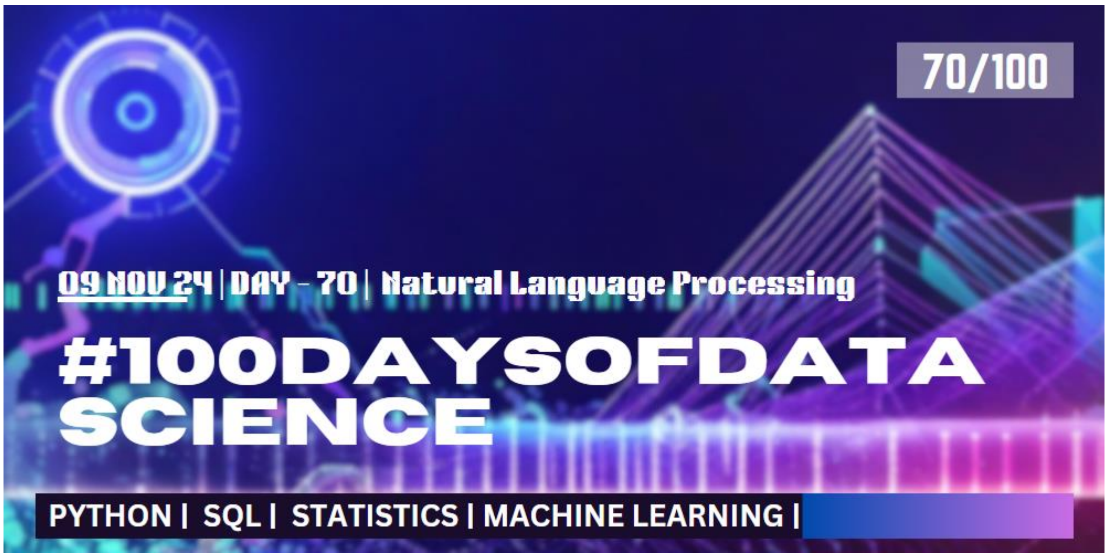

70/100
09 NOV 24 | DAY - 70 | Natural Language Processing
#100DAYSOFDATA SCIENCE
PYTHON | SQL | STATISTICS | MACHINE LEARNING |

Stemming and Lemmatization

What is Stemming?

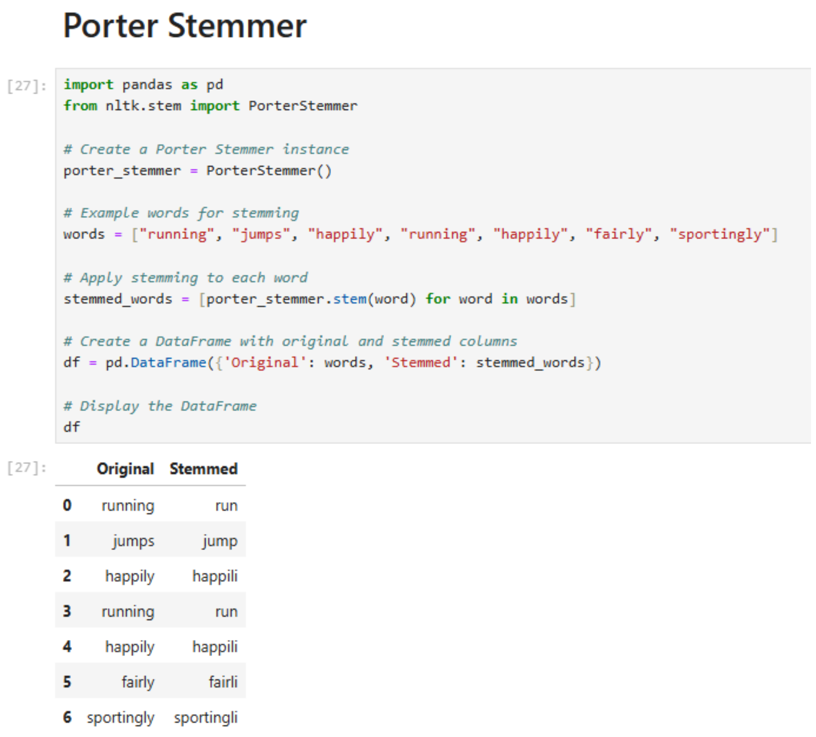
Stemming is a text normalization technique used in Natural Language Processing (NLP) to reduce words to their root or base form by removing prefixes and suffixes. Unlike lemmatization, stemming typically follows simple rules to truncate words without considering their meaning or context. This technique is widely used for basic NLP tasks where linguistic accuracy is less critical, like search engines, as it improves text processing speed by reducing word variations.

