UNIVERSIDADE CATÓLICA PORTUGUESA

BRAGA

Behavior Analysis Technologies

Session 2

Text Preprocessing

Applied Data Science

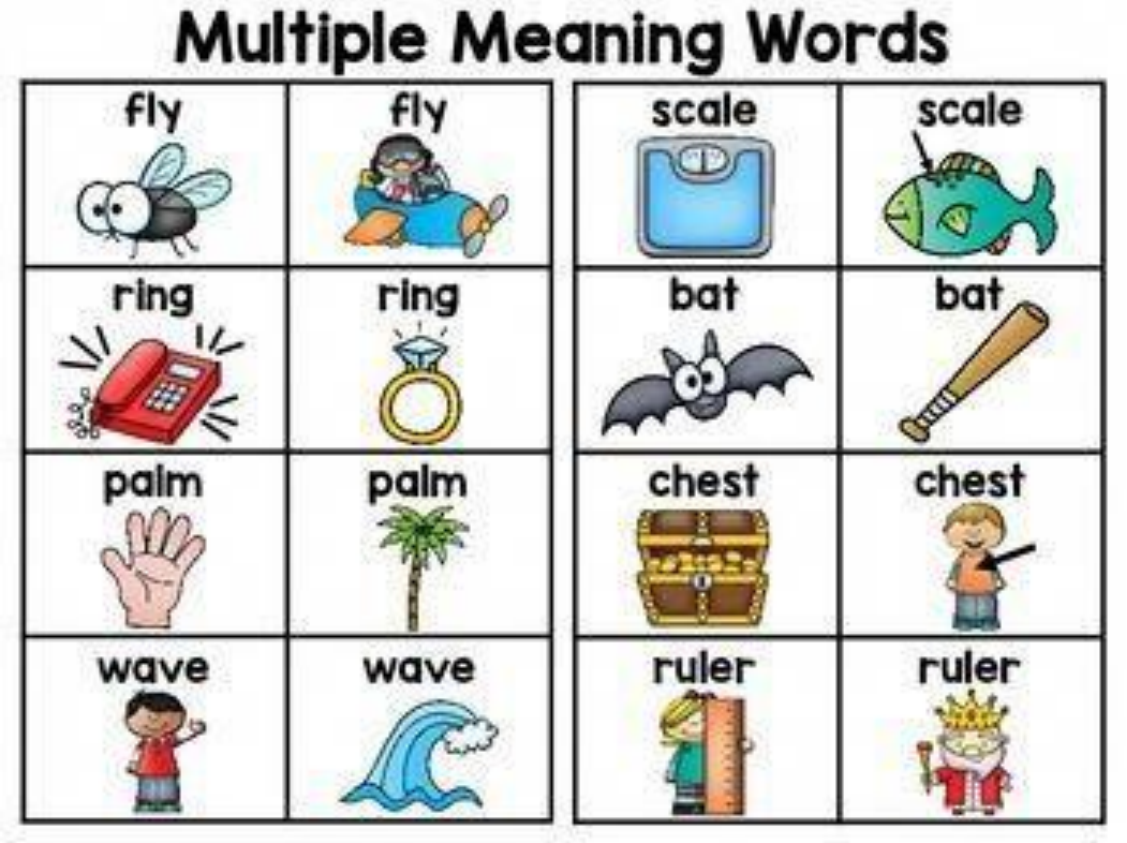
2024/2025

Question

What are the challenges in processing and analyzing natural language text?

Ambiguity in Language

- **Lexical Ambiguity:**
  - **Many words have multiple meanings.**



Ambiguity in Language

- **Syntatic Ambiguity:**
  - The **structure of sentences** can be ambiguous.

Ambiguity in Language

- **Semantic Ambiguity:**
  - It occurs when the **meaning of a sentence is unclear** because **the relationships between concepts or words are not well-defined**.

"Visiting relatives can be annoying."

- This could mean:
  - You find visiting your relatives annoying.
  - Your relatives, when they visit, are annoying.

