November 2023

Natural Language Processing

Objectives

- **Attempt to explain** What NLP is and how this is translated in the NLTK library.
- **Begin** to comprehend the **syntax** and process of the NLP process.
- **Feel more comfortable** engaging with the NLP content on Athena and beyond!

EXPLORE AI
ACADEMY

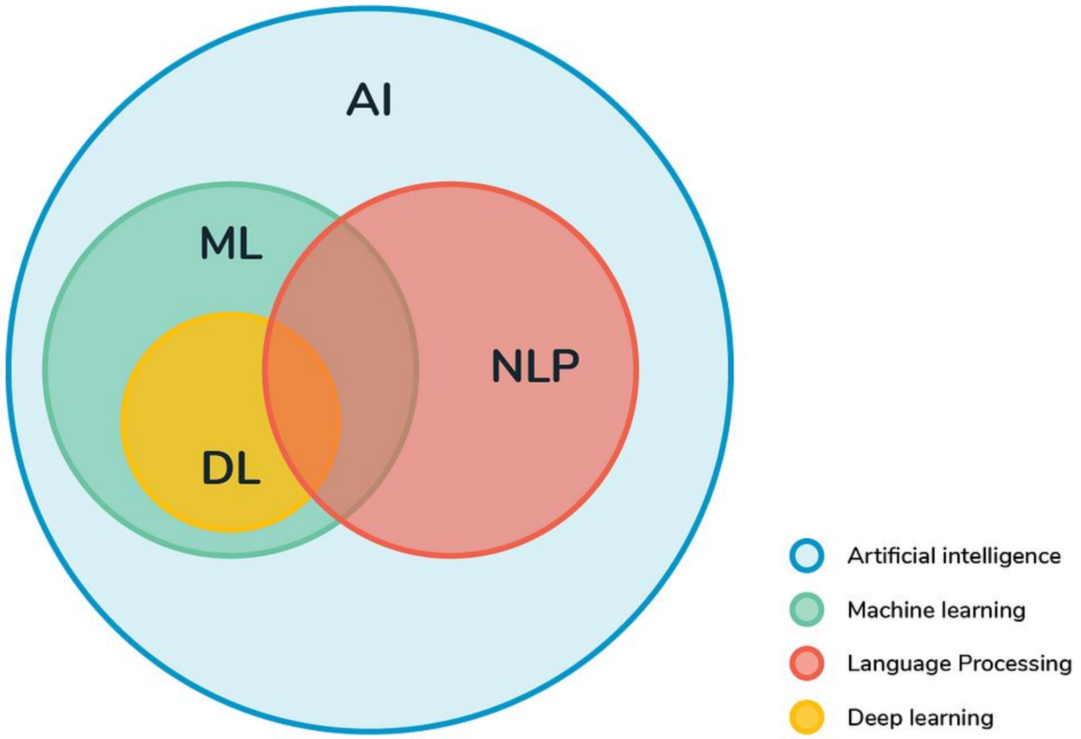
Natural Language Processing

- Natural Language Processing (NLP) is a subfield of artificial intelligence (AI) that focuses on the interaction between computers and human language.

- The primary goal of NLP is to enable machines to understand, interpret, and generate human language in a way that is both meaningful and contextually relevant.

- NLP involves a range of computational techniques and linguistic models to process and analyze natural language data.

Natural Language Processing



EXPLORE AI
ACADEMY

Text Cleaning

- Text cleaning in NLP involves preprocessing raw text data to enhance its quality for analysis.

- This includes **lowercasing**, **tokenization**, **removing special characters**, **punctuation**, and **stop words**, as well as **stemming** and **lemmatization**.

- These steps standardize the text, reduce noise, and facilitate more effective NLP algorithms.

- The aim is to produce a “tidy” dataset that **boosts the accuracy** and **efficiency**. This involves **removing irrelevant details** and **enhancing the meaningful content** in the text.

Removing Noise

Removing Noise

- Removing noise in Natural Language Processing (NLP) involves the process of eliminating irrelevant or unnecessary information from raw text data to enhance the quality and effectiveness of subsequent analyses.

- Removing **special characters** and **punctuation**, and **emails** are some of the examples one can find useful removals.

- A useful tool is the **Regular Expressions Library**

EXPLORE AI ACADEMY

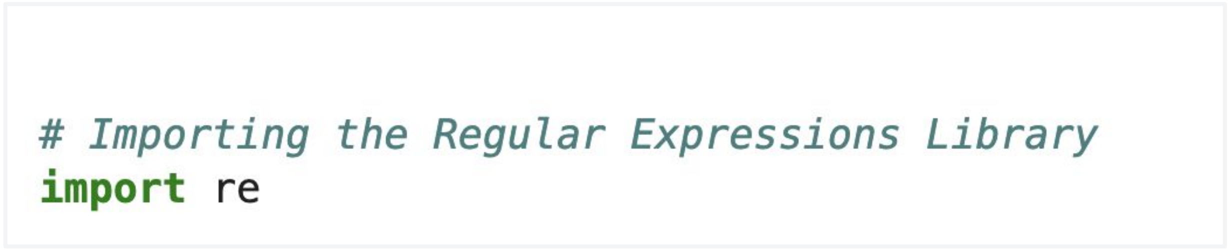
Regular Expression

Regular Expression Library

- A **RegEx**, or **Regular Expression**, is a sequence of characters that forms a search pattern.

