Unit:1

Introduction to NLP, NLP tasks in syntax, semantics, and pragmatics. Applications such as information extraction, question answering, and machine translation. The problem of ambiguity. The role of machine learning. Brief history of the field.

The essence of Natural Language Processing lies in making computers understand the natural language. That's not an easy task though.

Computers can understand the structured form of data like spreadsheets and tables in the database, but human languages, texts, and voices form an unstructured category of data, and it becomes difficult for the computer to understand it, and there is the need for Natural Language Processing.

It has various steps which will give us the desired output(maybe not in a few rare cases) at the end.

Step 1: Sentence Segmentation

Breaking the piece of text in various sentences.

Step 2: Word Tokenization

Breaking the sentence into individual words called as tokens

Step 3: Predicting Parts of Speech for each token

Predicting whether the word is a noun, verb, adjective, adverb, pronoun, etc. This will help to understand what the sentence is talking about. This can be achieved by feeding the tokens(and the words around it) to a pre-trained part-of-speech classification model

Step 4: Lemmatization

Feeding the model with the root word.

##'Play' and 'Playing' should be considered as same

Step 5: Identifying stop words

There are various words in the English language that are used very frequently like 'a', 'and', 'the' etc

What is Syntax?

A natural language typically follows a hierarchical structure, and contains the following components:

Sentences

Clauses

Phrases

Words

Some of the syntactic categories of a natural language are as follows:

Sentence(S)

Noun Phrase(NP)

Determiner(Det)

Verb Phrase(VP)

Prepositional Phrase(PP)

Verb(V)

Noun(N)

Example:

The syntax tree for the sentence given below is as follows:

I drive a car to my college.

Abbreviation Meaning

CC coordinating conjunction

CD cardinal digit

DT determiner

EX existential there

FW foreign word

IN preposition/subordinating conjunction

JJ This NLTK POS Tag is an adjective (large)

JJR adjective, comparative (larger)

JJS adjective, superlative (largest)

LS list market

MD modal (could, will)

NN noun, singular (cat, tree)

NNS noun plural (desks)

NNP proper noun, singular (sarah)

NNPS proper noun, plural (indians or americans)

PDT predeterminer (all, both, half)

POS possessive ending (parent\ ‘s)

PRP personal pronoun (hers, herself, him, himself)

PRP$ possessive pronoun (her, his, mine, my, our)

RB adverb (occasionally, swiftly)

RBR adverb, comparative (greater)

RBS adverb, superlative (biggest)

RP particle (about)

TO infinite marker (to)

UH interjection (goodbye)

VB verb (ask)

VBG verb gerund (judging)

VBD verb past tense (pleaded)

VBN verb past participle (reunified)

VBP verb, present tense not 3rd person singular(wrap)

VBZ verb, present tense with 3rd person singular (bases)

WDT wh-determiner (that, what)

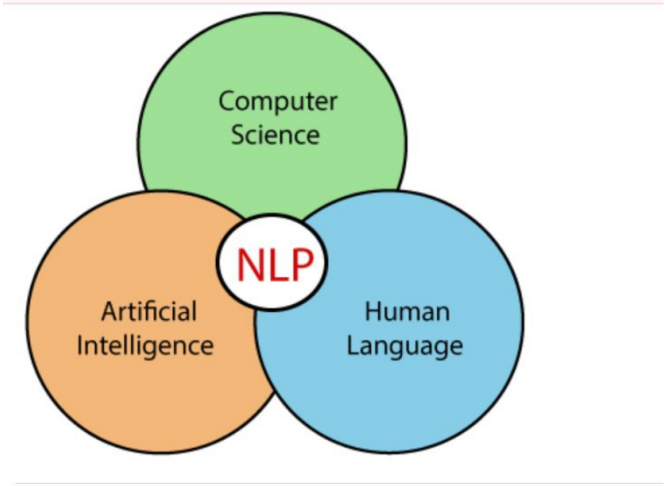
WP wh- pronoun (who)

WRB wh- adverb (how)

What is NLP?

NLP stands for Natural Language Processing, which is a part of Computer Science, Human language, and Artificial Intelligence.

