Regular Expression

& Python RegEx

Basic Text Processing

A NLP Tool - NLTK

Dr. Liao

1/28/2020

Overview

- Regular Expressions
  - Definitions, Types, Engines, Examples
  - Python RegEx Code Demos & Hands-On Practice in Class
- Basic Text Processing
  - Text Preprocessing
    - Tokenization & Segmentation
    - Normalization – Stemming & Lemmatization
    - Noise Removal - Stop Words Removal
  - Parts of Speech (POS) Tagging
  - N-Grams
  - Vectorization - Bag of Words (BOW) & TF-IDF
  - In-Class Hands-On Programming with Python & NLTK

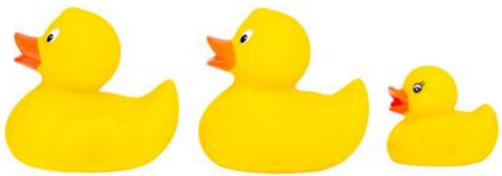
Regular Expression

Python Code Examples
Hands-On Practice in Class

Dr. Liao

1/28/2020

Regular Expressions



- **Questions?**
  - How to search for any of these words in each group?
    - 1) duck, ducks, Duck, Ducks
    - 2) goose, geese, Goose, Geese

- Definitions
  - A formal language for specifying text strings
    - Defined by Stanford Univ. Prof. Dan Jurafsky
  - A sequence of characters that define a search pattern
    - from Wikipedia

These images from the internet

Regular expressions (cont.)

- A powerful and standardized way of searching, replacing, and parsing text with complex patterns of characters
- Capturing text patterns
  - Rule-based or statistical methods

Uses of Regular Expressions in NLP

- Simple but powerful tools for large corpus analysis and ‘shallow’ processing
  - What word is most likely to begin a sentence?
  - What word is most likely to begin a question?
  - In your own email, are you more or less polite than the people you correspond with?
- They allow us to:
  - Obtain word frequency and co-occurrence statistics
  - Build simple interactive applications (e.g., **Eliza**)
  - Recognize date, time, money… expressions
  - Recognize Named Entities (NE): people names, company names
  - Do morphological analysis
- Regular expressions define **regular** languages or sets

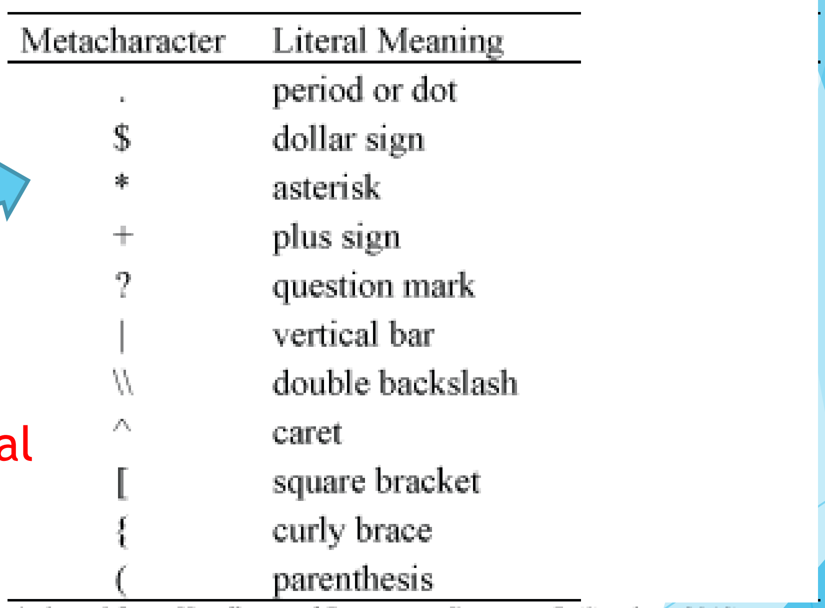
Adopted from Dr. Uzuner’s NLP590 Course Slide

Regular Expression Engines

- **Regex Engines**
  - Software
    - Python, R, Perl, .NET, Java, Javascript, PHP, …
  - Different regular expression engines are NOT fully compatible with each other
  - Online RegEx test engines
    - **Regex101.com**

