Machine Learning Lab

Week #3 - 25 Sept '24

The AI Society at Arizona State University

Objective

By the end of this workshop, you will understand

How to:

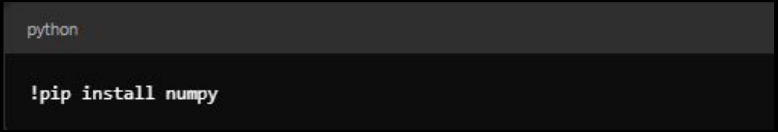
- How to handle text data.
- Feature Extraction from text data.
- TF-IDF creation and Visualization.
- And Colab Implementation

Quick View

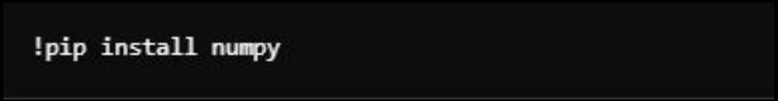
In Jupyter notebooks,

- The “!” symbol followed by a command is used to run shell (bash) commands directly from within a notebook cell.
- It allows you to execute system commands as if you were working in a terminal.

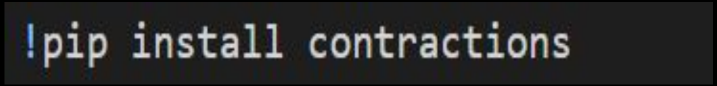
Examples:S



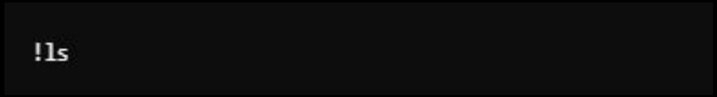
```python
!pip install numpy
```



```
!pip install numpy
```



```
!pip install contractions
```



```
!ls
```

Feature Transformation

Feature Transformation is the process of converting the text data into a numerical format that machine learning model can understand and process effectively .

Steps Involved: Cleaning

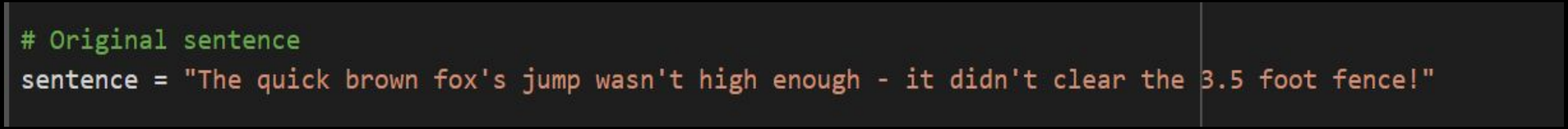
Cleaning Involves removing unnecessary elements from the text that don’t contribute to its meaning

- Removing HTML Tags:
  - Removing HTML tags is necessary to clean the text data of any makeup language, ensuring that only the actual content is processed.
  - This step prevents HTML elements from interfering with the analysis and potentially skewing the results.

- Removing special characters:
  - It helps standardize the text by keeping only letters and spaces, which simplifies further processing and analysis.
  - This step reduces noise in the data and focuses on the core textual content that is most relevant for sentiment analysis

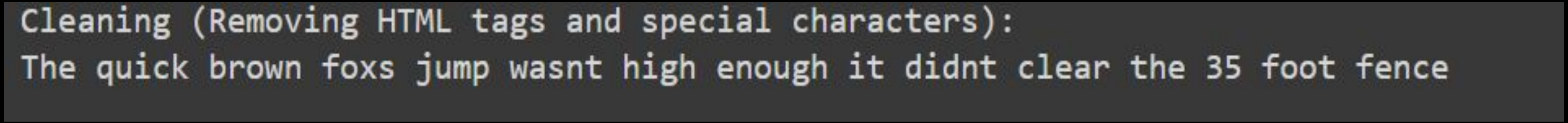
Steps Involved: Cleaning

Example:



```
# Original sentence
sentence = "The quick brown fox's jump wasn't high enough - it didn't clear the 3.5 foot fence!"
```

$\downarrow$



```
Cleaning (Removing HTML tags and special characters):
The quick brown foxs jump wasnt high enough it didnt clear the 35 foot fence
```

