

# Natural Language Processing (AI615)

Prepared by Dr. Ashraf Elsayed

## Lecture 1



# Natural Language Processing

## Lecture 1

Prepared by  
**Dr. Ashraf Elsayed**

# Introduction to Natural Language Processing

# What is natural language processing?

- Natural language processing (NLP) refers to the branch of computer science—and more specifically, the branch of [artificial intelligence or AI](#)—concerned with giving computers the ability to understand text and spoken words in much the same way human beings can.

تشير معالجة اللغة الطبيعية NLP إلى فرع من فروع علوم الحاسب - وبشكل أكثر تحديدًا، فرع من الذكاء الاصطناعي - الذي يهتم بمنح أجهزة الحاسب القدرة على فهم النصوص والكلمات المنطوقة بنفس الطريقة التي يستطيع بها البشر.

- NLP combines computational linguistics—rule-based modeling of human language—with statistical, machine learning, and deep learning models. Together, these technologies enable computers to process human language in the form of text or voice data and to ‘understand’ its full meaning, complete with the speaker or writer’s intent and sentiment.

- ❑ NLP drives computer programs that translate text from one language to another,
- ❑ Respond to spoken commands,
- ❑ Summarize large volumes of text rapidly—even in real time.

# NLP use cases

- **Spam detection**
- **Machine translation**
- **Virtual agents and chatbots:** [Virtual agents](#) such as Apple's Siri and Amazon's Alexa
- **Social media sentiment analysis**
- **Text summarization**

- **Classifiers:** classify a set of document into categories, (as spam filters)
- **Information Retrieval:** find relevant documents to a given query.
- **Information Extraction:** Extract useful information from resumes; discover names of people and events they participate in, from a document.
- **Machine Translation:** translate text from one human language into another
- **Question Answering:** find answers to natural language questions in a text collection or database...
- **Summarization:** Produce a readable summary, e.g., news about oil today.
- **Sentiment Analysis:** identify people opinion on a subjective.
- **Speech Processing:** book a hotel over the phone, TTS (for the blind)
- **Spelling checkers, grammar checkers, auto-filling, ..... and more**

# Language Technology

mostly solved

## Spam detection

Let's go to Agra! ✓

Buy Vvvvvvvv ✗

## Part-of-speech (POS) tagging

ADJ ADJ NOUN VERB ADV

Colorless green ideas sleep furiously.

## Named entity recognition (NER)

PERSON ORG LOC

Einstein met with UN officials in Princeton

making good progress

## Sentiment analysis

Best roast chicken in San Francisco! 👍

The waiter ignored us for 20 minutes. 👎

## Coreference resolution

Carter told Mubarak he shouldn't run again.

## Word sense disambiguation (WSD)

I need new batteries for my *mouse*.

## Parsing

I can see Alcatraz from the window!

## Machine translation (MT)

第13届上海国际电影节开幕...

The 13<sup>th</sup> Shanghai International Film Festival...

## Information extraction (IE)

You're invited to our dinner party, Friday May 27 at 8:30

Party  
May 27  
add

still challenging

## Question answering (QA)

Q. How effective is ibuprofen in reducing fever in patients with acute febrile illness?

## Paraphrase

XYZ acquired ABC yesterday

ABC has been taken over by XYZ

## Summarization

The Dow Jones is up

The S&P500 jumped

Housing prices rose

Economy is good

## Dialog

Where is Citizen Kane playing in SF?



Castro Theatre at 7:30. Do you want a ticket?



# Ambiguity makes NLP hard

- Ambiguity at multiple levels
- Word senses: **bank** (finance or river ?)
- Part of speech: **chair** (noun or verb ?)
- Syntactic structure: **I can see a man with a telescope**
- Multiple: **I made her duck**

- NLP applications require several NLP analyses:
  - Word tokenization
  - Sentence boundary detection
  - Part-of-speech (POS) tagging
    - to identify the part-of-speech (e.g. noun, verb) of each word
  - Named Entity (NE) recognition
    - to identify proper nouns (e.g. names of person, location, organization; domain terminologies)
  - Parsing
    - to identify the syntactic structure of a sentence
  - Semantic analysis
    - to derive the meaning of a sentence

# Part-Of-Speech (POS) Tagging

- POS tagging is a process of assigning a POS or lexical class marker to each word in a sentence (and all sentences in a corpus).