It is the technology that is used by machines to understand, analyse, manipulate, and interpret human's languages. It helps developers to organize knowledge for performing tasks such as translation, automatic summarization, Named Entity Recognition (NER), speech recognition, relationship extraction, and topic segmentation.



History of NLP

(1940-1960) - Focused on Machine Translation (MT)

The Natural Languages Processing started in the year 1940s.

1948 - In the Year 1948, the first recognisable NLP application was introduced in Birkbeck College, London.

1950s - In the Year 1950s, there was a conflicting view between linguistics and computer science. Now, Chomsky developed his first book syntactic structures and claimed that language is generative in nature.

In 1957, Chomsky also introduced the idea of Generative Grammar, which is rule based descriptions of syntactic structures.

(1960-1980) - Flavored with Artificial Intelligence (AI)

In the year 1960 to 1980, the key developments were:

Augmented Transition Networks (ATN)

Augmented Transition Networks is a finite state machine that is capable of recognizing regular languages.

Case Grammar

Case Grammar was developed by Linguist Charles J. Fillmore in the year 1968. Case Grammar uses languages such as English to express the relationship between nouns and verbs by using the preposition.

In Case Grammar, case roles can be defined to link certain kinds of verbs and objects.

For example: "Neha broke the mirror with the hammer". In this example case grammar identify Neha as an agent, mirror as a theme, and hammer as an instrument.

In the year 1960 to 1980, key systems were:

SHRDLU

SHRDLU is a program written by Terry Winograd in 1968-70. It helps users to communicate with the computer and moving objects. It can handle instructions such as "pick up the green ball" and also

computer and moving objects. It can handle instructions such as "pick up the green boll" and also answer the questions like "What is inside the black box." The main importance of SHRDLU is that it shows those syntax, semantics, and reasoning about the world that can be combined to produce a system that understands a natural language.

LUNAR

LUNAR is the classic example of a Natural Language database interface system that is used ATNs and Woods' Procedural Semantics. It was capable of translating elaborate natural language expressions into database queries and handle 78% of requests without errors.

1980 - Current

Till the year 1980, natural language processing systems were based on complex sets of hand-written rules. After 1980, NLP introduced machine learning algorithms for language processing.

In the beginning of the year 1990s, NLP started growing faster and achieved good process accuracy, especially in English Grammar. In 1990 also, an electronic text introduced, which provided a good resource for training and examining natural language programs. Other factors may include the availability of computers with fast CPUs and more memory. The major factor behind the advancement of natural language processing was the Internet.

Now, modern NLP consists of various applications, like speech recognition, machine translation, and machine text reading. When we combine all these applications then it allows the artificial intelligence to gain knowledge of the world. Let's consider the example of AMAZON ALEXA, using this robot you can ask the question to Alexa, and it will reply to you.

Advantages of NLP

NLP helps users to ask questions about any subject and get a direct response within seconds.

NLP offers exact answers to the question means it does not offer unnecessary and unwanted information.

NLP helps computers to communicate with humans in their languages.

It is very time efficient.

Most of the companies use NLP to improve the efficiency of documentation processes, accuracy of documentation, and identify the information from large databases.

Disadvantages of NLP

A list of disadvantages of NLP is given below:

NLP may not show context.

NLP is unpredictable

NLP may require more keystrokes.

NLP is unable to adapt to the new domain, and it has a limited function that's why NLP is built for a single and specific task only.

Components of NLP

There are the following two components of NLP -

1. Natural Language Understanding (NLU)

Natural Language Understanding (NLU) helps the machine to understand and analyse human language by extracting the metadata from content such as concepts, entities, keywords, emotion, relations, and semantic roles.

NLU mainly used in Business applications to understand the customer's problem in both spoken and written language.

NLU involves the following tasks -

It is used to map the given input into useful representation.

It is used to analyze different aspects of the language.

2. Natural Language Generation (NLG)

Natural Language Generation (NLG) acts as a translator that converts the computerized data into natural language representation. It mainly involves Text planning, Sentence planning, and Text Realization.

Difference between NLU and NLG

Difference between NLU and NLG

NLU	NLG
NLU is the process of reading and interpreting language.	NLG is the process of writing or generating language.
It produces non-linguistic outputs from natural language inputs.	It produces constructing natural language outputs from non-linguistic inputs.