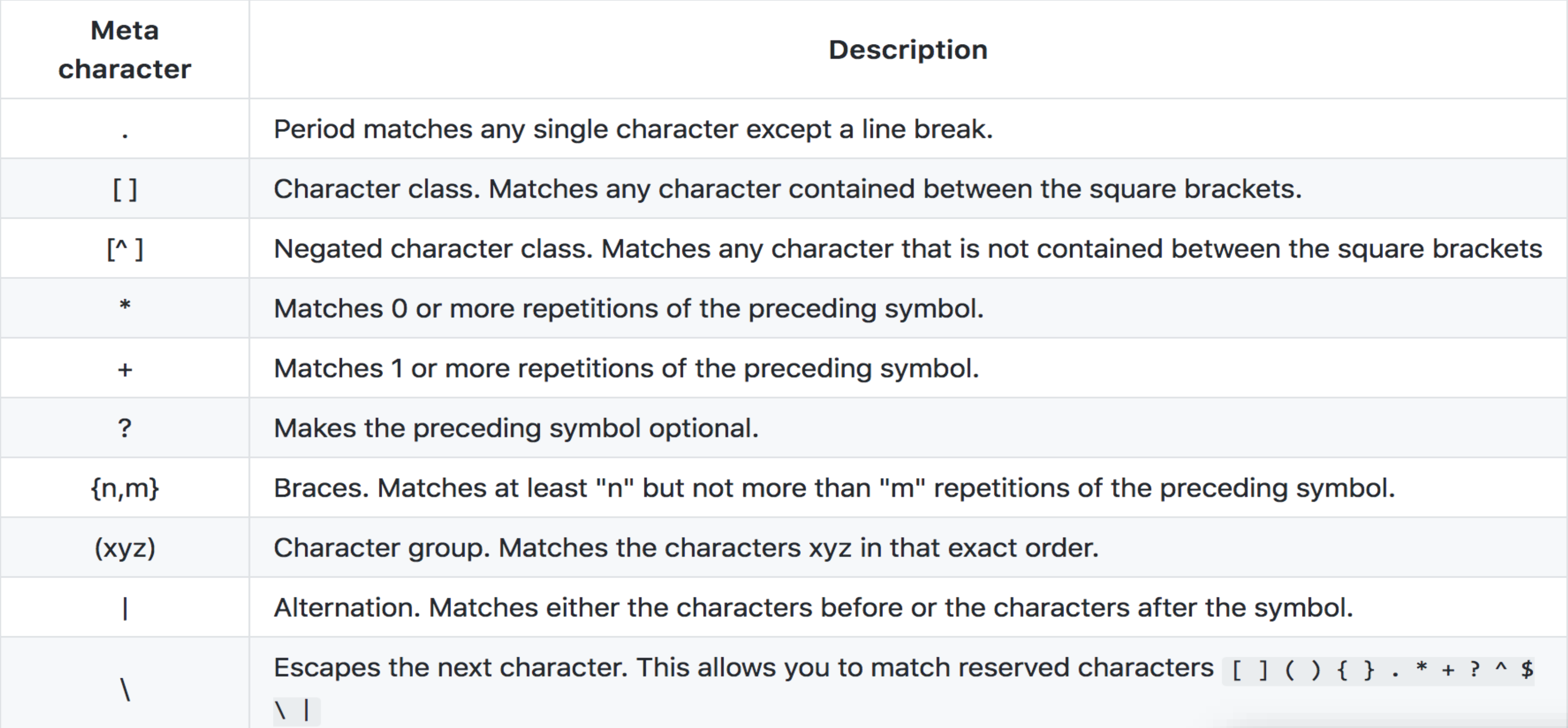
Types of Regular Expressions

- **Literals**- Normal text characters
  - Match the occurrences of the character in the string
    - E.g.: Sally is a dog.
- **Metacharacters** - Special characters
  - flexible to search
  - 12 characters have special meanings in regex
    - **Escape character - **the backslash ****
      - Treat a subsequent metacharacter as a literal
      - E.g. “2+3=5”
        - “2\+3=5” (correct regex)
    - Most of them are errors when used alone.

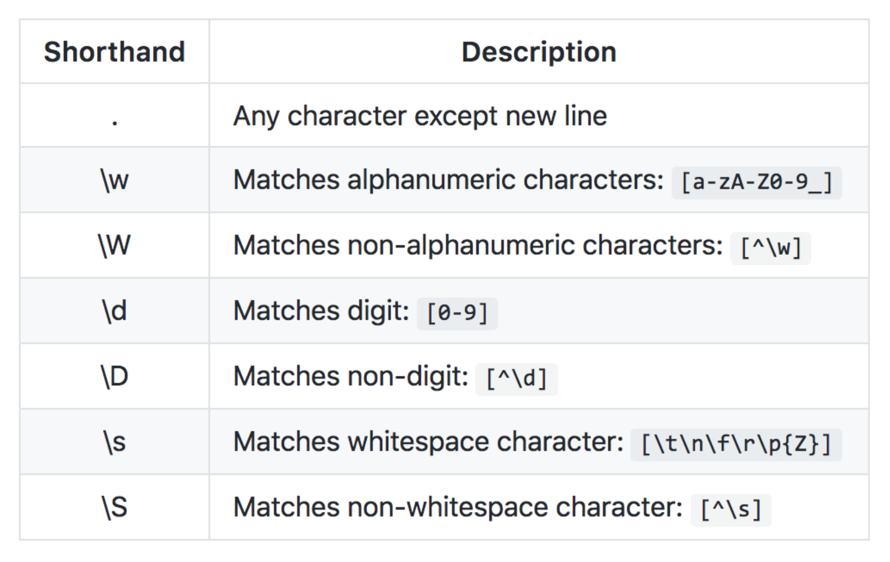


Metacharacter	Literal Meaning
.	period or dot
$	dollar sign
*	asterisk
+	plus sign
?	question mark
|	vertical bar
\\	double backslash
^	caret
[	square bracket
{	curly brace
(	parenthesis

*adapted from *Handling and Processing Strings in R* (Sanchez, 2013)



Meta character	Description
.	Period matches any single character except a line break.
[ ]	Character class. Matches any character contained between the square brackets.
[^]	Negated character class. Matches any character that is not contained between the square brackets
*	Matches 0 or more repetitions of the preceding symbol.
+	Matches 1 or more repetitions of the preceding symbol.
?	Makes the preceding symbol optional.
{n,m}	Braces. Matches at least "n" but not more than "m" repetitions of the preceding symbol.
(xyz)	Character group. Matches the characters xyz in that exact order.
|	Alternation. Matches either the characters before or the characters after the symbol.
\	Escapes the next character. This allows you to match reserved characters `[ ] ( ) { } . * + ? ^ $ \ |`
^	Matches the beginning of the input.
$	Matches the end of the input.

