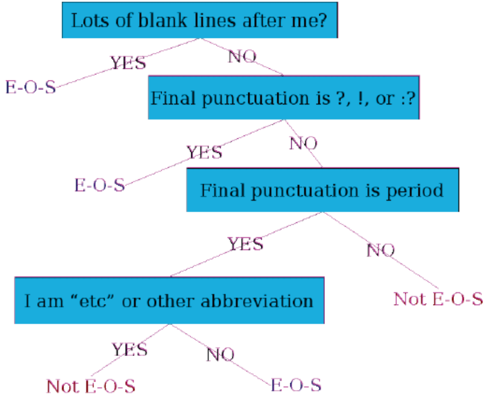
- Decides EndOfSentence/NotEndOfSentence
- Classifiers can be: hand-written rules, regular expressions, or machine learning

# Sentence Segmentation: Decision Tree Example

Decision Tree: Is this word the end-of-sentence (E-O-S)?

# Sentence Segmentation: Decision Tree Example

Decision Tree: Is this word the end-of-sentence (E-O-S)?



# Other Important Features

# *Other Important Features*

- Case of word with ".": Upper, Lower, Cap, Number

# Other Important Features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number

# Other Important Features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number
- Numeric Features

# Other Important Features

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number
- Numeric Features
  - Length of word with "."

# *Other Important Features*

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number
- Numeric Features
  - Length of word with "."
  - Probability (word with "." occurs at end-of-sentence)

# *Other Important Features*

- Case of word with ".": Upper, Lower, Cap, Number
- Case of word after ".": Upper, Lower, Cap, Number
- Numeric Features
  - Length of word with "."
  - Probability (word with "." occurs at end-of-sentence)
  - Probability (word after "." occurs at beginning-of-sentence)

# *Implementing Decision Trees*

# Implementing Decision Trees

- Just an if-then-else statement

# Implementing Decision Trees

- Just an if-then-else statement
- Choosing the features is more important

# *Implementing Decision Trees*

- Just an if-then-else statement
- Choosing the features is more important
- For numeric features, thresholds are to be picked

# *Implementing Decision Trees*

- Just an if-then-else statement
- Choosing the features is more important
- For numeric features, thresholds are to be picked
- With increasing features including numerical ones, difficult to set up the structure by hand

# *Implementing Decision Trees*

- Just an if-then-else statement
- Choosing the features is more important
- For numeric features, thresholds are to be picked
- With increasing features including numerical ones, difficult to set up the structure by hand
- Decision Tree structure can be learned using machine learning over a training corpus

# Implementing Decision Trees

- Just an if-then-else statement
- Choosing the features is more important
- For numeric features, thresholds are to be picked
- With increasing features including numerical ones, difficult to set up the structure by hand
- Decision Tree structure can be learned using machine learning over a training corpus

### Basic Idea

Usually works top-down, by choosing a variable at each step that best splits the set of items.
Popular algorithms: ID3, C4.5, CART

# Other Classifiers

# Other Classifiers

The questions in the decision tree can be thought of as features, that could be exploited by any other classifier:

# Other Classifiers

The questions in the decision tree can be thought of as features, that could be exploited by any other classifier:

- Support Vector Machines
- Logistic regression
- Neural Networks

# Word Tokenization

### *What is Tokenization?*

Tokenization is the process of segmenting a string of characters into words.

# Word Tokenization

### What is Tokenization?

Tokenization is the process of segmenting a string of characters into words.

*I have a can opener; but I can't open these cans.*

### Word Token

- An occurrence of a word
- For the above sentence, 11 word tokens.

### Word Type

- A different realization of a word
- For the above sentence, 10 word types.

# Tokenization in practice

- NLTK Toolkit (Python)
- Stanford CoreNLP (Java)
- Unix Commands

# Word Tokenization

# Word Tokenization

## Issues in Tokenization

- Finland's $\rightarrow$ Finland Finlands Finland's ?
- What're, I'm, shouldn't $\rightarrow$ What are, I am, should not ?
- San Francisco $\rightarrow$ one token or two?
- m.p.h. $\rightarrow$ ??

# Word Tokenization

### Issues in Tokenization

- Finland's $\rightarrow$ Finland Finlands Finland's ?
- What're, I'm, shouldn't $\rightarrow$ What are, I am, should not ?
- San Francisco $\rightarrow$ one token or two?
- m.p.h. $\rightarrow$ ??

For information retrieval, use the same convention for documents and queries

# Handling Hyphenation

Hyphens can be

# Handling Hyphenation

Hyphens can be

### *End-of-Line Hyphen*

Used for splitting whole words into part for text justification.