Variability and Flexibility of Language

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- **Synonyms:** Different words can express the same concept (e.g., "car" vs. "automobile").

- **Polysemy:** A single word can have multiple related meanings (e.g., "mouse" could refer to an animal or a computer device).

- **Paraphrasing:** The same idea can be expressed in numerous ways (e.g., "He completed the project ahead of schedule." and "He finished the work earlier than planned.").

Context Dependence

- **Contextual Understanding:** The meaning of a word or phrase often depends on surrounding words (e.g., "bat" can be a flying mammal or a sports equipment). Systems must grasp local context to understand the true meaning.

- **Long-Distance Dependencies:** Words in a sentence may influence each other even if they are far apart, complicating syntactic and semantic analysis.

Informality and Noise in Text

- **Informal Language:** Aight, text data be hella messy with mad slang, abbrevs, and all that emoji biz. Grammar's all over the place, makin' it a total pain to figure out what's what. 😅 ⬜ 🧮

- **Typos and errors** human generated text is often noisy with typos grammatical mistakes or incomplete sentences handling this requires sophisticated error correction or tolerance mechanisms

Informality and Noise in Text

- **Informal Language:** Text data often includes informal language (e.g., social media posts, chats) with slang, abbreviations, emojis, and non-standard grammar, making it hard to analyze accurately.

- **Typos and Errors:** Human-generated text is often noisy, with typos, grammatical mistakes, or incomplete sentences. Handling this requires sophisticated error correction or tolerance mechanisms.

High Dimensionality

- **Sparse Representation:** Text data, especially in large corpora, is often represented as high-dimensional vectors. Most of these dimensions are sparse, which makes computations slow and can lead to overfitting.

Lack of Structure

- **Unstructured Nature of Text:** Unlike structured data (e.g., databases), text data doesn't follow a predefined schema, making it harder to index, search, and analyze. Extracting useful information from free-text requires sophisticated techniques to identify structure.