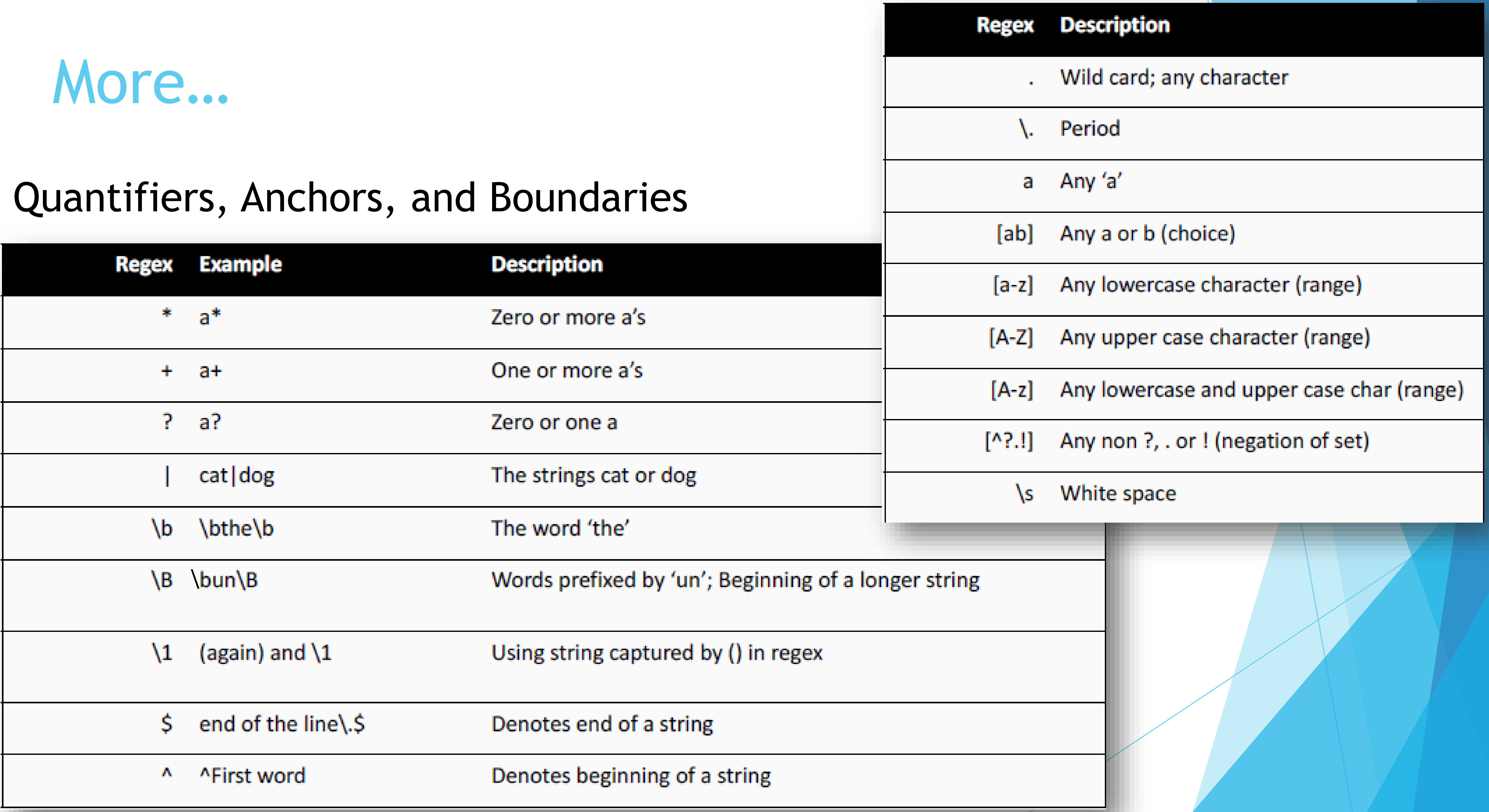


Shorthand	Description
.	Any character except new line
\w	Matches alphanumeric characters: `[a-zA-Z0-9_]`
\W	Matches non-alphanumeric characters: `[^\w]`
\d	Matches digit: `[0-9]`
\D	Matches non-digit: `[^\d]`
\s	Matches whitespace character: `[\t\n\f\r\p{Z}]`
\S	Matches non-whitespace character: `[^\s]`

The tables from the "Regular Expressions — The Last Guide"

More…

Quantifiers, Anchors, and Boundaries



Regex	Example	Description
*	a*	Zero or more a’s
+	a+	One or more a’s
?	a?	Zero or one a
|	cat|dog	The strings cat or dog
\b	\bthe\b	The word ‘the’
\B	\bun\B	Words prefixed by ‘un’; Beginning of a longer string
\1	(again) and \1	Using string captured by () in regex
$	end of the line\.$	Denotes end of a string
^	^First word	Denotes beginning of a string

Regex	Description
.	Wild card; any character
\.	Period
a	Any ‘a’
[ab]	Any a or b (choice)
[a-z]	Any lowercase character (range)
[A-Z]	Any upper case character (range)
[A-z]	Any lowercase and upper case char (range)
[^?.!]	Any non ?, . or ! (negation of set)
\s	White space

These two tables from Dr. Uzuner’s NLP590 Course materials

Regular Expression Examples

- The cat is in the hat.
  - The -> The cat is in the hat.
  - the -> The cat is in the hat.
  - [tT]he -> The cat is in the hat.
  - h?at -> The cat is in the hat.
  - .at -> The cat is in the hat.
  - at. -> The cat is in the hat.
  - at[.] -> The cat is in the hat.
    - A period inside a character set means a **literal period**.

Let's test in
regex101.com
& Python Code

All the corresponding Python code examples are shown in class.

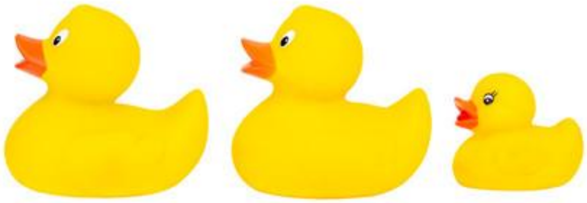
Regular Expression Examples (Cont.)

- The car is parked in the garage #3552.
  - [a-z]*-> The car is parked in the garage #3552.
  - [cpg]ar -> The car is parked in the garage #3552.
    - Or (c|p|g)ar
  - [^c]ar -> The car is parked in the garage #3552.
  - [0-9]{2} -> The car is parked in the garage #3552.
  - [0-9]{1,3} -> The car is parked in the garage #3552.
  - [0-9]{2,} -> The car is parked in the garage #3552.
  - [0-9]+ -> The car is parked in the garage #3552.
  - \d+ -> The car is parked in the garage #3552.

All the corresponding Python code examples are shown in class.

Regular Expression Examples (Cont.)

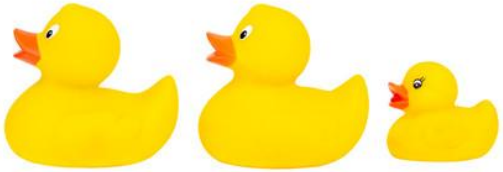
- **Questions?**
  - How to search for any of these words in each group?
    - 1) duck, ducks, Duck, Ducks
    - 2) goose, geese, Goose, Geese



Regular Expression Examples (Cont.)

- **Questions?**
  - How to search for any of these words in each group?
    - **1) duck, ducks, Duck, Ducks**
    - 2) goose, geese, Goose, Geese

- **Answer for 1)**



  - Let’s test in regex101.com
    - Example 1:
      - **Ducker**'s family has four ducks, seven duck toys, and one T-shirt with big "Ducks" words.
  - **\b[dD]ucks?\b**

The corresponding Python code examples are shown in class.

Regular Expression Examples (Cont.)

- **Questions?**
  - How to search for any of these words in each group?
    - 1) duck, ducks, Duck, Ducks
    - **2) goose, geese, Goose, Geese**

- **Answer for 2)**
  - Let’s test in regex101.com
    - Example 2:
      - The goose parents led a line of their baby geese across the river.
  - **\b[gG](oo|ee)se\b**

The corresponding Python code examples are shown in class.

More Questions

- What is the regex to identify all words that begin with ha, or hah, hahh, hahhh, etc. regardless of the h's?
  - \bhah*\B -> ha hah hahhsdd hahhhhaaaa hahhhhhhaaaadfdsf
  - But if \bhah*\b, what will you find?
    - -> ha hah hahhsdd hahhhhaaaa hahhhhhhaaaadfdsf
- What is the regex to identify all the word box and its plural form?
  - \b[bB]ox(es)?\b
- What is the regex to identify ier and ier phrases such as: happier and happier, or fuzzier and fuzzier
  - \b(.+)ier\b

The corresponding Python code examples are shown in class.

More…

- Let’s see Dr. Liao’s NLTK code examples & tutorials for more details for regular expressions & Chatbot…

Regular Expressions in Python
Hands-On Practice in Class

- Regular Expressions Python Code Examples & Tutorials for Text Processing/NLP
  - Dr. Liao wrote them **particularly** for
    - this course learning
    - **Assignments and final project** examples
- All programming tutorials & code example demos
  - Using Jupyter Lab in class

