16.5.3

# NLP Core Concepts

**You're** starting to piece together why it's so important to have NLP. Now Jennifer has asked you to look into some of the core concepts in NLP so that you have a good foundation to build on.

Before we can write algorithms to help computers understand language information, there are some key concepts and use cases that we need to know: tokenization, normalization, part-of-speech tagging, natural language generation, bag-of-words model, n-grams, and text similarity. This section will cover each of these concepts.

## Tokenization

**Tokenization** is the concept of splitting a document or sentence into small subsets of data that can be analyzed. It's essentially the building block of most NLP use cases. Tokenization can be performed by word or sentence.

To tokenize by word, you take a given sentence or document and split it into a list of words, with each sentence or word representing a token. A sentence tokenized by word would look like the following:

Original sentence: `I am enjoying learning about NLP.`

Tokenized by word: `['I', 'am', 'enjoying', 'learning', 'about', 'NLP', '.']`

To tokenize by sentence, you would provide a document with at least one sentence. Ideally, you would use more than one sentence, as tokenization splits a document up by sentences. Here's an example of a two-sentence document and how it would be split up. Notice how each sentence is a separate string:

Original sentence: `There is a concept in NLP called tokenization. There are two types of tokenization: word and sentence.`

Tokenized by sentence: `['There is a concept in NLP called tokenization.', 'There are two types of tokenization: word and sentence.']`

You can tokenize using NLTK or spaCy libraries, which we'll cover later.

## Normalization

**Normalization** is the concept of taking misspelled words and converting them into their original form. This is another building block in NLP in that it helps get the text to a readable form and allows us to create other use cases on top of it. Essentially, it makes it easier for us to create NLP programs, and it improves the output of those programs. There are numerous ways to accomplish this, but we'll focus on just two practices: stemming and lemmatization. Stemming and lemmatization are similar in that they both remove the suffix from a word, but there are some differences in how smooth or rough the cutoff tends to be:

- **Stemming** removes the suffix from a word and reduces it to its original form. This serves as a “rough” cut off the end of the word. An example of stemming might be to reduce “horses” to “horse” and “ponies” to “poni.” As seen here, the truncated form is not always a real word.
- **Lemmatization** removes the suffix from a word and reduces it to its original form. Lemmatization tends to be a “smoother” cut off the end of the word. It tries to return to the original root word. In contrast to stemming, lemmatization always returns a real word. For example, the word “am” might be lemmatized to “be.” While stemming is a blunt instrument that follows abstract rules regardless of real-world usage, lemmatization performs a similar process but reduces words to their root. Lemmatization accomplishes this by using a lexicon (a specialized dictionary) of words and their variant forms.

## Part-of-Speech Tagging

When researching or practicing NLP, you'll often hear of **part-of-speech tagging** (PoS), which can be helpful for a variety of different models in NLP. PoS tagging is the concept of finding each word's part of speech in a given document, as the example below illustrates:

| | |
|---|---|
| **Word** | **PoS and Tags** |
| I | Personal pronoun (PRP) |
| enjoy | Verb, non-third person (VBP) |
| biking | Verb, gerund, present participle (VBG) |
| on | Preposition or subordinating conjunction (IN) |
| the | Determiner (DT) |
| trails | Noun, plural (NNS) |

If you want to try this, we will do so using the Natural Language Toolkit library. To install follow the steps below.

1. Activate your environment

   `pip install nltk`

2. Install NLTK
3. Install additional NLTK tools

   `python -m nltk.downloader popular`

Go ahead and create a new Python file. You can add this code to get the PoS tags for the text you provide:

```
import nltk
from nltk import word_tokenize
text = word_tokenize("I enjoy biking on the trails")
output = nltk.pos_tag(text)
print(output)
```

When you run this file, it will print out the PoS tags for each word as shown in the table above.

**SKILL DRILL**

Go ahead and run this code with a sentence of your choice.

## Natural Language Generation

**Natural language generation** is a growing field in NLP that entails writing code in such a way that it will generate new text. Popular examples include chatbots, automated custom reports, and custom webpage content. In this module, we won't write code to generate new text.

Chatbots are easy to create and so are increasingly used in data science, especially NLP. Most of the technology already exists to create meaningful content from natural language generation.

## Bag-of-Words

When we're building datasets for NLP, we need to consider how our program will interact with the text document we provide. If we create a **bag-of-words (BoW)** model (i.e., the most frequent words), we can build models from that.

The basic idea behind this model is this: We have a document of words, but we don't care about the order of the words. We can count these words and create models based on how frequently they appear.

## n-gram

In NLP, there is an **n-gram** method, which is a sequence of items from a given text. With n-gram, you create groupings of items from the text of size 1 or more. The following n-grams are common:

- **Unigram** is an n-gram of size 1.
- **Bigram** is an n-gram of size 2.
- **Trigram** is an n-gram of size 3.

For example, a unigram for the sentence “I like pizzas” would be “I,” “like,” and “pizzas.” Its bigram would break up the sentence into sequential two-word chunks: “I like” and “like pizzas.” The trigram would be “I like pizzas.”

Of course, for longer sentences, you can always use larger groupings than just size 3, such as “four-gram,” “five-gram,” and so on.

N-grams can be used for a variety of NLP tasks, but most involve text mining or extraction. You can also use n-grams for spellcheck and text summarization.

## Text Similarity

Another popular use case for NLP is determining document or sentence similarity. These are important use cases, because they can tell us a lot about a document and its contents. There are a number of ways to evaluate text similarity, with varying levels of difficulty. Many of the technologies that we'll discuss have the ability to compare documents against one another.

Retake