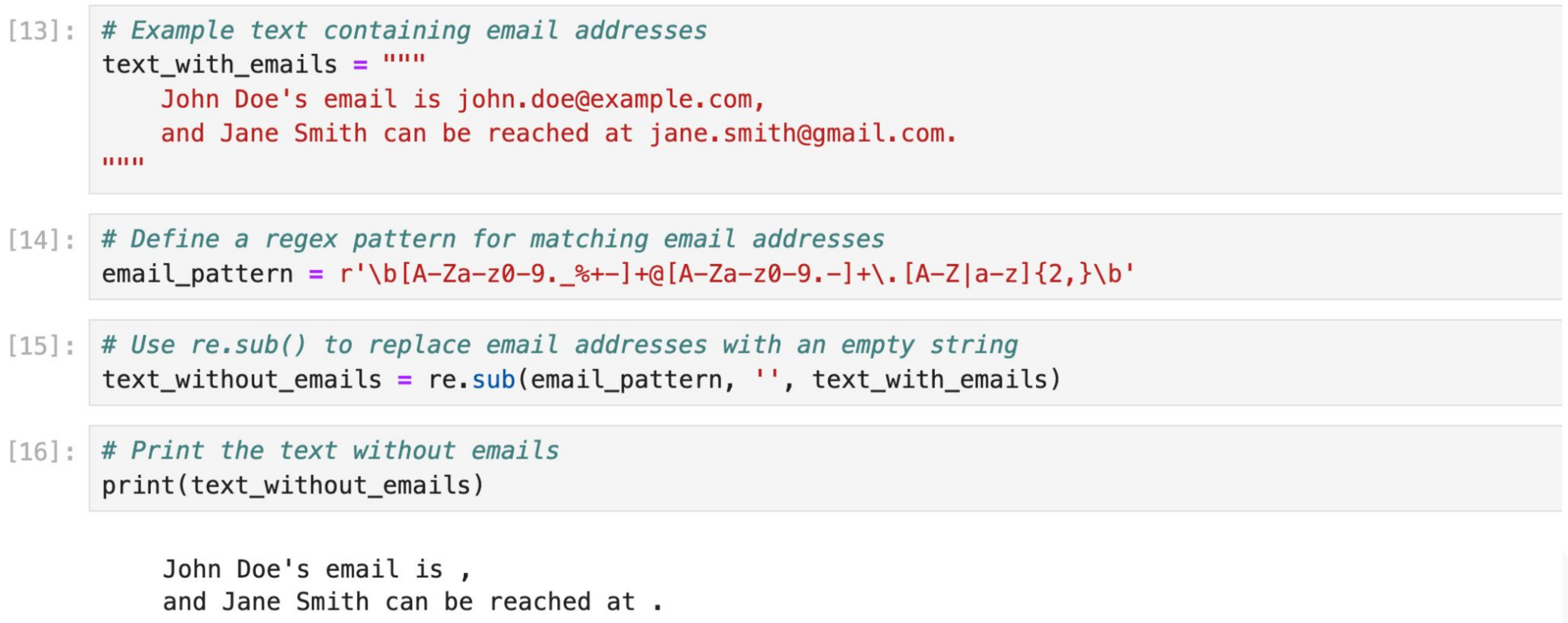
- **RegEx** can be used to check if a string contains the specified search pattern.

- Regular expressions are powerful tools for pattern matching and **string manipulation**.



```python
# Importing the Regular Expressions Library
import re
```

Removing Emails



[13]:
```
# Example text containing email addresses
text_with_emails = """
    John Doe's email is john.doe@example.com,
    and Jane Smith can be reached at jane.smith@gmail.com.
"""
```

[14]:
```
# Define a regex pattern for matching email addresses
email_pattern = r'\b[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\.[A-Z|a-z]{2,}\b'
```

[15]:
```
# Use re.sub() to replace email addresses with an empty string
text_without_emails = re.sub(email_pattern, '', text_with_emails)
```

[16]:
```
# Print the text without emails
print(text_without_emails)
```

```
    John Doe's email is ,
    and Jane Smith can be reached at .
```

Removing Emails

- In this example, the regex pattern **^[a-zA-Z0-9._%+-]+@[a-zA-Z0-9.-]+\.[a-zA-Z]{2,}$** breaks down as follows:

- **^[a-zA-Z0-9._%+-]+**: Start of the line, followed by one or more alphanumeric characters, dots, underscores, percent signs, plus signs, or hyphens.

- **@**: The at symbol.

- **[a-zA-Z0-9.-]+**: One or more alphanumeric characters, dots, or hyphens.