*This paper describes MIMIC, an adaptive mixed initia-tive spoken dialogue system that provides movie show-time information.*

# *Handling Hyphenation*

Hyphens can be

### *End-of-Line Hyphen*

Used for splitting whole words into part for text justification.

*This paper describes MIMIC, an adaptive mixed initia-tive spoken dialogue system that provides movie show-time information.*

### *Lexical Hyphen*

Certain prefixes are offen written hyphenated, e.g. co-, pre-, meta-, multi-, etc.

# Handling Hyphenation

Hyphens can be

### End-of-Line Hyphen

Used for splitting whole words into part for text justification.
*This paper describes MIMIC, an adaptive mixed initia-tive spoken dialogue system that provides movie show-time information.*

### Lexical Hyphen

Certain prefixes are offen written hyphenated, e.g. co-, pre-, meta-, multi-, etc.

### Sententially Determined Hyphenation

Mainly to prevent incorrect parsing of the phrase. Some possible usages:

- Noun modified by an 'ed'-verb: *case-based, hand-delivered*
- Entire expression as a modifier in a noun group: *three-to-five-year direct marketing plan*

# *Language Specific Issues: French and German*

*French*

**l’ensemble**: want to match with un ensemble

# *Language Specific Issues: French and German*

### *French*

**l'ensemble**: want to match with un ensemble

### *German*

Noun coumpounds are not segmented

- Lebensversicherungsgesellschaftsangestellter
- 'life insurance company employee'
- Compound splitter required for German information retrieval

# Language Specific Issues: Chinese and Japanese

No space between words

莎拉波娃现在居住在美国东南部的佛罗里达。

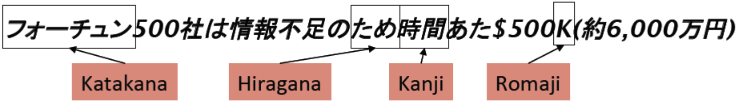
| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 莎拉波娃 | 现在 | 居住 | 在 | 美国 | 东南部 | 的 | 佛罗里达 |
| Sharapova | now | lives in | | US | southeastern | | Florida |

No space between words

莎拉波娃现在居住在美国东南部的佛罗里达。

| 莎拉波娃 | 现在 | 居住 | 在 | 美国 | 东南部 | 的 | 佛罗里达 |
|---|---|---|---|---|---|---|---|
| Sharapova | now | lives | in | US | southeastern | | Florida |

**Japanese:** further complications with multiple alphabets intermingled.



# *Language Specific Issues: Sanskrit*

सत्यम्ब्रूयात्प्रियम्ब्रूयान्नब्रूयात्सत्यमप्रियम्प्रियञ्चनानृतम्ब्रूयादेषधर्मःसनातनः

*satyaṃbrūyātpriyaṃbrūyānnabrūyātsatyamapriyaṃpriyaṃcanānṛtambrūyād-eṣadharmaḥsanātanaḥ*.
"One should tell the truth, one should say kind words; one should neither tell harsh truths, nor flattering lies; this is a rule for all times."

सत्यम्ब्रूयात्प्रियम्ब्रूयान्नब्रूयात्सत्यमप्रियम्प्रियश्चनानृतम्ब्रूयादेषधर्मःसनातनः

*satyaṃbrūyātpriyaṃbrūyānnabrūyātsatyamapriyaṃpriyaṃcanānṛtambrūyād-eṣadharmaḥsanātanaḥ*.

"One should tell the truth, one should say kind words; one should neither tell harsh truths, nor flattering lies; this is a rule for all times."

**Segmented Text:**

*satyam brūyāt priyam brūyāt na brūyāt satyam apriyam priyam ca na anṛtam brūyāt eṣaḥ dharmaḥ sanātanaḥ*.

# Longest Words

| Max | Language (non scientific) |
|---|---|
| **431** | **Sanskrit** *(Longest)* |
| 173 | Greek |
| 136 | Afrikaans |
| 85 | Māori |
| 79 | German |
| 74 | Turkish |
| 64 | Icelandic |
| 56 | Hungarian |
| 54 | Spanish |
| 49 | Dutch |
| 46 | Malay |
| 45 | English |
| 44 | Romanian |
| 42 | Georgian |
| 41 | Czech |
| 39 | Bulgarian |
| 39 | Lithuanian |
| 36 | Kazakh |
| 33 | Norwegian |
| 32 | Tagalog |
| 32 | Polish |
| 30 | Serbian |
| 30 | Montenegrin |
| 30 | Italian |
| 30 | Croatian |