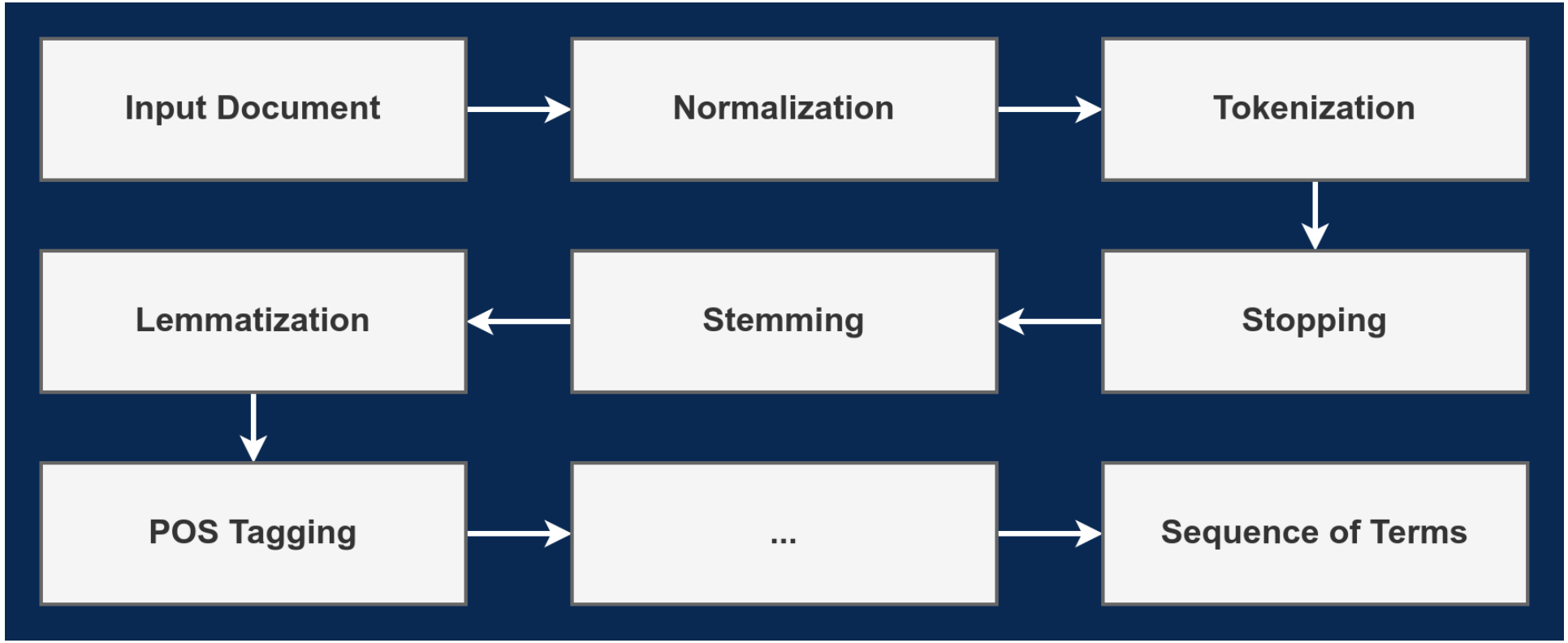
And Others

- **Subjectivity and Sentiment**
- **Language Evolution**
- **Multilingualism**
- **Scalability and Efficiency**

Question

How is raw text prepared for machine learning models?

The Text Preprocessing Pipeline



Text Normalization

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Normalization is the process of **standardizing text to reduce variation and improve consistency across the data** before analysis.

- It transforms text to a **common format**, making it easier to process in downstream tasks.

- Common tasks: "I'm fine, don't worry."
  - **Lowecasing:** "i'm fine, don't worry."
  - **Removing punctuation:** "im fine dont worry"
  - **Expanding contratctions:** "i am fine do not worry"

Exercise: Text Normalization

- Text normalization with Python:
  - Follow the instruction os the script *text_normalization.py*
  - Implement functions to:
    1. Lowercase text
    2. Remove punctuation from text
    3. Expand contractions

Tokenization

- Tokenization is the process of breaking down text into smaller, manageable units called tokens.
- Splitting is usually done along white spaces, punctuation marks, or other predefined rules.

- Types of tokens:
  - Word tokens: Breaking text into individual words.
    - "Word tokens" → ["Word", "tokens"]
  - Subword Tokens: Breaking text into smaller units than words.
    - "unhappy" → ["un", "happy"]
  - Character tokens:
    - "character" → ['c', 'h', 'a', 'r', 'a', 'c', 't', 'e', 'r']

Tokenization

- Tokenization sounds easy but can be surprisingly complex!
  - **Ambiguity**: Words with multiple meanings or contexts.
  - **Compound Words**: Identifying and splitting multi-word expressions (e.g., "New York").
  - **Special Characters**: Important part of tags, URLs, email addresses, etc (e.g., C++, C#).
  - **Numbers** can be important: nokia 3310, top 10 courses, Rua 25 de Abril.
  - **Periods** can occur in numbers, abbreviations, URLs, ends of sentences, and other situations: john.doe@example.com, weight: 68.5 kg, report.pdf.

Exercise: Tokenization

- Tokenization with Python:
  - Follow the instruction os the script *tokenization.py*
  - Implement a function that tokenizes text based on some condition.
    - Use the implemented function to split text based on:
      - Spaces
      - "#" character
  - Implement a function that tokenizes text based on punctuation.

Stopword Removal

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Function words that have little meaning apart from other words:
  - Articles: "the," "a," "an";
  - Demonstratives: "this," "that," "these," "those";
  - Propositions: "in," "on," "at," "by";
  - Conjunctions: "and," "but," "or," "so";
  - Pronouns: "he," "she," "it," "they";
  - Auxiliary verbs: "is," "are," "was," "were".

- These are considered **stopwords** and are generaly removed.

- In general, these constitute the most common words on documents.