Key Aspects of Stemming:

1. **Speed:** Fast as it doesn't require complex linguistic analysis.
2. **Simplicity:** Reduces words to root forms but may not always produce real words.
3. **Use Cases:** Ideal for simple applications like keyword-based search and indexing.

Types of Stemmers

1. **Porter Stemmer:** A widely used English stemmer that applies a series of rules to iteratively remove suffixes from words. Known for its balance between accuracy and simplicity.
   - *Example:* "connects," "connected," "connecting" all become "connect."



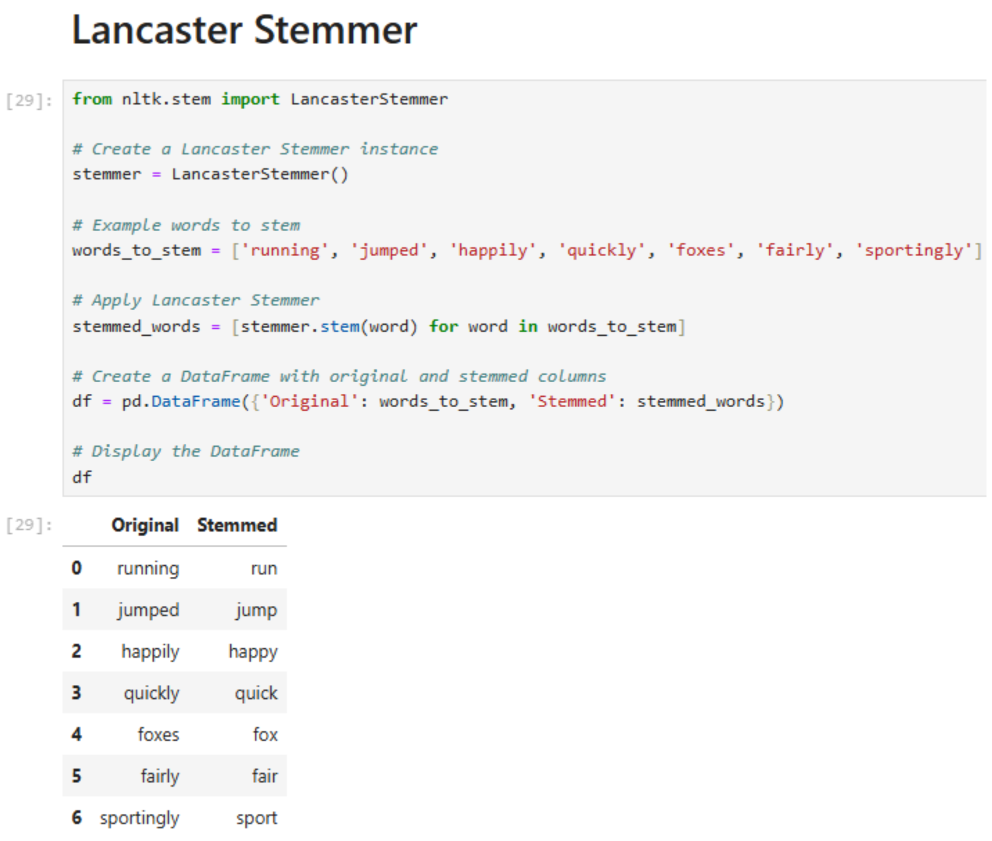
Porter Stemmer

[27]:
```python
import pandas as pd
from nltk.stem import PorterStemmer

# Create a Porter Stemmer instance
porter_stemmer = PorterStemmer()

# Example words for stemming
words = ["running", "jumps", "happily", "running", "happily", "fairly", "sportingly"]

# Apply stemming to each word
stemmed_words = [porter_stemmer.stem(word) for word in words]

# Create a DataFrame with original and stemmed columns
df = pd.DataFrame({'Original': words, 'Stemmed': stemmed_words})

# Display the DataFrame
df
```

[27]:

	Original	Stemmed
0	running	run
1	jumps	jump
2	happily	happili
3	running	run
4	happily	happili
5	fairly	fairli
6	sportingly	sportingli

2. **Lancaster Stemmer:** An aggressive stemmer that produces shorter stems, often reducing words more drastically than others. Suitable for tasks where speed is prioritized.
   - *Example:* "connected" becomes "connect," but "connection" may also become "connect."