Input:       the lead paint is unsafe

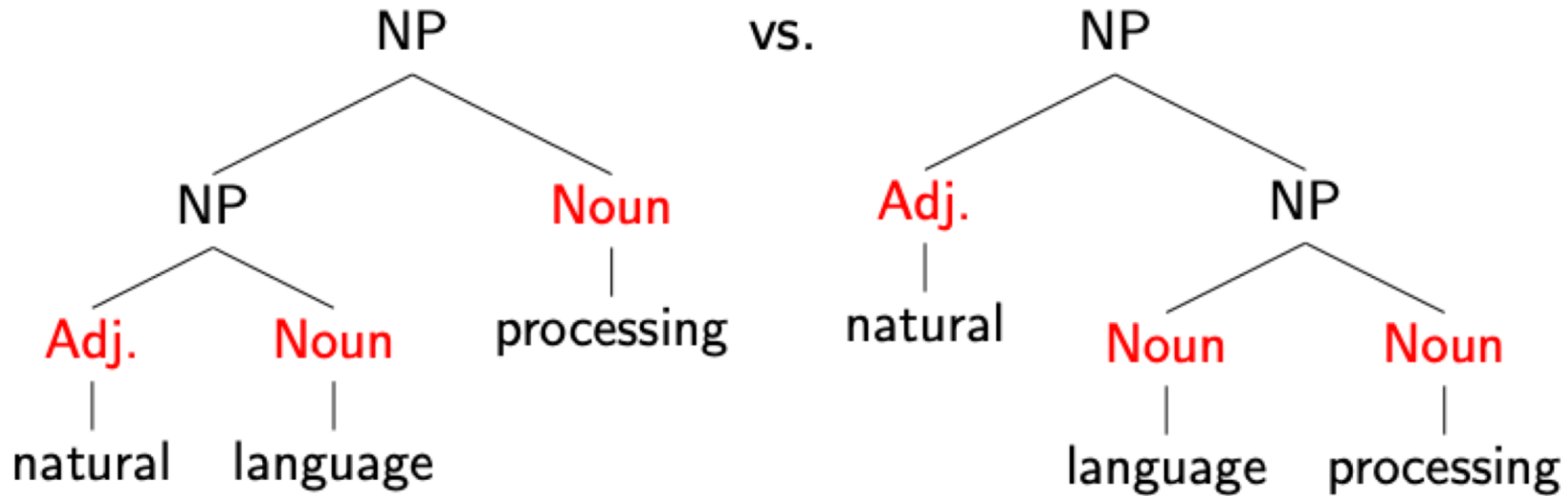
Output:     the/Det lead/N paint/N is/V unsafe/Adj

# Named Entity Recognition (NER)

- NER is to process a text and identify named entities in a sentence
  - e.g. “U.N. official Ekeus heads for Baghdad.”

[ORG U.N. ] official [PER Ekeus ] heads for [LOC Baghdad ] .

# Parsing



# Basic Text Processing

# Basic Text Processing

## Regular Expressions

# Regular expressions

- A formal language for specifying text strings
- How can we search for any of these?
  - woodchuck
  - woodchucks
  - Woodchuck
  - Woodchucks



# Regular Expressions: Disjunctions

- Letters inside square brackets []

| Pattern      | Matches              |
|--------------|----------------------|
| [wW]oodchuck | Woodchuck, woodchuck |
| [1234567890] | Any digit            |

- Ranges [A-Z]

| Pattern | Matches              |                                         |
|---------|----------------------|-----------------------------------------|
| [A-Z]   | An upper case letter | <u>D</u> renched Blossoms               |
| [a-z]   | A lower case letter  | <u>m</u> y beans were impatient         |
| [0-9]   | A single digit       | Chapter <u>1</u> : Down the Rabbit Hole |

# Regular Expressions: Negation in Disjunction

- Negations `[^Ss]`
  - Carat means negation only when first in []

| Pattern             | Matches                  |                                    |
|---------------------|--------------------------|------------------------------------|
| <code>[^A-Z]</code> | Not an upper case letter | O <u>y</u> fn pripetchik           |
| <code>[^Ss]</code>  | Neither 'S' nor 's'      | <u>I</u> have no exquisite reason" |
| <code>[^e^]</code>  | Neither e nor ^          | <u>L</u> ook here                  |
| <code>a^b</code>    | The pattern a carat b    | Look up <u>a^b</u> now             |

# Regular Expressions: More Disjunction

- Woodchucks is another name for groundhog!
- The pipe | for disjunction

| Pattern                                | Matches              |
|----------------------------------------|----------------------|
| <code>groundhog woodchuck</code>       |                      |
| <code>yours mine</code>                | yours<br>mine        |
| <code>a b c</code>                     | = <code>[abc]</code> |
| <code>[gG]roundhog [Ww]oodchuck</code> |                      |