Python Regular Expressions

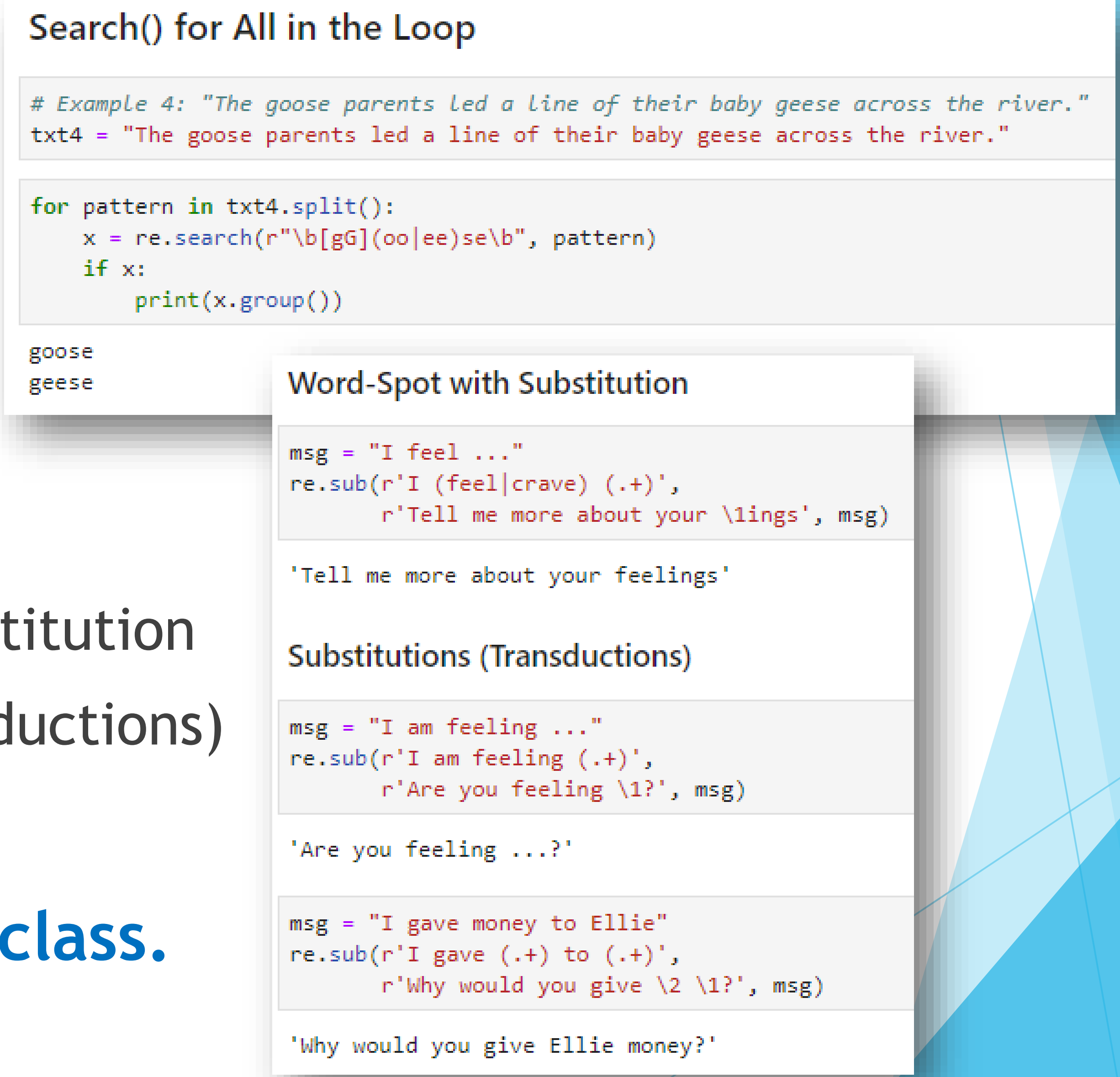
Function	Description
findall	Returns a list containing all matches
Search/Match	Returns a Match object if there is a match anywhere in the string
split	Returns a list where the string has been split at each match
sub	Replaces one or many matches with a string

- Python RegEx References
- 3rd Party **regex** module for Python

Modifying Text

- Key Steps
  - Search
    - to see if match
  - Substitution
    - Word-spot with substitution
    - Substitutions (Transductions)

Let’s see more code examples in Python in class.



Search() for All in the Loop

```
# Example 4: "The goose parents led a line of their baby geese across the river."
txt4 = "The goose parents led a line of their baby geese across the river."
```

```
for pattern in txt4.split():
    x = re.search(r"\b[gG](oo|ee)se\b", pattern)
    if x:
        print(x.group())
```

```
goose
geese
```

Word-Spot with Substitution

```
msg = "I feel ..."
re.sub(r'I (feel|crave) (.+)',
       r'Tell me more about your \1ings', msg)
```

```
'Tell me more about your feelings'
```

Substitutions (Transductions)

```
msg = "I am feeling ..."
re.sub(r'I am feeling (.+)',
       r'Are you feeling \1?', msg)
```

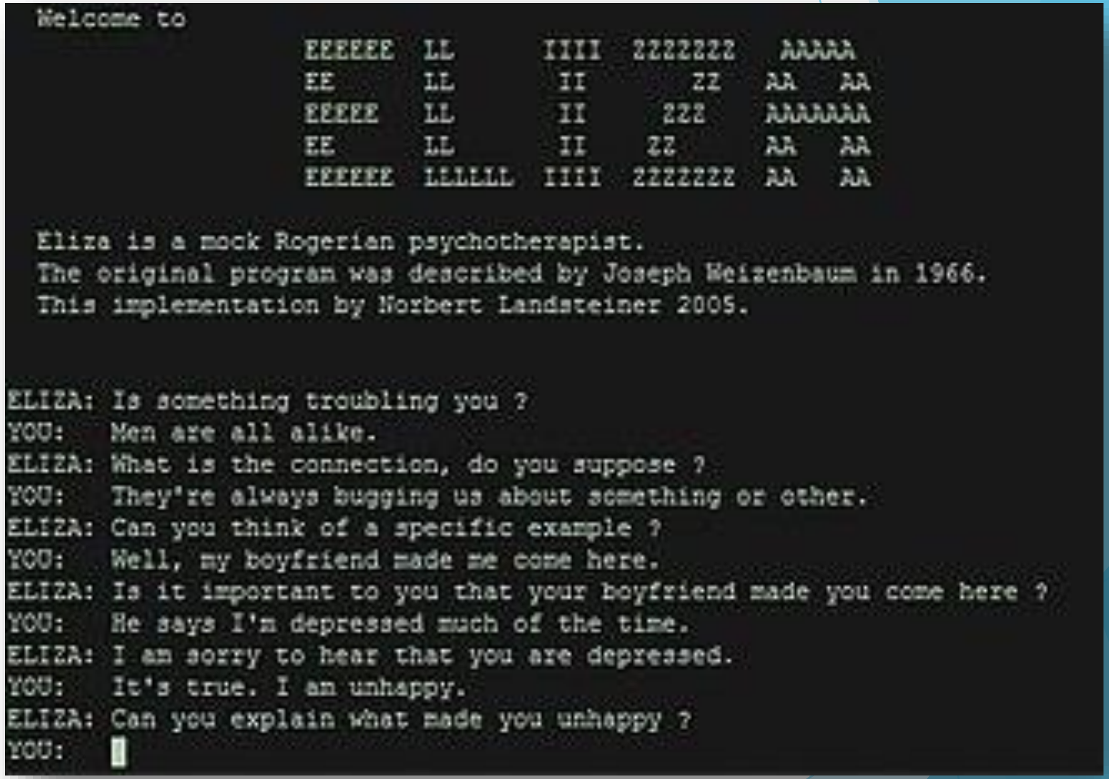
```
'Are you feeling ...?'
```

```
msg = "I gave money to Ellie"
re.sub(r'I gave (.+) to (.+)',
       r'Why would you give \2 \1?', msg)
```

```
'Why would you give Ellie money?'
```