Applications of NLP

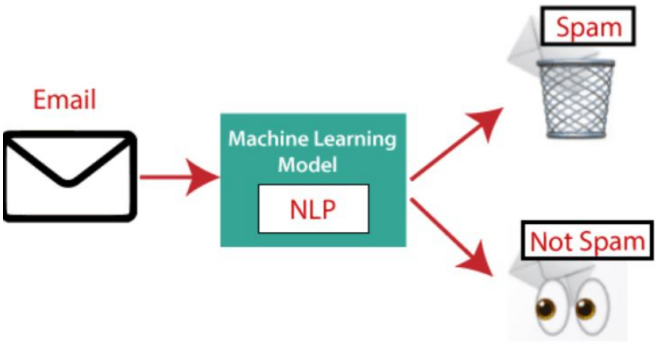
There are the following applications of NLP -

1. Question Answering

Question Answering focuses on building systems that automatically answer the questions asked by humans in a natural language.

2. Spam Detection

Spam detection is used to detect unwanted e-mails getting to a user's inbox.



3. Sentiment Analysis

Sentiment Analysis is also known as opinion mining. It is used on the web to analyse the attitude, behaviour, and emotional state of the sender. This application is implemented through a combination of NLP (Natural Language Processing) and statistics by assigning the values to the text (positive, negative, or natural), identify the mood of the context (happy, sad, angry, etc.)

4. Machine Translation

Machine translation is used to translate text or speech from one natural language to another natural language.

Example: Google Translator

5. Spelling correction

Microsoft Corporation provides word processor software like MS-word, PowerPoint for the spelling correction.

6. Speech Recognition

Speech recognition is used for converting spoken words into text. It is used in applications, such as mobile, home automation, video recovery, dictating to Microsoft Word, voice biometrics, voice user interface, and so on.

7. Chatbot

Implementing the Chatbot is one of the important applications of NLP. It is used by many companies to provide the customer's chat services.

8. Information extraction

Information extraction is one of the most important applications of NLP. It is used for extracting structured information from unstructured or semi-structured machine-readable documents.

9. Natural Language Understanding (NLU)

It converts a large set of text into more formal representations such as first-order logic structures that are easier for the computer programs to manipulate notations of the natural language processing.

How to build an NLP pipeline

There are the following steps to build an NLP pipeline -

Step1: Sentence Segmentation

Sentence Segment is the first step for building the NLP pipeline. It breaks the paragraph into separate sentences.

Example: Consider the following paragraph -

Independence Day is one of the important festivals for every Indian citizen. It is celebrated on the 15th of August each year ever since India got independence from the British rule. The day celebrates independence in the true sense.

Sentence Segment produces the following result:

1."Independence Day is one of the important festivals for every Indian citizen."

2."It is celebrated on the 15th of August each year ever since India got independence from the British rule."

3."This day celebrates independence in the true sense."

Step2: Word Tokenization

Word Tokenizer is used to break the sentence into separate words or tokens.

Example:

Data Science Department offers Corporate Training, Summer Training, Online Training, and Winter Training.

Word Tokenizer generates the following result:

"Data" "Science" "Department " "JavaTpoint", "offers", "Corporate", "Training", "Summer", "Training", "Online", "Training", "and", "Winter", "Training", "."

Step3: Stemming

Stemming is used to normalize words into its base form or root form.

For example, celebrates, celebrated and celebrating, all these words are originated with a single root word "celebrate."

The big problem with stemming is that sometimes it produces the root word which may not have any meaning.

For Example, intelligence, intelligent, and intelligently, all these words are originated with a single root word "intelligen." In English, the word "intelligen" do not have any meaning

Step 4: Lemmatization

Lemmatization is quite similar to the Stamming. It is used to group different inflected forms of the word, called Lemma.

The main difference between Stemming and lemmatization is that it produces the root word, which has a meaning.

For example: In lemmatization, the words intelligence, intelligent, and intelligently has a root word intelligent, which has a meaning.

