Welcome to the CoGrammar Natural Language Processing I

The session will start shortly...

Questions? Drop them in the chat. We'll have dedicated moderators answering questions.

CoGrammar

Data Science Session Housekeeping

- The use of disrespectful language is prohibited in the questions, this is a supportive, learning environment for all - please engage accordingly. **(Fundamental British Values: Mutual Respect and Tolerance)**
- No question is daft or silly - **ask them!**
- There are **Q&A sessions** midway and at the end of the session, should you wish to ask any follow-up questions. Moderators are going to be answering questions as the session progresses as well.
- If you have any questions outside of this lecture, or that are not answered during this lecture, please do submit these for upcoming Academic Sessions. You can submit these questions here: **Questions**

Data Science Session Housekeeping cont.

- For all **non-academic questions**, please submit a query: **www.hyperiondev.com/support**
- Report a **safeguarding** incident: **www.hyperiondev.com/safeguardreporting**
- We would love your **feedback** on lectures: **Feedback on Lectures**

Skills Bootcamp
8-Week Progression Overview

Fulfil 4 Criteria to Graduation

✅ Criterion 1: Initial Requirements

Timeframe: First 2 Weeks
Guided Learning Hours (GLH): Minimum of 15 hours
Task Completion: First four tasks

Due Date: 24 March 2024

✅ Criterion 2: Mid-Course Progress

60 Guided Learning Hours

Data Science - **13 tasks**
Software Engineering - **13 tasks**
Web Development - **13 tasks**

Due Date: 28 April 2024

CoGrammar

Skills Bootcamp Progression Overview

✅ Criterion 3: Course Progress

Completion: All mandatory tasks, including Build Your Brand and resubmissions by study period end
Interview Invitation: Within 4 weeks post-course
Guided Learning Hours: Minimum of 112 hours by support end date (10.5 hours average, each week)

✅ Criterion 4: Demonstrating Employability

Final Job or Apprenticeship Outcome: Document within 12 weeks post-graduation
Relevance: Progression to employment or related opportunity

CoGrammar

SKILLS FOR LIFE

SKILLS BOOTCAMPS

Department for Education

CoGrammar

Natural Language Processing I

May 2024

Learning Objectives

- Understand the basics of **natural language processing (NLP)** and the challenges of working with natural language data.
- Understand how to work with **text** in Python.
- Utilise **regular expressions** for pattern searching in text.
- Set up **SpaCy** as a tool for NLP pipeline.

Learning Objectives

- Understand the fundamental **text preprocessing** concepts of **stemming**, **lemmatisation**, **stop words**, and **tokenisation**.
- Understand advanced text processing concepts of **parts-of-speech (POS) tagging** and **named entity recognition (NER)**.

Natural Language Processing

Introduction

CoGrammar

Natural Language Processing

- **Natural Language Processing (NLP):** pivotal multidisciplinary technology in artificial intelligence to **enable computers to understand, process, and create human language** (textual, speech, and audio data).

- NLP allows humans and machines to interact seamlessly with **unstructured data.**

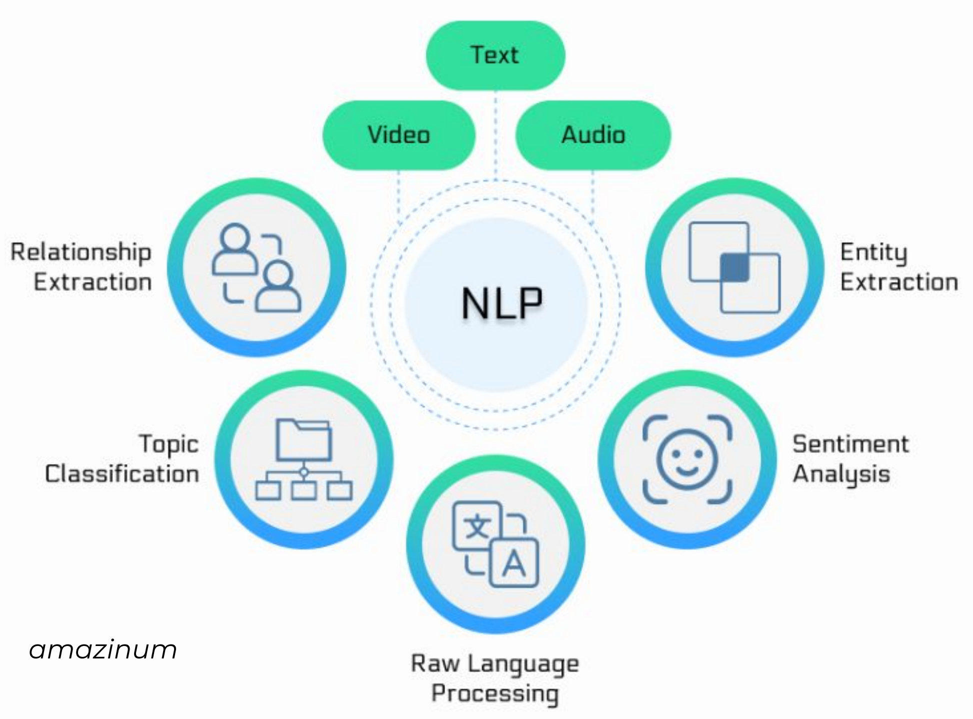
- **Applications**: automated customer support, virtual assistants, email filtering, machine translation, sentiment analysis, speech recognition, chatbots, text classification, real-time translation of languages.

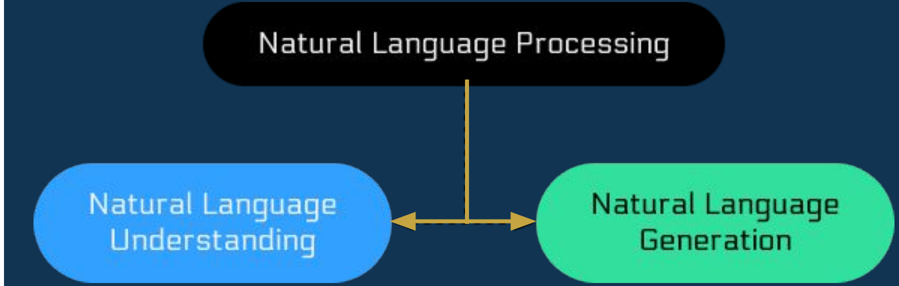
Structured vs Unstructured Data

Structured Data	Unstructured Data
Fits neatly into a database, defined and organised format, relational integrity	Lacks a predefined structure or taxonomy
Organised, easy to retrieve and analyse, enables quick decision making	Varied nature and lack of organisation makes it complex, more challenging to categorise or analyse
Goes into data warehouse	More complex storage (data lakes)
Uses basic tools like spreadsheets, SQL	Requires advanced tools **NLP**, ML
Ex: Financial records, customer information, inventory databases	Ex: Social media feeds, emails, audio, videos, customer reviews, sensor data

CoGrammar

NLP Areas and Components




Natural Language Processing
Natural Language Understanding
Natural Language Generation

NLU

Process of reading and interpreting language.

Produces non-linguistic outputs from natural language inputs.