Steps Involved: Normalization

Normalization in text preprocessing is a crucial step that involves standardizing the text to ensure consistency across the dataset

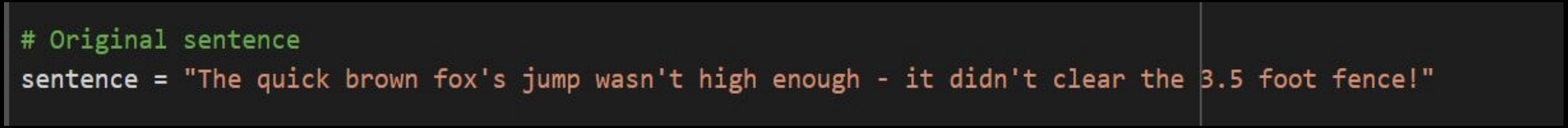
- Case Folding: Converting all text to lowercase (as done in the notebook) or uppercase.
- Unicode Normalization: Standardizing Unicode representations.
- Spelling Correction: Fixing common misspellings.

Significance :

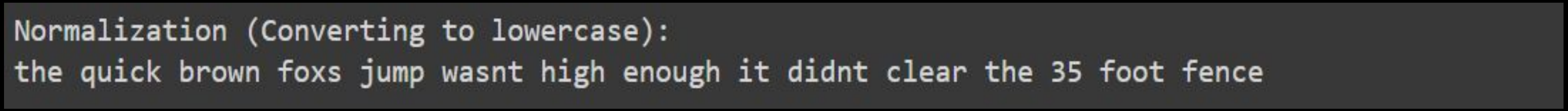
- Reduces the vocabulary size by treating variations of the same word as one.
- Improves consistency in the text data, which can lead to better model performance.
- Helps in reducing noise and irrelevant variations in the text.

Steps Involved: Normalization

Example:



```
# Original sentence
sentence = "The quick brown fox's jump wasn't high enough - it didn't clear the 3.5 foot fence!"
```

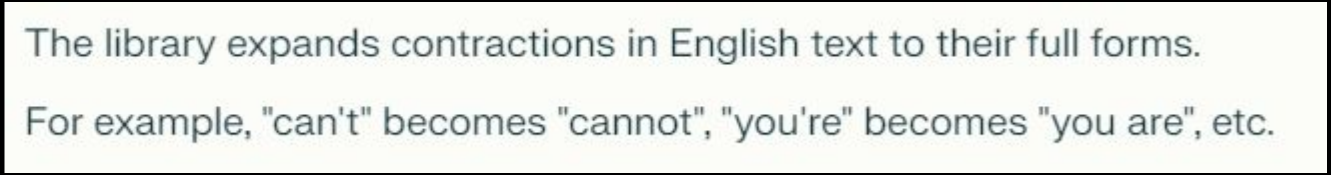


```
Normalization (Converting to lowercase):
the quick brown foxs jump wasnt high enough it didnt clear the 35 foot fence
```

Steps Involved: Contraction

The 'contractions' library is a tool for expanding contractions in text. Contractions are shortened forms of words or phrases, such as "don't" for "do not" or "I'm" for "I am".

- Adding contraction expansion could be an additional step to further normalize and standardize the text data before tokenization and other processing steps. This could potentially improve the quality of the input data for the sentiment analysis task.



The library expands contractions in English text to their full forms.

For example, "can't" becomes "cannot", "you're" becomes "you are", etc.

Steps Involved: Removing Stop Words

Removing stop words is a text preprocessing technique that involves eliminating common words from the text that do not contribute significantly to its meaning or sentiment.

- **Predefined Stopword Lists:** Using pre-compiled lists like NLTK's (as in the notebook).
- **Custom Stopword Lists:** Creating domain-specific stopword lists.
- **Frequency-based Stopword Removal:** Removing words based on their frequency in the corpus.
- **Part-of-Speech Based Removal:** Removing words based on their grammatical role.