# *Longest Words*

Compound word composed of 431 letters, from the Varadāmbikā Pariṅaya Campū by Tirumalāmba

निरन्तरान्धकारिता-दिगन्तर-कन्दलदमन्द-सुधारस-बिन्दु-सान्द्रतर-घनाघन-वृन्द-सन्देहकर-स्यन्दमान-मकरन्द-बिन्दु-बन्धुरतर-माकन्द-तरु-कुल-तल्प-कल्प-मृदुल-सिकता-जाल-जटिल-मूल-तल-मरुवक-मिलदलघु-लघु-लय-कलित-रमणीय-पानीय-शालिका-बालिका-करार-विन्द-गलन्तिका-गलदेला-लवङ्ग-पाटल-घनसार-कस्तूरिकातिसौरभ-मेदुर-लघुतर-मधुर-शीतलतर-सलिलधारा-निराकरिष्णु-तदीय-विमल-विलोचन-मयूख-रेखापसारित-पिपासायास-पथिक-लोकान्

# *Word Tokenization in Chinese or Sanskrit*

Also called '**Word Segmentation**'.

# *Word Tokenization in Chinese or Sanskrit*

Also called '**Word Segmentation**'.

### *Greedy Algorithm for Chinese*

**Maximum Matching (Greedy Algorithm)**

- Start a pointer at the beginning of the string
- Find the largest word in dictionary that matches the string starting at pointer
- Move the pointer over the word in string

*Think of the cases when word segmentation would be required for English Text.*

# Word Tokenization in Chinese or Sanskrit

Also called '**Word Segmentation**'.

*Greedy Algorithm for Chinese*

**Maximum Matching (Greedy Algorithm)**

- Start a pointer at the beginning of the string
- Find the largest word in dictionary that matches the string starting at pointer
- Move the pointer over the word in string

*Think of the cases when word segmentation would be required for English Text.*

Finding constituent words in a compound hashtags: #ThankYouSachin, #musicmonday etc.

# *Text Segmentation for Sanskrit*

[1]

### *General assumption behind the design*

Sentences from Classical Sanskrit may be generated by a regular relation $R$ of the Kleene closure $W^*$ of a regular set $W$ of *words* over a finite alphabet $\Sigma$.

[1] http://sanskrit.inria.fr

# *Text Segmentation for Sanskrit*

[1]

***General assumption behind the design***

Sentences from Classical Sanskrit may be generated by a regular relation $R$ of the Kleene closure $W^*$ of a regular set $W$ of *words* over a finite alphabet $\Sigma$.

- $W$: vocabulary of (inflected) words (*padas*) and
- $R$: sandhi

[1] http://sanskrit.inria.fr

# Text Segmentation for Sanskrit

[1]

### General assumption behind the design

Sentences from Classical Sanskrit may be generated by a regular relation $R$ of the Kleene closure $W^*$ of a regular set $W$ of *words* over a finite alphabet $\Sigma$.

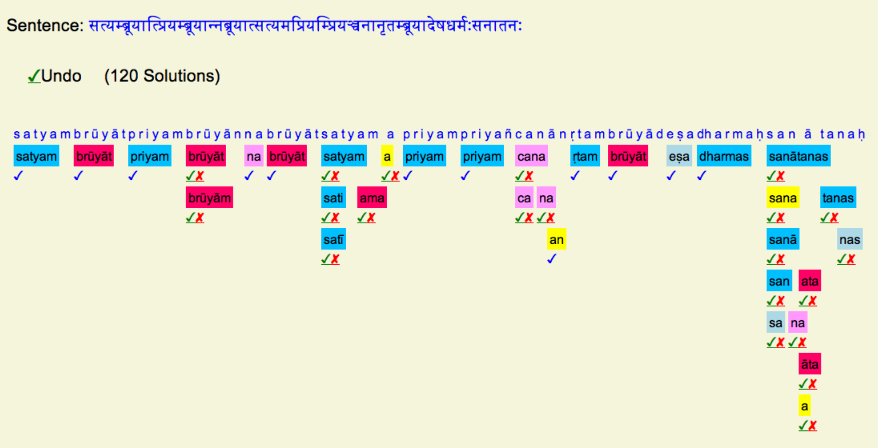
- $W$: vocabulary of (inflected) words (*padas*) and
- $R$: sandhi

### Analysis of a sentence

A candidate sentence $w$ is analyzed by inverting relation $R$ to produce a finite sequence $w_1, w_2, ...w_n$ of word forms, together with a proof that $w \in R(w_1 \cdot w_2 ... \cdot w_n)$.

[1] http://sanskrit.inria.fr

# Word Segmentation in Sanskrit



# *Normalization*

*Why to "normalize"?*

Indexed text and query terms must have the same form.

