Course Project: Sentiment Analysis of Two Shakespeare Characters Using Flair and GPT Chat Completion API

PCL 1 and PfL, Fall Semester 2025

Deadline for Part 1: 10.12.2025, 18:00 Deadline for Part 2: 22.12.2025, 18:00

This project is a major component of the course and is designed to help you apply your programming skills to a more complex, real-world problem. The 3 points awarded for this project reflect a workload approximately three times that of a regular exercise (1 point). It also introduces you to typical challenges in applied programming, such as integrating external tools and evaluating data quality on selected subsets.

Note: If you have questions or run into issues, please reach out via the course forum, email, or during the tutorial sessions. We are here to support you throughout the project.

Remarks on submission

- Working in pairs is **mandatory**. Only one student should submit the project on behalf of the team. Use the following naming convention for both parts:
  `username1_username2_(pcl1|pfl)_ex06_part_(1|2).zip`.
- Include the full names and matriculation numbers of both team members in the submission form.
- Add a **doc-string to every function**, along with type hints and concise comments. This not only helps the tutors understand your code, but also supports your own debugging and development process.
- **Grading:** You can earn a total of 3 points. Each part specifies the maximum points achievable. Full marks require not only correct implementation but also well-structured code, including meaningful comments, doc-strings, and appropriate use of functions and data structures. If you choose to implement an advanced improvement suggestion that goes beyond the required processing steps, it may be used to compensate for any missing points (up to the 3-point maximum).
- **Submission:** There are **two deadlines**. Submit Part 1 by **10.12.2025 at 18:00** via the designated OLAT block. Submit Part 2 by **22.12.2025 at 18:00**, also via OLAT. Late submissions may result in point deductions.
- **Important:** Full points can only be awarded if both parts are submitted on time.

Note on the Use of AI-assisted Code Generation

We recommend programming as much as possible manually to **train your coding skills for the exam**. If you are stuck, you may use ChatGPT or other code-generating tools, provided that their use is transparently documented in your code, in line with Faculty guidelines. You should be prepared to explain any part of the submitted code. If submitted content exceeds the expected course level, an individual oral exam may be required to verify your understanding, or points might be deducted.