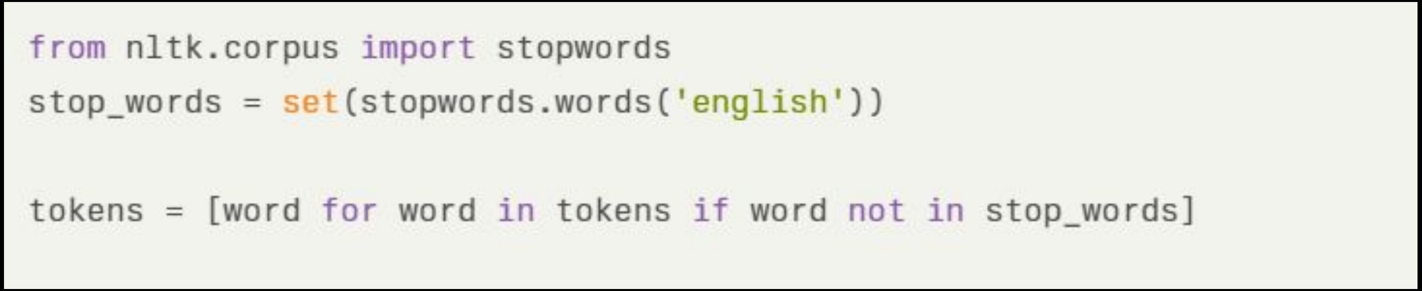
Steps Involved: Removing Stop Words

Significance:

- Reduces the dimensionality of the text data, making subsequent processing more efficient.
- Focuses the analysis on more meaningful words, potentially improving model performance.
- Can significantly reduce the size of the vocabulary, which is especially useful in tasks like TF-IDF vectorization.

Implementation:

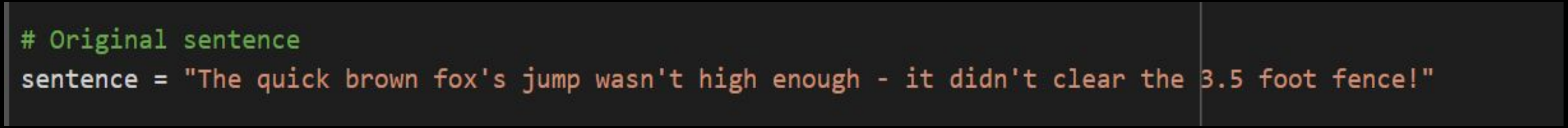
In the notebook, stopwords are removed using NLTK's list of English stopwords:



```
from nltk.corpus import stopwords
stop_words = set(stopwords.words('english'))

tokens = [word for word in tokens if word not in stop_words]
```

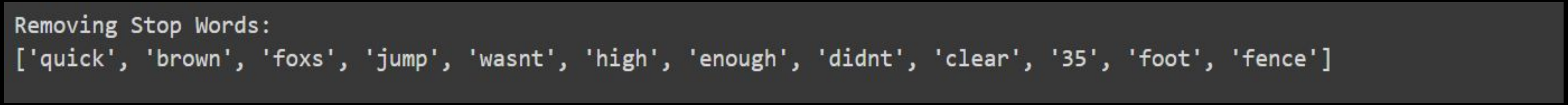
Steps Involved: Removing Stop Words

Example:



```
# Original sentence
sentence = "The quick brown fox's jump wasn't high enough - it didn't clear the 3.5 foot fence!"
```

$\downarrow$



```
Removing Stop Words:
['quick', 'brown', 'foxs', 'jump', 'wasnt', 'high', 'enough', 'didnt', 'clear', '35', 'foot', 'fence']
```

Steps Involved: Tokenization

Tokenization is the process of breaking down text into smaller units called tokens. This step is crucial for converting raw text into a format that can be easily processed and analyzed by machine learning algorithms.

- **Word Tokenization:** Splitting text into individual words (as done in the notebook).
- **Sentence Tokenization:** Splitting text into sentences.
- **N-gram Tokenization:** Creating tokens of N consecutive words.
- **Subword Tokenization:** Breaking words into smaller units.