Step 5: Identifying Stop Words

In English, there are a lot of words that appear very frequently like "is", "and", "the", and "a". NLP pipelines will flag these words as stop words. Stop words might be filtered out before doing any statistical analysis.

Example: He is a good boy.

Step 6: Dependency Parsing

Dependency Parsing is used to find that how all the words in the sentence are related to each other.

Step 7: POS tags

POS stands for parts of speech, which includes Noun, verb, adverb, and Adjective.

It indicates that how a word functions with its meaning as well as grammatically within the

It indicates that how a word functions with its meaning as well as grammatically within the sentences.

A word has one or more parts of speech based on the context in which it is used.

Example: "Google" something on the Internet.

In the above example, Google is used as a verb, although it is a proper noun.

```
[ ] Start coding or generate with AI.
```

Step 8: Named Entity Recognition (NER)

Named Entity Recognition (NER) is the process of detecting the named entity such as person name, movie name, organization name, or location.

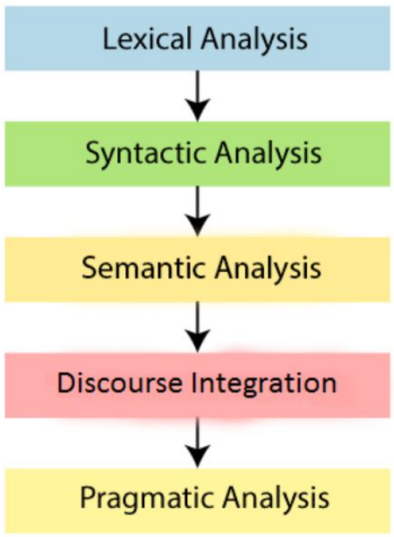
Example: Steve Jobs introduced iPhone at the Macworld Conference in San Francisco, California.

Step 9: Chunking

Chunking is used to collect the individual piece of information and grouping them into bigger pieces of sentences.

Phases of NLP

There are the following five phases of NLP:



1. Lexical Analysis and Morphological

The first phase of NLP is the Lexical Analysis. This phase scans the source code as a stream of characters and converts it into meaningful lexemes.

It divides the whole text into paragraphs, sentences, and words.

2. Syntactic Analysis (Parsing)

Syntactic Analysis is used to check grammar, word arrangements, and shows the relationship among the words.

Example: Agra goes to the Poonam

In the real world, Agra goes to the Poonam, does not make any sense, so this sentence is rejected by the Syntactic analyzer.

3. Semantic Analysis

Semantic analysis is concerned with the meaning representation.

It mainly focuses on the literal meaning of words, phrases, and sentences and it analyzes the text for meaningfulness

for meaningfulness.

For example - sentences such as “hot ice-cream” do not pass.

4. Discourse Integration

Discourse Integration depends upon the sentences that proceeds it and also invokes the meaning of the sentences that follow it.

For example - "This is not true". The word "This" in this sentence depends upon the context of the word "This" in the previous sentences.

5. Pragmatic Analysis

Pragmatic is the fifth and last phase of NLP. It helps you to discover the intended effect by applying a set of rules that characterize cooperative dialogues.

For Example: "Open the door" is interpreted as a request instead of an order.

One example of pragmatics in language would be if one person asked, "What do you want to eat?" and another responded, "Ice cream is good this time of year."

The second person did not explicitly say what they wanted to eat, but their statement implies that they want to eat ice cream.

Why NLP is difficult?

NLP is difficult because Ambiguity and Uncertainty exist in the language.

##Ambiguity

There are the following three ambiguity -

1. Lexical Ambiguity

Lexical Ambiguity exists in the presence of two or more possible meanings of the sentence within a single word.

Example:

Manya is looking for a match.

In the above example, the word match refers to that either Manya is looking for a partner or Manya is looking for a match. (Cricket or other match)

2. Syntactic Ambiguity

Syntactic Ambiguity exists in the presence of two or more possible meanings within the sentence.

Example:

I saw the girl with the binocular.

In the above example, did I have the binoculars? Or did the girl have the binoculars?

3.Referential Ambiguity

Referential Ambiguity exists when you are referring to something using the pronoun.

Example: Kiran went to Sunita. She said, "I am hungry."