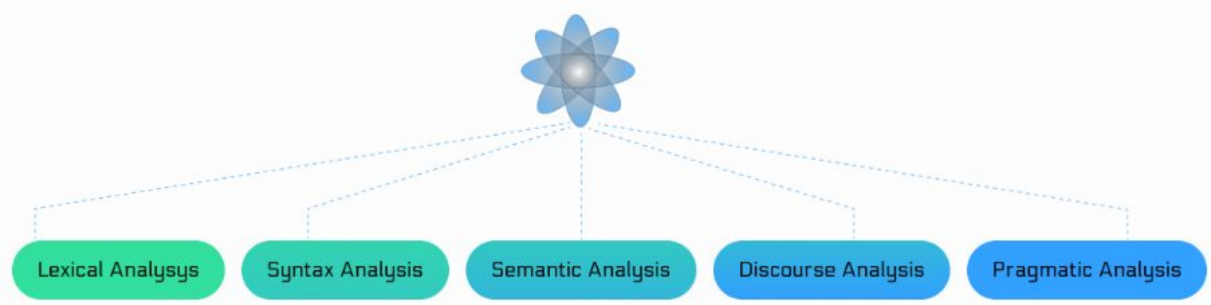
NLG

Process of writing or generating language.

Constructs natural language outputs from non-linguistic inputs.

NLP Levels

❖ **Morphological/Lexical analysis:** processing and understanding parts of speech. E.g., the word ‘character’ can be used as a noun or a verb.

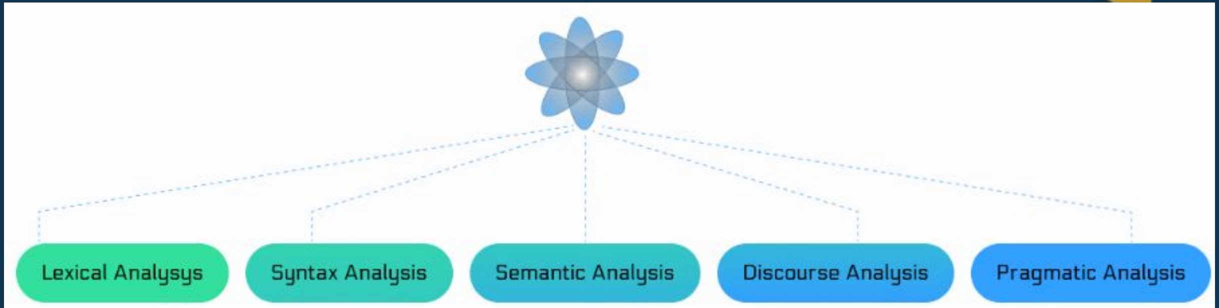


❖ **Syntax analysis:** understanding the sentence structure.
Correct Syntax: Sun rises in the east.
Incorrect Syntax: Rise in sun the east.

❖ **Semantic analysis:** understanding the literal meaning of the words, phrases, and sentences. E.g., ‘Hot ice-cream’ or ‘The apple ate the banana’ will be rejected by semantic analyser. ‘Red apple’ gives information about one object, hence treated as a single phrase.

NLP Levels

- **Discourse analysis:** understanding units larger than a single sentence utterance. E.g., “Julie is a bright student. She spends most of the time in the library.” Here, discourse assigns “she” to refer to “Julie”.



Lexical Analysys

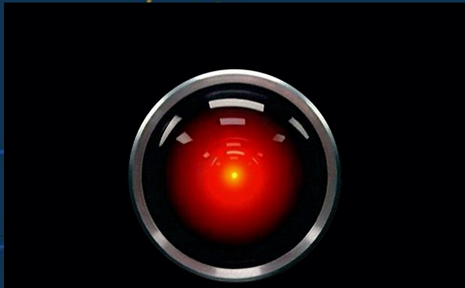
Syntax Analysis

Semantic Analysis

Discourse Analysis

Pragmatic Analysis

- **Pragmatic analysis:** using real-world knowledge to understand the bigger context of the sentence.
  Dave: Hal, switch to manual hibernation control.
  Hal 9000: I can tell from the tone of your voice, Dave, that you’re upset. Why don’t you take a stress pill and get rest.



2001: A Space Odyssey

Challenges with text handling

- Understand the **underlying intent** of the conversation **challenging** for a machine.

- Contextual words, phrases, and homonyms:
  - *I **ran** to the store because we **ran** out of milk.*

- Synonyms: *small, little, tiny, minute.*

- Errors in text and speech, detecting irony and sarcasm, colloquialisms and slang

- Low-resource languages, need for multilingual resources

CoGrammar

Challenges with text handling

- ❖ Ambiguity

  - ➢ **Lexical ambiguity:** a word that could be used as a verb, noun, or adjective.
  - ➢ **Semantic ambiguity**
    1. *Most of the time travellers worry about their luggage.*

    Without punctuation, hard to infer whether “time travellers” worry about their luggage or just “travellers.”

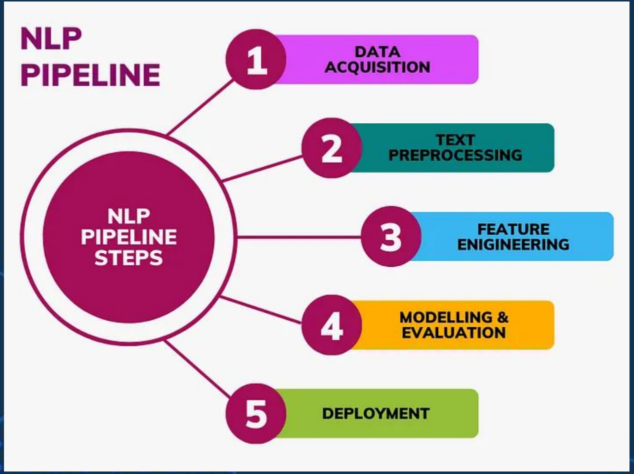
    2. *I saw the boy on the beach with binoculars.*

    Is this, I saw a boy through my binoculars or the boy had binoculars with him?
  - ➢ **Syntactic ambiguity:** The phrase with my binoculars could modify the verb, “saw,” or the noun, “boy.”

CoGrammar

NLP Pipeline

- **Data Acquisition** - obtaining raw textual data e.g. **built-in or public datasets**, collect (scraping) data from websites (not in the lecture)



- **Text Preprocessing** - refine raw text data for meaningful analysis
  - **Basic Cleaning:** eliminate irrelevant elements unnecessary for linguistic analysis (e.g., stripping out HTML tags, handling emojis, spell checks.
  - **Basic Preprocessing:** tokenisation, stemming/lemmatisation, stop-word removal.
  - **Advanced Preprocessing:** POS tagging, NER

CoGrammar

NLP Pipeline

- **Feature Engineering:** transforming **raw text data into numerical features** that machine learning models can comprehend and utilize effectively, e.g. bag-of-words, TF-IDF, word embeddings.

- **Modelling:** models are applied and evaluated using different approaches.

