```
import nltk
nltk.download('wordnet')
nltk.download('omw-1.4')

import spacy
from nltk.corpus import wordnet

nlp = spacy.load("en_core_web_sm")

def extract_noun_phrases_meaning(text):
    doc = nlp(text)
    noun_phrases = [chunk.text for chunk in doc.noun_chunks]

    meanings = {}
    for phrase in noun_phrases:
        synsets = wordnet.synsets(phrase)
        if synsets:
            meanings[phrase] = synsets[0].definition()
        else:
            meanings[phrase] = "No definition found"

    return meanings

text = "The quick brown fox jumps over the lazy dog."
meanings = extract_noun_phrases_meaning(text)
print(meanings)
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data] Downloading package omw-1.4 to /root/nltk_data...
{'The quick brown fox': 'No definition found', 'the lazy dog': 'No definition found'}
```

```
!pip install transformers

from transformers import pipeline

nlp = pipeline("ner", model="dbmdz/bert-large-cased-finetuned-conll03-english")

text = "John bought a car. He loves his new car."
results = nlp(text)
for entity in results:
    print(entity)
```

```
Requirement already satisfied: transformers in /usr/local/lib/python3.10/dist-packages (4.44.2)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from transformers)
Requirement already satisfied: huggingface-hub<1.0,>=0.23.2 in /usr/local/lib/python3.10/dist-packages
Requirement already satisfied: numpy>=1.17 in /usr/local/lib/python3.10/dist-packages (from transformer
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.10/dist-packages (from transfo
Requirement already satisfied: pyyaml>=5.1 in /usr/local/lib/python3.10/dist-packages (from transformer
Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.10/dist-packages (from trans
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from transformers)
Requirement already satisfied: safetensors>=0.4.1 in /usr/local/lib/python3.10/dist-packages (from tran
Requirement already satisfied: tokenizers<0.20,>=0.19 in /usr/local/lib/python3.10/dist-packages (from
Requirement already satisfied: tqdm>=4.27 in /usr/local/lib/python3.10/dist-packages (from transformers
Requirement already satisfied: fsspec>=2023.5.0 in /usr/local/lib/python3.10/dist-packages (from huggin
Requirement already satisfied: typing-extensions>=3.7.4.3 in /usr/local/lib/python3.10/dist-packages (f
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (fro
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requ
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requ
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_token.py:89: UserWarning:
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 (https://huggingface.co/
You will be able to reuse this secret in all of your notebooks.
Please note that authentication is recommended but still optional to access public models or datasets.
  warnings.warn(
config.json: 100%                                   998/998 [00:00<00:00, 51.8kB/s]
model.safetensors: 100%                             1.33G/1.33G [00:17<00:00, 112MB/s]
Some weights of the model checkpoint at dbmdz/bert-large-cased-finetuned-conll03-english were not used
- This IS expected if you are initializing BertForTokenClassification from the checkpoint of a model tr
- This IS NOT expected if you are initializing BertForTokenClassification from the checkpoint of a mode
tokenizer_config.json: 100%                         60.0/60.0 [00:00<00:00, 1.78kB/s]
vocab.txt: 100%                                     213k/213k [00:00<00:00, 2.90MB/s]
/usr/local/lib/python3.10/dist-packages/transformers/tokenization_utils_base.py:1601: FutureWarning: `c
  warnings.warn(
{'entity': 'I-PER', 'score': 0.98807204, 'index': 1, 'word': 'John', 'start': 0, 'end': 4}
```

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

def evaluate_coherence(text):
    sentences = text.split(". ")
    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(sentences)

    coherence_scores = []
    for i in range(len(sentences) - 1):
        score = cosine_similarity(tfidf_matrix[i], tfidf_matrix[i + 1])[0][0]
        coherence_scores.append(score)

    avg_coherence = sum(coherence_scores) / len(coherence_scores) if coherence_scores else 0
    return avg_coherence

text = "The weather was cold. Snow covered the streets. People wore warm coats."
coherence_score = evaluate_coherence(text)
print("Coherence Score:", coherence_score)
```

```
Coherence Score: 0.08081817896262673
```

```
def recognize_dialog_act(utterance):
    if utterance.lower().startswith(("how", "what", "why", "where", "who", "when")):
        return "Question"
    elif any(word in utterance.lower() for word in ["yes", "no", "sure", "okay"]):
        return "Answer/Agreement"
    elif utterance.lower().startswith(("please", "could you", "would you")):
        return "Request"
    elif utterance.lower().startswith("thank"):
        return "Thanking"
    else:
        return "Statement"

dialog = ["What time is it?", "It's 3 PM.", "Thanks!", "Could you help me?"]
for utterance in dialog:
    act = recognize_dialog_act(utterance)
    print(f"Utterance: '{utterance}' -> Dialog Act: {act}")
```

```
Utterance: 'What time is it?' -> Dialog Act: Question
Utterance: 'It's 3 PM.' -> Dialog Act: Statement
Utterance: 'Thanks!' -> Dialog Act: Thanking
Utterance: 'Could you help me?' -> Dialog Act: Request
```

```
import openai

openai.api_key = "sk-proj-QkCCim66-7VqQq5nTMHn_zH6A2jB-qQwcD77ZZgVVsr4xOEJUJl3U0Nl_RIgUrYI9H1I4-ng2kT3BlbkFJ

def generate_text(prompt):
    response = openai.Completion.create(
        engine="text-davinci-003",
        prompt=prompt,
        max_tokens=50
    )
    return response.choices[0].text.strip()

prompt = "Once upon a time in a faraway land"
generated_text = generate_text(prompt)
print(generated_text)
```

```
---------------------------------------------------------------------------
APIRemovedInV1                            Traceback (most recent call last)
<ipython-input-10-b2481e14e7c0> in <cell line: 17>()
     15 # Example usage
     16 prompt = "Once upon a time in a faraway land"
---> 17 generated_text = generate_text(prompt)
     18 print(generated_text)
```

Next steps: Explain error

1 frames

```
from transformers import pipeline

translator = pipeline("translation_en_to_fr")

def translate_to_french(text):
    translation = translator(text, max_length=50)
    return translation[0]['translation_text']

english_text = "Hello, how are you?"
french_text = translate_to_french(english_text)
print(french_text)
```

```
No model was supplied, defaulted to google-t5/t5-base and revision 686f1db (https://huggingface.co/goog
Using a pipeline without specifying a model name and revision in production is not recommended.
config.json: 100%                                    1.21k/1.21k [00:00<00:00, 19.3kB/s]
model.safetensors: 100%                              892M/892M [00:12<00:00, 151MB/s]
generation_config.json: 100%                         147/147 [00:00<00:00, 8.06kB/s]
spiece.model: 100%                                   792k/792k [00:00<00:00, 11.1MB/s]
tokenizer.json: 100%                                 1.39M/1.39M [00:00<00:00, 4.42MB/s]
Bonjour, comment êtes-vous?
```