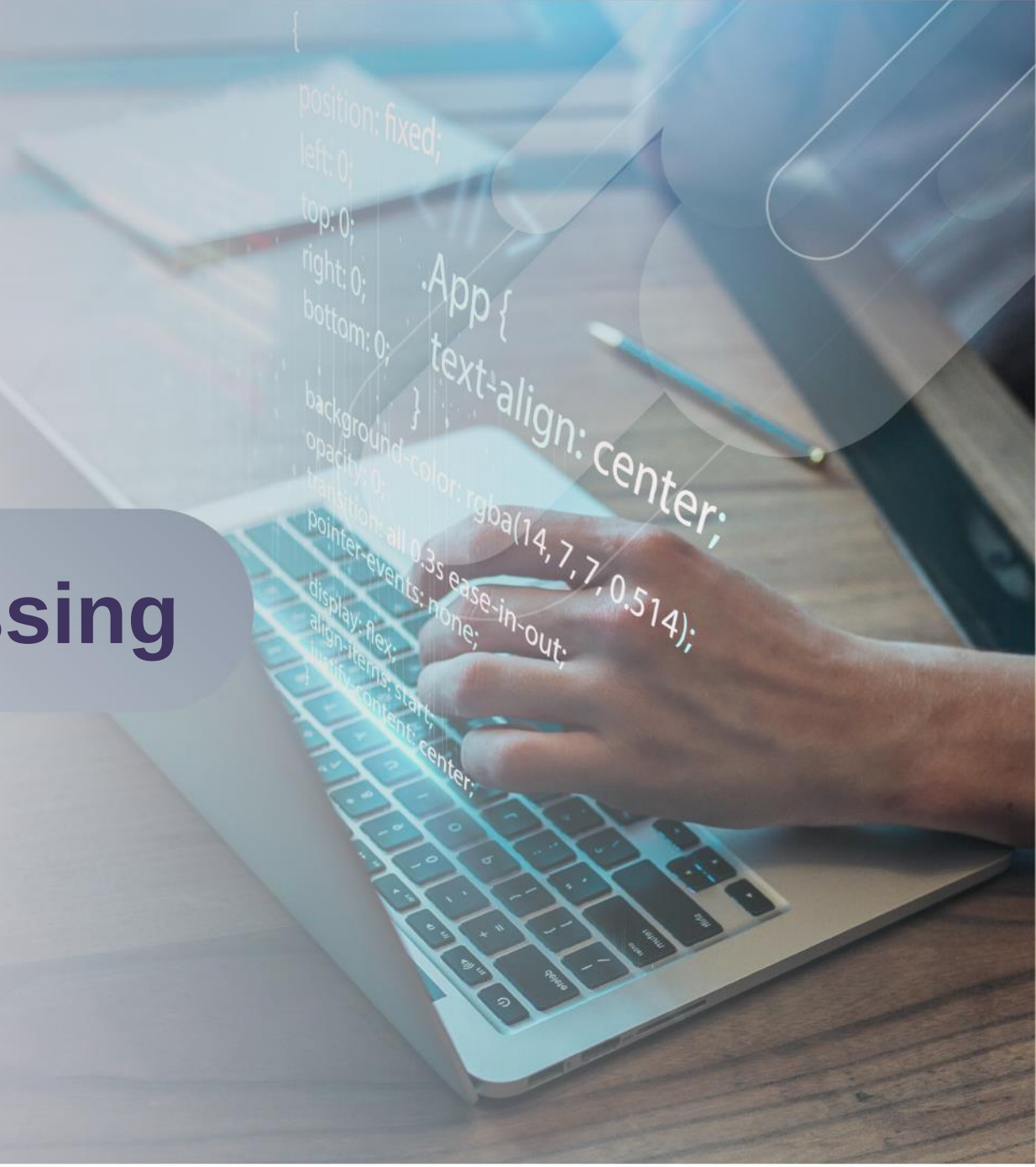
NANYANG TECHNOLOGICAL UNIVERSITY SINGAPORE

EE6405

Natural Language Processing



Dr. S. Supraja
NTU Electrical and Electronic Engineering

Content

Week	Topics	
1	Introduction to Natural Language Processing (NLP)	**Traditional Machine Learning**
2	Linguistic Feature Extraction	
3	Term Weighting Schemes and Topic Modelling	
4	Traditional Machine Learning Methods and NLP Applications	
5	Evaluation Metrics and Word Embeddings	**Deep Learning**
6	Neural Language Models	
7	Transformers	
8	Hyperparameter Tuning	**New NLP Trends**
9	Transformer-based Large Language Models	
10	Explore the Landscape: A Survey of NLP Applications Across Diverse Industries	
11	In-Depth Analysis: Deep-dive into NLP Applications	