Lancaster Stemmer

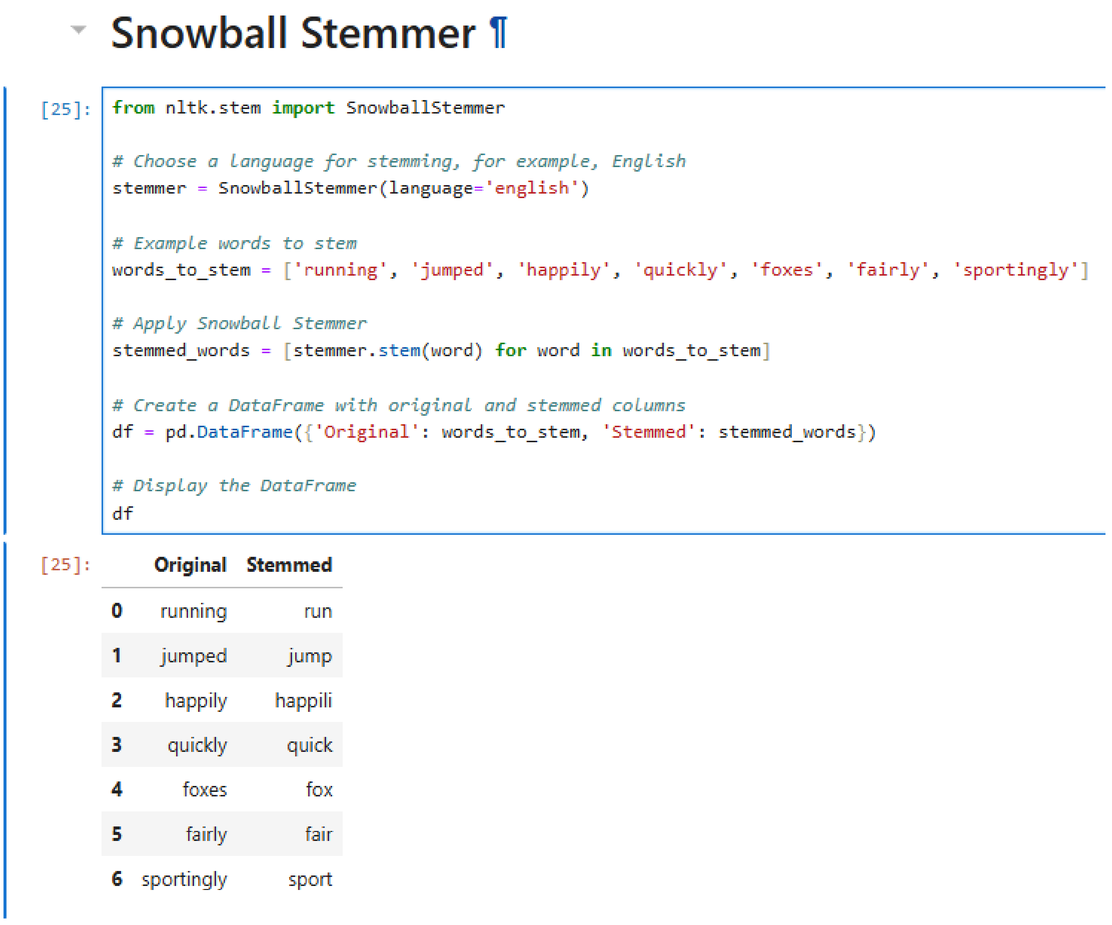
[29]:

```python
from nltk.stem import LancasterStemmer

# Create a Lancaster Stemmer instance
stemmer = LancasterStemmer()

# Example words to stem
words_to_stem = ['running', 'jumped', 'happily', 'quickly', 'foxes', 'fairly', 'sportingly']

# Apply Lancaster Stemmer
stemmed_words = [stemmer.stem(word) for word in words_to_stem]

# Create a DataFrame with original and stemmed columns
df = pd.DataFrame({'Original': words_to_stem, 'Stemmed': stemmed_words})

# Display the DataFrame
df
```

