Course Project: Automatic Plot Analysis

PCL 1, Fall Semester 2023

Deadline for Part 0: 13.12.23, 18:00
Deadline for part 1-4: 26.12.23, 18:00

This project is a major effort in this programming course (remember, the course is worth 9 ECTS points, 50% more than normal courses) to apply your coding skills to a more complex problem. The number of exercise points (3 points) you can earn with this project should be proportional to the amount of work you put in the normal exercises (1 point). With this project, we also want to prepare you for the typical situation of plotting your data to see patterns, trends, and insights that are not immediately apparent from raw data alone. This skill is crucial in many fields, where visualizing data effectively can lead to a more profound understanding and better decision-making based on the analyzed information.
Note: We don't want you to suffer unnecessarily! If there are any questions, please post a question in the forum, write an email or ask us the tutorial or office hours!

Remarks on submission

- Learning partnerships in pairs are **mandatory**. Both students have to submit the project and name it `username1_username2_pcl1_course_project.zip`
- Please state the first and last names of both students on the submission form.
- **Add a docstring and type hints to every function** and add comments to your code where helpful. They help the tutors understand what you are thinking, but they can also help you find and fix possible bugs.
- **Grading**: In total you can earn 3 points. For each part, the maximum number of points is specified. To achieve full points in each implementation part, the quality of your comments and doc strings as well as the functions and data structures you choose are considered.
- **Hand in:** There are **two submission deadlines** for this project. You will have to hand in Part 0 until **December 13th at 18:00**. Please send an email with the subject "Project, Part0" to the tutors. Hand in the **full** exercise via OLAT by **December 26th at 18:00**. Make sure to hand in on time!
- **Important:** To receive full points, you have to hand in on time on both of the deadlines.

Introduction

In plot analysis, understanding language, sentiment, and character interaction is crucial. The project explores these aspects using text mining and sentiment analysis. Utilizing natural language processing, with a focus on named entity recognition (NER) and sentiment analysis, your goal is to reveal and illustrate the dynamics of characters and emotions in your chosen literary text.

This project involves analyzing a book from the Project Gutenberg library. Your aim is to identify and analyze the distribution of **named entities**, especially characters, and the **sentiments** related to them in the narrative. The goal of this project isn't just to test out NLP methods, but also to gain a deeper understanding of the text's structure and underlying feelings. Furthermore, your job is to create visual representations of the relationships and sentiment trends among entities, providing an illustrative and easy-to-understand overview of the narrative's emotional and interactional dynamics.

Part 0: Book Selection

Important: Please, submit by December 13th, 6 pm via email to the tutors with "Project, Part0" as the subject.

Objective: To identify a suitable book for in-depth analysis, you should first examine multiple works using simple methods.

a) Select 3 books from the Project Gutenberg library

Choose books with a rich distribution of named entities and sentiments. Opt for English-written texts in genres like drama or action. While making your selection, consider the narrative structure and the presence of diverse characters and emotional expressions. Feel free to ask ChatGPT to give you some recommendations.

b) Text Clean-up and Chapter Splitting

Gutenberg books typically come with headers and footers containing information not relevant to our analysis. To help with this, we've provided a file named `gutenberg_cleanup.py` containing a pre-implemented clean-up function. Your task is to extend this script by writing a function to split the book into chapters and save each chapter in a separate file within a folder named after the book title. Follow these steps to adapt and use the script:

a) **Implementing Chapter Splitting:** Enhance the `gutenberg_cleanup.py` script by adding a function to split the text into chapters. After cleaning the text, this function should divide the book into chapters and save each one separately.

b) **Organizing Files by Book Title:** Each book should have its dedicated folder named `{BookTitle}`. Within this folder, save the cleaned full text as `BookTitle_clean.txt` and create a subfolder named `chapters` to store individual chapter files.

c) **Refactoring for Arguments:** Modify the script to accept command-line arguments, allowing you to specify the file path of the book to be processed. This avoids hard-coding file paths and makes the script more flexible.

d) **Executing the Script:** Run the script from the command line for each of your selected books, passing the file path as an argument. Ensure that the script performs both cleaning and chapter splitting operations and saves the results in the appropriate `{BookTitle}` folder.

This setup ensures you work with clean and well-organized text data, with each book's chapters neatly stored in separate files within their respective folders, enhancing the manageability of subsequent analysis tasks.

c) Use grep for a Preliminary Chapter-Based Sentiment and Entity Analysis

Conduct initial analysis using grep[1], focusing on a chapter-by-chapter basis to capture the progression of entities and sentiments throughout the book.

- **Identifying Named Entities and Sentiments by Chapter:** Apply grep to each chapter to detect named entities and sentiment expressions, aiding in understanding the development of the narrative.

[1] This is thought as an exercise for working with grep to quickly explore data.