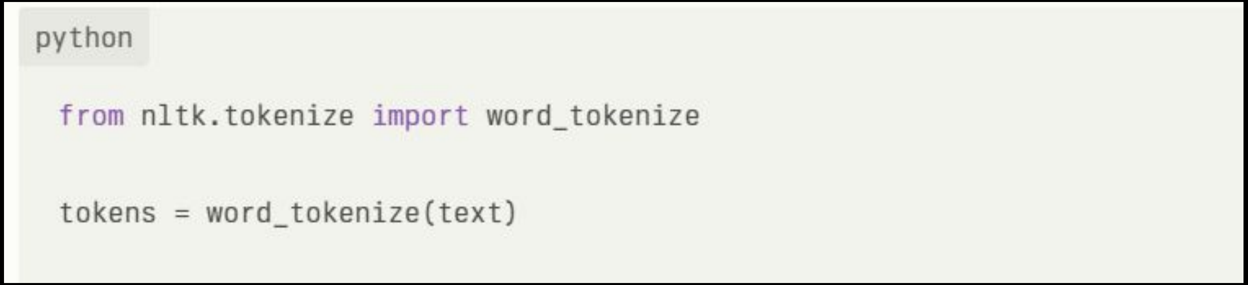
Steps Involved: Tokenization

Significance:

- Breaks down text into meaningful units for analysis.
- Enables the creation of numerical representations of text (e.g., for TF-IDF vectorization).
- Facilitates further preprocessing steps like stopword removal and lemmatization.

Implementation:

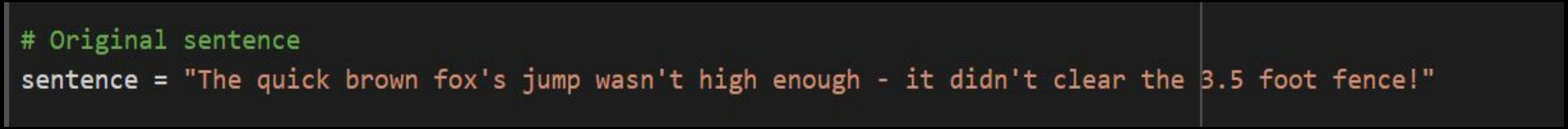
In the notebook, tokenization is performed using NLTK's word_tokenize function:



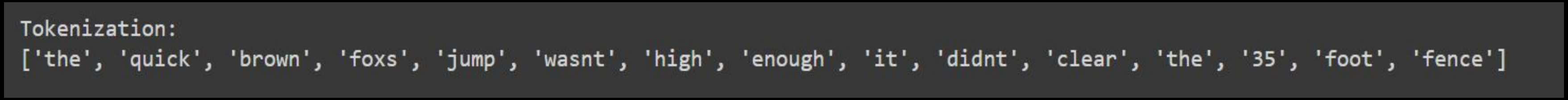
```python
from nltk.tokenize import word_tokenize

tokens = word_tokenize(text)
```

Steps Involved: Tokenization

Example:



```
# Original sentence
sentence = "The quick brown fox's jump wasn't high enough - it didn't clear the 3.5 foot fence!"
```



```
Tokenization:
['the', 'quick', 'brown', 'foxs', 'jump', 'wasnt', 'high', 'enough', 'it', 'didnt', 'clear', 'the', '35', 'foot', 'fence']
```

TF-IDF

Term Frequency (TF) : Measures how often a term appears in a document.

Inverse Document Frequency (IDF) : Weights the term based on its rarity across all documents.

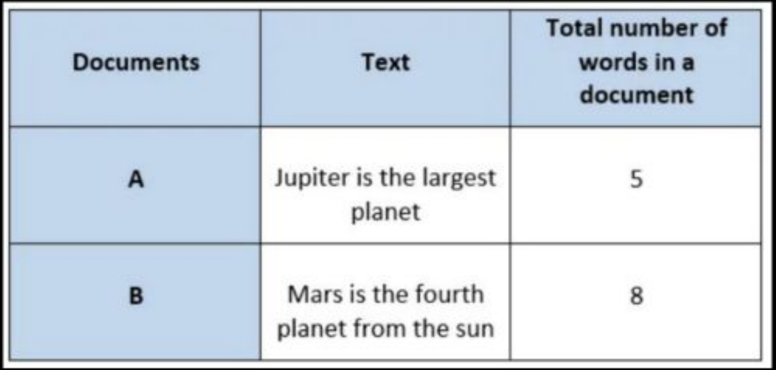
$$TF\text{-}IDF(t,d) = TF(t,d) * IDF(t)$$

Helps prioritize important, unique words in documents while downplaying common terms.