Course Objectives

By the end of this course, you will be able to:

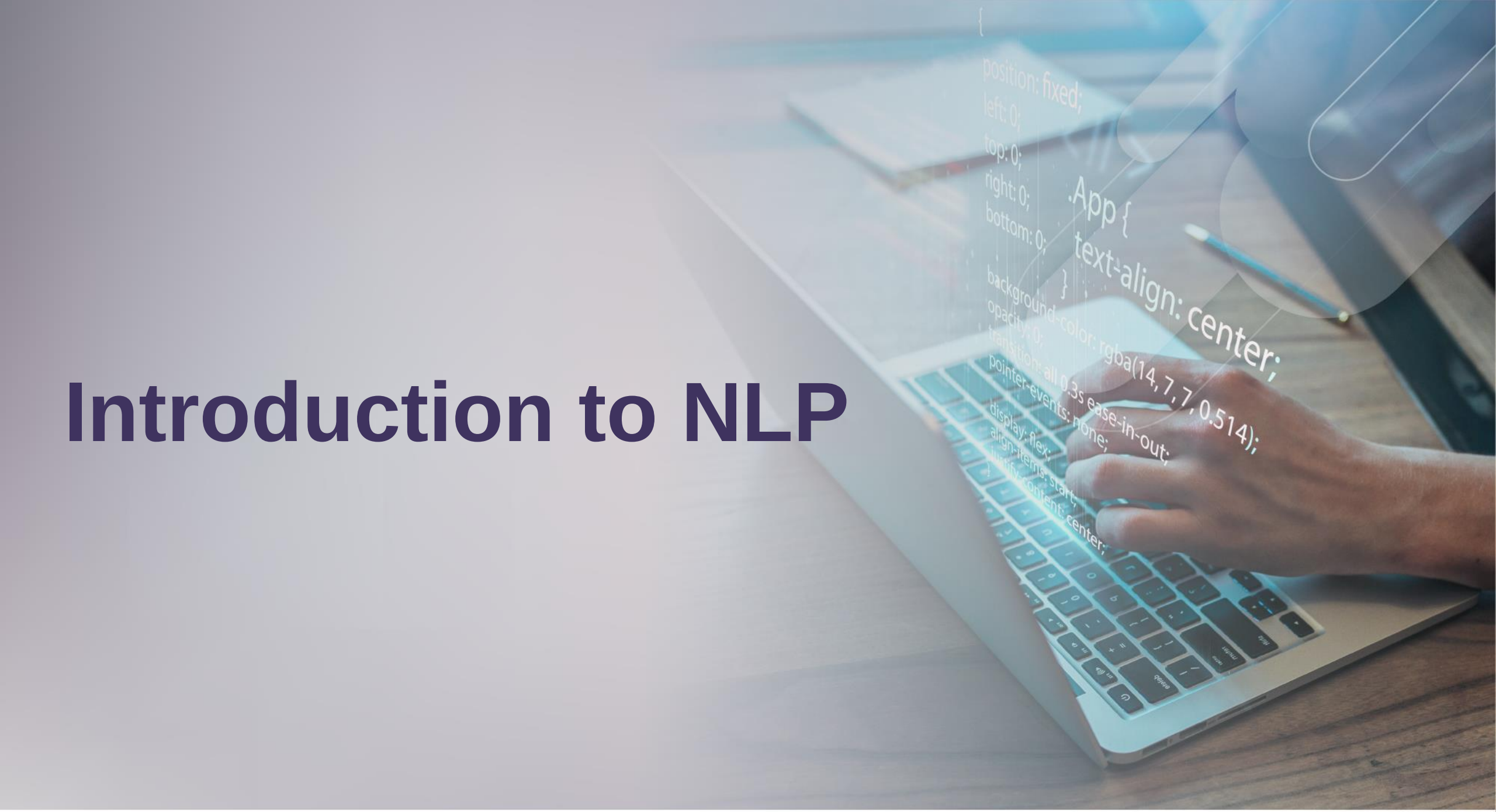
- Identify the appropriateness of NLP preprocessing techniques in different contexts.
- Explain the mathematical derivations of traditional NLP methods such as term weighting schemes, feature extraction techniques, and topic modeling.
- Execute simple NLP tasks such as classification and clustering on small-scale problems and evaluate the algorithm performance.
- Implement NLP algorithms in Python programming language.
- Distinguish between the theoretical concepts of traditional and deep neural network-based NLP techniques.