[29]:

	Original	Stemmed
0	running	run
1	jumped	jump
2	happily	happy
3	quickly	quick
4	foxes	fox
5	fairly	fair
6	sportingly	sport

3. **Snowball Stemmer:** An extension of Porter Stemmer, it offers improved performance and supports multiple languages, making it ideal for multilingual NLP applications.
   - *Example:* Similar to Porter but adapted for other languages.



Snowball Stemmer ¶

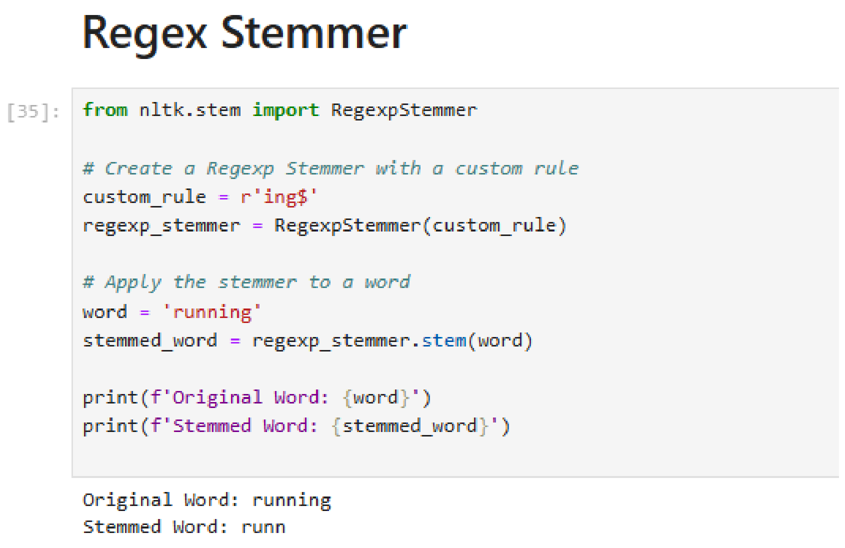
[25]:

```python
from nltk.stem import SnowballStemmer

# Choose a language for stemming, for example, English
stemmer = SnowballStemmer(language='english')

# Example words to stem
words_to_stem = ['running', 'jumped', 'happily', 'quickly', 'foxes', 'fairly', 'sportingly']

# Apply Snowball Stemmer
stemmed_words = [stemmer.stem(word) for word in words_to_stem]

# Create a DataFrame with original and stemmed columns
df = pd.DataFrame({'Original': words_to_stem, 'Stemmed': stemmed_words})

# Display the DataFrame
df
```

[25]:

	Original	Stemmed
0	running	run
1	jumped	jump
2	happily	happili
3	quickly	quick
4	foxes	fox
5	fairly	fair
6	sportingly	sport

4. **Regex-Based Stemmer:** Uses custom-defined regular expressions to remove affixes based on specific patterns. Allows for flexibility, especially with specialized or structured text.
   - *Example:* You could use a regex to remove "-ing" endings from words like "running."



Regex Stemmer

```
[35]: from nltk.stem import RegexpStemmer

      # Create a Regexp Stemmer with a custom rule
      custom_rule = r'ing$'
      regexp_stemmer = RegexpStemmer(custom_rule)

      # Apply the stemmer to a word
      word = 'running'
      stemmed_word = regexp_stemmer.stem(word)

      print(f'Original Word: {word}')
      print(f'Stemmed Word: {stemmed_word}')
```