Stopword Removal

a	did	herself	not	the	we've
about	didn't	him	of	their	were
above	do	himself	off	theirs	weren't
after	does	his	on	them	what
again	doesn't	how	once	themselves	what's
against	doing	how's	only	then	when
all	don't	i	or	there	when's
am	down	i'd	other	there's	where
an	during	i'll	ought	these	where's
and	each	i'm	our	they	which
any	few	i've	ours	they'd	while
are	for	if	ourselves	they'll	who
aren't	from	in	out	they're	who's
as	further	into	over	they've	whom
at	had	is	own	this	why
be	hadn't	isn't	same	those	why's
because	has	it	shan't	through	with
been	hasn't	it's	she	to	won't
before	have	its	she'd	too	would
being	haven't	itself	she'll	under	wouldn't
below	having	let's	she's	until	you
between	he	me	should	up	you'd
both	he'd	more	shouldn't	very	you'll
but	he'll	most	so	was	you're
by	he's	mustn't	some	wasn't	you've
can't	her	my	such	we	your
cannot	here	myself	than	we'd	yours
could	here's	no	that	we'll	yourself

When was the first computer invented?

How do I install a hard disk drive?

How do I use Adobe Photoshop?

Where can I learn more about computers?

How to download a video from YouTube

What is a special character?

How do I clear my Internet browser history?

How do you split the screen in Windows?

How do I remove the keys on a keyboard?

How do I install a hard disk drive?

Exercise: Stopword Removal

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Stopword Removal with Python:
  - Follow the instruction os the script *stopword_removal.py*
  - Implement a function that removes stopwords from a sequence of tokens, given a set of stopwords.

Stemming

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Process of **reducing words to their root** or base form.
  - **Inflectional forms**: Variations of a word that indicate grammatical changes such as tense, number, or case.
    - "run," "runs," "running" (all forms of the verb "run")
    - "cat," "cats" (singular and plural forms)
  - **Derivational forms**: Variations that change the meaning or part of speech of a word, often by adding prefixes or suffixes.
    - "happy," "happiness" (noun derived from the adjective "happy")
    - "connect," "connection" (noun derived from the verb "connect")
- In most cases, these have the same or very similar meanings.

Stemming

- Basic types of stemmers:
  - Algorithmic
  - Dictionary-based
  - Hybrid algorithmic-dictionary