- **\.**: A literal dot.

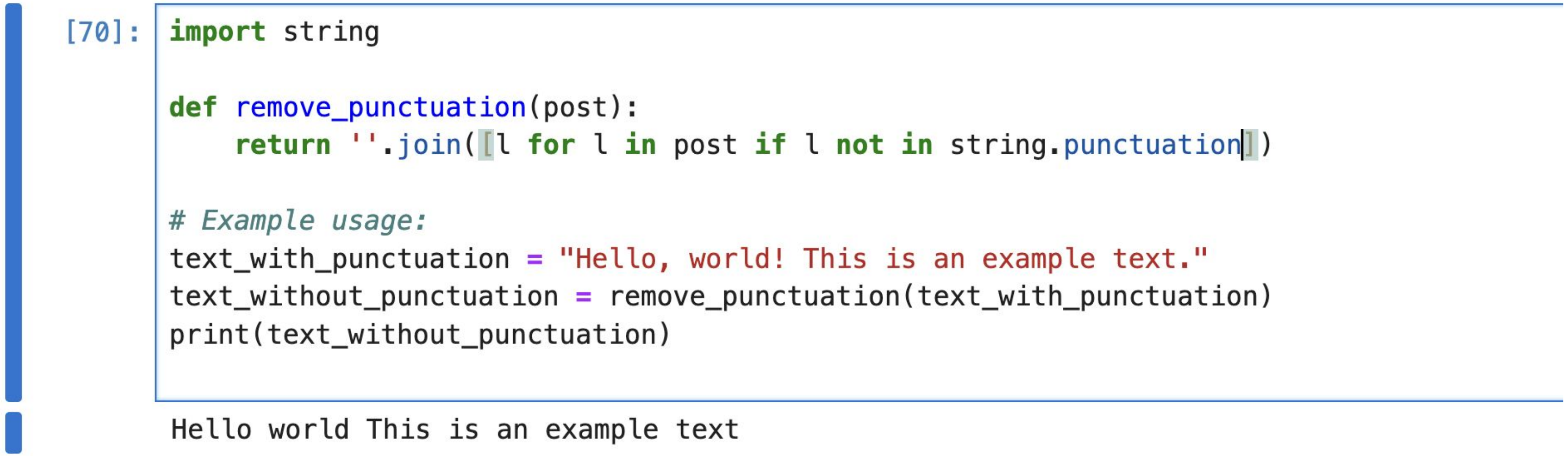
- **[a-zA-Z]{2,}$**: Two or more alphabetic characters at the end of the line.

EXPLORE AI ACADEMY

Removing Punctuations and Special Characters

Removing Punctuations and Special Characters

Removing Punctuations and Special Characters



```
[70]: import string

      def remove_punctuation(post):
          return ''.join([l for l in post if l not in string.punctuation])

      # Example usage:
      text_with_punctuation = "Hello, world! This is an example text."
      text_without_punctuation = remove_punctuation(text_with_punctuation)
      print(text_without_punctuation)

      Hello world This is an example text
```

Tokenization

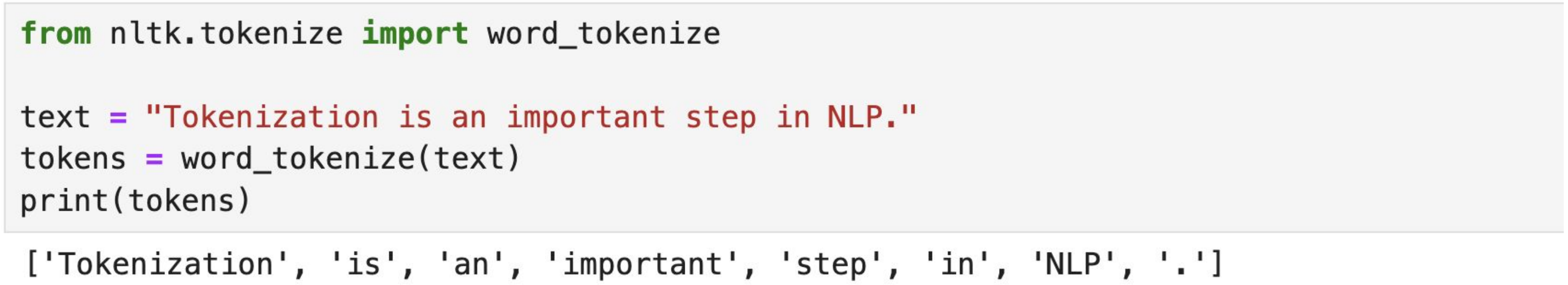
Tokenization

- Tokenization in Natural Language Processing (NLP) is the process of breaking down a text into smaller units, typically words or phrases, referred to as tokens.

- **Tokens** are the building blocks for various NLP tasks, allowing machines to process and analyze human language.

- Tokenization is a crucial **preprocessing step** for tasks such as *text analysis*, *sentiment analysis*, *machine translation*, and more.

