```
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

documents = [
    "Natural language processing is a field of study in artificial intelligence .",
    "NLP techniques are used in various applications like machine translation and sentiment
    "The development of NLP tools and libraries has made text analysis easier.",

q]
query = "NLP techniques are used in various applications like machine translation and senti
tfidf_vectorizer = TfidfVectorizer()
tfidf_matrix = tfidf_vectorizer.fit_transform(documents)
query_tfidf = tfidf_vectorizer.transform([query])
cosine_similarities = cosine_similarity(query_tfidf, tfidf_matrix).flatten()
document_ranks = sorted(range(len(cosine_similarities)), key=lambda i: cosine_similarities[
print("Ranked Documents:")
for rank, index in enumerate(document_ranks):
    print(f"Rank {rank + 1}: {documents[index]}")
```

```
Ranked Documents:
Rank 1: NLP techniques are used in various applications like machine translation
Rank 2: The development of NLP tools and libraries has made text analysis easier
Rank 3: Natural language processing is a field of study in artificial intelligen
```

```
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import wordNetLemmatizer
nltk.download('punkt')
text = "The quick brown foxes are jumping over the lazy dogs."
words = nltk.word_tokenize(text)
stemmer = PorterStemmer()
stemmed_words = [stemmer.stem(word) for word in words]
print("Original Words:", words)
print("Stemmed Words:", stemmed_words)
```

```
Original Words: ['The', 'quick', 'brown', 'foxes', 'are', 'jumping', 'over', 'th
Stemmed Words: ['the', 'quick', 'brown', 'fox', 'are', 'jump', 'over', 'the', 'l
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

```
from transformers import MarianMTModel , MarianTokenizer
def translate_text(text , target_lang="ar"):
 model_name = f'Helsinki-NLP/opus-mt-en-{target_lang}'
 model = MarianMTModel.from_pretrained(model_name)
 tokenizer = MarianTokenizer.from_pretrained(model_name)
 inputs = tokenizer(text, return_tensors="pt",padding = True ,Truncate = True)
 outputs = model.generate(**inputs)
 translated_text = tokenizer.decode(outputs[0],skip_special_tokens=True)
return translated_text
if __name__ =="__main__":
  english_text = "Harsha"
  german_text=translate_text(english_text)
  print("translate Text :")
  print(german_text)
```

```
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:88: User
The secret `HF_TOKEN` does not exist in your Colab secrets.
To authenticate with the Hugging Face Hub, create a token in your settings tab (
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access publ
  warnings.warn(
config.json: 100%                                        1.39k/1.39k [00:00<00:00, 51.7kB/s]
pytorch_model.bin: 100%                                    308M/308M [00:10<00:00, 28.7MB/s]
/usr/local/lib/python3.10/dist-packages/torch/_utils.py:831: UserWarning: TypedS
  return self.fget.__get__(instance, owner)()
generation_config.json: 100%                                 293/293 [00:00<00:00, 4.68kB/s]
tokenizer_config.json: 100%                                  44.0/44.0 [00:00<00:00, 967B/s]
source.spm: 100%                                         801k/801k [00:00<00:00, 19.7MB/s]
target.spm: 100%                                         917k/917k [00:00<00:00, 10.3MB/s]
vocab.json: 100%                                         2.12M/2.12M [00:00<00:00, 34.0MB/s]
/usr/local/lib/python3.10/dist-packages/transformers/models/marian/tokenization_
  warnings.warn("Recommended: pip install sacremoses.")
Keyword arguments {'Truncate': True} not recognized.
translate Text :
```

```
pos_tags=[]
for word in text.split():
  tagged = False
  for pattern , pos_tag in pattern:
    if re.match(pattern,word,re.I):
      pos_tags.append((word,pos_tag))
      tagged = True
      break
    if not tagged:
      pos_tags.append((word , 'Unknown'))
    for words , pos_tag in pos_tags:
      print(f"{word}:{pos_tag}")
```