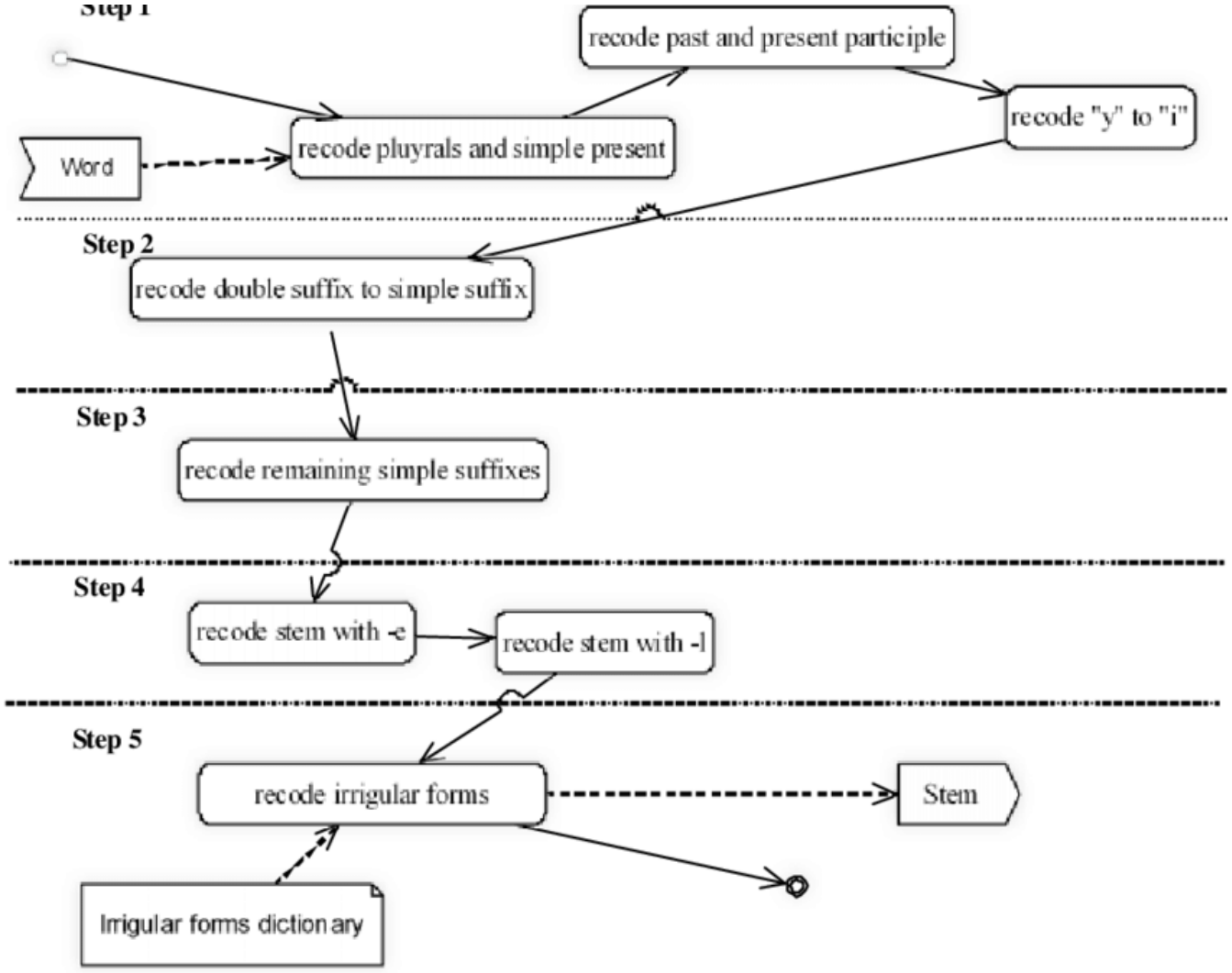
Suffix-s Stemmer

- Assumes that any word ending with an ‘s’ is plural
  - dogs → dog; cars → car
- Cannot detect many plural relationships (**false negatives**)
  - centuries → century
- In rare cases it detects a relationship where it does not exist (**false positives**)
  - bus → bu

Porter Stemmer

- Most popular algorithmic stemmer
- Consists of 5 steps, each step containing a set of rules for removing suffixes
- Produces stems not words
- Makes a number of errors and difficult to modify

Porter Stemmer



Karaa, W. B. A. (2013). A New Stemmer to Improve Information Retrieval. In International Journal of Network Security & Its Applications (Vol. 5, Issue 4, pp. 143–154). Academy and Industry Research Collaboration Center (AIRCC). https://doi.org/10.5121/ijnsa.2013.5411

Krovetz Stemmer

- Hybrid algorithmic-dictionary
- Word checked in dictionary
  - If present, either left alone or replaced with exception stems
  - If not present, word is checked for suffixes that could be removed
- After removal, dictionary is checked again
- Produces words not stems

Stemmer Comparison

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Original text

Document will describe marketing strategies carried out by U.S. companies for their agricultural chemicals, report predictions for market share of such chemicals, or report market statistics for agrochemicals, pesticide, herbicide, fungicide, insecticide, fertilizer, predicted sales, market share, stimulate demand, price cut, volume of sales

Porter stemmer

market strateg carr compan agricultur chemic report predict market share chemic report market statist agrochem pesticid herbicid fungicid insecticid fertil predict sale stimul demand price cut volum sale

Krovetz stemmer

marketing strategy carry company agriculture chemical report prediction market share chemical report market statistic agrochemic pesticide herbicide fungicide insecticide fertilizer predict sale stimulate demand price cut volume sale

Exercise: Stemming

- Stemming with Python:
  - Follow the instruction os the script *stemming.py*
  - Implement the Suffix-s Stemmer.