Course Objectives

By the end of this course, you will be able to:

- Formulate the construction of word embeddings and model training using deep neural network-based architecture such as RNN, Seq2Seq, Attention mechanism, and transformers.
- Describe the working principles of pre-trained language models.
- Design NLP-based projects as a team to solve a real-life application by employing any techniques learnt along with fine-tuning various models.

Introduction to NLP



What is NLP?

NLP represents a facet of artificial intelligence focussed on examining, comprehending, and producing human languages as they are naturally spoken and written.

Serves as a bridge between human language and computers. Some key purposes include:

Language Understanding

Allowing computers comprehend the meaning and context of human language.

Sentiment Analysis

Analysing the sentiment or emotion behind written text or spoken words.

Machine Translation

Facilitating the automatic translation of text from one language to another.

Language Generation

Enabling Computers to generate human-like text or speech.

NLP Around Us

A Brief History of NLP

1906 Ferdinand de Saussure proposes ‘Language as a Science’, deriving “meaning” from within a language’s relationships and contrasts.

1916 Cours de Linguistique Générale is published, introduces structuralist approach to languages.

1957 Noam Chomsky proposes that for a computer to understand a language, the sentence structure would have to be changed.

1980s Steady shift from rule-based text processing(symbolic) to statistical models(stochastic) capable of making soft, probabilistic decisions.

1997 RNNs introduced, later put into use for NLP in 2007.

2003 Yoshio Bengio proposes first neural language model with a feed-forward neural network - Birth of neural NLP.

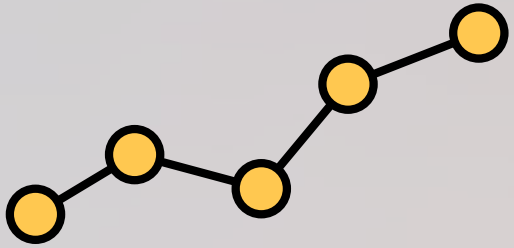
2011 Apple’s Siri becomes one of the first successful NLP/AI assistants.

Approaches to NLP

Symbolic

Rule based approach, based on lexica and semantics.

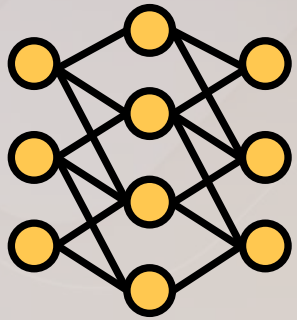
Stochastic



Probabilistic language models.

Examples include N-grams, Bayes Theorem, K-Means

Neural Model



Neural network approach to NLP.

Examples include BERT, LSTMs, GCNs etc.

Preprocessing Techniques

Why do we need them?

NOISE REDUCTION
Remove special characters, punctuation, and irrelevant information to clean the data.

TOKENIZATION
Break text into smaller units (words/subwords) for better analysis.

NORMALIZATION
Standardise words (lowercase, stemming, lemmatization) for reduced vocabulary size.

STOP WORD REMOVAL
Eliminate common words with little semantic value to reduce noise.

HANDLING OOV WORDS
Replace out-of-vocabulary words with unknown tokens or use subword tokenization.

SENTENCE SEGMENTATION
Separate text into sentences for individual analysis.

FEATURE ENGINEERING
Extract linguistic features (n-grams, POS tags) to enhance understanding.

