Natural Language Toolkit(nltk)

```
#Tokenizer and Stemmer
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.stem import PorterStemmer
text = "Prof. Sasikala teaches Python."
print(word_tokenize(text))      # ['Prof.', 'Sasikala', 'teaches', 'Python', '.', ...]
stemmer = PorterStemmer()
print(stemmer.stem("teaching"))   # teach
print(stemmer.stem("studies"))    # studi
#Lemmatizer
from nltk.stem import WordNetLemmatizer
import nltk
nltk.download('wordnet')
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("studies", pos="v"))  # study
# PoS Tagging
nltk.download('averaged_perceptron_tagger')
tokens = word_tokenize("Engineer designs wearable devices.")
print(nltk.pos_tag(tokens))
# [('Engineer', 'NNP'), ('designs', 'VBZ'), ('wearable', 'JJ'), ('devices', 'NNS'), ('.', '.')]
#Named Entity Recognition
nltk.download('maxent_ne_chunker')
nltk.download('words')
tagged = nltk.pos_tag(tokens)
entities = nltk.chunk.ne_chunk(tagged)
print(entities)
#Accessing Corpora
from nltk.corpus import wordnet
```

```
syns = wordnet.synsets("teach")

print(syns[0].definition())  # impart skills or knowledge to
```

```
import spacy
# Load SpaCy's English model
nlp = spacy.load("en_core_web_sm")
# Sample text
text = """Prof. Sasikala visited Chennai, then flew to New York for a conference.
She also stopped by Paris and Tokyo before returning to India."""

# Process the text
doc = nlp(text)

# Extract places directly
places = [ent.text for ent in doc.ents if ent.label_ in ("GPE", "LOC")]

print("Identified Places:", set(places))
```

A simple named entity recognition for the identification of places

```
import nltk


# Download necessary resources
nltk.download('punkt')
nltk.download('maxent_ne_chunker')
nltk.download('words')
nltk.download('averaged_perceptron_tagger')


# Sample document
text = """Professor visited Chennai, then flew to New York for a conference.
He also stopped by Paris and Tokyo before returning to India."""


# Step 1: Tokenize and tag
sentences = nltk.sent_tokenize(text)
tokenized_sentences = [nltk.word_tokenize(sent) for sent in sentences]
tagged_sentences = [nltk.pos_tag(sent) for sent in tokenized_sentences]


# Step 2: Named Entity Chunking
chunked_sentences = nltk.ne_chunk_sents(tagged_sentences, binary=False)


# Step 3: Extract location entities
def extract_places(tree):
    places = []
    for subtree in tree:
        if hasattr(subtree, 'label') and subtree.label() in ('GPE', 'LOCATION'):
            place = " ".join([leaf[0] for leaf in subtree.leaves()])
            places.append(place)
    return places


# Step 4: Collect all places
all_places = []
for tree in chunked_sentences:
    all_places.extend(extract_places(tree))


print("Identified Places:", set(all_places))
```