Photo D. Fletcher

# Regular Expressions: ? \* + .

| Pattern | Matches                    |                                                        |
|---------|----------------------------|--------------------------------------------------------|
| colou?r | Optional previous char     | <u>color</u> <u>colour</u>                             |
| oo*h!   | 0 or more of previous char | <u>oh!</u> <u>ooh!</u> <u>oooh!</u><br><u>ooooh!</u>   |
| o+h!    | 1 or more of previous char | <u>oh!</u> <u>ooh!</u> <u>oooh!</u><br><u>ooooh!</u>   |
| baa+    |                            | <u>baa</u> <u>baaa</u> <u>baaaa</u><br><u>baaaaa</u>   |
| beg.n   | Exactly 1 char             | <u>begin</u> <u>begun</u> <u>begun</u><br><u>beg3n</u> |



Stephen C Kleene

Kleene \*, Kleene +

# Regular Expressions: Anchors ^ \$

| Pattern                 | Matches                      |
|-------------------------|------------------------------|
| <code>^[A-Z]</code>     | <u>P</u> alo Alto            |
| <code>^[^A-Za-z]</code> | <u>1</u> "Hello"             |
| <code>\. \$</code>      | The end <u>.</u>             |
| <code>. \$</code>       | The end?<br>The end <u>!</u> |

# Example

- Find me all instances of the word “the” in a text.

the

Misses capitalized examples

[tT]he

Incorrectly returns other or theology

[^a-zA-Z][tT]he[^a-zA-Z]

# Example

```
\w = [A-Za-z0-9_]  
\W = [^a-zA-Z0-9_]  
\d = [0-9]  
\D = [^0-9]
```

{ } use to repeat the previous group

Example:

[0-9]{4} matches 2345 1112 4565

[A-Z]{2,4} matches AD ACC ASDF

[0-9]{3,} matches any text with at least three digits

e.g. ~~34~~ 567 6789 666778

# Grouping

- We use () to group characters
- Example

( [A-Z] [0-9] ) {2}      matches ~~A7~~      A5V6  
does not match ~~A55~~ ~~B56~~ ~~RD5~~

# Errors

- The process we just went through was based on **fixing two kinds of errors**
  - Matching strings that we should not have matched (**there, then, other**)
    - **False positives (Type I)**
  - Not matching things that we should have matched (The)
    - **False negatives (Type II)**

## Errors cont.

- In NLP we are always dealing with these kinds of errors.
- Reducing the error rate for an application often involves two efforts:
  - Increasing accuracy or precision (minimizing false positives)
  - Increasing coverage or recall (minimizing false negatives).

# Summary

- Regular expressions play a surprisingly large role
  - Sophisticated sequences of regular expressions are often the first model for any text processing tasks
- For many hard tasks, we use machine learning classifiers
  - But regular expressions are used as features in the classifiers
  - Can be very useful in capturing generalizations

# Basic Text Processing

## Words and Corpora

# How many words?

- I do uh main- mainly business data processing
  - Fragments, filled pauses
- Seuss's **cat** in the hat is different from other **cats**!
  - **Lemma**: same stem and may be different part of speech
    - **cat** and **cats** = same lemma
  - **Wordform**: the full inflected surface form
    - **cat** and **cats** = different wordforms

# How many words?

they lay back on the San Francisco grass and looked at the stars and their

- **Type**: an element of the vocabulary.
- **Token**: an instance of that type in running text.
- How many?
  - 15 tokens (or 14)
  - 13 types (or 12) (or 11?)

# How many words?

**$N$**  = number of tokens

**$V$**  = vocabulary = set of types

$|V|$  is the size of the vocabulary

Church and Gale (1990):  $|V| > O(N^{1/2})$

|                                       | Tokens = $N$ | Types = $ V $ |
|---------------------------------------|--------------|---------------|
| Switchboard<br>phone<br>conversations | 2.4 million  | 20 thousand   |
| Shakespeare                           | 884,000      | 31 thousand   |
| Google N-grams                        | 1 trillion   | 13 million    |

# Corpora

Words don't appear out of nowhere!

A text is produced by

- a specific writer(s),
- at a specific time,
- in a specific variety,
- of a specific language,
- for a specific function.

# Corpora vary along dimension like

- **Language:** 7097 languages in the world
- **Variety**, like African American Language varieties.
  - AAE Twitter posts might include forms like "*iont*" (*I don't*)
- **Code switching**, e.g., Spanish/English, Hindi/English:
  - S/E: Por primera vez veo a @username actually being hateful! It was beautiful:)  
*[For the first time I get to see @username actually being hateful! it was beautiful:]*
  - H/E: dost tha or ra- hega ... dont worry ... but dherya rakhe  
*["he was and will remain a friend ... don't worry ... but have faith"]*
- **Genre:** newswire, fiction, scientific articles, Wikipedia
- **Author Demographics:** writer's age, gender, ethnicity,

# Corpus datasheets

## **Motivation:**

- Why was the corpus collected?
- By whom?
- Who funded it?