Converting grep Results to JSON

After saving the grep results in `.txt` format, the next step is to convert these results into structured JSON files, separating entities and sentiments while associating them with their corresponding chapters.

- **Structuring Data:** Create a Python script to read the `.txt` files and organize the data into JSON format. This should result in two JSON files per chapter: one for entities and one for sentiments.
- **Conversion Process:** In your script, read each `ChapterNumber_Entities.txt` and `ChapterNumber_Sentiments.txt` file, parse the data, and structure it into JSON objects. Include chapter identifiers in the JSON structure.
- **JSON File Naming and Organization:** Name the JSON files in a way that reflects their content and chapter, like `BookTitle_Chapter1_Entities.json` and `BookTitle_Chapter1_Sentiments.json`. Organize these files in the respective book folders.

Example Python Script for Data Conversion: Create a Python script that performs the following steps:

a) Read each `.txt` file for entities and sentiments.

b) Parse the contents to structure the data into a Python dictionary.

c) Save the dictionary as a JSON file.

This script will be a crucial part of your data preprocessing and should be carefully designed to handle the specific format of your grep results.

Important: Do not remove or change the skeleton code provided in `part0.py`. You may not use any external or built-in libraries besides the ones given in the file. You may define more functions if you want to.

Deliverables: Submit the following:

- A main folder containing subfolders for each book, with JSON files for entities and sentiments per chapter.
- The Python script used for converting `.txt` files to JSON.
- The `discussion.txt` file documenting your analysis process.

By converting your grep results into JSON format, you'll facilitate further analysis and visualization in the later parts of your project, making use of structured and easily accessible data.

Part 1: Named Entity Recognition using spaCy (15 points)

Objective: Utilize spaCy's Named Entity Recognition (NER) capabilities for detailed identification of the main characters in the chosen book, enhancing accuracy and depth of entity detection.

a) Setting Up spaCy for NER

- Install spaCy and its necessary dependencies in your Python environment.
- While installing SpaCy, choose the settings for your device, then choose English for the trained pipeline, and select efficiency.
- Please read through SpaCy's documentation if you want explanations and examples for all of its features.
- Conduct initial tests with sample text to ensure accurate entity identification.

b) Focused NER on Main Characters in the Selected Book

- Utilize spaCy to process the text of the selected book, with a specific focus on entities classified as PERSON. Prioritize main characters, capturing their presence and significance within the narrative.
- Implement a clustering technique to group different names or aliases referring to the same character. For example, a character like Jane Eyre might be referred to as "Janet" or "Jane Elliott" in the book. Ensure that all these references are clustered under the main character's identity.
- Document each identified main character with additional information like the context of mention (sentence or paragraph) and frequency of occurrence within the book.
- Store these details in a structured JSON file. Each entry should include the main character's name, context, and frequency.
- After processing, review the stored data and note down observations related to the main characters. Pay attention to any surprising inclusions, exclusions, or misclassifications by spaCy.

JSON File Structure for Focused NER Results: Organize the NER results in a JSON file, ensuring the structure enables easy analysis of the main characters. Each main character should be a distinct entry, with details about their occurrences in terms of context, location in the text, and frequency. For example:

```
{
  "main_characters": [
    {
      "name": "CharacterName1",
      "aliases": ["Alias1", "Alias2"],
      "occurrences": [
        {
          "sentence": "Context of mention.",
          "chapter": "Chapter number",
          "position": {"start": startIndex, "end": endIndex}
        },
        // More occurrences
      ]
    },
    // More main characters
  ]
}
```

Deliverables:

- A JSON file, `BookTitle_MainCharacters_NER.json`, containing detailed information about the named main characters and their aliases.
- Your code for the focused NER task, as a Python script (`.py`) or a Jupyter Notebook (`.ipynb`), with annotations on your workflow, challenges encountered, and specific choices made during the NER process.
- A txt file, `NER_MainCharacters_Observations.txt`, noting your findings and observations about the main characters from the JSON file.

This part of the project emphasizes your ability to apply NLP techniques for targeted entity recognition, focusing on the main characters and their various names or aliases, and your skills in documenting and critically analyzing the NER process.

Part 2: Entity-Specific Sentiment Analysis (15 points)

Objective: Implement a simple yet effective method for analyzing and quantifying the sentiments associated with the main characters of your story, focusing on sentence-level and chapter-level sentiment aggregation.

a) Choosing a Sentiment Analysis Method

- Evaluate and select a sentiment analysis tool suitable for your analysis. Consider Flair, spaCy, or the NRCLex library. To aid you in your choices, you should consider reading the documentation or blogposts[2] pertaining to Sentiment Analysis for these libraries.
- Explore using a sentence-internal, distance-weighted approach to better understand the sentiment in relation to the proximity of entities.