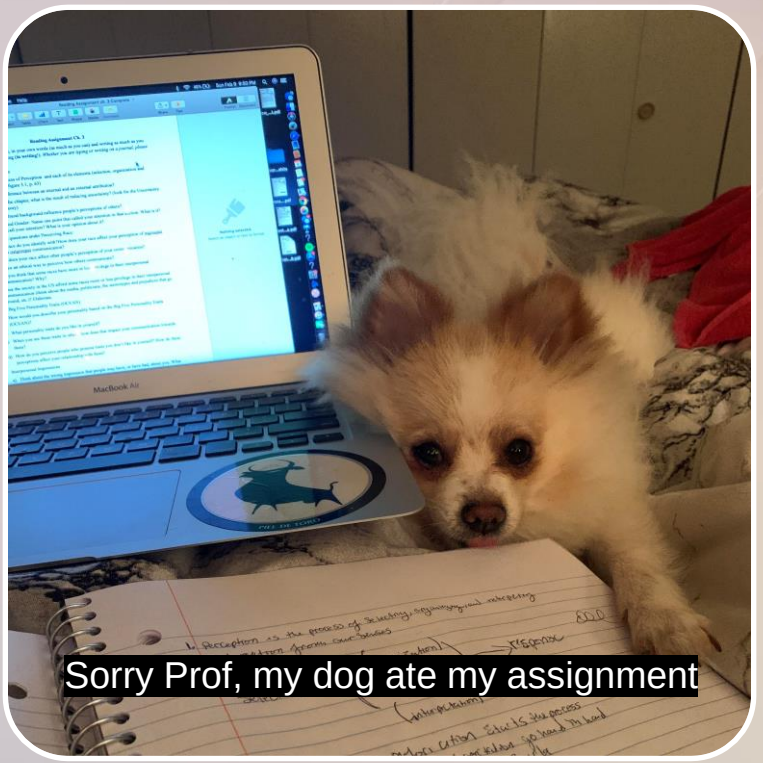
Preprocessing Techniques – RegEx

Regular Expressions (RegEx)

- A string of text that lets you create patterns that help match, locate, and manage text.

Example
Find the following strings within a block of text:

- pomeranian
- pomeranians
- Pomeranian
- Pomeranians



Preprocessing Techniques – RegEx

Metacharacters

Metacharacters are characters that serve special functions within a RegEx.

Special Sequences

A special sequence is a \ followed by a character that match a specific set of characters.

Sets

A set is a set of characters inside a pair of square brackets [ ] with a special meaning

Preprocessing Techniques – RegEx – Metacharacters

Character	Description	Examples
[ ]	A set of characters	“[a-m]”
\	Special Sequence/Escape	“\.”
.	Any character (except newline)	“c.t”
^	Starts With	“^hello”
$	Ends With	“planet$”
*	Zero or more occurences	“he.*o”
+	One or more occurences	“he.+o”
?	Zero or one occurences	“colou?r”
{}	Exactly the specified no. of occurences	“he.{2}o”
|	Either or	“start|stop”
()	Matches exact characters	“(hello)+”

Preprocessing Techniques – RegEx – Special Sequences

Character	Description	Examples
\A	Returns a match if the specified characters are at the beginning of the string	“\AThe”
\b	Returns a match where the specified characters are at the beginning or at the end of a word	“\bain” | “ain\b”
\B	Returns a match where the specified characters are present, but NOT at the beginning (or at the end) of a word	“\Bain” | “ain\B”
\d	Returns a match where the string contains digits (numbers from 0-9)	“\d”
\D	Returns a match where the string DOES NOT contain digits	“\D”
\s	Returns a match where the string contains a white space character	“\s”
\S	Returns a match where the string DOES NOT contain a white space character	“\S”
\w	Returns a match where the string contains any word characters (characters from a to Z, digits from 0-9, and the underscore _ character)	“\w”
\W	Returns a match where the string DOES NOT contain any word characters	“\W”
\Z	Returns a match if the specified characters are at the end of the string	“Spain\Z”

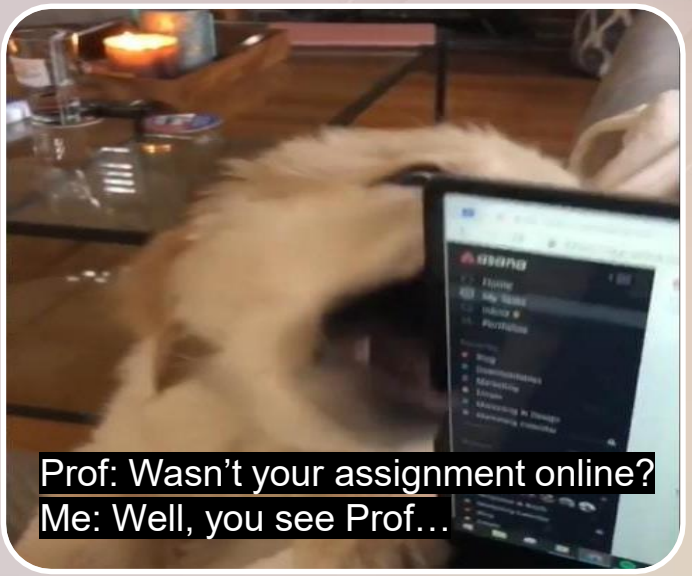
Preprocessing Techniques – RegEx – Sets

Character	Description
[ntu]	Returns a match where one of the specified characters (n, t, or u) is present
[a-d]	Returns a match for any lower-case character alphabetically between a and d
[^ntu]	Returns a match for any character EXCEPT n, t and u
[0123]	Returns a match where the specified digits (0, 1, 2 or 3) are present
[0-9]	Returns a match for digits between 0-9
[0-5][0-9]	Returns a match for any two-digit numbers from 00 and 59
[a-zA-Z]	Returns a match for any character alphabetically between a and z, lower case OR upper case
[+]	Returns a match for the char ‘+” (no special meaning here)

