TALOS ERA CHAIR IN ARTIFICIAL INTELLIGENCE FOR HUMANITIES AND SOCIAL SCIENCES

ΠΑΝΕΠΙΣΤΗΜΙΟ ΚΡΗΤΗΣ
UNIVERSITY OF CRETE

A beginner's notebook on NLP

Dimitris Bilianos
Postdoctoral Researcher

Funded by the European Union

Horizon ERA Chair TALOS AI4SSH Project funded by the European Commission Grant Agreement n° 101087269, https://cordis.europa.eu/project/id/101087269

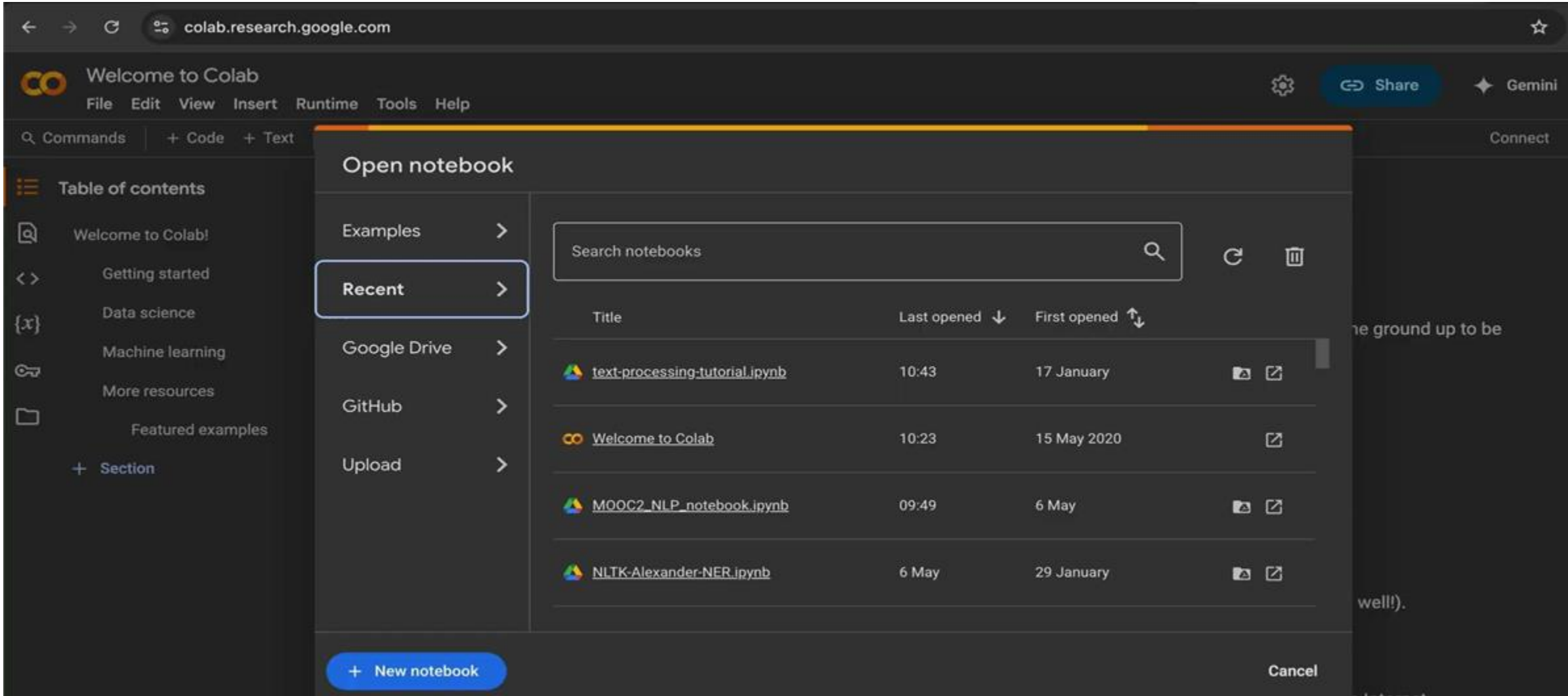
Welcome to Natural Language Processing with Python!

Over the next few minutes, we'll learn how to:

- **Open** and **read text** from .txt files
- Calculate some simple but **useful metrics** to understand our text better, like how many words and characters it contains.
- Perform a few basic preprocessing steps to **clean up our text** and make it easier to analyze.

→We'll be using Python and Google Colab, which provide a fantastic environment for writing and running code. Let's dive in and see how we can unlock the power hidden within text!

Loading a file into our Python environment



Reading our text file

```python
# Open the text file in read mode ('r')
with open('my_text_file.txt', 'r') as file:
    text_content = file.read()

# Now, the entire content of the file is stored in the 'text_content' variable
print(text_content)
```

Explanation of the code block:

- All the lines that are green and start with the # symbol are just comments
- open('my_text_file.txt', 'r'): This line tries to open **a file named my_text_file.txt.** The 'r' tells Python we want to **read** from this file.
- with ... as file:: This way of handling files ensures that the file is **automatically closed** even if errors occur.
- text_content = file.read(): This **reads** the entire content of the opened **file** and **stores it in a variable** called text_content.
- print(text_content): This will **display the text that was read** from the file once we press the play button on the left side of the code block.