In the above sentence, We do not know that who is hungry, either Kiran or Sunita.

NLP APIs

Natural Language Processing APIs allow developers to integrate human-to-machine communications and complete several useful tasks such as speech recognition, chatbots, spelling correction, sentiment analysis, etc.

A list of NLP APIs is given below:

IBM Watson API

IBM Watson API combines different sophisticated machine learning techniques to enable developers to classify text into various custom categories.

It supports multiple languages, such as English, French, Spanish, German, Chinese, etc. With the help of IBM Watson API, you can extract insights from texts, add automation in workflows, enhance search, and understand the sentiment.

The main advantage of this API is that it is very easy to use.

Pricing: Firstly, it offers a free 30 days trial IBM cloud account. You can also opt for its paid plans.

Chatbot API

Chatbot API allows you to create intelligent chatbots for any service.

It supports Unicode characters, classifies text, multiple languages, etc.

It is very easy to use. It helps you to create a chatbot for your web applications.

Pricing: Chatbot API is free for 150 requests per month. You can also opt for its paid version, which starts from $100 to 5,000 per month.

Speech to text API

Speech to text API is used to convert speech to text

Pricing: Speech to text API is free for converting 60 minutes per month. Its paid version starts form $500 to 1,500 per month.

Sentiment Analysis API

Sentiment Analysis API is also called as 'opinion mining' which is used to identify the tone of a user (positive, negative, or neutral)

Pricing: Sentiment Analysis API is free for less than 500 requests per month. Its paid version starts form $19 to 99 per month.

Translation API by SYSTRAN

The Translation API by SYSTRAN is used to translate the text from the source language to the target language. We can use its NLP APIs for language detection, text segmentation, named entity recognition, tokenization, and many other tasks.

Pricing: This API is available for free. But for commercial users, We need to use its paid version.

Text Analysis API by AYLIEN

Text Analysis API by AYLIEN is used to derive meaning and insights from the textual content. It is available for both free as well as paid from$119 per month. It is easy to use.

Pricing: This API is available free for 1,000 hits per day. We can also use its paid version, which starts from $199 to S1, 399 per month.

Cloud NLP API

The Cloud NLP API is used to improve the capabilities of the application using natural language processing technology. It allows us to carry various natural language processing functions like sentiment analysis and language detection. It is easy to use.

Pricing: Cloud NLP API is available for free.

Google Cloud Natural Language API

Google Cloud Natural Language API allows you to extract beneficial insights from unstructured text.

This API allows you to perform entity recognition, sentiment analysis, content classification, and syntax analysis in more the 700 predefined categories. It also allows you to perform text analysis in multiple languages such as English, French, Chinese, and German.

Pricing: After performing entity analysis for 5,000 to 10,000,000 units, you need to pay $1.00 per 1000 units per month.

NLP Libraries

Scikit-learn: It provides a wide range of algorithms for building machine learning models in Python.

Natural language Toolkit (NLTK): NLTK is a complete toolkit for all NLP techniques.

Pattern: It is a web mining module for NLP and machine learning.

TextBlob: It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase

TextBlob: It provides an easy interface to learn basic NLP tasks like sentiment analysis, noun phrase extraction, or pos-tagging.

Quepy: Quepy is used to transform natural language questions into queries in a database query language.

SpaCy: SpaCy is an open-source NLP library which is used for Data Extraction, Data Analysis, Sentiment Analysis, and Text Summarization.

Gensim: Gensim works with large datasets and processes data streams.

A RegEx, or Regular Expression, is a sequence of characters that forms a search pattern.

RegEx can be used to check if a string contains the specified search pattern.

RegEx Module

Python has a built-in package called re, which can be used to work with Regular Expressions.