- **Evaluation:** comprehensively gauge model performance

- **Deployment:** transition of the developed model from the development environment to a production environment

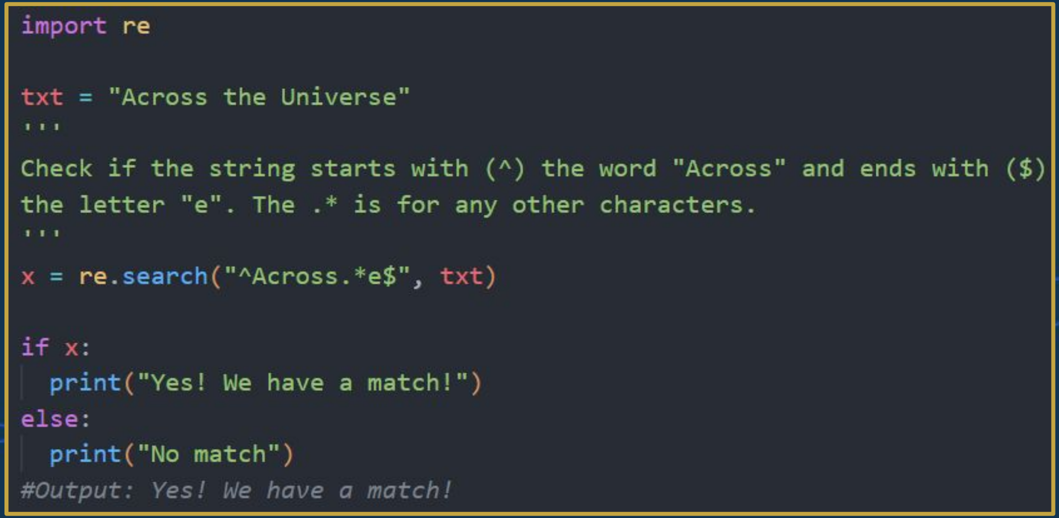
CoGrammar

Text Cleaning

Regular Expressions

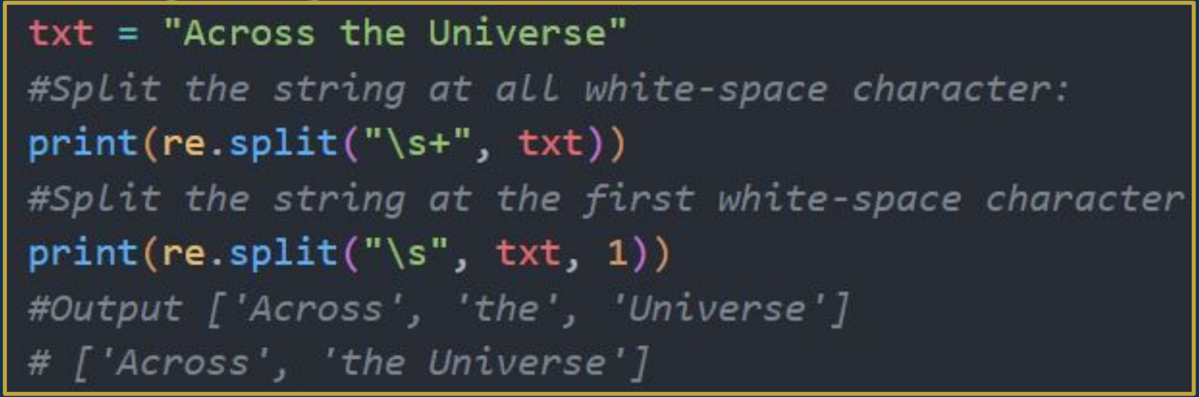
Regular Expressions

- **Regular expressions** or **RegEx** is a sequence of characters mainly used to find or replace patterns embedded in the text.
- Strings with a **special syntax.**
- Allow to **match patterns** in other strings.
- **Applications**: Find all weblinks in a document, parse email addresses, remove/replace unwanted characters.



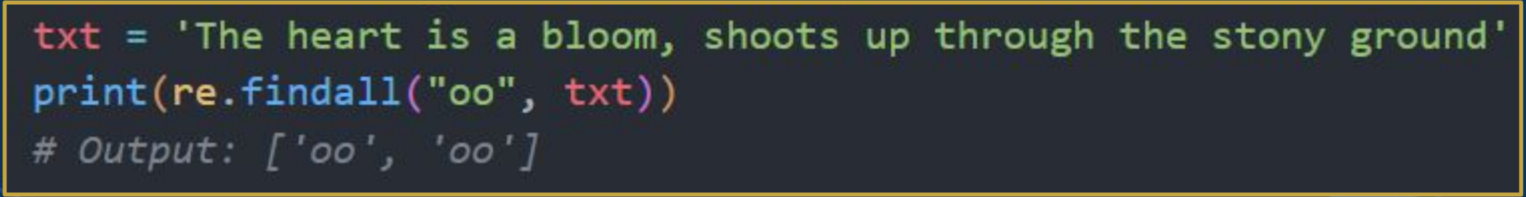
```python
import re

txt = "Across the Universe"
'''
Check if the string starts with (^) the word "Across" and ends with ($)
the letter "e". The .* is for any other characters.
'''
x = re.search("^Across.*e$", txt)

if x:
  print("Yes! We have a match!")
else:
  print("No match")
#Output: Yes! We have a match!
```

Regular Expressions

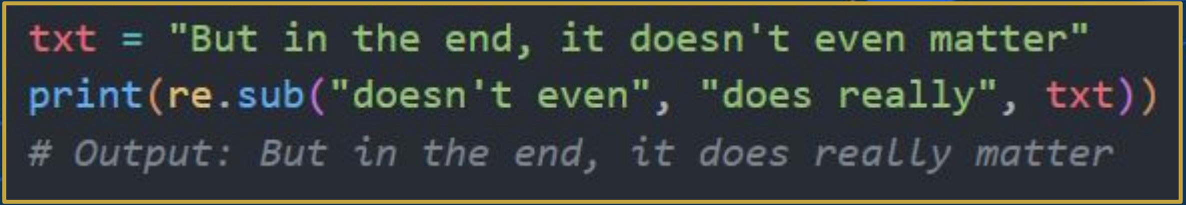


```
txt = "Across the Universe"
#Split the string at all white-space character:
print(re.split("\s+", txt))
#Split the string at the first white-space character
print(re.split("\s", txt, 1))
#Output ['Across', 'the', 'Universe']
# ['Across', 'the Universe']
```

Please see cheat sheet for more options



```
txt = 'The heart is a bloom, shoots up through the stony ground'
print(re.findall("oo", txt))
# Output: ['oo', 'oo']
```



```
txt = "But in the end, it doesn't even matter"
print(re.sub("doesn't even", "does really", txt))
# Output: But in the end, it does really matter
```

CoGrammar

NLP tools

spaCy

CoGrammar

spaCy

- **spaCy**, written in Python and Cython, is an open-source software library for NLP.