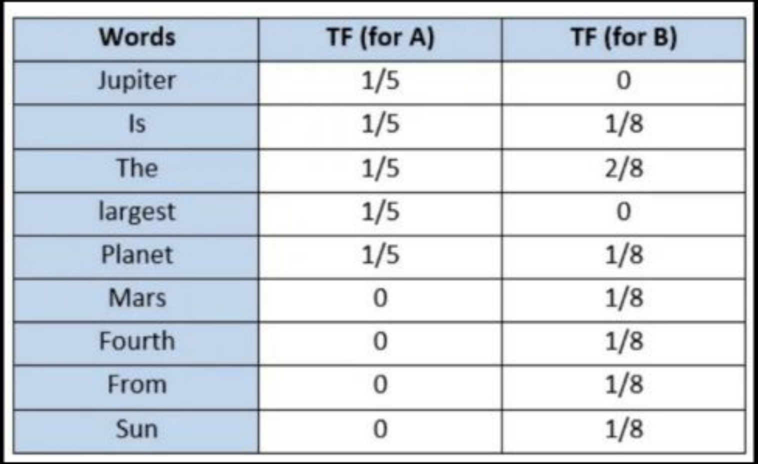
Term Frequency

$$\mathrm{TF}(t, d) = \frac{\text{Frequency of } t \text{ in } d}{\text{Total words in } d}$$

Example :



Documents	Text	Total number of words in a document
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8



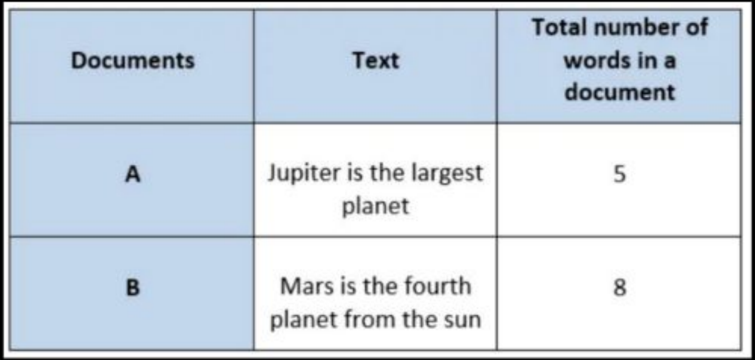
Words	TF (for A)	TF (for B)
Jupiter	1/5	0
Is	1/5	1/8
The	1/5	2/8
largest	1/5	0
Planet	1/5	1/8
Mars	0	1/8
Fourth	0	1/8
From	0	1/8
Sun	0	1/8

Source Images : https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

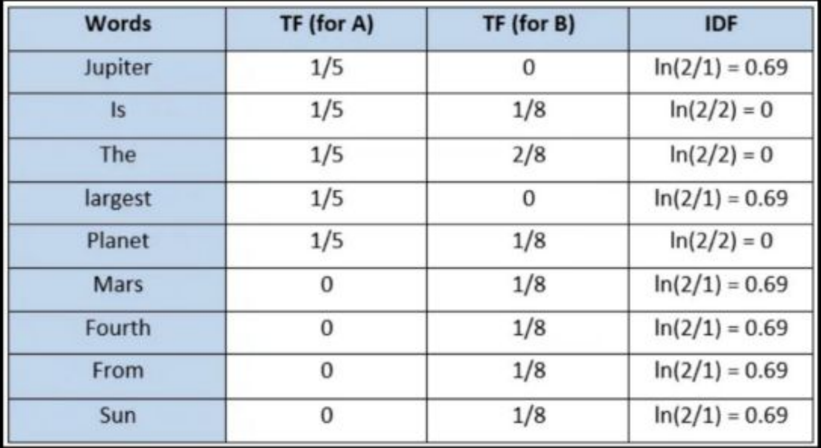
Inverse Document Frequency

$$\mathrm{IDF}(t) = \log\left(\frac{\text{Total Documents}}{1 + \text{Number of Documents containing } t}\right)$$