Eliza

- ELIZA is an early natural language processing computer program created from 1964 to 1966 at the MIT Artificial Intelligence Laboratory by Joseph Weizenbaum. (Wikipedia)
  - Original Paper (Scanned PDF)
- “Eliza is the most important chatbot dialogue system in the history.”
  - Dan Jurafsky’s NLP Book 3Ed.
- Domain Knowledge:
  - Artificial Intelligence
    - Expert System
    - Natural Language Processing



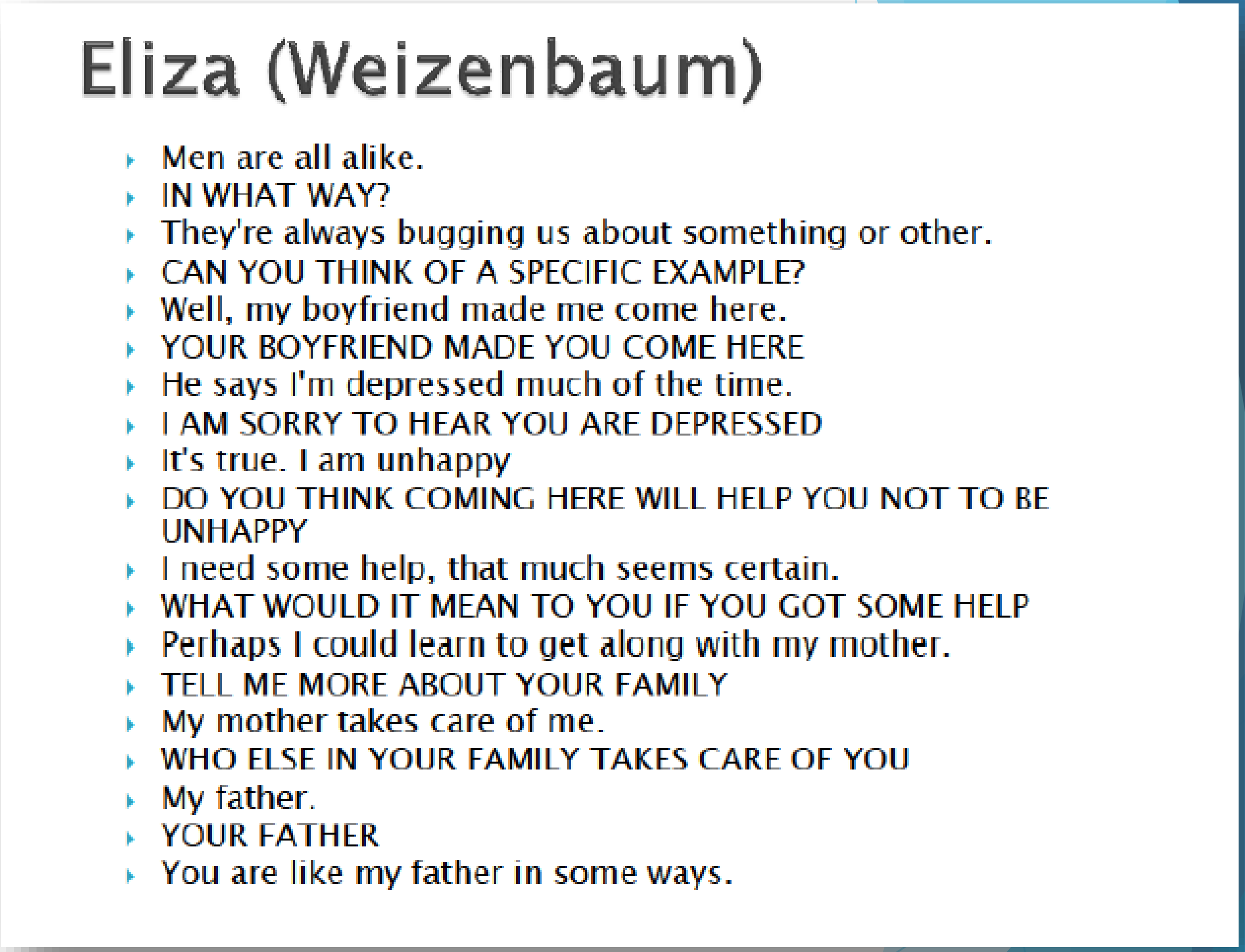
```
 Welcome to
                    EEEEEE  LL      IIII  ZZZZZZZ   AAAAA
                    EE      LL       II       ZZ   AA   AA
                    EEEEE   LL       II     ZZZ    AAAAAAA
                    EE      LL       II    ZZ      AA   AA
                    EEEEEE  LLLLLL  IIII  ZZZZZZZ  AA   AA

 Eliza is a mock Rogerian psychotherapist.
 The original program was described by Joseph Weizenbaum in 1966.
 This implementation by Norbert Landsteiner 2005.


ELIZA: Is something troubling you ?
YOU:   Men are all alike.
ELIZA: What is the connection, do you suppose ?
YOU:   They're always bugging us about something or other.
ELIZA: Can you think of a specific example ?
YOU:   Well, my boyfriend made me come here.
ELIZA: Is it important to you that your boyfriend made you come here ?
YOU:   He says I'm depressed much of the time.
ELIZA: I am sorry to hear that you are depressed.
YOU:   It's true. I am unhappy.
ELIZA: Can you explain what made you unhappy ?
YOU:   
```

Eliza

- In 1966,
  - **Joseph Weizenbaum** developed Eliza a Rogerian* psychotherapist.
- Simple program
  - uses pattern matching to process the input and translate it into suitable outputs.
- Succeeded
  - because Eliza didn’t need to *know* anything in order to simulate a Rogerian psychotherapist.



Eliza (Weizenbaum)

- Men are all alike.
- IN WHAT WAY?
- They're always bugging us about something or other.
- CAN YOU THINK OF A SPECIFIC EXAMPLE?
- Well, my boyfriend made me come here.
- YOUR BOYFRIEND MADE YOU COME HERE
- He says I'm depressed much of the time.
- I AM SORRY TO HEAR YOU ARE DEPRESSED
- It's true. I am unhappy
- DO YOU THINK COMING HERE WILL HELP YOU NOT TO BE UNHAPPY
- I need some help, that much seems certain.
- WHAT WOULD IT MEAN TO YOU IF YOU GOT SOME HELP
- Perhaps I could learn to get along with my mother.
- TELL ME MORE ABOUT YOUR FAMILY
- My mother takes care of me.
- WHO ELSE IN YOUR FAMILY TAKES CARE OF YOU
- My father.
- YOUR FATHER
- You are like my father in some ways.

Amazing thing:

People **believed** that ELIZA understood them and their problems!

* related to the system of therapy or the theory of personality of Carl Rogers.