Tokenization



```python
from nltk.tokenize import word_tokenize

text = "Tokenization is an important step in NLP."
tokens = word_tokenize(text)
print(tokens)
```

```
['Tokenization', 'is', 'an', 'important', 'step', 'in', 'NLP', '.']
```

Stemming

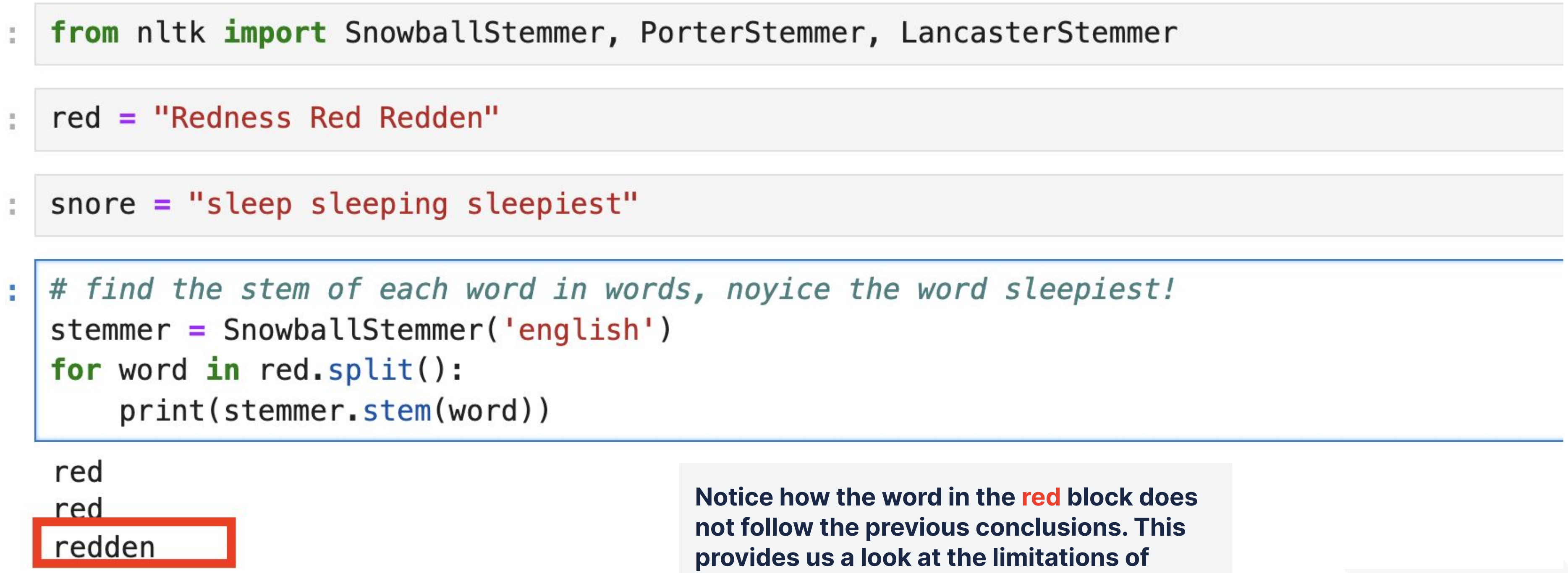
Stemming

- **Text Normalization Technique:** Stemming in Natural Language Processing (NLP) is a text normalization method that reduces words to their base or root form, known as the "stem."

- **Simplifying Vocabulary:** The primary purpose of stemming is to simplify and standardize the vocabulary used in text.

- **Enhancing Text Analysis Efficiency:** Stemming aids in improving the efficiency of NLP tasks, such as information retrieval and search engines, by treating variations of words as equivalent.

Stemming



```
from nltk import SnowballStemmer, PorterStemmer, LancasterStemmer
```

```
red = "Redness Red Redden"
```

```
snore = "sleep sleeping sleepiest"
```

```
# find the stem of each word in words, noyice the word sleepiest!
stemmer = SnowballStemmer('english')
for word in red.split():
    print(stemmer.stem(word))
```

```
red
red
redden
```

Notice how the word in the red block does not follow the previous conclusions. This provides us a look at the limitations of stemming. This is where one can look at an alternative...