- U.S.A. and USA should be matched

# Normalization

*Why to "normalize"?*

Indexed text and query terms must have the same form.

- U.S.A. and USA should be matched
- We implicitly define equivalence classes of terms

# *Case Folding*

- Reduce all letters to lower case

# Case Folding

- Reduce all letters to lower case
- Possible exceptions (Task dependent):
  - Upper case in mid sentence, may point to named entities (e.g. General Motors)

# *Case Folding*

- Reduce all letters to lower case
- Possible exceptions (Task dependent):
  - Upper case in mid sentence, may point to named entities (e.g. General Motors)
  - For MT and inforamtion extraction, some cases might be helpful (*US* vs. *us*)

# Lemmatization

- Reduce inflections or variant forms to base form:
  - am, are, is $\rightarrow$ be
  - car, cars, car's, cars' $\rightarrow$ car
- Have to find the correct dictionary headword form

# Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

# Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

*Morphemes are divided into two categories*

- Stems: The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions

# *Morphology*

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

*Morphemes are divided into two categories*

- Stems: The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions
  - Prefix: un-, anti-, etc (a-, ati-, pra- etc.)

# Morphology

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

*Morphemes are divided into two categories*

- Stems: The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions
  - Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  - Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)

# *Morphology*

Morphology studies the internal structure of words, how words are built up from smaller meaningful units called **morphemes**

*Morphemes are divided into two categories*

- Stems: The core meaning bearing units
- Affixes: Bits and pieces adhering to stems to change their meanings and grammatical functions
  - Prefix: un-, anti-, etc (a-, ati-, pra- etc.)
  - Suffix: -ity, -ation, etc (-taa, -ke, -ka etc.)
  - Infix: '*n*' in '*vindati*' (he knows), as contrasted with *vid* (to know).

# Stemming

- Reducing terms to their stems, used in information retrieval

# *Stemming*

- Reducing terms to their stems, used in information retrieval
- Crude chopping of affixes
  - language dependent

- Reducing terms to their stems, used in information retrieval
- Crude chopping of affixes
  - language dependent
  - *automate(s), automatic, automation* all reduced to *automat*

*for example compressed and compression are both accepted as equivalent to compress.*

for exampl compress and compress ar both accept as equival to compress

# *Porter's algorithm*

### *Step 1a*

- sses $\rightarrow$ ss (caresses $\rightarrow$ caress)
- ies $\rightarrow$ i (ponies $\rightarrow$ poni)
- ss $\rightarrow$ ss (caress $\rightarrow$ caress)
- s $\rightarrow$ $\phi$ (cats $\rightarrow$ cat)

# Porter's algorithm

## Step 1a

- sses $\rightarrow$ ss (caresses $\rightarrow$ caress)
- ies $\rightarrow$ i (ponies $\rightarrow$ poni)
- ss $\rightarrow$ ss (caress $\rightarrow$ caress)
- s $\rightarrow$ $\phi$ (cats $\rightarrow$ cat)

## Step 1b

- (*v*)ing $\rightarrow$ $\phi$ (walking $\rightarrow$ walk, king $\rightarrow$

# Porter's algorithm

### Step 1a

- sses $\rightarrow$ ss (caresses $\rightarrow$ caress)
- ies $\rightarrow$ i (ponies $\rightarrow$ poni)
- ss $\rightarrow$ ss (caress $\rightarrow$ caress)
- s $\rightarrow$ $\phi$ (cats $\rightarrow$ cat)

### Step 1b

- (*v*)ing $\rightarrow$ $\phi$ (walking $\rightarrow$ walk, king $\rightarrow$ king)
- (*v*)ed $\rightarrow$ $\phi$ (played $\rightarrow$ play)

# *Porter's algorithm*

### *Step 2*

- ational $\rightarrow$ ate (relational $\rightarrow$ relate)
- izer $\rightarrow$ ize (digitizer $\rightarrow$ digitize)
- ator $\rightarrow$ ate (operator $\rightarrow$ operate)

# Porter's algorithm

### Step 2

- ational $\rightarrow$ ate (relational $\rightarrow$ relate)
- izer $\rightarrow$ ize (digitizer $\rightarrow$ digitize)
- ator $\rightarrow$ ate (operator $\rightarrow$ operate)

### Step 3

- al $\rightarrow$ $\phi$ (revival $\rightarrow$ reviv)
- able $\rightarrow$ $\phi$ (adjustable $\rightarrow$ adjust)
- ate $\rightarrow$ $\phi$ (activate $\rightarrow$ activ)