- **Fast and intuitive**, top contender for **beginners NLP tasks.**

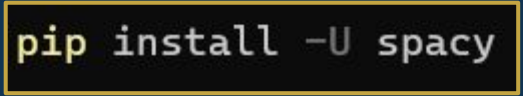
- Specifically designed to be an useful library for implementing **production-ready systems.**

- In contrast, **natural language toolkit (NLTK)** is more comprehensive than spaCy, allows in-depth customization and implementation of specific algorithms for advanced **research projects.**

CoGrammar

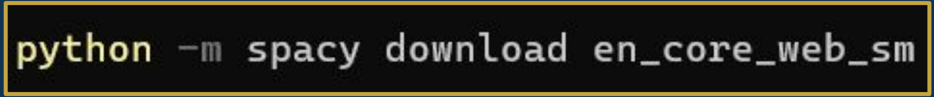
spaCy

- ❖ spaCy can be installed using pip



```
pip install -U spacy
```

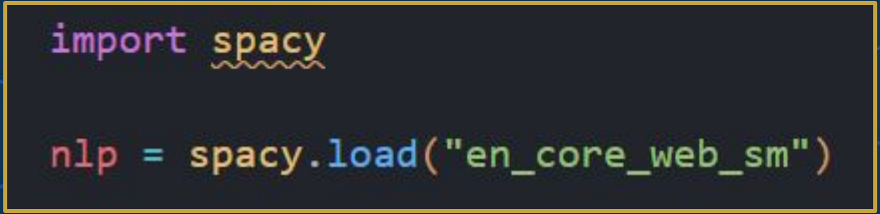
- ❖ If a **trained pipeline** is available for a language, it can be download using the spacy download command.
  Here the spaCy's trained pipelines for **English language** can be installed as Python packages



```
python -m spacy download en_core_web_sm
```

[spaCy models and languages](#)

- ❖ Once downloaded, the model can be imported as



```
import spacy

nlp = spacy.load("en_core_web_sm")
```

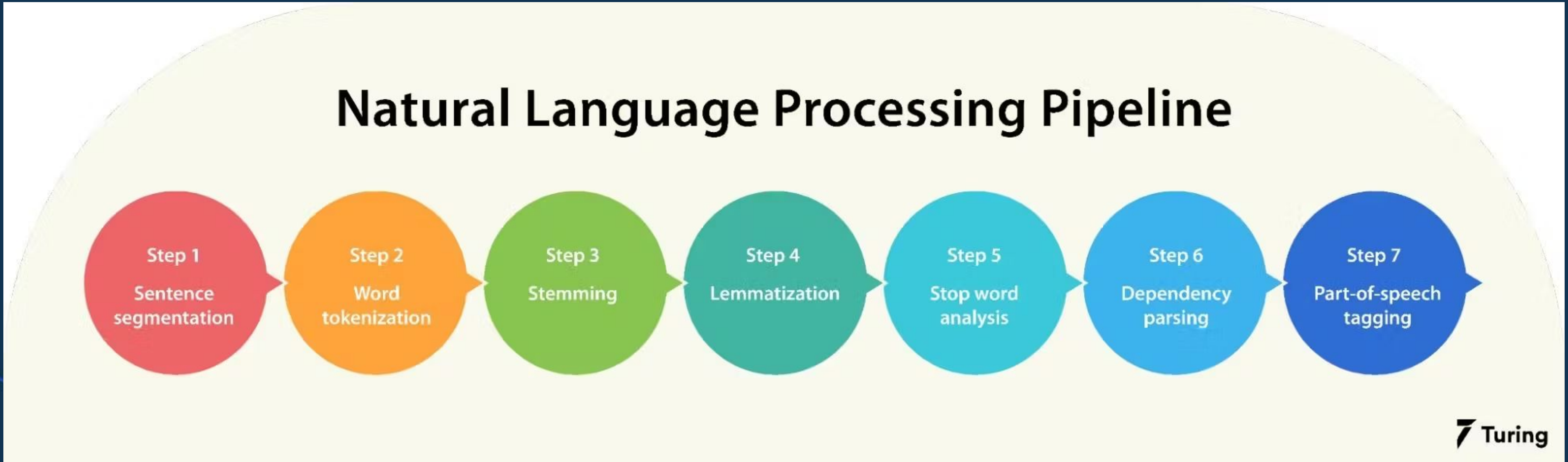
spaCy VSCode extension

NLP Pipeline

Text Preprocessing

CoGrammar

NLP Pipeline

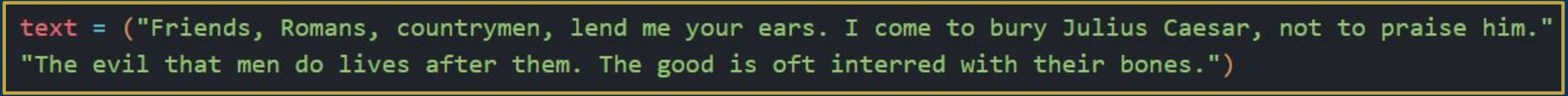


Natural Language Processing Pipeline

CoGrammar

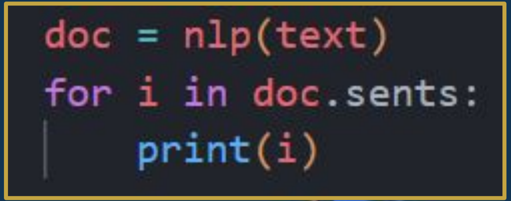
Sentence Fragmentation

Sentence Fragmentation is the first step in NLP pipeline, divides the entire paragraph into **different sentences** for better understanding.



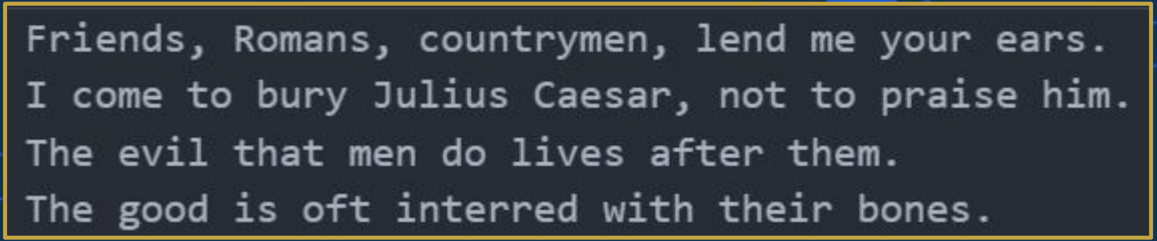
```
text = ("Friends, Romans, countrymen, lend me your ears. I come to bury Julius Caesar, not to praise him."
"The evil that men do lives after them. The good is oft interred with their bones.")
```

Sentence fragmentation using spaCy



```
doc = nlp(text)
for i in doc.sents:
    print(i)
```

Output

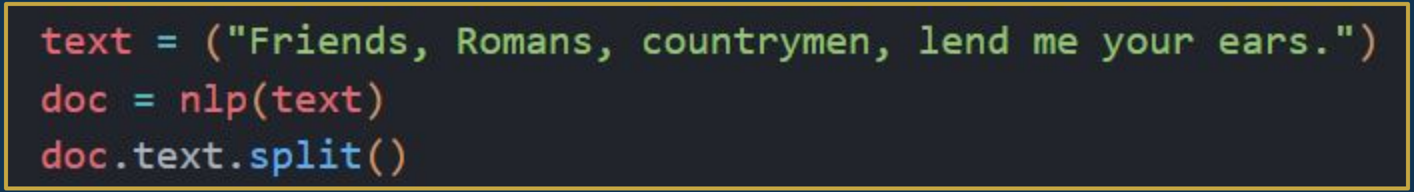


```
Friends, Romans, countrymen, lend me your ears.
I come to bury Julius Caesar, not to praise him.
The evil that men do lives after them.
The good is oft interred with their bones.
```