Lemmatization

- Lemmatization is the process of **reducing a word to its base or dictionary form (lemma)**.

- Unlike stemming, which strips prefixes or suffixes, lemmatization **uses a dictionary to map a word to its canonical form**.

- Examples:
  - "running"
    - Lemma: "run" (as a verb)
  - "better"
    - Lemma: "good" (when used as na adjective)

Exercise: Lemmatization

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Lemmatization with Python:
  - Follow the instruction os the script *lemmatization.py*
  - Implement a function that performns lemmatization based on a dictionary.

POS Tagging

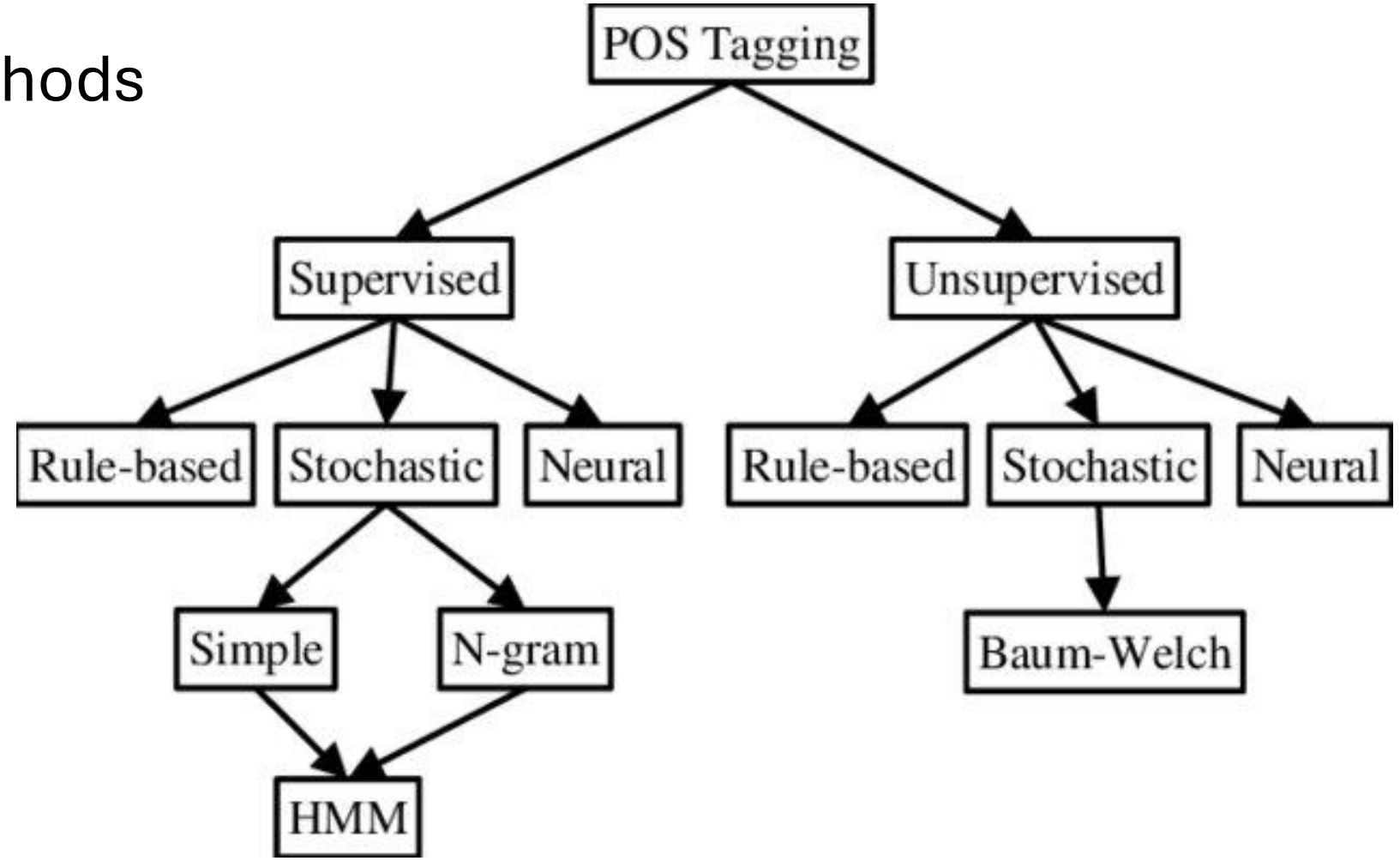
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Part-of-Speech (POS) Tagging is the process of **labeling each word** in a sentence **with its part of speech** (noun, verb, adjective, etc.) based on both its **definition and context** within the sentence.