**Situation:** In what situation was the text written?

**Collection process:** If it is a subsample how was it sampled?  
Was there consent? Pre-processing?

- **+Annotation process, language variety, demographics, etc.**

# Basic Text Processing

## Word tokenization

# Text Normalization

- Every NLP task needs to do text normalization:
  1. Segmenting/tokenizing words in running text
  2. Normalizing word formats
  3. Segmenting sentences in running text

# Space-based tokenization

- A very simple way to tokenize
  - For languages that use space characters between words
    - Arabic, Cyrillic, Greek, Latin, etc., based writing systems
  - Segment off a token between instances of spaces
- Unix tools for space-based tokenization
  - The "tr" command
  - Given a text file, output the word tokens and their frequencies

# Simple Tokenization in UNIX

- Given a text file, output the word tokens and their frequencies

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

```
1945 A
 72 AARON
 19 ABBESS
  5 ABBOT
... ..
25 Aaron
 6 Abate
 1 Abates
 5 Abbess
 6 Abbey
 3 Abbot
.... ..
```

Change all non-alpha to newlines

Sort in alphabetical order

Merge and count each type

# The first step: tokenizing

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

THE

SONNETS

by

William

Shakespeare

From

fairest

creatures

We

...

# The second step: sorting

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

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

# More counting

- Merging upper and lower case

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

- Sorting the counts

```
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
```

What happened here?

# Issues in Tokenization

- Can't just blindly remove punctuation:
  - m.p.h., Ph.D., AT&T, cap'n
  - prices (\$45.55)
  - dates (01/02/06)
  - URLs (<http://www.stanford.edu>)
  - hashtags ([#nlproc](#))
  - email addresses ([someone@cs.colorado.edu](mailto:someone@cs.colorado.edu))
- Clitic: a word that doesn't stand on its own
  - "are" in [we're](#), French "je" in [j'ai](#), "le" in [l'honneur](#)
- When should multiword expressions (MWE) be words?
  - [New York](#), [rock 'n' roll](#)

# Tokenization in NLTK

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', '...']
```

# Tokenization in languages without spaces

Many languages (like Chinese, Japanese, Thai) don't use spaces to separate words!

How do we decide where the token boundaries should be?

# How to do word tokenization in Chinese?

- 姚明进入总决赛 “Yao Ming reaches the finals”



# How to do word tokenization in Chinese?

- 姚明进入总决赛 “Yao Ming reaches the finals”

- 3 words?

- 姚明 进入 总决赛

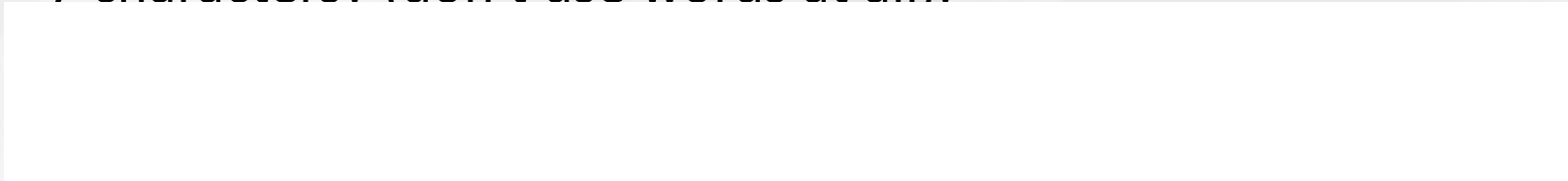
- YaoMing reaches finals

- 5 words?

- 姚 明 进入 总 决赛

- Yao Ming reaches overall finals

- 7 characters? (don't use words at all):



# How to do word tokenization in Chinese?

- 姚明进入总决赛 “Yao Ming reaches the finals”

- 3 words?

- 姚明 进入 总决赛

- YaoMing reaches finals

- 5 words?

- 姚 明 进入 总 决赛

- Yao Ming reaches overall finals

- 7 characters? (don't use words at all):

- 姚 明 进 入 总 决 赛

- Yao Ming enter enter overall decision game

# Word tokenization / segmentation

So in Chinese it's common to just treat each character (zi) as a token.

- So the **segmentation** step is very simple

In other languages (like Thai and Japanese), more complex word segmentation is required.

- The standard algorithms are neural sequence models trained by supervised machine learning.



# Thank You