Import the re module:

RegEx in Python

When you have imported the re module, you can start using regular expressions:

Search the string to see if it starts with "The" and ends with "Spain":

```
[7] import re

  txt = "The rain in Spain"
  x = re.search("^The.*Spain$", txt)

  if x:
    print("YES! We have a match!")
  else:
    print("No match")
```

```
YES! We have a match!
```

RegEx Functions

The re module offers a set of functions that allows us to search a string for a match:

Function Description

findall Returns a list containing all matches

search 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

Metacharacters

Metacharacters are characters with a special meaning:

[] A set of characters

```
[14] import re
txt = "The rain in Spain"
#Find all lower case characters alphabetically between "T" and "n":
x = re.findall("[T-n]", txt)
print(x)
```

```
['T', 'h', 'e', 'a', 'i', 'n', 'i', 'n', 'a', 'i', 'n']
```

\ Signals a special sequence (can also be used to escape special characters)

"\d"

```
[15] import re
txt = "That will be 59 dollars"
#Find all digit characters:
x = re.findall("\d", txt)
print(x)
```

```
['5', '9']
```

. Any character (except newline character) "he..o"

```
[ ] import re
txt = "hello planet"
x = re.findall("he..o", txt)
print(x)
```

Search for a sequence that starts with "he", followed by two (any) characters, and an "o":

^ Starts with "^hello"

Check if the string starts with 'hello':

```
import re
txt = "hello planet"
x = re.findall("^hello", txt)
if x:
  print("Yes, the string starts with 'hello'")
else:
  print("No match")
```

```
Yes, the string starts with 'hello'
```

Endswith"
" *planet*

Check if the string ends with 'planet':

```
[18] import re

txt = "hello planet"


x = re.findall("planet$", txt)
if x:
  print("Yes, the string ends with 'planet'")
else:
  print("No match")
```

Yes, the string ends with "planet"

Double-click (or enter) to edit

* Zero or more occurrences "he.*o"

```
[19] import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed by 0 or more  (any) characters, and an "o":
x = re.findall("he.*o", txt)
print(x)
```

```
['hello']
```

+ One or more occurrences "he.+o"

```
[20] import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed by 1 or more  (any) characters, and an "o":
x = re.findall("he.+o", txt)
print(x)
```

```
['hello']
```

? Zero or one occurrences "he.?o"

This time we got no match, because there were not zero, not one, but two characters between "he" and the "o"

```
[21] import re

txt = "hello planet"

#Search for a sequence that starts with "he", followed by 0 or 1  (any) character, and an "o":

x = re.findall("he.?o", txt)

print(x)

#This time we got no match, because there were not zero, not one, but two characters between "he" and t
```

```
[]
```

{} Exactly the specified number of occurrences "he.{2}o"

```
[23] import re
txt = "hello planet"
#Search for a sequence that starts with "he", followed excactly 2 (any) characters, and an "o":
x = re.findall("he.{3}o", txt)
print(x)
```

```
[]
```

| Either or "falls|stays"

```
[24] import re

txt = "The rain in Spain falls mainly in the plain!"

#Check if the string contains either "falls" or "stays":

x = re.findall("falls|stays", txt)

print(x)

if x:
  print("Yes, there is at least one match!")
```

```
else:
  print("No match")
```

```
['falls']
Yes, there is at least one match!
```

The findall() Function

The findall() function returns a list containing all matches.

```
#Example
#Print a list of all matches:
import re
txt = "The rain in Spain"
x = re.findall("ai", txt)
print(x)
```

```
['ai', 'ai']
```

The list contains the matches in the order they are found.

If no matches are found, an empty list is returned:

```
#Example
#Return an empty list if no match was found:

import re

txt = "The rain in Spain"
x = re.findall("Portugal", txt)
print(x)
```

```
[]
```

The search() Function

The search() function searches the string for a match, and returns a Match object if there is a match.

If there is more than one match, only the first occurrence of the match will be returned:

Example

Search for the first white-space character in the string:

```
import re
txt = "The rain in Spain"
x = re.search("\s", txt)
print("The first white-space character is located in position:", x.start())
```

```
The first white-space character is located in position: 3
```

Double-click (or enter) to edit

The split() Function

The split() function returns a list where the string has been split at each match:

Example

Split at each white-space character:

```
import re

txt = "The rain in Spain"
x = re.split("\s", txt)
print(x)
```

```
['The', 'rain', 'in', 'Spain']
```

We can control the number of occurrences by specifying the maxsplit parameter:

Example

Split the string only at the first occurrence:

```
[29] import re
     txt = "The rain in Spain"
     x = re.split("\s", txt, 1)
     print(x)
```

```
['The', 'rain in Spain']
```