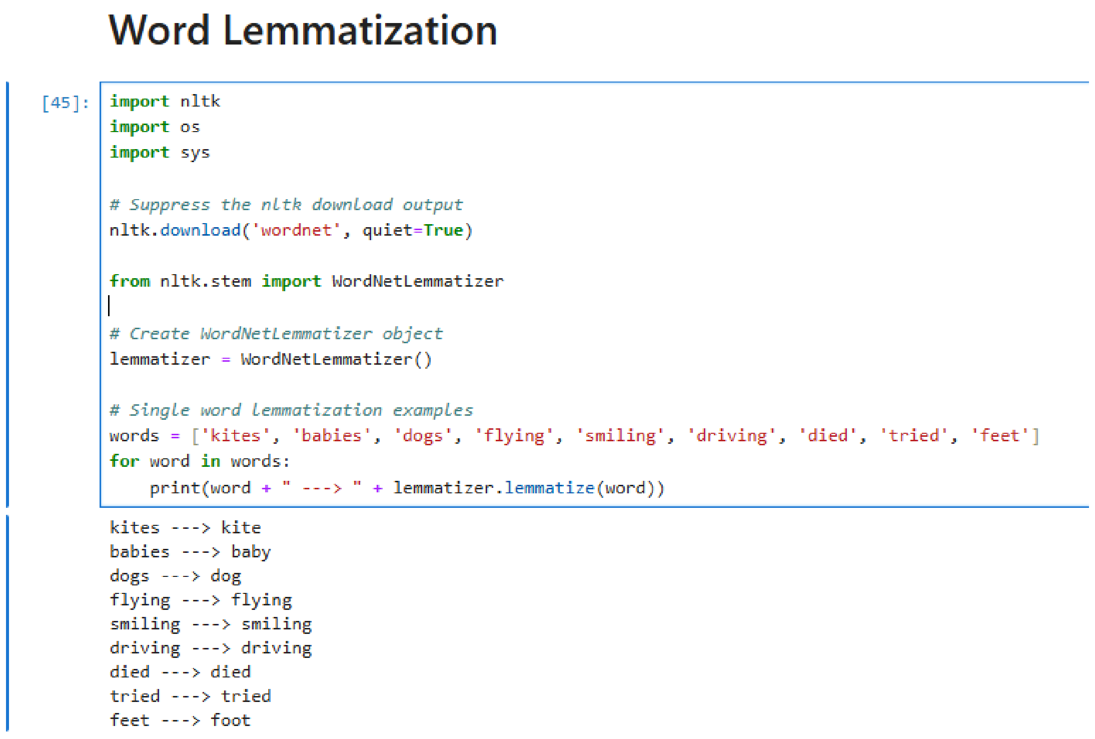
```
Original Word: running
Stemmed Word: runn
```

What is Lemmatization?
Lemmatization is a text normalization technique that reduces words to their **lemma** or dictionary form by considering the context and part of speech. Unlike stemming, lemmatization requires linguistic analysis, which makes it more accurate but computationally intensive. Lemmatization is often used in NLP tasks where understanding the exact meaning and form of a word is crucial, like sentiment analysis or entity recognition.
Key Aspects of Lemmatization:

1. **Accuracy:** Produces meaningful words based on dictionary forms.
2. **Contextual Awareness:** Accounts for part of speech, making it more precise.
3. **Use Cases:** Ideal for tasks that require high linguistic accuracy, such as language translation.

Types of Lemmatization

1. **Word Lemmatization:** Reduces individual words to their base or dictionary forms based on grammatical context, ensuring linguistic accuracy.
   - *Example:* "am," "are," and "is" all lemmatize to "be."