EXPLORE AI ACADEMY

Lemmatization

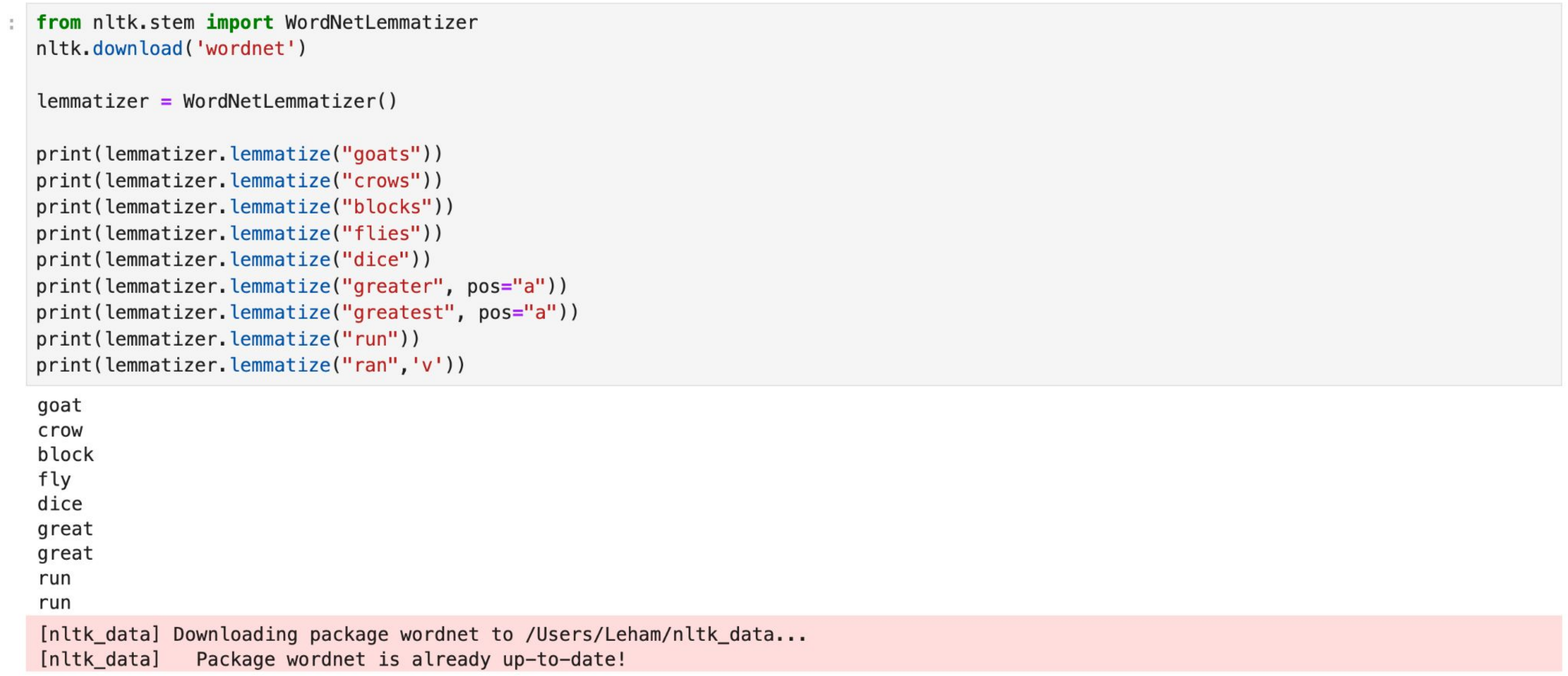
Lemmatization

- **Linguistic Root Extraction**: Lemmatization in Natural Language Processing (NLP) involves extracting the linguistic *root* or **lemma** of a word, providing its canonical form.

- **Precision in Text Normalization**: Unlike stemming, lemmatization considers the word's context and aims to return a meaningful base form, reducing words to their dictionary or linguistic root.

- **Improved Accuracy and Interpretability**: Lemmatization enhances the accuracy and interpretability of NLP analyses by maintaining the integrity of words. It is particularly useful in applications where maintaining the exact grammatical form is crucial.