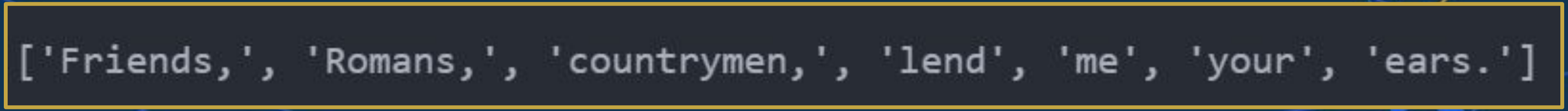
CoGrammar

Tokenisation

Word tokenisation breaks the sentence into **separate words** or **tokens**. This helps understand the context of the text.



```
text = ("Friends, Romans, countrymen, lend me your ears.")
doc = nlp(text)
doc.text.split()
```



```
['Friends,', 'Romans,', 'countrymen,', 'lend', 'me', 'your', 'ears.']
```

CoGrammar

Stemming

- **Stemming** normalises words into their **base or root form**, helps to **predict** the **parts of speech** for each token, involves **stripping** the **prefixes/suffixes** from words to **get their stem.**

- For example, converting the word “walking” to “walk”.

- Another example, “intelligently”, “intelligence”, and “intelligent”, all these words originate from a single root word “intelligen”. However, in English there is no such word as “intelligen”.

- Stemming chops off the part of word by assuming that the result is the expected word, **not grammar based**, hence **inaccurate**.

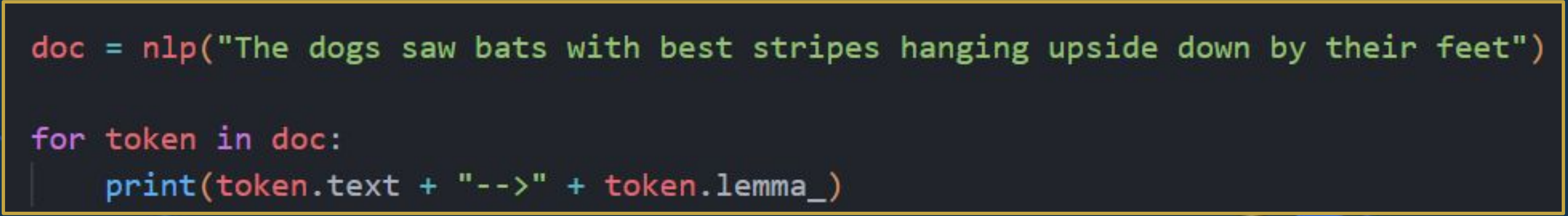
- **spaCy does not provide a built-in function for stemming** as its inaccuracy is not suitable for production level use.

CoGrammar

Lemmatisation

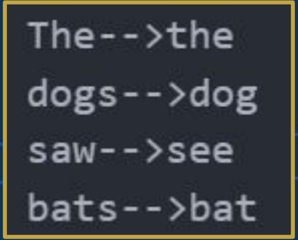
Lemmatisation **removes inflectional endings** and returns the canonical form of a word or **lemma**. Similar to stemming except that the lemma is an **actual word.**

For example, ‘playing’ and ‘plays’ are forms of the word ‘play’. Hence, play is the lemma of these words. Unlike a stem (recall ‘intelligen’), ‘play’ is a proper word.

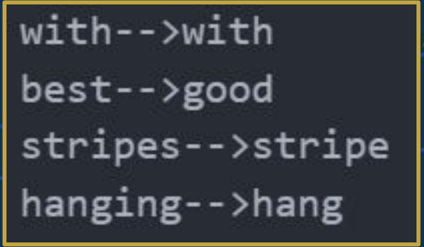


```
doc = nlp("The dogs saw bats with best stripes hanging upside down by their feet")

for token in doc:
    print(token.text + "-->" + token.lemma_)
```

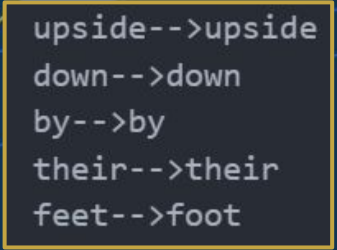
Output



```
The-->the
dogs-->dog
saw-->see
bats-->bat
```



```
with-->with
best-->good
stripes-->stripe
hanging-->hang
```



```
upside-->upside
down-->down
by-->by
their-->their
feet-->foot
```

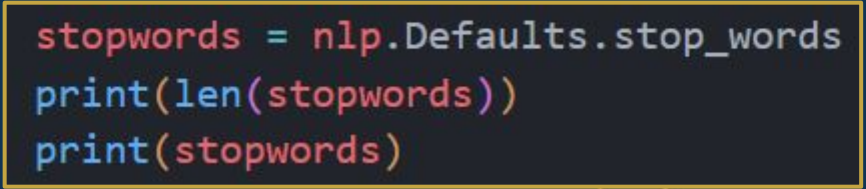
CoGrammar

Stop words

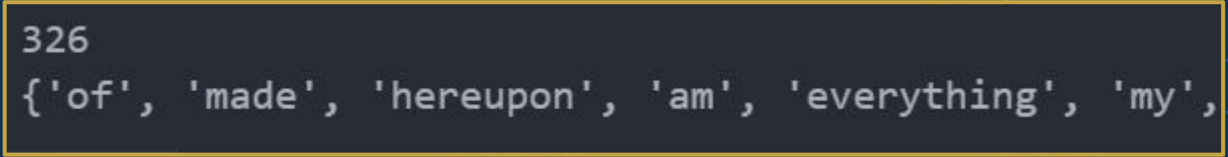
- Consider the **importance** of each and every word in a given sentence.

