Introduction to Natural Language Processing

Lecture 1: Text Tokenization

Harito

September 15, 2025

What is NLP?

- Natural Language Processing (NLP) is a field of AI that enables computers to understand, interpret, and generate human language.
- It combines computer science, artificial intelligence, and linguistics.
- Two main sub-fields:
  - Natural Language Understanding (NLU)
  - Natural Language Generation (NLG)

The NLP Pipeline

A typical NLP project follows a pipeline of steps:

1. **Text Acquisition**
2. **Text Preprocessing**

The NLP Pipeline

A typical NLP project follows a pipeline of steps:

1. **Text Acquisition**
2. **Text Preprocessing**
3. Feature Engineering
4. Model Building
5. Evaluation
6. Deployment

Today, we focus on a key preprocessing step: Tokenization.

What is Tokenization?

Definition

Tokenization is the process of breaking down a stream of text into smaller units called **tokens**. These tokens can be words, characters, or subwords.

What is Tokenization?

Definition

Tokenization is the process of breaking down a stream of text into smaller units called **tokens**. These tokens can be words, characters, or subwords.

Example:

- *Input Text:* "NLP is fascinating!"
- *Output Tokens:* ‘["NLP", "is", "fascinating", "!"]‘

It is the very first step for many downstream tasks.

Tokenization Strategies

How do we decide where to split the text?

- **Whitespace Tokenization**: Split the text by spaces. It's simple but often insufficient.
  - ‘”Hello, world”‘ -> ‘[”Hello,”, ”world”]‘ (Punctuation is attached!)

Tokenization Strategies

How do we decide where to split the text?

- **Whitespace Tokenization**: Split the text by spaces. It's simple but often insufficient.
  - ‘”Hello, world”‘ -> ‘[”Hello,”, ”world”]‘ (Punctuation is attached!)
- **Punctuation-based Tokenization**: Treat punctuation as separate tokens.
  - ‘”Hello, world”‘ -> ‘[”Hello”, ”,”, ”world”]‘ (Better!)

Tokenization Strategies

How do we decide where to split the text?

- **Whitespace Tokenization**: Split the text by spaces. It's simple but often insufficient.
  - ‘”Hello, world”‘ -> ‘[”Hello,”, ”world”]‘ (Punctuation is attached!)
- **Punctuation-based Tokenization**: Treat punctuation as separate tokens.
  - ‘”Hello, world”‘ -> ‘[”Hello”, ”,”, ”world”]‘ (Better!)
- **Regular Expressions (Regex)**: Use patterns to define what a token is. Very powerful and flexible.

Python Example

Here is a simple implementation using regex.

```
import re

def regex_tokenize(text):
    # Convert to lowercase
    text = text.lower()
    # Regex to find all word characters or any non-whitespace
    # \w+ -> one or more word characters (letters, numbers, _
    # |    -> OR
    # [^\s] -> any character that is not a whitespace
    # For this lab, we can use a simpler one:
    # \w+|[^\w\s]
    tokens = re.findall(r'\w+|[^\w\s]', text)
    return tokens
```

```python
# --- Test ---
sentence = "Let's see how it handles 123 numbers!"
print(regex_tokenize(sentence))

# Output:
# ["let's", 'see', 'how', 'it',
# 'handles', '123', 'numbers', '!']
```

Next Steps

Now, it's time for the lab!

Objective:

- Implement a Tokenizer interface.
- Create your own ‘SimpleTokenizer‘ and ‘RegexTokenizer‘.