Preprocessing Techniques – RegEx

Some examples:

Pattern	Matches
[pP]omeranian	Pomeranian, pomeranian
pomeranians?	pomeranian, pomeranians
pomeranian|retriever	pomeranian OR retriever
pomeranian+	pomeranian, pomeraniannn
p.meranian	pemeranian, p0meranian
pomeranian$	pomeranian.



Preprocessing Techniques – RegEx

RegEx Functions

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

Example

```
import re
string='Harper is a good girl.'
re.sub('..g.*d','the goodest', string)
```

```
'Harper is the goodest girl.'
```

Preprocessing Techniques – NLTK

Natural Language Toolkit (NLTK)

- NLTK(Natural Language Toolkit) is the primary python API for NLP.
- Contains useful functions for preprocessing, which will be covered later.
- Examples include:

Tokenization	Stop word removal	Lemmatization/ Stemming	Tagging POS (parts of speech)

```
>pip install nltk
```

Preprocessing Techniques – Tokenization

- Tokenization refers to separating a piece of text into smaller ‘tokens’.
- Tokens can be words, characters or subwords(n-gram).
- The most common form of tokenization is to use space as a delimiter, resulting in individual words forming tokens.

```
from nltk.tokenize import word_tokenize
string='''
One ring to rule them all,
one ring to find them, one ring to bring them all,
and in the darkness, bind them.
'''
words = word_tokenize(string)
print(words)
```

```
['One', 'ring', 'to', 'rule', 'them', 'all', ',', 'one', 'ring', 'to', 'find', 'them', ',', 'one', 'ring', 'to', 'bring', 'them', 'all',
',', 'and', 'in', 'the', 'darkness', ',', 'bind', 'them', '.']
```

Preprocessing Techniques – Stemming

What is Stemming?

- It is the process of reducing inflected words to their ‘stem’. For example:

- Stemming removes the last few characters of a given word to obtain a shorter form. (Note, sometimes this form may not carry any meaning, such as the above case.)

Preprocessing Techniques – Stemming

Advantages

- **Improved model performance:**
  Stemming reduces the number of unique words that the algorithm needs to process.
- **Grouping of similar words:**
  Words with a similar meaning can be grouped together even if they possess different forms.
- **Reduces complexity of text:**
  Stemming reduces the size of the vocabulary, hence making texts easier to analyze and understand.

Disadvantages

- **Overstemming:**
  The algorithm may reduce unrelated words to the same word stem.
  E.g.: university, universal, universe == universi.
- **Understemming:**
  The algorithm does not reduce the word enough, resulting in synonyms bearing different stems.
  E.g.: alumnus, alumni, alumnae are not reduced to the same stem.
- **Language Challenges:**
  As the target language's morphology, spelling, and character encoding get more complicated, stemmers become more difficult to design.
  E.g. French, having a larger number of verb inflections, will require a more complicated stemmer than English.

Preprocessing Techniques – Stemming

```
import nltk
nltk.download('punkt')
from nltk.stem import PorterStemmer
from nltk.tokenize import word_tokenize
from functools import reduce
ps=PorterStemmer()
```

```
string='''
From the tip of his wand burst the silver doe.
She landed on the office floor, bounded once across the office, and soared out of the window.
Dumbledore watched her fly away, and as her silvery glow faded he turned back to Snape, and his eyes were fı
“After all this time?”
“Always,” said Snape.”
'''
words = word_tokenize(string)
stemmed_string = reduce(lambda x, y: x + " " + ps.stem(y), words, "")
print(stemmed_string)
```

```
 from the tip of hi wand burst the silver doe . she land on the offic floor , bound onc across the offic , and soar out of the window . du
mbledor watch her fli away , and as her silveri glow fade he turn back to snape , and hi eye were full of tear . “ after all thi time ? ”
“ alway , ” said snape . ”
```

Preprocessing Techniques – Stemming

Input

From the tip of his wand burst the silver doe.

She landed on the office floor, bounded once across the office, and soared out of the window.