- In English, some words appear **more frequently** than others such as "is", "a", "the", "and". As they appear often, the NLP pipeline flags them as **stop words.** They are **filtered out** so as to focus on more important words.

spaCy has **326** default stopwords
(output shows only a few)



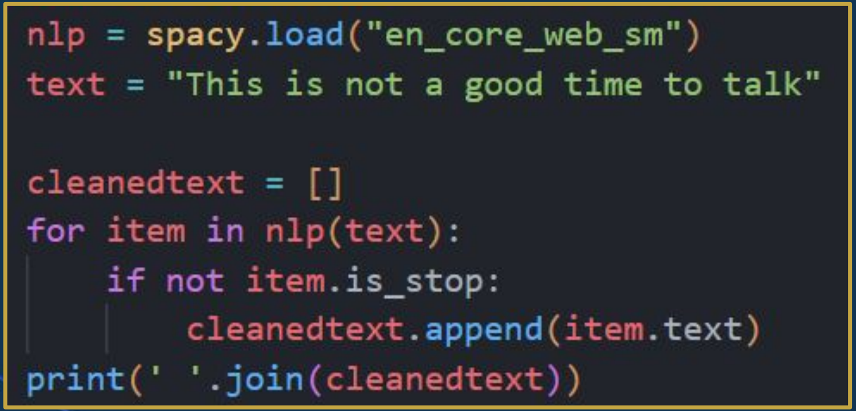
```
stopwords = nlp.Defaults.stop_words
print(len(stopwords))
print(stopwords)
```



```
326
{'of', 'made', 'hereupon', 'am', 'everything', 'my',
```

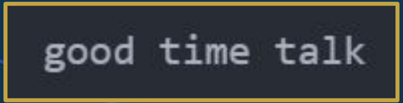
Stop words

Remove stop words from text



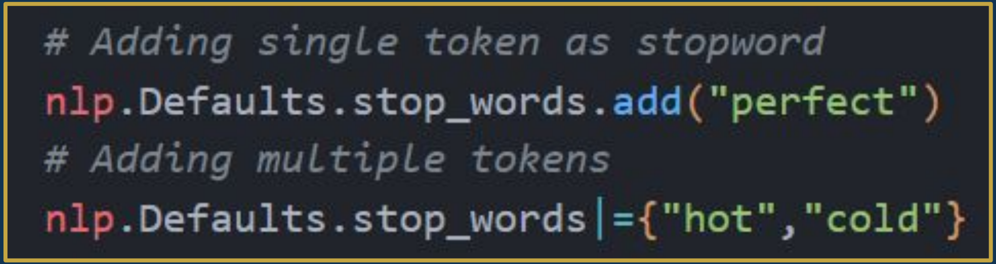
```python
nlp = spacy.load("en_core_web_sm")
text = "This is not a good time to talk"

cleanedtext = []
for item in nlp(text):
    if not item.is_stop:
        cleanedtext.append(item.text)
print(' '.join(cleanedtext))
```

Output

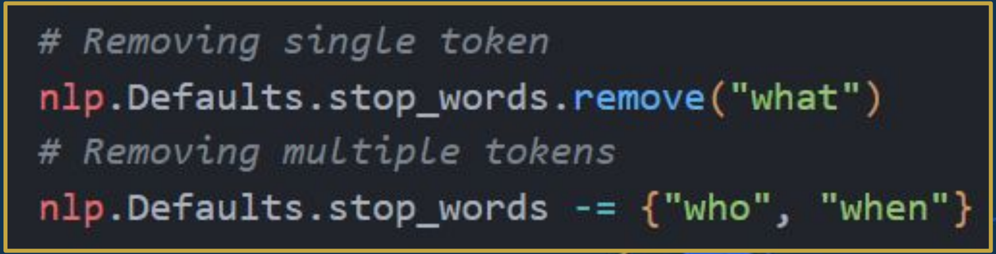


```
good time talk
```

Add/remove stop words



```python
# Adding single token as stopword
nlp.Defaults.stop_words.add("perfect")
# Adding multiple tokens
nlp.Defaults.stop_words|={"hot","cold"}
```



```python
# Removing single token
nlp.Defaults.stop_words.remove("what")
# Removing multiple tokens
nlp.Defaults.stop_words -= {"who", "when"}
```

CoGrammar

NLP Pipeline

Advanced Text Preprocessing

CoGrammar

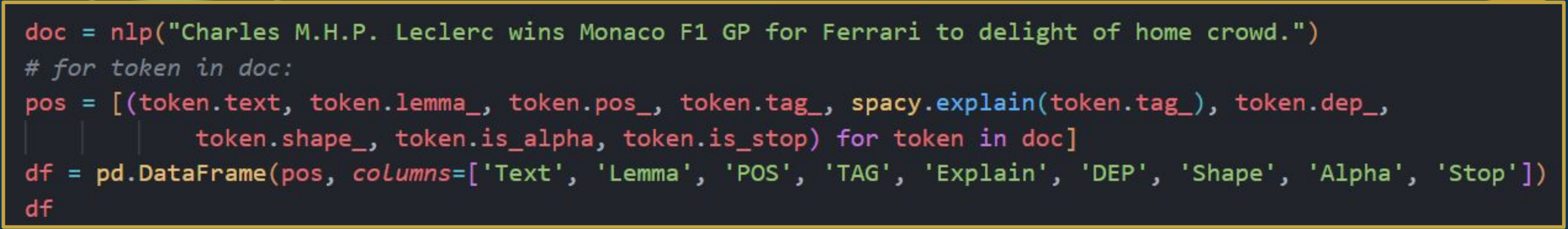
Parts Of Speech Tagging

- **Parts of speech (POS)** depicts how a specific word is utilized in a sentence, giving each word in a text a **grammatical category,** such as nouns, pronoun, verbs, adjectives, adverbs, prepositions, conjunctions, interjections.

- To **understand grammatical structure** of a sentence, **disambiguate words with multiple meanings** (e.g., "bank" can have multiple meanings), improve accuracy of NLP tasks, facilitate research in linguistics.

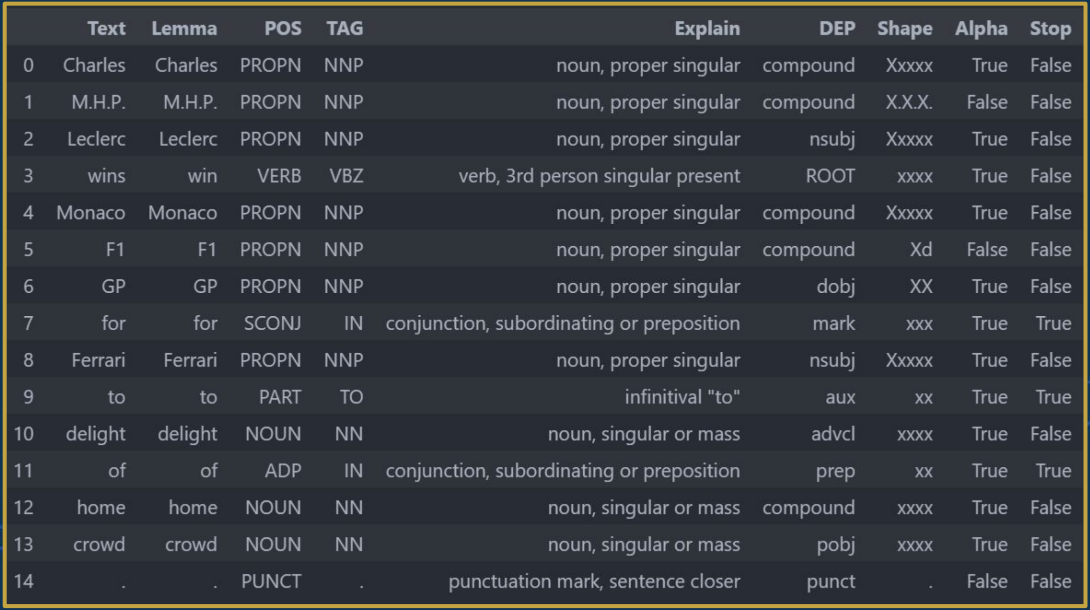
- Essential for assigning a **syntactic category,** needed for text summarization, sentiment analysis, machine translation.