Adopted from Dr. Uzuner’s NLP590 Course Slide

1st Programming Assignment

- Eliza Chatbot
  - Write an Eliza program in Python
  - Team work
  - Submit in two weeks

- How to Build an Eliza Chatbot?

How to Build an Eliza Chatbot?

- Joseph Weizenbaum’s Original Paper (Scanned PDF)
- Chapter 24 of Dan Jurafsky’s NLP Book 3Ed.
- How to Build Eliza Chatterbot - A Program that can Chat with Humans
- AIT590 Course materials and Python code examples
- Note that:
  - Eliza may not understand what you say.
  - How to deal with the following situations?
    - If you input greeting statements,
    - If you do not input anything or keep typing Enter,
    - If your input does not match any in-built sentence framework
    - If no context is found,
    - If you repeat yourself,
    - If…

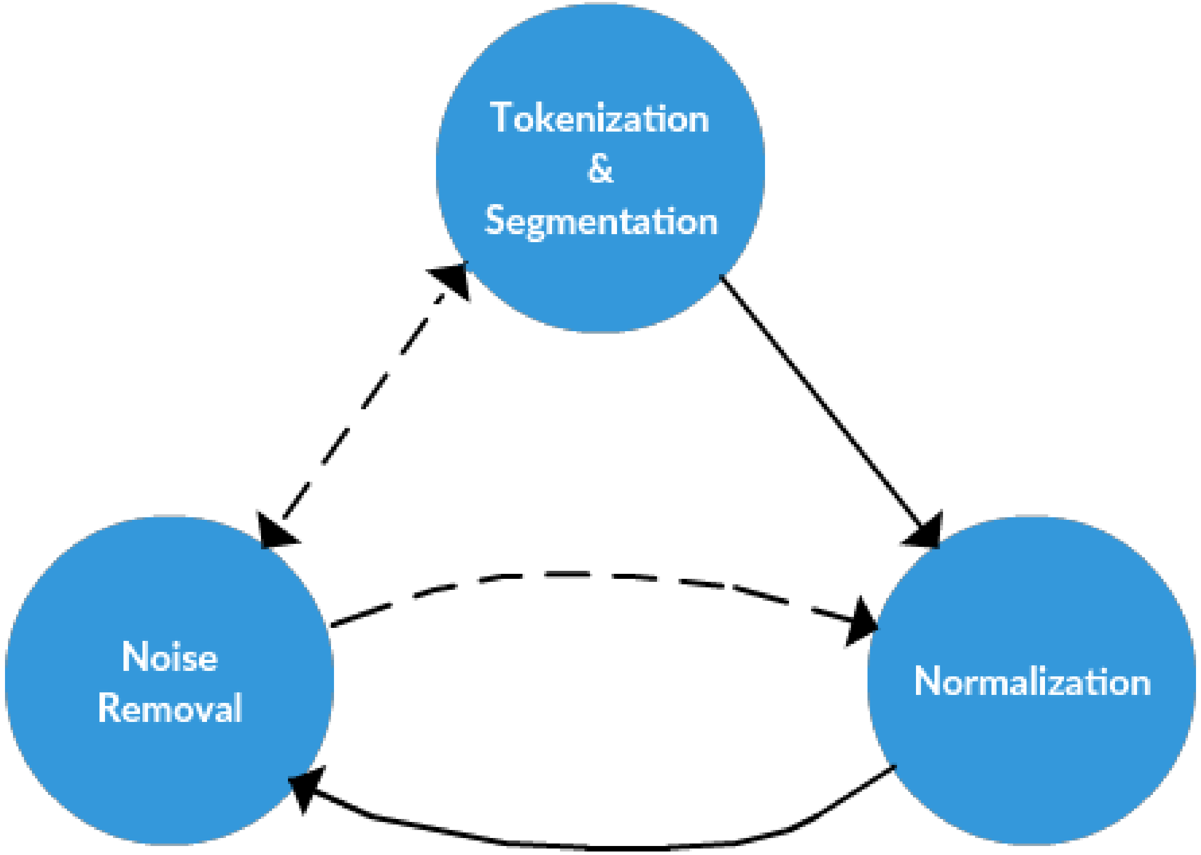
Basic Text Processing

A NLP Tool - NLTK

Dr. Liao

Text Preprocessing

- Tokenization
- Normalization
- Noise Removal

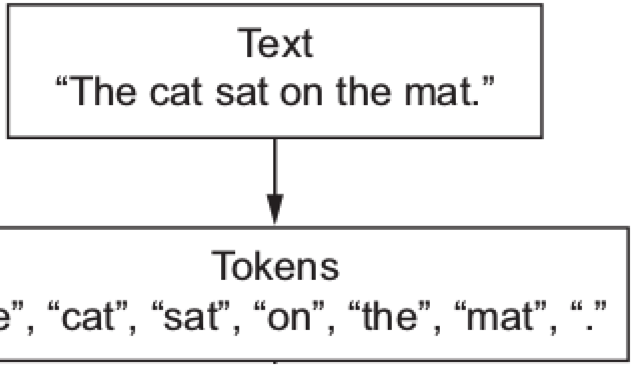


https://www.kdnuggets.com/2017/12/general-approach-preprocessing-text-data.html

Tokenization & Segmentation

- Tokenization
  - Breaking up text document into small pieces or individual words called **tokens**
- Segmentation
  - Breaking down into a larger chunk than tokens ((e.g. paragraphs or sentences)

- Types of Tokenization (NLTK)
  - Sentence Tokenization
  - Word Tokenization



Text
“The cat sat on the mat.”

Tokens
“the”, “cat”, “sat”, “on”, “the”, “mat”, “.”

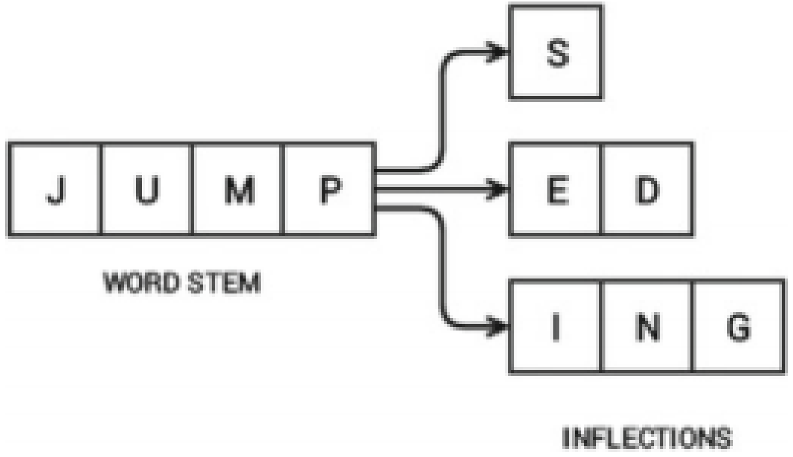
- *NLTK demos will be shown in the in-class programming with code examples later on.*

Normalization

- Stemming
- Lemmatization
- Everything Else

Normalization - Stemming

- The process of reducing a word to its stem/root word.
- Reduces inflection in words (e.g. ‘help’, ’helping’, ’helped’, ’helpful’) to their root form (e.g. ‘help’)
- removes the morphological affixes from words, leaving only the word stem



Word stem and its inflections
(Source: Text Analytics with Python, Apress/Springer 2016)

Normalization (cont.)