Dumbledore watched her fly away, and as her silvery glow faded he turned back to Snape, and his eyes were full of tears.

“After all this time?”

“Always,” said Snape.”

Output

from the tip of hi wand burst the silver doe . she land on the offic floor , bound onc across the offic , and soar out of the window . dumbledor watch her fli away , and as her silveri glow fade he turn back to snape , and hi eye were full of tear . “ after all thi time ? ” “ alway , ” said snape . ”

Preprocessing Techniques – Lemmatization

What is Lemmatization?

- It is the process of reducing inflected words to their ‘lemma’(dictionary form). For example:

- As opposed to stemming, lemmatization aims to remove ONLY the inflectional ends of words, returning its dictionary form.

Preprocessing Techniques – Lemmatization

Advantages

- **Accuracy**
  As the word's Part-Of-Speech(POS) is taken into consideration, the word's context is considered when lemmatizing the word, resulting in a more accurate shortening of the word.

Disadvantages

- **Time-consuming**
  Lemmatization is slow and time-consuming when compared to stemming, as morphological analysis is conducted on each word to derive its meaning.

Preprocessing Techniques – Lemmatization

```
import nltk
nltk.download('wordnet')
from nltk.stem import WordNetLemmatizer
from nltk.tokenize import word_tokenize
nltk.download('averaged_perceptron_tagger')
lemmatizer= WordNetLemmatizer()
from nltk.corpus import wordnet
```

```
def pos_tagger(nltk_tag):
    if nltk_tag.startswith('J'):
        return wordnet.ADJ
    elif nltk_tag.startswith('V'):
        return wordnet.VERB
    elif nltk_tag.startswith('N'):
        return wordnet.NOUN
    elif nltk_tag.startswith('R'):
        return wordnet.ADV
    else:
        return None
```

Preprocessing Techniques – Lemmatization

```
string = '''
From the tip of his wand burst the silver doe.
She landed on the office floor, bounded once across the office, and soared out of the window.
Dumbledore watched her fly away, and as her silvery glow faded he turned back to Snape, and his eyes were f
“After all this time?”
“Always,” said Snape.”
'''
pos_tagged = nltk.pos_tag(nltk.word_tokenize(string))

wordnet_tagged = list(map(lambda x: (x[0], pos_tagger(x[1])), pos_tagged))
print(wordnet_tagged)

lemmatized_sentence = []

for word, tag in wordnet_tagged:
    if tag is None:
        lemmatized_sentence.append(word)
    else:
        lemmatized_sentence.append(lemmatizer.lemmatize(word, tag))
lemmatized_sentence = " ".join(lemmatized_sentence)

print(lemmatized_sentence)
```

Preprocessing Techniques – Lemmatization

Input

From the tip of his wand burst the silver doe.

She landed on the office floor, bounded once across the office, and soared out of the window.

Dumbledore watched her fly away, and as her silvery glow faded he turned back to Snape, and his eyes were full of tears.

“After all this time?”

“Always,” said Snape.”