Lemmatization



```
from nltk.stem import WordNetLemmatizer
nltk.download('wordnet')

lemmatizer = WordNetLemmatizer()

print(lemmatizer.lemmatize("goats"))
print(lemmatizer.lemmatize("crows"))
print(lemmatizer.lemmatize("blocks"))
print(lemmatizer.lemmatize("flies"))
print(lemmatizer.lemmatize("dice"))
print(lemmatizer.lemmatize("greater", pos="a"))
print(lemmatizer.lemmatize("greatest", pos="a"))
print(lemmatizer.lemmatize("run"))
print(lemmatizer.lemmatize("ran",'v'))
```

```
goat
crow
block
fly
dice
great
great
run
run
```

```
[nltk_data] Downloading package wordnet to /Users/Leham/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Stop Words

Stop Words

the
of
a
in
on
my
her
for
to
and
moon
with
from
rain
snow
day
night
autumn
winter
morning
an
back
river
train
me
is
at
all
as
sun
blue
spring
leaves
through
mountain
summer
another
that
between
first
clouds
cold
hand
dream
white
tree
one
street
old
red
fog
full
still
its
storm
christmas
wind
home
sky
light
over
time
life
bed
garden
flowers
late
deep
eyes
sound
silence
off
was
too
way
only
open
cloud
his
you
this
not
no
up
so
sea
heat
smell
coffee
crows
dusk
haiku
away
sunset
black
new
little
hands
into
after
stars
love
two
are
trees
shadow
lost
tide
evening
like
dark
when
green
butterfly
your
more
before
last
about
how
long
just
fresh
end
empty
words
window
what
dreams
wind
field
face
he
call
down
beach
our
high
they
ice
never
dead
alone
she
blossoms
where
mother
water
mist
warm
shadows
falling
now
see
side
sunday

Stop Words

- Stop words in Natural Language Processing (NLP) are common words, such as "the," "and," and "is," that are frequently used but often add little semantic meaning to a text.

- The primary purpose of stop words removal is to reduce noise in text data during preprocessing, focusing on more meaningful words for analysis. By eliminating these common words, the analysis can emphasize content-rich terms.

- Removing stop words not only refines the quality of data for tasks like text classification and sentiment analysis but also contributes to computational efficiency by reducing the dimensionality of the dataset.

EXPLORE AI
ACADEMY

Text Extraction

- Text extraction in Natural Language Processing (NLP) involves **isolating and retrieving relevant information from unstructured text data**.

- Examples of text extraction are numerous however, the for the scope of this discussion we will be looking at **n-grams** and **bag of words.** These methods help one find **patterns** in the text that increase the likelihood of a model predicting more accurately.

N-grams

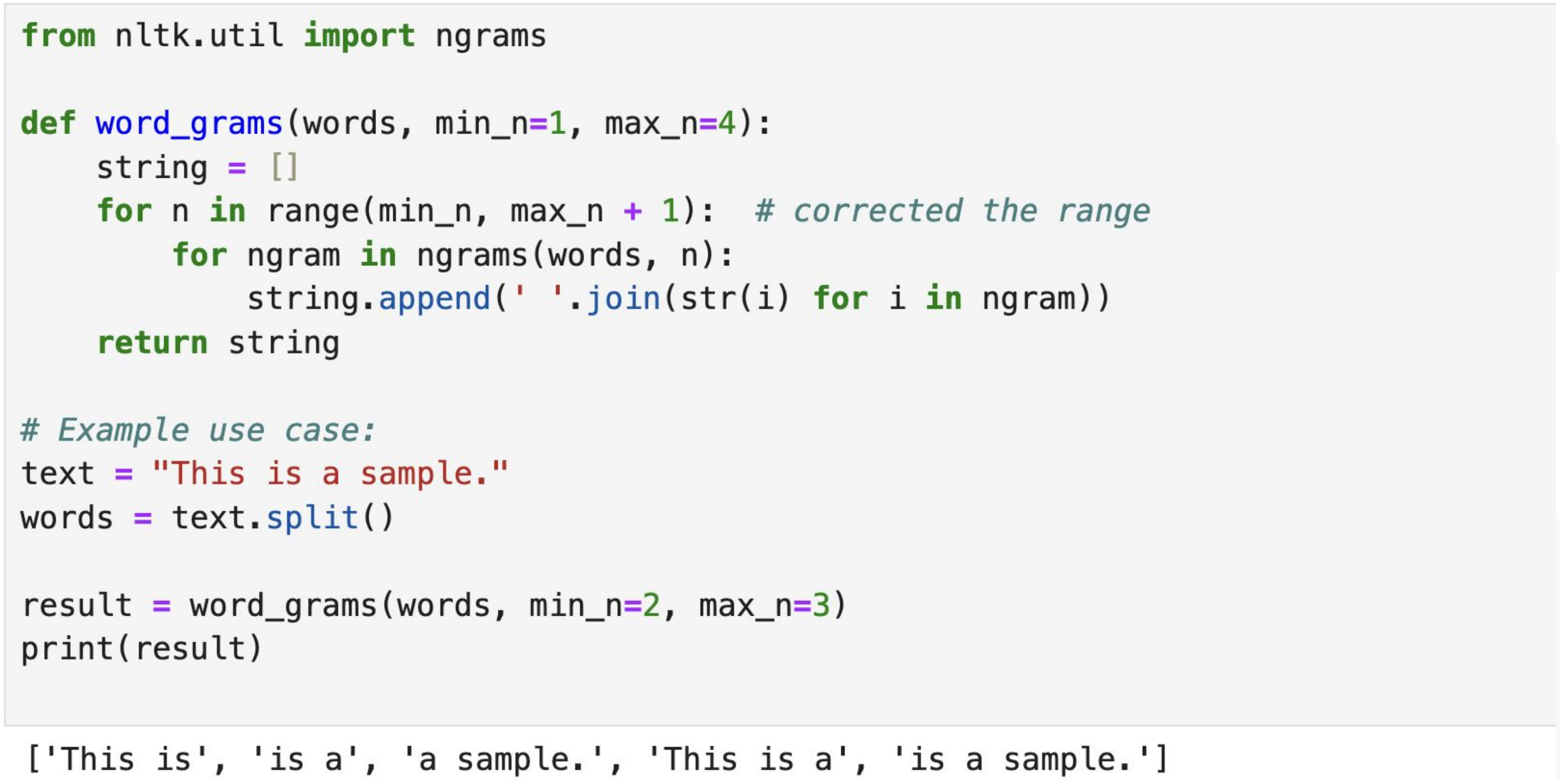
N-grams

- N-grams in Natural Language Processing (NLP) are sequential word combinations of 'n' length, where 'n' represents the number of words in each **grouping**.

- N-grams provide contextual insights by capturing relationships between adjacent words, helping in language modeling and understanding the structure of textual data.

- Widely used in tasks like machine translation, speech recognition, and sentiment analysis, n-grams contribute to the understanding of patterns and dependencies within a sequence of words, allowing algorithms to gain a more nuanced understanding of the context in which words appear.

N-grams