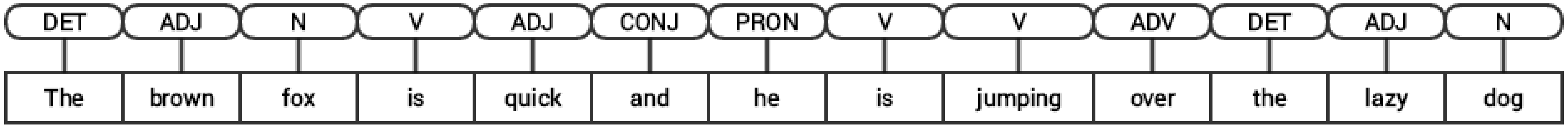
- Stemming
- Lemmatization
  - Related to stemming
  - The difference -> capture canonical forms based on a word's lemma
    - e.g. better → good
- Everything Else
  - substitution or removal
    - set all characters to lowercase or uppercase
    - remove numbers (or convert numbers to textual representations)
    - remove punctuation
    - strip white space (also generally part of tokenization)
    - remove default stop words (English)

Stop Word Removal

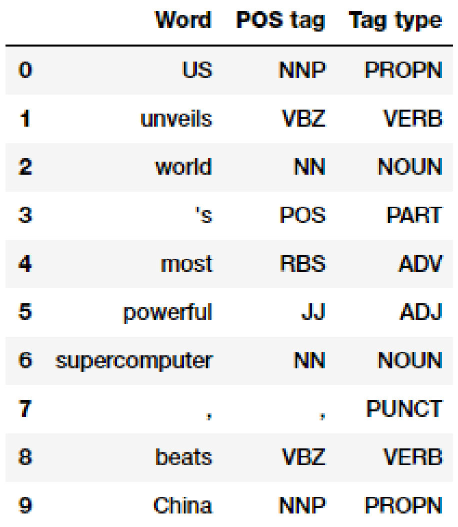
- **Stop words** are common words that do not contribute much of the information in a text document.
  - Words like ‘the’, ‘is’, ‘a’ have less value and add noise to the text data.

Parts of Speech (POS) Tagging

- Each word in a sentence can be classified into classes
  - such as verbs, adjectives, nouns, etc.
- POS Tagging is a process of tagging words in a sentence to particular part-of-speech, based on its definition and context in the sentence.

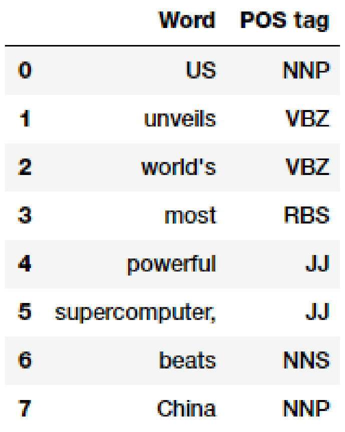


DET	ADJ	N	V	ADJ	CONJ	PRON	V	V	ADV	DET	ADJ	N
The	brown	fox	is	quick	and	he	is	jumping	over	the	lazy	dog



	Word	POS tag	Tag type
0	US	NNP	PROPN
1	unveils	VBZ	VERB
2	world	NN	NOUN
3	's	POS	PART
4	most	RBS	ADV
5	powerful	JJ	ADJ
6	supercomputer	NN	NOUN
7	,	,	PUNCT
8	beats	VBZ	VERB
9	China	NNP	PROPN

SpaCy POS tagging



	Word	POS tag
0	US	NNP
1	unveils	VBZ
2	world's	VBZ
3	most	RBS
4	powerful	JJ
5	supercomputer,	JJ
6	beats	NNS
7	China	NNP

NLTK POS tagging

POS Online

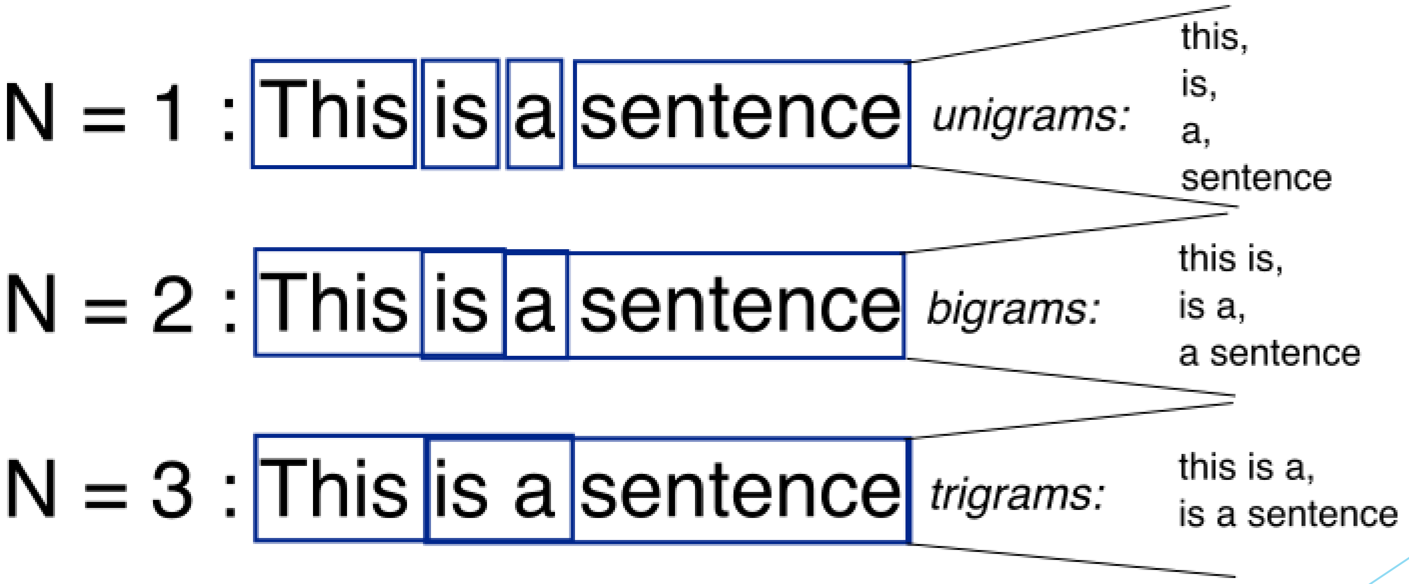
based on
the Stanford University Part-Of-Speech-Tagger

Examples: POS tagging a news headline

Web source

N-Grams

- N-grams are the combination of multiple words used together
- can be used when we want to preserve sequence information in the document, like what word is likely to follow the given one.
- Unigrams don't contain any sequence information because each word is taken individually.