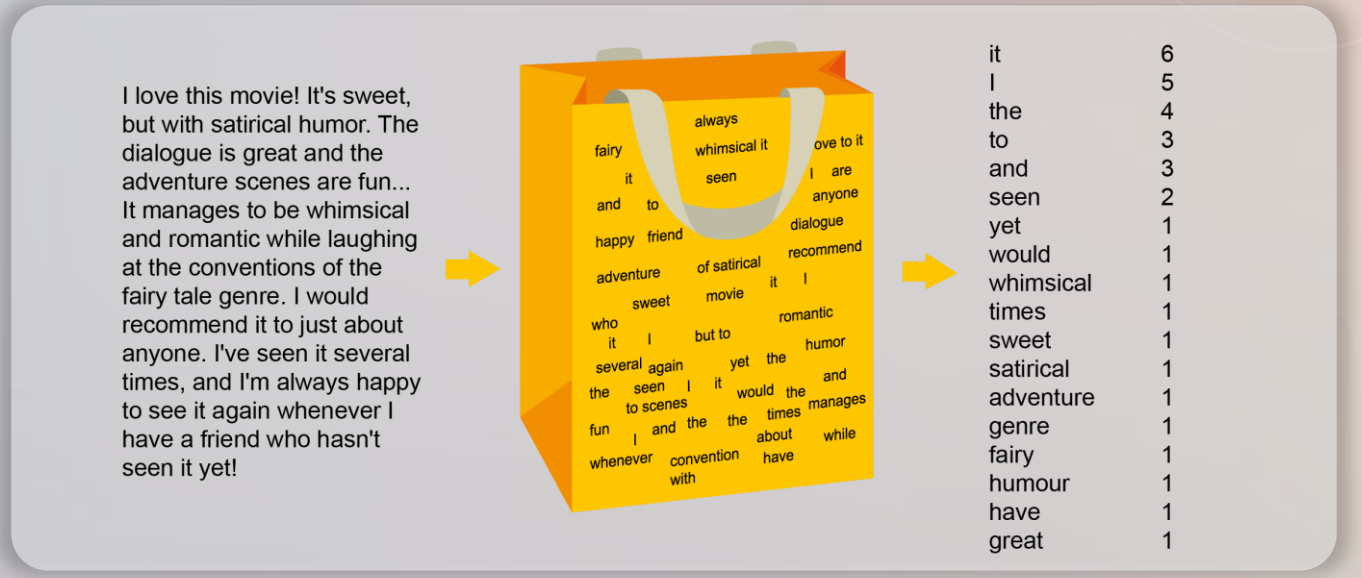
Output

From the tip of his wand burst the silver doe . She land on the office floor , bound once across the office , and soar out of the window . Dumbledore watch her fly away , and as her silvery glow fade he turn back to Snape , and his eye be full of tear . “ After all this time ? ” “ Always , ” say Snape . ”

Preprocessing Techniques – BOW

Bag-Of-Words (BOW)

- A representation of text that describes the occurrence of words within a document.
- Only word counts are retained, grammatical details and word order are discarded.
- Considered a ‘bag’ as all information about order and grammar are discarded.
- Converts unstructured text into structured data in the form of fixed length vectors.



Preprocessing Techniques – BOW

Worked Example

Sentence 1: “Never gonna give you up, never gonna let you down.”

Sentence 2: “Never gonna run around and desert you.”

Step 1

Tokenize the sentences, convert all letters to lowercase, remove punctuation and remove stop words.

We get:

Sentence 1: “never gon na give never gon na let”

Sentence 2: “never gon na run desert”

```
import nltk
import re
import numpy as np
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

text = """Never gonna run around and desert you."""

dataset = nltk.word_tokenize(text)
for i in range(len(dataset)):
    dataset[i] = dataset[i].lower()
    dataset[i] = re.sub(r'\W', ' ', dataset[i])
    dataset[i] = re.sub(r'\s+', ' ', dataset[i])
filtered_sentence = [w for w in dataset if not w.lower() in stop_words]
```

Preprocessing Techniques – BOW

Worked Example

Sentence 1: “never gon na give never gon na let”

Sentence 2: “never gon na run desert”

Step 2

Go through the all the words and make a list of available vocabulary.

- never
- gon
- na
- give
- let
- run
- desert

Preprocessing Techniques – BOW

Worked Example

Sentence 1: “never gon na give never gon na let”

Sentence 2: “never gon na run desert”

Step 3

Since we now have a 7 word long vocabulary, we can use a fixed-length document representation of 7, with one position in the vector to score each word.

Sentence 1

Word	Frequency
never	2
gon	2
na	2
give	1
let	1
run	0
desert	0

Sentence 2

Word	Frequency
never	1
gon	1
na	1
give	0
let	0
run	1
desert	1

Preprocessing Techniques – BOW

Worked Example

Sentence 1: “never gon na give never gon na let”

Sentence 2: “never gon na run desert”

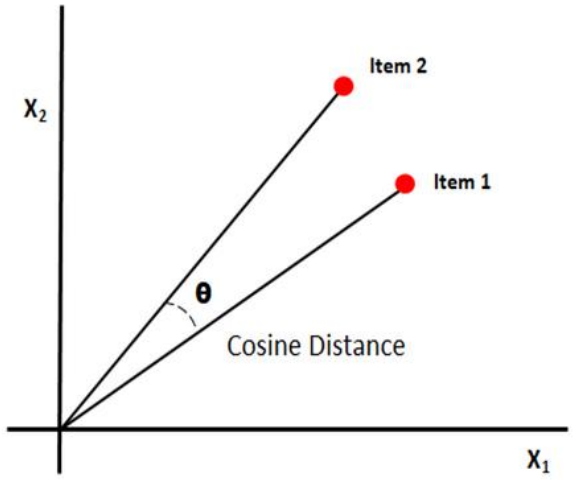
Putting the sentences in vector form, we get:

- Sentence 1: [2,2,2,1,1,0,0]
- Sentence 2: [1,1,1,0,0,1,1]

With these vectors, we can now make comparisons between the sentences.