Getting a feel for our text: Basic metrics

```python
# Define a function to calculate basic text metrics
def calculate_metrics(text): # we define a function named calculate_metrics
                             # that takes a single argument, text, which represents the text to analyze
    word_list = text.split()  # Break the text into a list of words using spaces as the delimiter
    num_words = len(word_list)  # Total word count
    num_chars = len(text)  # Total character count including spaces and punctuation
    unique_words = len(set(word_list))  # Unique word count
    '''
    The set() function creates a set from the word_list.
    A set is a data structure that can only contain unique elements.
    Any duplicate elements in the original list are removed when you create a set
    '''
    return num_chars, num_words, unique_words # To be used to output the metrics

```

Explanation of the code block:

The code above defines a function that takes one argument, text, which represents the text to analyze.

This function breaks the text into words and passes them into a word list. It then calculates the number of words in this list as well as the overall number of characters in the text. Then, it creates a set of words, removing any duplicate elements, such as words that appear more than once. Finally, the function is instructed to output the three calculated values

Time to put our function into action

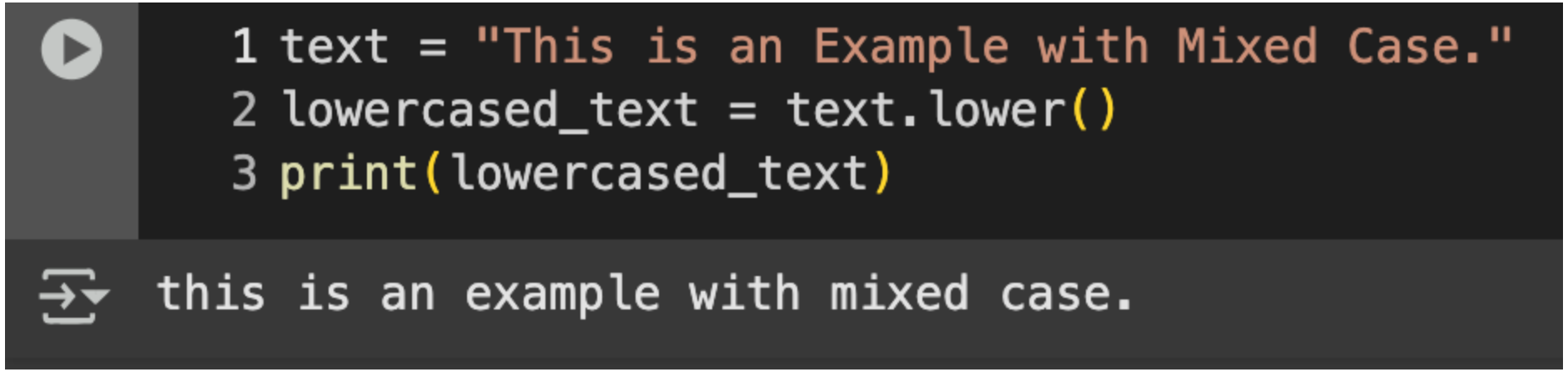
```
1 text_to_analyze = "This is just a simple example to be used as a demonstration"
2 char_count, word_count, unique_count = calculate_metrics(text_to_analyze)
3
4 print(f"Total characters: {char_count}")
5 print(f"Total words: {word_count}")
6 print(f"Number of unique words: {unique_count}")
```

```
Total characters: 59
Total words: 12
Number of unique words: 11
```

- Understanding these basic metrics is a **fundamental first step** in exploring any text data.
- They can give you a **quick sense** of the text's length and vocabulary richness

Cleaning Up Our Text: Basic Preprocessing

- Converting all the text to **lowercase** is a common first step.
- Why do we do this? Because computers treat "The" and "the" as different words.
- By converting everything to lowercase, we ensure that these variations are counted as the same word, which is usually what we want for analysis.



Wrapping up!

In this brief introduction, we've covered some **fundamental concepts** and **practical techniques:**

- We learned how to **open and read text data** from .txt files using Python's built-in capabilities.
- We explored how to **calculate basic metrics** like the total number of characters, words, and unique words in a text, giving us an initial understanding of its size and vocabulary.
- Finally, we touched upon **basic preprocessing**, including **converting text to lowercase**, which helps in preparing text for more advanced analysis.

What’s next

This is just the **beginning of your NLP journey**. There's a vast and fascinating world to explore, such as:

- Explore **more preprocessing techniques**: Learn about lemmatization, handling punctuation, and dealing with different text encodings.
- Use **Python libraries** such as NLTK, for tasks such as **part-of-speech tagging** (identifying the grammatical role of each word) and **named entity recognition** (identifying people, places, organizations, etc.).
- Get started with **text analysis** tasks: Begin exploring techniques like **sentiment analysis** (determining the emotional tone of text) and **topic modeling** (discovering the main topics in a collection of documents).