```
import re
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
from nltk.corpus import stopwords
from nltk import pos_tag, ne_chunk
from nltk.stem import WordNetLemmatizer
from autocorrect import Speller

pip install autocorrect

Collecting autocorrect
  Downloading autocorrect-2.6.1.tar.gz (622 kB)
     ---------------------------------------- 0.0/622.8 kB ? eta
-:--:--
     - -------------------------------- 30.7/622.8 kB 435.7 kB/s eta
0:00:02
     ---------------- ----------------- 297.0/622.8 kB 3.1 MB/s eta
0:00:01
     ---------------------------------------- 622.8/622.8 kB 3.9 MB/s eta
0:00:00
  Preparing metadata (setup.py): started
  Preparing metadata (setup.py): finished with status 'done'
Building wheels for collected packages: autocorrect
  Building wheel for autocorrect (setup.py): started
  Building wheel for autocorrect (setup.py): finished with status
'done'
  Created wheel for autocorrect: filename=autocorrect-2.6.1-py3-none-
any.whl size=622375
sha256=13bf28041b15bffb5ca4e7e2555be1cc9fcd40e08a48536bdf6f9712b4c32da
9
  Stored in directory: c:\users\deep salunkhe\appdata\local\pip\cache\
wheels\5e\90\99\807a5ad861ce5d22c3c299a11df8cba9f31524f23ae6e645cb
Successfully built autocorrect
Installing collected packages: autocorrect
Successfully installed autocorrect-2.6.1
Note: you may need to restart the kernel to use updated packages.

nltk.download('punkt')  # For tokenization
nltk.download('stopwords')  # For stop word removal
nltk.download('averaged_perceptron_tagger')  # For POS tagging
nltk.download('maxent_ne_chunker')  # For Named Entity Recognition
nltk.download('words')  # Required for NER
nltk.download('wordnet')  # For lemmatization

[nltk_data] Downloading package punkt to C:\Users\Deep
[nltk_data]     Salunkhe\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping tokenizers\punkt.zip.
[nltk_data] Downloading package stopwords to C:\Users\Deep
[nltk_data]     Salunkhe\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\Deep Salunkhe\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping taggers\averaged_perceptron_tagger.zip.
[nltk_data] Downloading package maxent_ne_chunker to C:\Users\Deep
[nltk_data]     Salunkhe\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping chunkers\maxent_ne_chunker.zip.
[nltk_data] Downloading package words to C:\Users\Deep
[nltk_data]     Salunkhe\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\words.zip.
[nltk_data] Downloading package wordnet to C:\Users\Deep
[nltk_data]     Salunkhe\AppData\Roaming\nltk_data...
```

```
True
```

```
def preprocess_corpus(corpus):
    # Sentence segmentation: Split the corpus into individual
sentences
    sentences = sent_tokenize(corpus)

    processed_sentences = []

    for sentence in sentences:
        # Lowercasing: Convert all text to lowercase
        sentence = sentence.lower()

        # Number handling: Remove all digits
        # Note: This simple approach removes all numbers. Modify as
needed.
        sentence = re.sub(r'\d+', '', sentence)

        # Tokenization: Split sentence into individual words
        tokens = word_tokenize(sentence)

        # Stop word removal: Remove common words that don't carry much
meaning
        stop_words = set(stopwords.words('english'))
        tokens = [token for token in tokens if token not in
stop_words]

        # Spelling correction: Correct misspelled words
        # Note: This uses the autocorrect library and may need
adjustments
        spell = Speller()
        tokens = [spell(token) for token in tokens]

        # Text normalization: Lemmatization
        # Convert words to their base or dictionary form
        lemmatizer = WordNetLemmatizer()
        tokens = [lemmatizer.lemmatize(token) for token in tokens]
```

```
        # Named Entity Recognition (NER)
        # First, perform Part-of-Speech (POS) tagging
        pos_tags = pos_tag(tokens)
        # Then, identify named entities
        named_entities = ne_chunk(pos_tags)

        # Reconstruct the sentence from processed tokens
        processed_sentence = ' '.join(tokens)
        processed_sentences.append(processed_sentence)

    # Join all processed sentences back into a single text
    processed_corpus = ' '.join(processed_sentences)

    # Return both the processed corpus and the named entities
    return processed_corpus, named_entities

corpus = """
Lorem Ipsum is simply dummy text of the printing and typesetting
industry. Lorem Ipsum has been the industry's standard dummy text ever
since the 1500s, when an unknown printer took a galley of type and
scrambled it to make a type specimen book. It has survived not only
five centuries, but also the leap into electronic typesetting,
remaining essentially unchanged. It was popularised in the 1960s with
the release of Letraset sheets containing Lorem Ipsum passages, and
more recently with desktop publishing software like Aldus PageMaker
including versions of Lorem Ipsum.
"""

# Apply preprocessing to the example corpus
processed_corpus, named_entities = preprocess_corpus(corpus)

print("Processed Corpus:")
print(processed_corpus)

print("\nNamed Entities:")
print(named_entities)

# Note: The output will show the processed text with lowercased words,
# removed stop words and numbers, corrected spellings, and lemmatized
words.
# Named entities will be displayed separately in a tree structure
```

```
Processed Corpus:
lore ipsum simply dummy text printing typesetting industry . lore
ipsum industry 's standard dummy text ever since , unknown printer
took galley type scrambled make type specimen book . survived five
century , also leap electronic typesetting , remaining essentially
unchanged . popularised release letraset sheet containing lore ipsum
passage , recently desktop publishing software like album pacemaker
including version lore ipsum .
```

```
Named Entities:
(S
  popularised/JJ
  release/NN
  letraset/JJ
  sheet/NN
  containing/VBG
  lore/JJR
  ipsum/JJ
  passage/NN
  ,/,
  recently/RB
  desktop/VBD
  publishing/NN
  software/NN
  like/IN
  album/NN
  pacemaker/NN
  including/VBG
  version/NN
  lore/RB
  ipsum/NN
  ./.)
```