```python
from nltk.util import ngrams

def word_grams(words, min_n=1, max_n=4):
    string = []
    for n in range(min_n, max_n + 1):  # corrected the range
        for ngram in ngrams(words, n):
            string.append(' '.join(str(i) for i in ngram))
    return string

# Example use case:
text = "This is a sample."
words = text.split()

result = word_grams(words, min_n=2, max_n=3)
print(result)
```

```
['This is', 'is a', 'a sample.', 'This is a', 'is a sample.']
```

Bag of Words

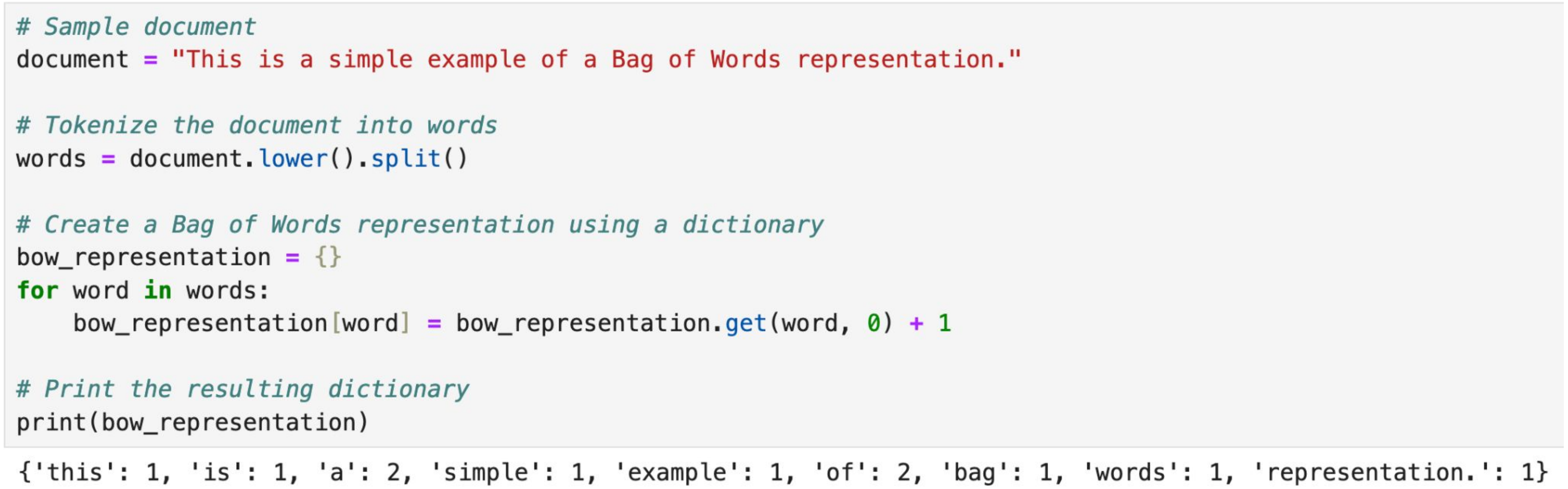
The Bag of Words (BoW)

- The Bag of Words (BoW) model in Natural Language Processing (NLP) is a text representation technique that represents a document as an **unordered set of words, disregarding grammar and word order**.

- BoW **focuses on the frequency of words in a document, creating a sparse matrix or a dictionary** where each unique word is a feature, and its frequency is the corresponding value.

- Commonly used in tasks like text classification and sentiment analysis, **BoW simplifies text (for models)** allowing machine learning algorithms to analyze and categorize text **based on the presence and frequency of words**.

The Bag of Words (BoW)



```python
# Sample document
document = "This is a simple example of a Bag of Words representation."

# Tokenize the document into words
words = document.lower().split()

# Create a Bag of Words representation using a dictionary
bow_representation = {}
for word in words:
    bow_representation[word] = bow_representation.get(word, 0) + 1