- How POS Tagging Works?
  - **Input:** A sentence or text is provided for tagging.
  - **Tokenization:** The sentence is split into individual words or tokens.
  - **Tagging Algorithm**: Each token is tagged with its corresponding part of speech based on context.

POS Tagging

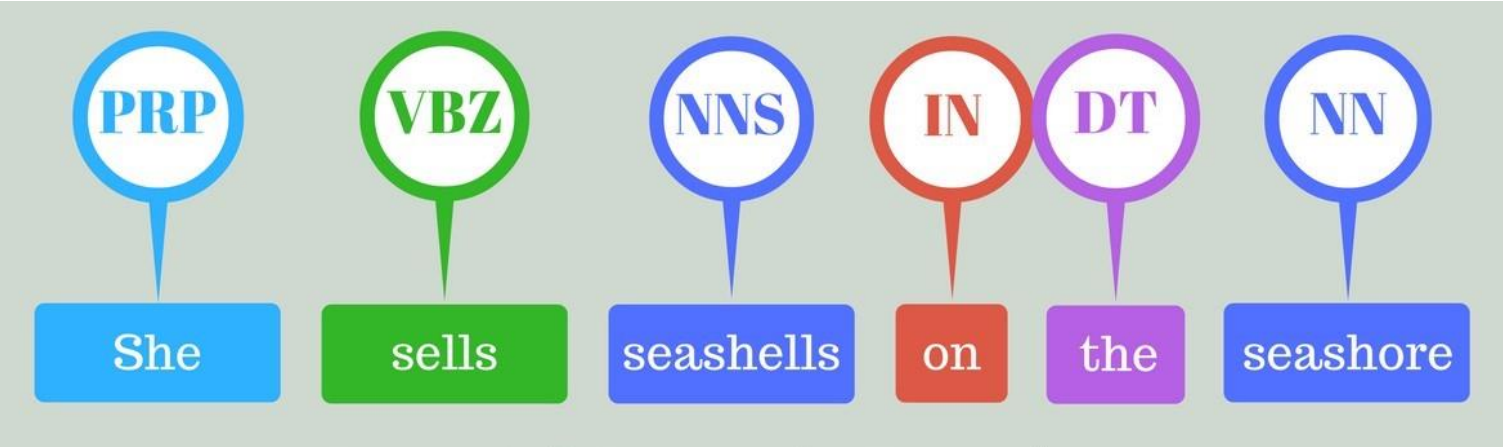
- Tagging Methods



https://www.researchgate.net/publication/331684946_PART_OF_SPEECH_TAGGER_FOR_ARABIC_TEXT_BASED_SUPPORT_VECTOR_MACHINES_A_REVIEW

POS Tagging

CC	Coordinating conjunction	NNS	Noun, plural	UH	Interjection
CD	Cardinal number	NNP	Proper noun, singular	VB	Verb, base form
DT	Determiner	NNPS	Proper noun, plural	VBD	Verb, past tense
EX	Existential there	PDT	Predeterminer	VBG	Verb, gerund or present participle
FW	Foreign word	POS	Possessive ending	VBN	Verb, past participle
IN	Preposition or subordinating conjunction	PRP	Personal pronoun	VBP	Verb, non-3rd person singular present
JJ	Adjective	PRP$	Possessive pronoun	VBZ	Verb, 3rd person singular present
JJR	Adjective, comparative	RB	Adverb	WDT	Wh-determiner
JJS	Adjective, superlative	RBR	Adverb, comparative	WP	Wh-pronoun
LS	List item marker	RBS	Adverb, superlative	WP$	Possessive wh-pronoun
MD	Modal	RP	Particle	WRB	Wh-adverb
NN	Noun, singular or mass	SYM	Symbol		
		TO	to		