N = 1 : This is a sentence *unigrams:* this, is, a, sentence

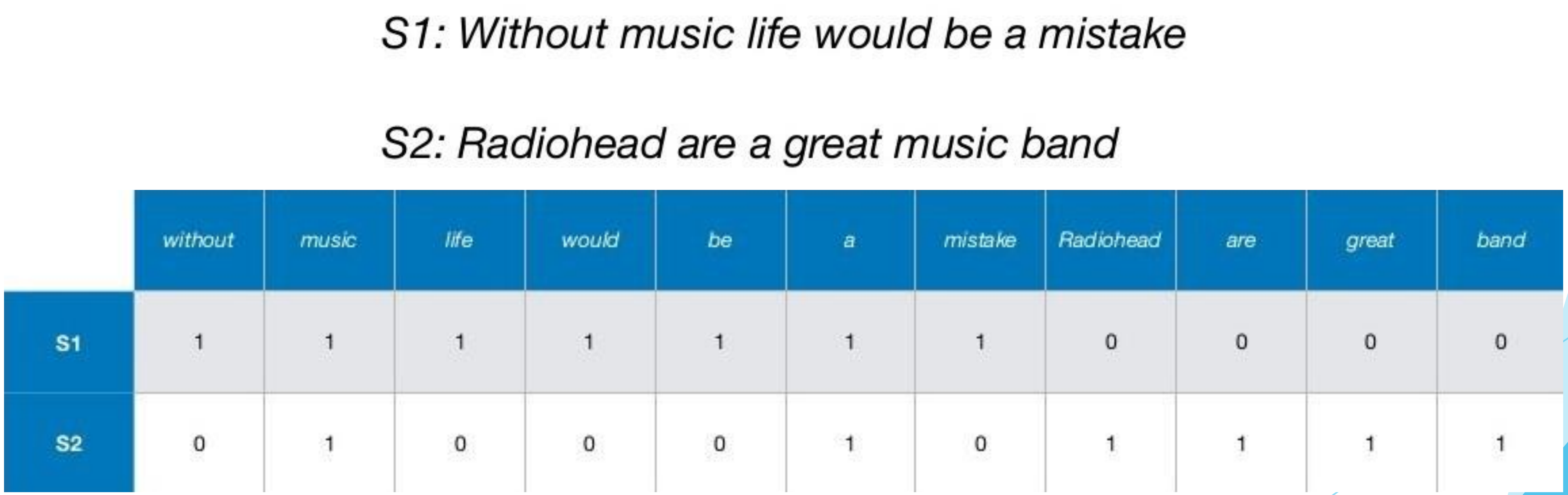
N = 2 : This is a sentence *bigrams:* this is, is a, a sentence

N = 3 : This is a sentence *trigrams:* this is a, is a sentence

Image Source

Vectorization

- Definition:
  - The process of converting text into numbers
- Method 1: Bag of Words (BOW)



S1: Without music life would be a mistake

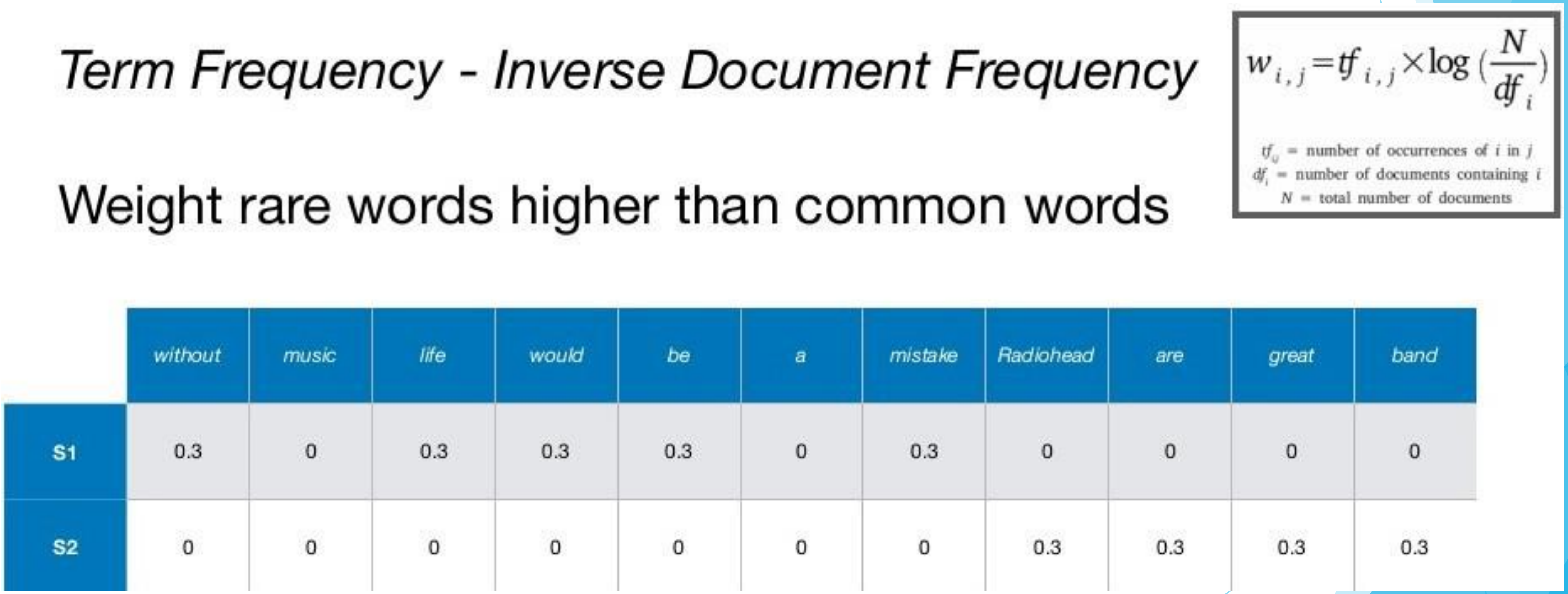
S2: Radiohead are a great music band

	without	*music*	*life*	*would*	*be*	*a*	*mistake*	*Radiohead*	*are*	*great*	*band*
S1	1	1	1	1	1	1	1	0	0	0	0
S2	0	1	0	0	0	1	0	1	1	1	1

Image Source

Vectorization

- Method 2: TF-IDF



Term Frequency - Inverse Document Frequency

Weight rare words higher than common words

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

tf_{ij} = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

	without	*music*	*life*	*would*	*be*	*a*	*mistake*	*Radiohead*	*are*	*great*	*band*
S1	0.3	0	0.3	0.3	0.3	0	0.3	0	0	0	0
S2	0	0	0	0	0	0	0	0.3	0.3	0.3	0.3

Image Source

More…

- Let’s see Dr. Liao’s NLTK code examples & tutorials for more details for text processing…

NLP with NLTK in Python Hands-On Programming in Class

- NLP with NLTK in Python Code Examples & Tutorials for Text Analysis / NLP
  - Dr. Liao wrote them **particularly** for
    - this course learning
    - **Assignments and final project** examples
- All programming tutorials & code example demos
  - Using online Jupyter Lab in class
- **More** NLTK code examples & tutorials in coming classes…