The sub() Function

The sub() function replaces the matches with the text of your choice:

Example

Replace every white-space character with the number 9:

```
import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt)
print(x)
```

```
The9rain9in9Spain
```

We can control the number of replacements by specifying the count parameter:

Example

Replace the first 2 occurrences:

```
[31] import re

txt = "The rain in Spain"
x = re.sub("\s", "9", txt, 2)
print(x)
```

```
The9rain9in Spain
```

The Match object has properties and methods used to retrieve information about the search, and the result:

.span() returns a tuple containing the start-, and end positions of the match.

.string returns the string passed into the function

.group() returns the part of the string where there was a match

Example

Print the position (start- and end-position) of the first match occurrence.

The regular expression looks for any words that starts with an upper case "S":

```
[38] import re
     txt = "The rain in Spain"
     x = re.search(r"\bS\w+", txt)
     print(x.span())
```

```
(12, 17)
```

Regular Expressions

Retrieve order number from given number

- ^ Asserts position at start of the string

- Asserts position at start of the string
- `(` Denotes the start of a capturing group
- `\d` Numerical digit, 0, 1, 2, ... 9. Etc.
- `{1,2}` one to two times.
- `)` You guessed it - Closes the group.
- `$` Assert position at end of the string

Parenthesis actually denotes a capturing group, not that they are part of the syntax itself.

[58]
```
import re
chat1='codebasics: Hello, I am having an issue with my order # 412889912'
pattern = '[\d{10}]'
matches = re.findall(pattern, chat1)
matches
```

```
['4', '1', '2', '8', '8', '9', '9', '1', '2']
```

[73]
```
import re
chat1='codebasics: Hello, I am having an issue with my order # 412889912'
pattern = '\d+'
matches = re.findall(pattern, chat1)
matches
```

```
['412889912']
```

[ ]
```
import re
chat1='codebasics: Hello, I am having an issue with my order # 412889912'
pattern = '\d+'
matches = re.findall(pattern, chat1)
matches
```

```
text='''
Born    Elon Reeve Musk
June 28, 1971 (age 50)
Pretoria, Transvaal, South Africa
Citizenship
South Africa (1971-present)
Canada (1971-present)
United States (2002-present)
Education       University of Pennsylvania (BS, BA)
Title
Founder, CEO and Chief Engineer of SpaceX
CEO and product architect of Tesla, Inc.
Founder of The Boring Company and X.com (now part of PayPal)
Co-founder of Neuralink, OpenAI, and Zip2
Spouse(s)
Justine Wilson

(m. 2000; div. 2008)
Talulah Riley

(m. 2010; div. 2012)

(m. 2013; div. 2016)
'''
```

Double-click (or enter) to edit

Start coding or generate with AI.

Double-click (or enter) to edit

Q.1 Get Age from above text

```
text='''
Born  Elon Reeve Musk
June 28, 1971 (age 50)
Pretoria, Transvaal, South Africa
Citizenship
South Africa (1971-present)
Canada (1971-present)
United States (2002-present)
```

```
Education University of Pennsylvania (BS, BA)
Title
Founder, CEO and Chief Engineer of SpaceX
CEO and product architect of Tesla, Inc.
Founder of The Boring Company and X.com (now part of PayPal)
Co-founder of Neuralink, OpenAI, and Zip2
Spouse(s)
Justine Wilson

(m. 2000; div. 2008)
Talulah Riley

(m. 2010; div. 2012)

(m. 2013; div. 2016)
'''
```

```
[78] def get_pattern_match(pattern, text):
         matches = re.findall(pattern, text)
         if matches:
             return matches[0]
```

```
[79] get_pattern_match(r'age (\d+)', text)
```

```
'50'
```

Q,2 get the name after boarn string

```
[80] get_pattern_match(r'Born(.*)\n', text).strip()
```

```
'Elon Reeve Musk'
```