Exercise: POS Tagging

- POS Tagging with Python:
  - Follow the instruction os the script *pos_tagging.py*
  - Implement a function that performns POS Tagging based on a dictionary.

The nltk package

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Tokenization:

```
from nltk.tokenize import sent_tokenize, word_tokenize

text = "Natural language processing (NLP) is a field of computer science, artificial intelligence and computational linguistics concerned with the interactions between
 computers and human (natural) languages, and, in particular, concerned with programming computers to fruitfully process large natural language corpora. Challenges in
 natural language processing frequently involve natural language understanding, natural language generation (frequently from formal, machine-readable logical forms),
 connecting language and machine perception, managing human-computer dialog systems, or some combination thereof."

print(sent_tokenize(text))
print(word_tokenize(text))
```

- And many more:
  - https://www.nltk.org/api/nltk.tokenize.html

The nltk package

- Stop Words:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

nltk.download('stopwords')
nltk.download('punkt_tab')
print(stopwords.words('english'))

example_sent = """This is a sample sentence,
                showing off the stop words filtration."""

stop_words = set(stopwords.words('english'))

word_tokens = word_tokenize(example_sent)
# converts the words in word_tokens to lower case and then checks whether
#they are present in stop_words or not
filtered_sentence = [w for w in word_tokens if not w.lower() in stop_words]
#with no lower case conversion
filtered_sentence = []

for w in word_tokens:
    if w not in stop_words:
        filtered_sentence.append(w)

print(word_tokens)
print(filtered_sentence)
```

Many other languages including portuguese!

The nltk package

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Stemming:

```
from nltk.stem import PorterStemmer

ps = PorterStemmer()

# choose some words to be stemmed
words = ["program", "programs", "programmer", "programming", "programmers"]

for w in words:
    print(w, " : ", ps.stem(w))
```

- And many more:
  - https://www.nltk.org/api/nltk.stem.html

The nltk package

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Lemmatization:

```
import nltk
from nltk.stem import WordNetLemmatizer

nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()

print("rocks :", lemmatizer.lemmatize("rocks"))
print("corpora :", lemmatizer.lemmatize("corpora"))

# a denotes adjective in "pos"
print("better :", lemmatizer.lemmatize("better", pos="a"))
```

The nltk package

- POS Tagging:

- https://www.nltk.org/api/nltk.tag.html

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import sent_tokenize

nltk.download('averaged_perceptron_tagger_eng')
stop_words = set(stopwords.words('english'))

# Dummy text
txt = "Sukanya, Rajib and Naba are my good friends. " \
    "Sukanya is getting married next year. " \
    "Marriage is a big step in one's life." \
    "It is both exciting and frightening. " \
    "But friendship is a sacred bond between people." \
    "It is a special kind of love between us. " \
    "Many of you must have tried searching for a friend "\
    "but never found the right one."

# sent_tokenize is one of instances of
# PunktSentenceTokenizer from the nltk.tokenize.punkt module

tokenized = sent_tokenize(txt)
for i in tokenized:

    # Word tokenizers is used to find the words
    # and punctuation in a string
    wordsList = nltk.word_tokenize(i)

    # removing stop words from wordList
    wordsList = [w for w in wordsList if not w in stop_words]

    #  Using a Tagger. Which is part-of-speech
    # tagger or POS-tagger.
    tagged = nltk.pos_tag(wordsList)

    print(tagged)
```

Exercise: nltk package

- **Take-home assignment:**
  - Test the previous nltk scripts
  - Explore the package!