Day63_NLP_P1_Natural_Language_Understanding_NLU_(ALL_IN_1)

August 13, 2025

1 NLU – Text Preprocessing Quick Reference

What is NLP?

Natural Language Processing (NLP) is a field of AI that enables computers to understand, interpret, and generate human language.

Two main parts of NLP:

1. **NLU – Natural Language Understanding**
   - Focuses on understanding meaning from text/speech.
   - Example tasks: sentiment analysis, entity recognition, question answering.
2. **NLG – Natural Language Generation**
   - Focuses on producing human-like language.
   - Example tasks: chatbots replying to users, AI writing articles.

This notebook covers **NLU** – the first part of NLP.

2 Tokenization

2.1 Word Tokenization

- Splits text into individual words and punctuation.
- **When to use:** Any NLP pipeline as the first step.

[1]:
```
from nltk.tokenize import word_tokenize
text = "Natural Language Understanding helps AI systems interpret human␣
 ↪language."
word_tokenize(text)
```

[1]:
```
['Natural',
 'Language',
 'Understanding',
 'helps',
 'AI',
 'systems',
 'interpret',
 'human',
```

```
 'language',
 '.']
```

2.2 Sentence Tokenization

- Splits text into sentences.
- **When to use:** Summarization, dialogue systems.

[2]:
```
from nltk.tokenize import sent_tokenize
sent_tokenize(text)
```

[2]:
```
['Natural Language Understanding helps AI systems interpret human language.']
```

2.3 Paragraph Tokenization

- Splits text into paragraphs by blank lines.
- **When to use:** Document-level analysis.

[3]:
```
from nltk.tokenize import blankline_tokenize
blankline_tokenize(text)
```

[3]:
```
['Natural Language Understanding helps AI systems interpret human language.']
```

2.4 Whitespace Tokenizer

- Splits by spaces/tabs, punctuation remains.
- **When to use:** Emails, URLs, structured text.

[4]:
```
from nltk.tokenize import WhitespaceTokenizer
WhitespaceTokenizer().tokenize(text)
```

[4]:
```
['Natural',
 'Language',
 'Understanding',
 'helps',
 'AI',
 'systems',
 'interpret',
 'human',
 'language.']
```

2.5 WordPunct Tokenizer

- Splits words and punctuation separately.
- **When to use:** Emoticons, hashtags, code parsing.

[5]:
```
from nltk.tokenize import wordpunct_tokenize
wordpunct_tokenize(text)
```

[5]:
```
['Natural',
 'Language',
 'Understanding',
 'helps',
 'AI',
 'systems',
 'interpret',
 'human',
 'language',
 '.']
```

3 N-grams

- Groups words into sequences of N.
- **When to use:** Text prediction, context analysis.

[6]:
```
import nltk
tokens = nltk.word_tokenize(text)
```

3.1 Bigrams

[7]:
```
print(list(nltk.bigrams(tokens)))   # Bigrams
```

```
[('Natural', 'Language'), ('Language', 'Understanding'), ('Understanding',
'helps'), ('helps', 'AI'), ('AI', 'systems'), ('systems', 'interpret'),
('interpret', 'human'), ('human', 'language'), ('language', '.')]
```

3.2 Trigrams

[8]:
```
print(list(nltk.trigrams(tokens)))  # Trigrams
```

```
[('Natural', 'Language', 'Understanding'), ('Language', 'Understanding',
'helps'), ('Understanding', 'helps', 'AI'), ('helps', 'AI', 'systems'), ('AI',
'systems', 'interpret'), ('systems', 'interpret', 'human'), ('interpret',
'human', 'language'), ('human', 'language', '.')]
```

3.3 4-grams

[9]:
```
print(list(nltk.ngrams(tokens, 4))) # 4-grams
```

```
[('Natural', 'Language', 'Understanding', 'helps'), ('Language',
'Understanding', 'helps', 'AI'), ('Understanding', 'helps', 'AI', 'systems'),
('helps', 'AI', 'systems', 'interpret'), ('AI', 'systems', 'interpret',
'human'), ('systems', 'interpret', 'human', 'language'), ('interpret', 'human',
'language', '.')]
```

4 Stemming

- Reduces words to root form (may not be real words).
- **When to use:** Fast processing, search engines.

[10]:
```
from nltk.stem import PorterStemmer, LancasterStemmer, SnowballStemmer

words = ['running', 'runs', 'easily', 'fairly']
```

4.1 PorterStemmer

[11]:
```
print([PorterStemmer().stem(w) for w in words])
```

```
['run', 'run', 'easili', 'fairli']
```

4.2 LancasterStemmer

[12]:
```
print([LancasterStemmer().stem(w) for w in words])
```

```
['run', 'run', 'easy', 'fair']
```

4.3 SnowballStemmer

[13]:
```
print([SnowballStemmer('english').stem(w) for w in words])
```

```
['run', 'run', 'easili', 'fair']
```

5 Lemmatization

- Reduces words to their dictionary form, considering meaning.
- **When to use:** Accurate text analysis (sentiment, topic modeling).

[14]:
```
from nltk.stem import WordNetLemmatizer
wnl = WordNetLemmatizer()
wnl.lemmatize('running', pos='v')
```

[14]:
```
'run'
```

6 Stopwords Removal

- Removes common words that add little meaning.
- **When to use:** Text classification, clustering (if stopwords are not important).

[15]:
```
from nltk.corpus import stopwords
print(stopwords.words('english')[:10])
print("Stopwords count:", len(stopwords.words('english')))
```

```
['a', 'about', 'above', 'after', 'again', 'against', 'ain', 'all', 'am', 'an']
Stopwords count: 198
```

7 POS Tagging

- Assigns grammatical roles (noun, verb, adjective) to tokens.
- **When to use:** Syntax analysis, NER, grammar correction.

[16]:
```
import nltk
tokens = nltk.word_tokenize(text)
nltk.pos_tag(tokens)
```

[16]:
```
[('Natural', 'JJ'),
 ('Language', 'NNP'),
 ('Understanding', 'NNP'),
 ('helps', 'VBZ'),
 ('AI', 'NNP'),
 ('systems', 'NNS'),
 ('interpret', 'JJ'),
 ('human', 'JJ'),
 ('language', 'NN'),
 ('.', '.')]
```

8 Named Entity Recognition (NER)

- Identifies people, places, organizations, etc.
- **When to use:** Information extraction, knowledge graphs.

[18]:
```
# It will form a Tree
# from nltk import ne_chunk
# tokens = nltk.word_tokenize("The US president stays in the White House.")
# tags = nltk.pos_tag(tokens)
# ne_chunk(tags)
```

[19]:
```
from nltk import ne_chunk
from nltk.tree import Tree

tokens = nltk.word_tokenize("The US president stays in the White House.")
tags = nltk.pos_tag(tokens)
tree = ne_chunk(tags)

# Extract entities as (text, label)
entities = []
for subtree in tree:
    if isinstance(subtree, Tree):
        entity_text = " ".join(token for token, pos in subtree.leaves())
        entities.append((entity_text, subtree.label()))

print(entities)
```

```
[('US', 'GSP'), ('White House', 'FACILITY')]
```

Closing Notes

- Tokenization: First step for all NLP tasks.
- Whitespace vs WordPunct: Choose based on punctuation importance.
- N-grams: Add context but require more data for larger N.
- Stemming vs Lemmatization: Speed vs accuracy trade-off.
- Stopwords: Remove for efficiency unless meaningful in context.
- POS & NER: Provide structural and entity-level understanding of text.