E.g. One measure of similarity between the sentences would be the cosine similarity between these vectors

Cosine Distance/Similarity



$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}},$$

Preprocessing Techniques – N-grams

What is N-grams?

- A contiguous sequences of items that are collected from a sequence of text or speech corpus.
- The ‘n’ in n-grams specify the number of tokens considered.

Unigram: “I” , “am”, “the”, “one”, “who”, “knocks”

Bigram: “I am”, “the one”, “who knocks”

Trigram: “I am the”, “one who knocks”

... Up to N-grams

Preprocessing Techniques – N-grams

What is N-grams?

- N-grams are a probabilistic model that computes the probability of sentence or a given sequence of words.

  $P(W)=P(w_1, w_2, w_3, w_4,\ldots w_n)$ OR

  $P(w_5 \mid w_1, w_2, w_3, w_4)$

- To compute this probability, we use the chain rule of probability, namely:

$$P(B|A) = \frac{P(A,B)}{P(A)} \; or \; P(A,B) = P(A)P(B|A)$$

- With more variables, this becomes:

$$P(A,B,C,D)=P(A)P(B \mid A)P(C \mid A,B)P(D \mid A,B,C)$$

- As these sentences get longer, the computation for these probabilities becomes increasingly difficult.

Preprocessing Techniques – N-grams

What is N-grams?

- To simplify these probabilities, the Markov Assumption is utilised, namely:

$$P(w_n|w_1, w_2, w_3, \dots w_{n-1}) \approx P(w_n|w_{n-1}) - Unigram$$

$$P(w_n|w_1, w_2, w_3, \dots w_{n-1}) \approx P(w_n|w_{n-2}, w_{n-1}) - Bigram$$

$$P(w_n|w_1, w_2, w_3, \dots w_{n-1}) \approx P(w_n|w_{n-k} \dots, w_{n-1}) - k - gram$$

- Intuitively, this means the next word in a sentence can be predicted as a probability based on the *n* number of words that precedes it.

Preprocessing Techniques – N-grams

Worked Example

$$P(w_i|w_{i-1}) = \frac{c(w_{i-1}, w_i)}{c(w_{i-1})}$$

<s>This is a dog </s>

<s> This is a cat </s>

<s> I love my cat </s>

$$P(This|<s>) = \frac{2}{3} = 0.67 \quad P(I|<s>) = \frac{1}{3} = 0.33 \quad P(is|This) = \frac{2}{2} = 1 \quad P(a|is) = \frac{2}{2} = 1$$

$$P(dog|a) = \frac{1}{2} = 0.5 \qquad P(cat|a) = \frac{1}{2} = 0.5 \qquad P(</s>|cat) = \frac{2}{2} = 1$$

Summary

Key points discussed this week:

- **Preprocessing techniques including:**
  - **RegEx:**
    A sequence of characters used to match, search, and manipulate patterns in text strings.
  - **Stemming:**
    The process of reducing words to their base or root form to simplify text analysis and improve information retrieval in natural language processing.
  - **Lemmatization:**
    The process of converting words to their base or dictionary form (lemmas) to maintain the grammatical meaning and improve language understanding in natural language processing.

Summary

Key points discussed this week:

- **NLTK:**
  A Python library that provides tools and resources for processing and analysing human language data.
- **Basic language models including:**
  - **Bag of Words:**
    A simple and commonly used text representation model that converts documents into vectors, disregarding grammar and word order, by counting the frequency of each word.
  - **N-grams:**
    Contiguous sequences of *n* words in a text, used in language modeling and text analysis to capture local context and relationships between words.

References

- https://commons.wikimedia.org/wiki/File:Google_Translate_logo.svg
- https://upload.wikimedia.org/wikipedia/commons/thumb/0/04/ChatGPT_logo.svg/512px-ChatGPT_logo.svg.png
- https://media.sproutsocial.com/uploads/2023/07/Sentiment-analysis-HUB-Final.png
- https://m.media-amazon.com/images/I/41HeL2cy9LL.png
- https://www.reddit.com/r/Pomeranians/comments/f231oi/sorry_professor_tofu_ate_my_homework/?rdt=46489
- https://www.hindustantimes.com/it-s-viral/puppy-seems-to-give-the-dog-ate-my-homework-excuse-a-twist-by-trying-to-eat-a-laptop-watch/story-aNnCuoyxM4e9cMO0bnWSKN.html
- https://www.tyrrell4innovation.ca/author/jseal/page/2/