POS Tagging



```
doc = nlp("Charles M.H.P. Leclerc wins Monaco F1 GP for Ferrari to delight of home crowd.")
# for token in doc:
pos = [(token.text, token.lemma_, token.pos_, token.tag_, spacy.explain(token.tag_), token.dep_,
            token.shape_, token.is_alpha, token.is_stop) for token in doc]
df = pd.DataFrame(pos, columns=['Text', 'Lemma', 'POS', 'TAG', 'Explain', 'DEP', 'Shape', 'Alpha', 'Stop'])
df
```

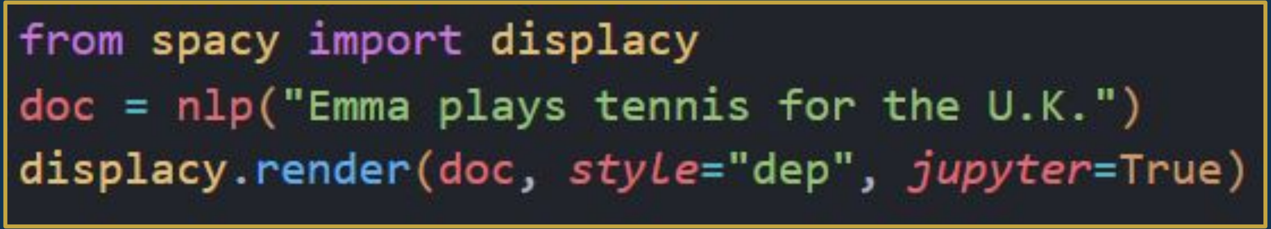
text: The original word text.
lemma: The base form of the word.
pos and **tag**: simple/detailed POS tag.
explain: More details for POS tag
dep: Syntactic dependency (relation between tokens.
shape: word shape (capitalization, punctuation, digits.)
is_alpha: Is token an alpha character
is_stop: Is token part of stop words



	Text	Lemma	POS	TAG	Explain	DEP	Shape	Alpha	Stop
0	Charles	Charles	PROPN	NNP	noun, proper singular	compound	Xxxxx	True	False
1	M.H.P.	M.H.P.	PROPN	NNP	noun, proper singular	compound	X.X.X.	False	False
2	Leclerc	Leclerc	PROPN	NNP	noun, proper singular	nsubj	Xxxxx	True	False
3	wins	win	VERB	VBZ	verb, 3rd person singular present	ROOT	xxxx	True	False
4	Monaco	Monaco	PROPN	NNP	noun, proper singular	compound	Xxxxx	True	False
5	F1	F1	PROPN	NNP	noun, proper singular	compound	Xd	False	False
6	GP	GP	PROPN	NNP	noun, proper singular	dobj	XX	True	False
7	for	for	SCONJ	IN	conjunction, subordinating or preposition	mark	xxx	True	True
8	Ferrari	Ferrari	PROPN	NNP	noun, proper singular	nsubj	Xxxxx	True	False
9	to	to	PART	TO	infinitival "to"	aux	xx	True	True
10	delight	delight	NOUN	NN	noun, singular or mass	advcl	xxxx	True	False
11	of	of	ADP	IN	conjunction, subordinating or preposition	prep	xx	True	True
12	home	home	NOUN	NN	noun, singular or mass	compound	xxxx	True	False
13	crowd	crowd	NOUN	NN	noun, singular or mass	pobj	xxxx	True	False
14	.	.	PUNCT	.	punctuation mark, sentence closer	punct	.	False	False

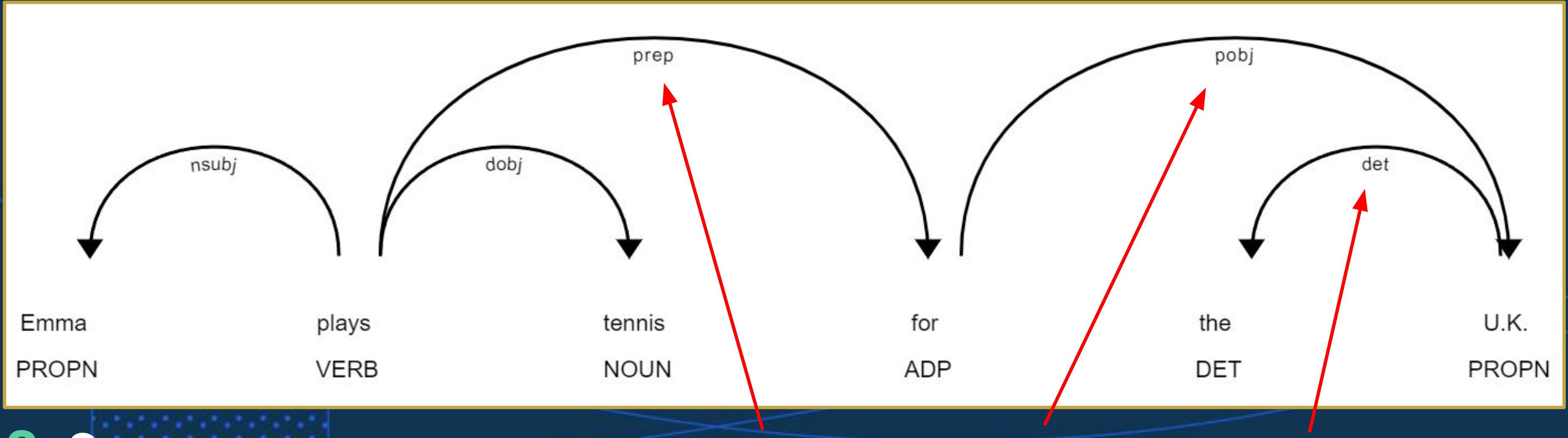
CoGrammar

Visualise POS Tagging



```python
from spacy import displacy
doc = nlp("Emma plays tennis for the U.K.")
displacy.render(doc, style="dep", jupyter=True)
```

https://spacy.io/usage/linguistic-features#pos-tagging



dep: Syntactic dependency (relation between tokens.

CoGrammar

Named Entity Recognition

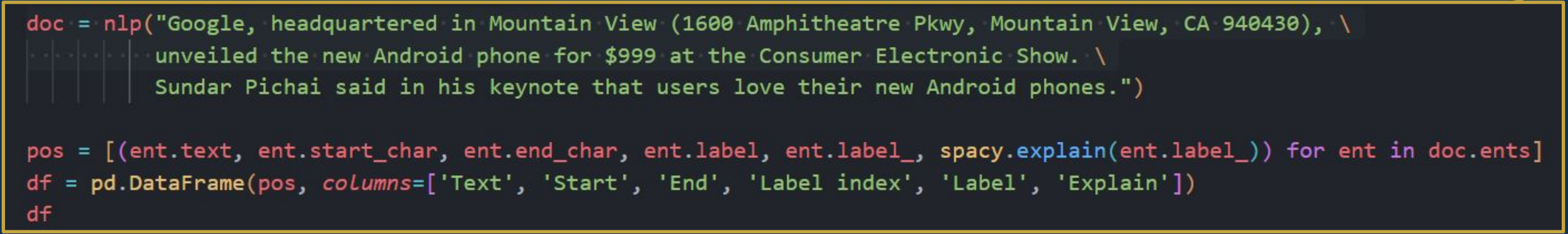
- **Named Entity Recognition (NER)** focuses on **identifying** and **classifying entities**, identifies key information in the text and classifies into a set of **predefined categories** (person names, organizations, locations, time expressions, quantities, percentages)

- **Ambiguity in classification**

  - **France (organization)** won the 1998 FIFA world cup vs The 1998 world cup happened in **France (location).**
  - **Washington (location)** is the capital of the US vs The first president of the US was **Washington (person)**.

