**Q.1 What are different frameworks used for NLP in Python?**

* Python ecosystem is diverse and supports a wide variety of libraries, frameworks, and for natural language processing and text analytics.
* Each **framework** contains various **methods, capabilities, and features** for
* operating on text,
* getting insights, and making the data ready for further analysis, like applying machine learning algorithms on pre-processed textual data.

1. **The Natural Language Toolkit(nltk) frame work**

* It is a complete platform that contains over 50 corpora and lexical resources, such as WordNet.
* it also provides the necessary tools, interfaces, and methods to process and analyse text data
* It comes with a suite of efficient modules for classification, tokenization, stemming, lemmatization, tagging, parsing, and semantic reasoning.
* It is widely used standard platform for any NLP project in the industry.

1. **spacy:**
   * It is a free, open-source library for NLP written in efficient, memory-managed Cython.
   * It provides industrial-strength natural language processing capabilities.
   * It is used for large-scale information extraction tasks.
   * It is fastest and works seamlessly with deep learning and machine learning frameworks like TensorFlow, PyTorch, Scikit-Learn, Gensim.
   * The best part is that spaCy has support for several languages and provides pretrained word vectors!
2. **gensim:** 
   * The gensim library has a rich set of capabilities for semantic analysis, including topic modeling and similarity analysis.
   * It also contains a Python port of Google's popular Word2Vec model, which is a neural network model implemented to learn distributed representations of words where similar words (semantic) occur close to each other.
   * Gensim can be used for semantic analysis as well as feature engineering!
3. **textblob:**

* This is another library that provides several capabilities, including text processing, phrase extraction, classification, POS tagging, text translation, and sentiment analysis.
* TextBlob makes a language translation and sentiment analysis very easy, by its extremely intuitive and easy to use API.

1. **Other frame works and libraries:** 
   * Scikit-Learn, NumPy, and SciPy stack, which are extremely useful for text feature engineering, handling feature sets in the form of matrices, and even performing popular machine learning tasks like similarity computation, text classification, and clustering.
   * deep learning and tensor-based libraries like PyTorch, TensorFlow, and Keras are also used to build advanced deep learning models based on deep neural nets, convnets, sequential, and generative models.

Python programs demonstrating fundamental concepts in Natural Language Processing (NLP) :

1. Phonology – Pronunciation and rhyming words
2. Morphology – Word structures (Lemmatization, POS tagging)
3. Syntax – Sentence structure and dependency parsing
4. Semantics – Word meanings and relationships
5. Pragmatics – Context-driven language interpretation

**1. Phonology (Study of Sound Patterns - Concepts Covered: Phonemes, rhyming words.)**

* Phonology deals with how sounds are structured in a language.
* We use pronouncing (for English phonetics) to extract phonemes and rhyming words.

import pronouncing *# Get phonemes for a word*

word = "language" phonemes = pronouncing.phones\_for\_word(word)

print(f"Phonemes for '{word}': {phonemes}") *# Find rhyming words*

rhymes = pronouncing.rhymes(word)

print(f"Words that rhyme with '{word}': {rhymes[:10]}")

**2. Morphology (Study of Word Formation - Tokenization, Lemmatization, POS tagging, Morphological analysis.)**

* Morphology involves breaking words into morphemes (smallest units of meaning).
* We use spacy for tokenization and lemmatization.

import spacy

nlp = spacy.load("en\_core\_web\_sm") *# Analyze morphology of a sentence*

sentence = "The cats are playing happily."

doc = nlp(sentence)

print("Token | Lemma | POS | Morphology")

print("-" \* 40)

for token in doc: print(f"{token.text:10} | {token.lemma\_:10} | {token.pos\_:6} | {token.morph}")

Understanding the Code

1. token.text: The actual word (token) in the sentence.
2. token.lemma\_: The base form (lemma) of the word.
3. token.pos\_: The part of speech (POS) category (e.g., noun, verb, adjective).
4. token.morph: Morphological features (e.g., tense, number, case, gender).

**Expected Output**

Token | Lemma | POS | Morphology

--------------------------------------------------------------------------------

The | the | DET | Definite=Def | PronType=Art

cats | cat | NOUN | Number=Plur

are | be | AUX | Mood=Ind |T ense=Pres | VerbForm=Fin

playing | play | VERB | Tense=Pres | VerbForm=Part

happily | happily | ADV |

. | . | PUNCT |

**Breaking Down the Output**

| **Token** | **Lemma** | **POS** | **Morphology** |
| --- | --- | --- | --- |
| **The** | the | DET (Determiner) | Definite article |
| **cats** | cat | NOUN | Plural |
| **are** | be | AUX (Auxiliary verb) | Present tense |
| **playing** | play | VERB | Present participle |
| **happily** | happily | ADV (Adverb) | No extra morphology |
| **.** | . | PUNCT (Punctuation) | No extra morphology |

**Explanation of Each Column**

1. **Token**: The word as it appears in the sentence.
2. **Lemma**: The base/dictionary form of the word.
   * "cats" → "cat"
   * "are" → "be"
   * "playing" → "play"
3. **POS**: Part of speech (POS) categorization.
   * DET (Determiner)
   * NOUN (Noun)
   * AUX (Auxiliary verb)
   * VERB (Verb)
   * ADV (Adverb)
   * PUNCT (Punctuation)
4. **Morphology**: Additional linguistic features.
   * Number=Plur → Indicates the noun is **plural**.
   * Tense=Pres → Indicates **present tense**.
   * VerbForm=Part → Indicates it’s a **participle form**.

**3. Syntax (Sentence Structure & Grammar – Sentence structure, dependency parsing.)**

**Syntax focuses on how words are arranged to form grammatically correct sentences. The program below uses dependency parsing with spacy.**

import spacy

from spacy import displacy

nlp = spacy.load("en\_core\_web\_sm")

sentence = "The quick brown fox jumps over the lazy dog."

doc = nlp(sentence) *# Visualize syntax tree*

displacy.serve(doc, style="dep")

**4. Semantics (Meaning of Words & Sentences – Word embeddings, semantic similarity.)**

**Semantics involves understanding word meanings. This program calculates word similarity using word embeddings (Word2Vec).**

import gensim.downloader as api *# Load a pre-trained Word2Vec model*

model = api.load("glove-wiki-gigaword-50")

word1 = "king"

word2 = "queen"

similarity = model.similarity(word1, word2)

print(f"Semantic similarity between '{word1}' and '{word2}': {similarity:.4f}")

**5. Pragmatics (Context & Implicature – Context-based analysis, sentiment analysis.)**

**Pragmatics deals with meaning in context. The program below classifies sentences based on sentiment to show contextual meaning.**

from textblob import TextBlob

sentences = [ "I love this movie!", "The food was terrible.", "It's raining outside, take an umbrella." ]

for sentence in sentences:

sentiment = TextBlob(sentence).sentiment.polarity

context = "Positive" if sentiment > 0 else "Negative" if sentiment < 0 else "Neutral"

print(f"Sentence: '{sentence}' | Sentiment: {context}")