module03-analyzing-text-content-natural-language-processing-30pct/3-1-3-computer-lab.qmd

List of Figures

List of Tables

descriptive analysis

The provided code chunk reads a collection of text files from a specified directory, cleans the text by removing non-ASCII characters, and then stores the cleaned and original text in a `pandas` DataFrame along with a unique ID and the filename. The DataFrame is then printed, displaying the organized data for further analysis.

New Code Chunk for Word Count and Character Count:

To extend the analysis, the following code chunk will add two new columns to the DataFrame: one for the word count and another for the character count of the cleaned text.

```
# Perform word count and character count on each cleaned text in the DataFrame
df['word_count'] = df['cleaned_text'].apply(lambda x: len(x.split()))
df['character_count'] = df['cleaned_text'].apply(lambda x: len(x))

# Select and print all columns except 'original_text' and 'cleaned_text'
columns_to_display = df.columns.difference(['original_text', 'cleaned_text'])
print(df[columns_to_display])
```

Explanation:

1. **Word Count**:
   - The `apply` function is used on the `cleaned_text` column to calculate the word count. The `lambda` function splits each cleaned text string into words using `split()` and then calculates the length of the resulting list using `len(x.split())`. This word count is stored in a new column `word_count`.
2. **Character Count**:
   - Similarly, the `apply` function calculates the character count by applying `len(x)` directly on the `cleaned_text` string. This counts the total number of characters (including spaces) and stores the result in a new column `character_count`.
3. **Updated DataFrame**:
   - The DataFrame is then printed again, now including the `word_count` and `character_count` columns, providing additional insights into the text length and structure.