# Print the resulting dictionary
print(bow_representation)
```

```
{'this': 1, 'is': 1, 'a': 2, 'simple': 1, 'example': 1, 'of': 2, 'bag': 1, 'words': 1, 'representation.': 1}
```

Sklearn Vectorizers

Sklearn Vectorizers

- SciKit Learn is an extensive library for machine learning projects, including several classifier and classifications algorithms, methods for training and metrics collection, and for preprocessing input data.

- In every NLP project, text needs to be vectorized in order to be processed by machine learning algorithms.

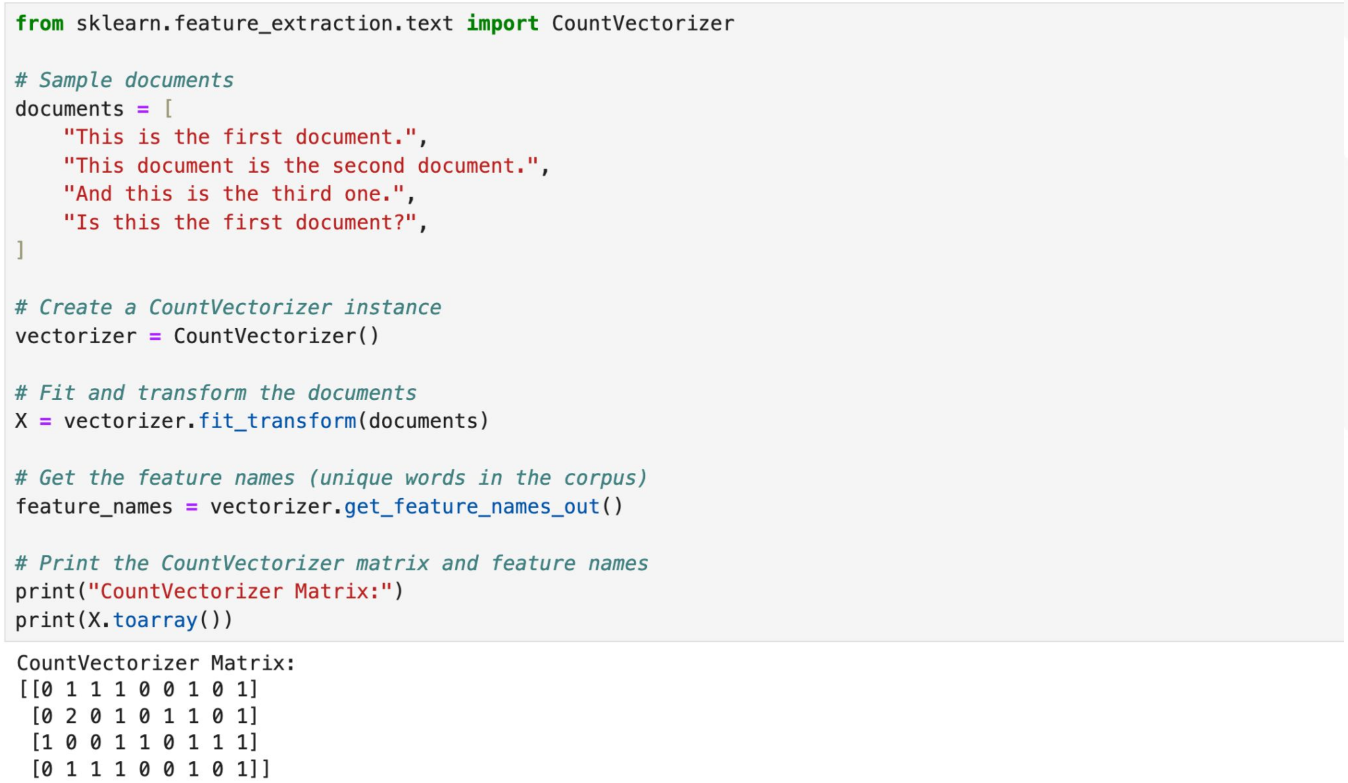
- These tools help you “skip” the above pre processes **reducing the need to manually process text data.**

Sklearn Vectorizers

Popular parameters of CountVectorizer include:

- **stop_words**: Specifies the words to be ignored during the tokenization process (e.g., "english" to remove common English stop words).
- **max_features**: Limits the number of features (words) to the specified number with the highest frequency.
- **ngram_range**: Specifies the range of n-grams to consider (e.g., ngram_range=(1, 2) considers unigrams and bigrams).

Sklearn Vectorizer - “Why reinvent the wheel if the hard work has been done ?”



```python
from sklearn.feature_extraction.text import CountVectorizer

# Sample documents
documents = [
    "This is the first document.",
    "This document is the second document.",
    "And this is the third one.",
    "Is this the first document?",
]

# Create a CountVectorizer instance
vectorizer = CountVectorizer()

# Fit and transform the documents
X = vectorizer.fit_transform(documents)

# Get the feature names (unique words in the corpus)
feature_names = vectorizer.get_feature_names_out()

# Print the CountVectorizer matrix and feature names
print("CountVectorizer Matrix:")
print(X.toarray())
```

```
CountVectorizer Matrix:
[[0 1 1 1 0 0 1 0 1]
 [0 2 0 1 0 1 1 0 1]
 [1 0 0 1 1 0 1 1 1]
 [0 1 1 1 0 0 1 0 1]]
```

EXPLORE AI
ACADEMY

Questions?

Objectives

- **Attempt to explain** What NLP is and how this is translated in the NLTK library.
- **Begin** to comprehend the **syntax** and process of the NLP process.
- **Feel more comfortable** engaging with the NLP content on Athena and beyond!