Q.1 Extract Concentration Risk Types. It will be a text that appears after "Concentration Risk:", In below example, your regex should extract these two strings

```
text = '''
Concentration of Risk: Credit Risk
Financial instruments that potentially subject us to a concentration of credit risk consist of cash, ca
restricted cash, accounts receivable, convertible note hedges, and interest rate swaps. Our cash balanc
or on deposit at high credit quality financial institutions in the U.S. These deposits are typically in
and December 31, 2020, no entity represented 10% or more of our total accounts receivable balance. The
hedges and interest rate swaps is mitigated by transacting with several highly-rated multinational bank
Concentration of Risk: Supply Risk
We are dependent on our suppliers, including single source suppliers, and the inability of these suppli
products in a timely manner at prices, quality levels and volumes acceptable to us, or our inability to
suppliers, could have a material adverse effect on our business, prospects, financial condition and ope
'''
```

```
['Credit Risk', 'Supply Risk']
```

Spacy Tokenization

```
[83] import spacy
```

Create blank language object and tokenize words in a sentence

```
[87] nlp = spacy.blank("en")

doc = nlp("Data Science Running Mtech Executive from 20-22.")

for token in doc:
    print(token)
```

```
Data
Science
Running
Mtech
Executive
from
20
-
22
.
```

Using index to grab tokens

```
[90] doc[8]
```

```
22
```

```
[91] token = doc[1]
     token.text
```

```
'Science'
```

Span object

```
[92] span = doc[0:5]
     span
```

```
Data Science Running Mtech Executive
```

Collecting email ids of students from students information sheet

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

```
[95] with open("/content/drive/My Drive/Colab Notebooks/Mtech-III(NLP)/student.txt") as f:
         text = f.readlines()
     text
```

```
['Dayton high school, 8th grade students information\n',
 '==================================================\n',
 '\n',
 'Name\tbirth day   \temail\n',
 '-----\t-----------\t------\n',
 'Virat   5 June, 1882    virat@kohli.com\n',
 'Maria\t12 April, 2001  maria@sharapova.com\n',
 'Serena  24 June, 1998   serena@williams.com \n',
 'Joe      1 May, 1997    joe@root.com\n',
 '\n',
 '\n',
 '\n']
```

Supporting Other langaues

```
[96] nlp = spacy.blank("hi")
     doc = nlp("भैया जी! 5000 ₹ उधार थे वो वापस देदो")
     for token in doc:
         print(token, token.is_currency)
```

```
भैया False
जी False
! False
5000 False
₹ True
उधार False
थे False
वो False
वापस False
देदो False
```

Q.2 Extract all Fees amount from data sceince below sentence along with currency.
Output should be,

two $

500 €

Fees Amount = "X gave fees two $ to Departemt for Mtech, Y gave 500 € to Departemt for ME"

```
[ ] Start coding or generate with AI.
```

Stemming in NLTK

```
[97] from nltk.stem import PorterStemmer
     stemmer = PorterStemmer()
```

```
words = ["eating", "eats", "eat", "ate", "adjustable", "rafting", "ability", "meeting"]
```

```
for word in words:
    print(word, "|", stemmer.stem(word))
```

```
eating | eat
eats | eat
eat | eat
ate | ate
adjustable | adjust
rafting | raft
ability | abil
meeting | meet
```

Lemmatization in Spacy

[99]
```
nlp = spacy.load("en_core_web_sm")

doc = nlp("Mando talked for 3 hours although talking isn't his thing")
doc = nlp("eating eats eat ate adjustable rafting ability meeting better")
for token in doc:
    print(token, " | ", token.lemma_)
```

```
eating  |  eat
eats  |  eat
eat  |  eat
ate  |  eat
adjustable  |  adjustable
rafting  |  raft
ability  |  ability
meeting  |  meeting
better  |  well
```

Customizing lemmatizer

[100]
```
nlp.pipe_names
```

```
['tok2vec', 'tagger', 'parser', 'attribute_ruler', 'lemmatizer', 'ner']
```

[101]
```
doc[6]
```

```
ability
```

Q.3 Convert the given text into it's base form using both stemming and lemmatization

text = """Latha is very multi talented girl.She is good at many skills like dancing, running, singing, playing.She also likes eating Pav Bhagi. she has a habit of fishing and swimming too.Besides all this, she is a wonderful at cooking too."""

```
Start coding or generate with AI.
```