Example :



Documents	Text	Total number of words in a document
A	Jupiter is the largest planet	5
B	Mars is the fourth planet from the sun	8



Words	TF (for A)	TF (for B)	IDF
Jupiter	1/5	0	ln(2/1) = 0.69
Is	1/5	1/8	ln(2/2) = 0
The	1/5	2/8	ln(2/2) = 0
largest	1/5	0	ln(2/1) = 0.69
Planet	1/5	1/8	ln(2/2) = 0
Mars	0	1/8	ln(2/1) = 0.69
Fourth	0	1/8	ln(2/1) = 0.69
From	0	1/8	ln(2/1) = 0.69
Sun	0	1/8	ln(2/1) = 0.69

Source Images : https://towardsdatascience.com/text-vectorization-term-frequency-inverse-document-frequency-tfidf-5a3f9604da6d

Advantages and Limitations of TF-IDF

Advantages

- Weighs words based on importance, discounts common words.
- Improves Search and Retrieval Relevance.

Limitations

- Context Ignorance
- Synonym Problem
- Cannot Handle Complex Relationships

Additional Resources

Python: https://www.w3schools.com/python/

Python Video: https://www.youtube.com/watch?v=_uQrJ0TkZlc

Libraries: https://www.geeksforgeeks.org/libraries-in-python/

Pandas: https://www.w3schools.com/python/pandas/default.asp

Join our Discord for more updates and resources!

https://www.bit.ly/AIS-Links

Test your knowledge!

Group Picture!

Steps Involved: Lemmatization

This process involves analyzing words morphologically and using vocabulary and morphological analysis to determine the root form of the word.

- **WordNet Lemmatization:** Using NLTK's WordNetLemmatizer (as in the notebook).
- **Part-of-Speech Aware Lemmatization:** Specifying the part of speech for more accurate lemmatization.
- **Language-Specific Lemmatization:** Using lemmatizers designed for specific languages.
- **Part-of-Speech Based Removal:** Removing words based on their grammatical role.

Steps Involved: Lemmatization

Significance:

- Reduces inflectional forms and sometimes derivationally related forms of a word to a common base form..
- Helps in consolidating different forms of the same word, reducing vocabulary size.
- Improves text analysis by treating related words as a single item.

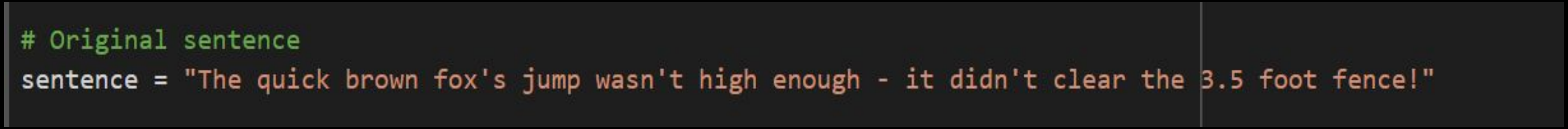
Implementation:

Lemmatization is performed using NLTK's WordNetLemmatizer:

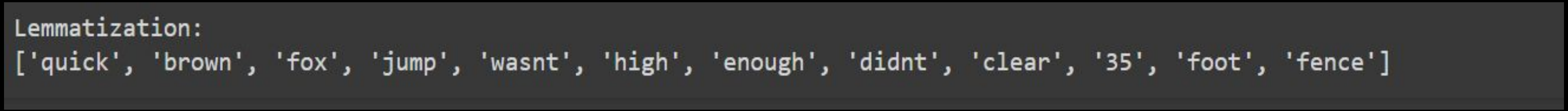
```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
tokens = [lemmatizer.lemmatize(word) for word in tokens]
```

Steps Involved: Lemmatization

Example:



```
# Original sentence
sentence = "The quick brown fox's jump wasn't high enough - it didn't clear the 3.5 foot fence!"
```



```
Lemmatization:
['quick', 'brown', 'fox', 'jump', 'wasnt', 'high', 'enough', 'didnt', 'clear', '35', 'foot', 'fence']
```