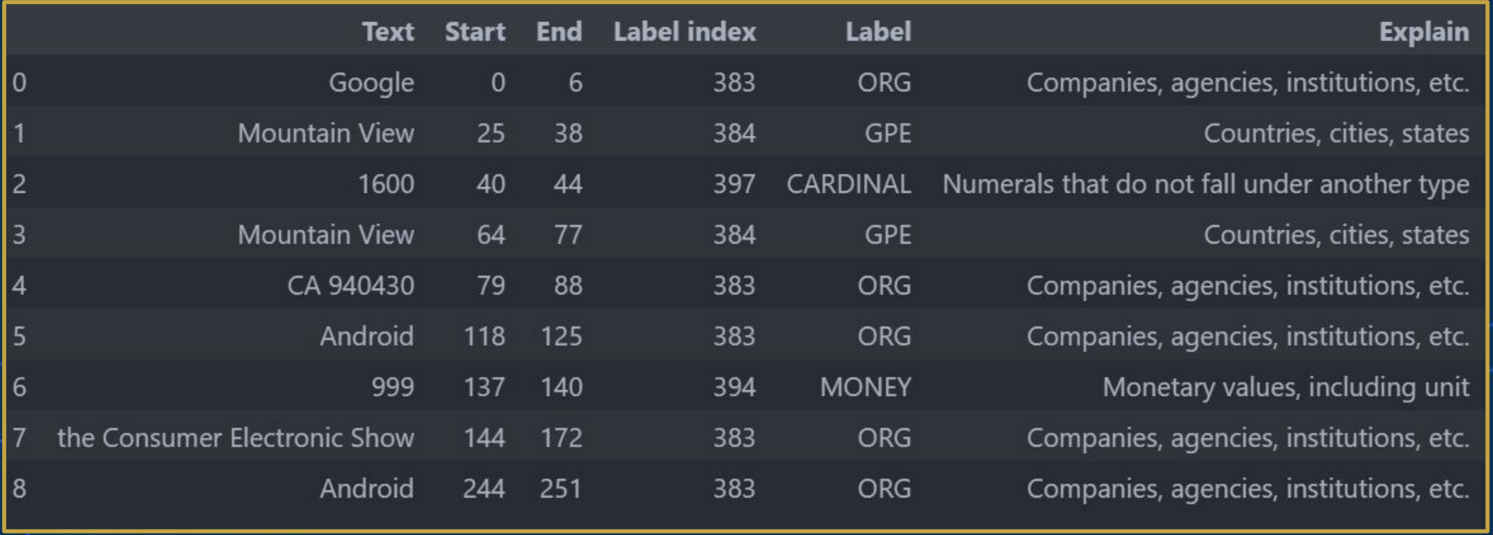
CoGrammar

NER



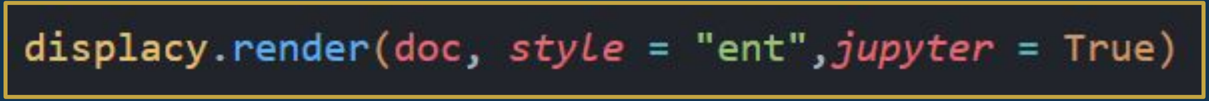
```
doc = nlp("Google, headquartered in Mountain View (1600 Amphitheatre Pkwy, Mountain View, CA 940430), \
          unveiled the new Android phone for $999 at the Consumer Electronic Show. \
          Sundar Pichai said in his keynote that users love their new Android phones.")

pos = [(ent.text, ent.start_char, ent.end_char, ent.label, ent.label_, spacy.explain(ent.label_)) for ent in doc.ents]
df = pd.DataFrame(pos, columns=['Text', 'Start', 'End', 'Label index', 'Label', 'Explain'])
df
```

Start and **End** Index of start/end of entity in the doc.

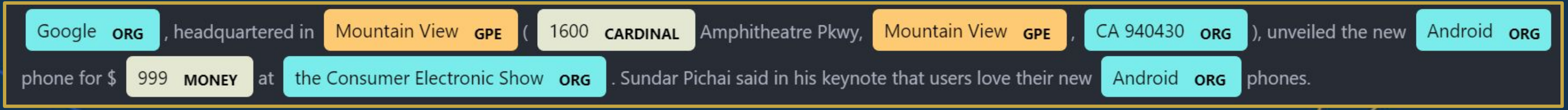


	Text	Start	End	Label index	Label	Explain
0	Google	0	6	383	ORG	Companies, agencies, institutions, etc.
1	Mountain View	25	38	384	GPE	Countries, cities, states
2	1600	40	44	397	CARDINAL	Numerals that do not fall under another type
3	Mountain View	64	77	384	GPE	Countries, cities, states
4	CA 940430	79	88	383	ORG	Companies, agencies, institutions, etc.
5	Android	118	125	383	ORG	Companies, agencies, institutions, etc.
6	999	137	140	394	MONEY	Monetary values, including unit
7	the Consumer Electronic Show	144	172	383	ORG	Companies, agencies, institutions, etc.
8	Android	244	251	383	ORG	Companies, agencies, institutions, etc.

Visualise NER



```
displacy.render(doc, style = "ent",jupyter = True)
```



Google ORG , headquartered in Mountain View GPE (1600 CARDINAL Amphitheatre Pkwy, Mountain View GPE , CA 940430 ORG), unveiled the new Android ORG phone for $ 999 MONEY at the Consumer Electronic Show ORG . Sundar Pichai said in his keynote that users love their new Android ORG phones.

https://spacy.io/usage/linguistic-features#named-entities

Summary and Next Steps

CoGrammar

Key points

- NLP needs extra processing steps compared to general machine learning pipelines as there are added challenges to natural language e.g. text data.

- Text cleaning: essential to prepare for NLP tasks. **Regular Expression** is used for searching strings of specific patterns to convert or remove them.

- Text Preprocessing includes **tokenisation, stemming** or **lemmatisation**, **stop-word removal, parts-of-speech tagging** and **named entity recognition.**

- Feature Engineering: represent text in numeric vectors for the ML algorithm to understand the text attribute.

- Model Building and Evaluation

Next Lecture

Questions and Answers

CoGrammar

Thank you for attending

SKILLS FOR LIFE
SKILLS BOOTCAMPS

Department for Education

CoGrammar