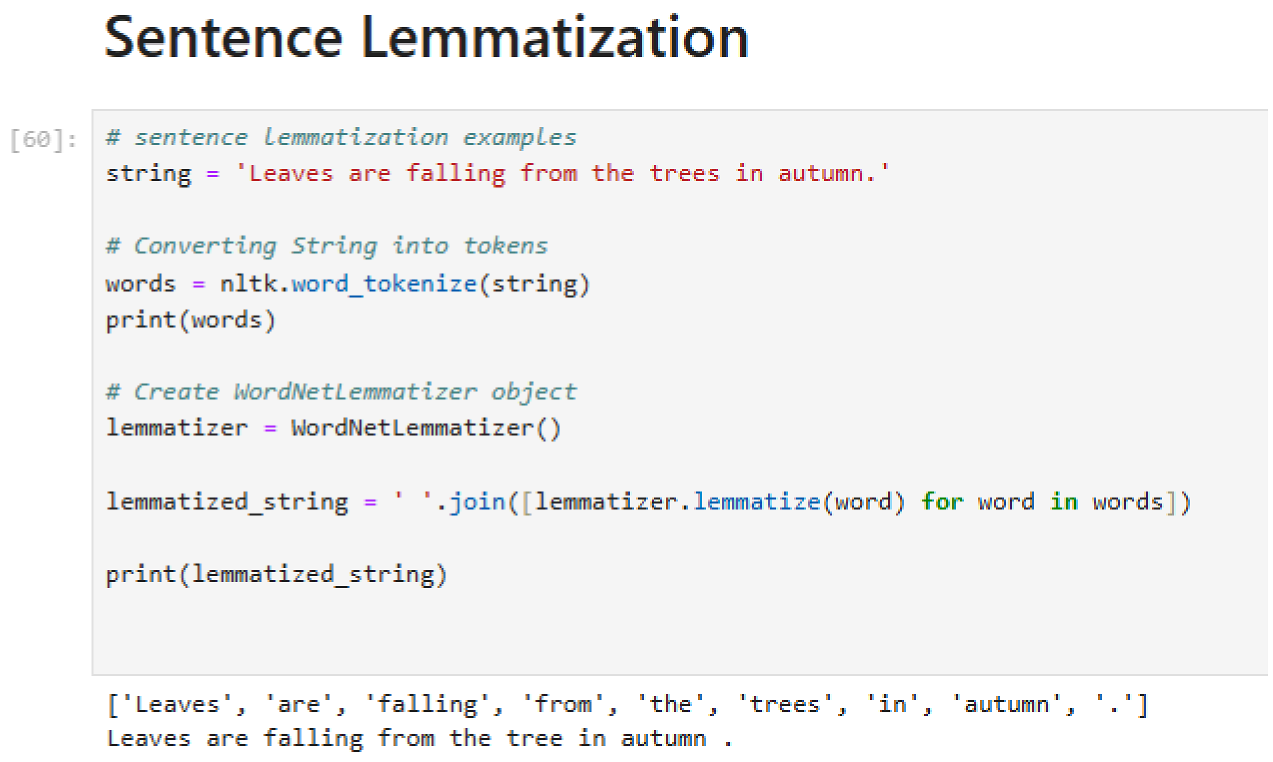


Word Lemmatization

```
[45]: import nltk
      import os
      import sys

      # Suppress the nltk download output
      nltk.download('wordnet', quiet=True)

      from nltk.stem import WordNetLemmatizer

      # Create WordNetLemmatizer object
      lemmatizer = WordNetLemmatizer()

      # Single word lemmatization examples
      words = ['kites', 'babies', 'dogs', 'flying', 'smiling', 'driving', 'died', 'tried', 'feet']
      for word in words:
          print(word + " ---> " + lemmatizer.lemmatize(word))
```

```
kites ---> kite
babies ---> baby
dogs ---> dog
flying ---> flying
smiling ---> smiling
driving ---> driving
died ---> died
tried ---> tried
feet ---> foot
```

2. **Sentence Lemmatization:** Applies lemmatization to each word in a sentence, taking into account the sentence's overall structure and meaning. Useful for tasks like summarization or translation, where understanding context is essential.
    - *Example:* For the sentence "The cats are running," lemmatization would convert "cats" to "cat" and "running" to "run," maintaining the sentence's grammatical integrity.



Sentence Lemmatization

```
[60]: # sentence Lemmatization examples
string = 'Leaves are falling from the trees in autumn.'

# Converting String into tokens
words = nltk.word_tokenize(string)
print(words)

# Create WordNetLemmatizer object
lemmatizer = WordNetLemmatizer()

lemmatized_string = ' '.join([lemmatizer.lemmatize(word) for word in words])

print(lemmatized_string)
```

```
['Leaves', 'are', 'falling', 'from', 'the', 'trees', 'in', 'autumn', '.']
Leaves are falling from the tree in autumn .
```

Stemming vs. Lemmatization: Choosing the Right Approach

- **Use Stemming** for quick, high-level tasks like information retrieval where precision isn’t critical.
- **Use Lemmatization** when word accuracy matters, especially in applications that require deep understanding of language.

By selecting the right approach, you can significantly enhance the effectiveness of NLP applications and ensure more accurate text analysis!