We will give more details during the coming tutorials.

b) Implementing Sentiment Analysis for Selected Main Characters

- Apply your chosen sentiment analysis tool to the text, focusing on areas where entities are identified. E.g. by simply computing a sentiment score for each sentence with a main character. Or a bit more complex by assigning sentiment scores to each words and by using the distance of entities to sentiment-laden words within a sentence (see typical distance-weighted formula for such an approach below). This focuses the analysis to a specific entity.
- Structure the results in a JSON file. Associate each entity with its sentiment scores, considering both their occurrence within individual sentences and across different chapters.
- Develop a method how to aggregate sentiment scores over chapters and the full book.

Example of a Distance-Weighted Sentiment Scoring Formula: Consider the sentence "Alice, though worried, admired the beautiful garden." For the entity "Alice," we identify the sentiment words "worried" ($pol(w) = -1$) and "beautiful" ($pol(w) = +1$). The distances in tokens are 2 and 5, respectively. Using the formula $score(e) = \Sigma_p \frac{pol(w)}{dist(e,w)}$, the sentiment score for "Alice" is calculated as $\frac{-1}{2} + \frac{1}{5} = -0.3$, indicating a slightly negative sentiment.

Deliverables:

- A JSON file named `BookTitle_Sentiment.json` summarizing the sentiment analysis. Detail sentiment scores for each entity, along with their context in the text.
- The code for sentiment analysis, in a Python script (`.py`) or Jupyter Notebook (`.ipynb`), with detailed annotations explaining your chosen methodology and any specific decisions made during the analysis.

This part of the project assesses your ability to apply sentiment analysis techniques creatively and document your analytical process effectively.

Part 3: Visualization of Results (5 points)

Objective: Visually represent the findings from the text mining process, focusing on the dynamics of entities and sentiments over time and narrative structure.

[2] https://neptune.ai/blog/sentiment-analysis-python-textblob-vs-vader-vs-flair

a) Selecting Visualization Tools

- Investigate and decide on an appropriate visualization tool. A basic option is Matplotlib. Seaborn or Yellowbrick produce nicer diagrams and plots.
- Ensure the chosen tools can effectively represent complex data, like time-series data or relationship networks.

b) Creating Visualizations

- Design visualizations that effectively depict the dynamics of entities over the narrative timeline and the evolution of sentiments across chapters.
- Create visualizations showing semantic similarity between entities based on sentiment scores or contextual usage.
- Make sure visualizations are easy to understand, provide useful information, and show the data well.
- For each of your chosen visualisations, you should have an example for your grep results and your results using more advanced methods.

Example: Visualizing Character Sentiment Over Time

To visualize changes in a character's sentiment throughout the narrative, use a line graph plotting chapters on the x-axis and sentiment scores on the y-axis.

Python Code for Sentiment Visualization: Below is a Python script using Matplotlib for this purpose:

```
import matplotlib.pyplot as plt
import json

# Load your sentiment analysis results
with open('BookTitle_Sentiment.json', 'r') as file:
    sentiment_data = json.load(file)

# Plotting sentiment for a character
chapters = [chapter for chapter, sentiment in sentiment_data['Alice'].items()]
sentiments = [sentiment for chapter, sentiment in sentiment_data['Alice'].items()]

plt.figure(figsize=(10, 6))
plt.plot(chapters, sentiments, marker='o', linestyle='-', color='b')
plt.title('Sentiment of Alice Over Time')
plt.xlabel('Chapter')
plt.ylabel('Sentiment Score')
plt.grid(True)
plt.show()
```

Deliverables:

- A collection of visualizations in formats like JPEG, PNG, or as a Jupyter Notebook, showing timelines, sentiment trends, and other graphical representations of the NER and sentiment analysis results.
- The code used for creating these visualizations, submitted either as a Python script (`.py`) or a Jupyter Notebook (`.ipynb`), with annotations explaining your visualization process and choices.

This part assesses your ability to visually interpret and communicate data, an essential skill for presenting complex analytical results.

Part 4: Reflection and Discussion (5 points)

Objective: Reflect on the project experience, insights gained, and consider the real-world application of these techniques.

a) Learning and Challenges

- Comparing the results of grep and the more in-depth NLP analysis, what are the key insights?
- You were asked to visualise both methods. How do they differ?
- Summarize key learnings, focusing on technical skills and literary insights.
- Briefly discuss major challenges and how you addressed them, particularly moving from basic grep searches to advanced NLP.

b) Analysis Insights and Real-World Applicability

- Describe significant insights from the book analysis and how advanced NLP tools enhanced your initial grep findings.
- Reflect on how these techniques and insights could apply in real-world contexts, like social media analysis or other literary works.

Deliverables:

- A concise `discussion_project.txt` file covering the learning experience, project insights, and real-world applicability.

This part encourages critical reflection on your journey through the project, from initial text analysis to complex NLP applications, and how these skills translate beyond the project.

Feedback

We welcome any feedback regarding this project. How long did you work on it? What parts were most challenging?

Good Luck and have fun with the project!