Example comment: `# function generated by ChatGPT with minor edits`

Introduction

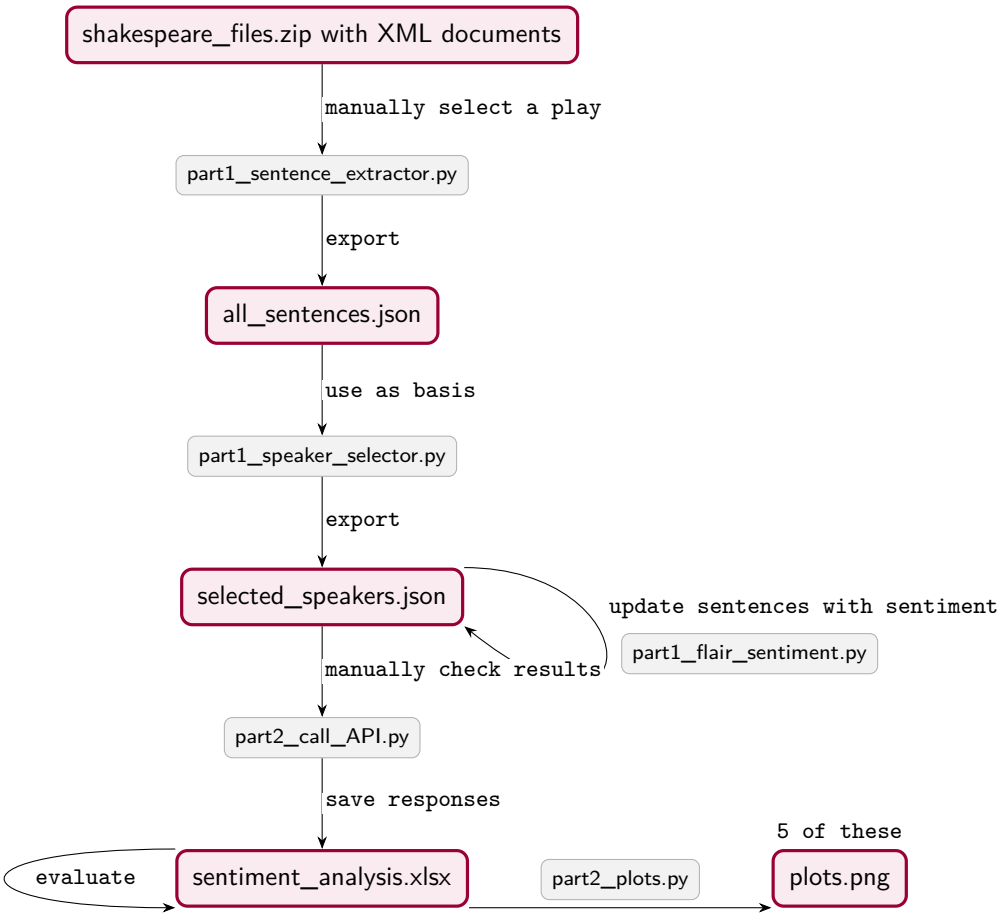

Figure 1: Data and processing flow of the project

Sentiment analysis refers to the automatic detection of emotional tone in textual data. In this project, you will use XML files of Shakespeare plays from the NLTK corpus to analyze how the sentiment and emotions of selected speakers evolve throughout a play. The project follows a structured pipeline: in Part 1, sentences are extracted from the XML files; next, main speakers are selected and their dialogues exported to a JSON file. Sentiment analysis is then applied using `flair`, followed by a more fine-grained emotion classification via the OpenAI Chat Completion API in Part 2, based on NRCLex's eight emotions (anger, anticipation, disgust, fear, joy, sadness, surprise, trust). The results are compiled into an Excel file for manual evaluation of a subset of the automatic analyses. Finally, visualizations on the full dataset are produced for aggregated interpretation.

Project Structure

Figure 1 illustrates the project workflow, outlining the data flow and key processing steps. Each step corresponds to a separate Python script that you will develop. Skeleton code is provided for each part, which you must complete and submit. Use the predefined functions as a basis and expand them where needed. You may add additional functions, but any removal of predefined functions must be clearly justified.

a) **Sentence Extraction:** Parse the selected XML Shakespeare file to extract all spoken sentences, along with metadata such as act, scene, sentence number, and speaker. Save the results to a JSON file.

b) **Speaker Selection:** Identify two main characters from the extracted data and export their dialogue to a second JSON file for further analysis.

c) **Sentiment Analysis:** Use `flair` to assign sentiment scores, then call the OpenAI Chat Completion API to annotate sentences with emotion labels. Store the results in an Excel file.

d) **Manual Evaluation:** Review a random subset of the API-annotated sentences (see Part 2 for details on annotation responsibilities and sample size).

e) **Plotting Sentiment Development:** Generate visualizations to illustrate changes in sentiment across the play.

The following sections provide detailed instructions for implementing each component of the project pipeline.

1 Part 1: Data Preprocessing and `flair` Sentiment Analysis

1.1 Parsing XML (1 Point)

1.1.1 Tips for Understanding the Corpus Format

The Shakespeare plays provided by NLTK are XML files. For guidance on the format and how to process them, refer to the provided Jupyter notebook.

1.1.2 Sentence Extraction

Your first task is to extract all spoken sentences from a selected Shakespeare XML file. Use the script `part1_sentence_extractor.py` for this step. You may choose any play from the corpus.

For each sentence, record the corresponding act, scene, speaker, and sentence number. Save the output in a JSON file named `all_sentences_{your_play}.json`. Consider the structure of your JSON carefully to ensure it supports the next steps in the pipeline.

1.1.3 Speaker Selection

From the extracted sentences, select two main characters for analysis. Each character must appear in every act and speak **at least 50 sentences**. Use the script `part1_speaker_selector.py` for this step to verify these conditions programmatically.

Export the selected speakers and their associated sentences to a second JSON file, which will serve as input for the sentiment analysis: `selected_speakers_{your_play}.json`.

1.2 Sentiment Analysis with `flair` (0.5 Point)

Apply sentiment analysis using `flair` on the JSON file containing the selected characters and their sentences. Annotate each sentence with its sentiment score and update the JSON structure accordingly.

Use the script `part1_flair_sentiment.py` for this step.

2 Part 2: LLM-based Sentiment and Emotion Analysis, Manual Evaluation and Visualization

2.1 OpenAI API Analysis, Evaluation and Visualization (1 Point)

2.1.1 OpenAI Chat Completion Sentiment and Emotion Analysis

After applying sentiment analysis with `flair`, use the script `part2_call_API.py` for this step to analyze all sentences with a sentiment confidence score of approximately 0.9 or higher. Starter code and an initial prompt are provided for guiding the sentiment and emotion analysis using structured JSON output from GPT-4o-mini or GPT-4o.

Your task is to refine and improve this prompt so the language model can perform the sentiment and emotion classification effectively. For emotions, use NRCLex's eight categories: anger, anticipation, disgust, fear, joy, sadness, surprise, trust.

Save the `flair`- and GPT-annotated sentences, along with their metadata (act, speaker, etc.), in an Excel file using the `openpyxl` package. Name the output file: `sentiment_analysis_{your_play}.xlsx`.

2.1.2 Manual Evaluation

Manually evaluate the annotated sentences to verify whether the assigned sentiment and emotion labels reflect the intended meaning of the text.

Each team member must annotate at least 25 randomly selected sentences *per speaker* in separate columns of the Excel file.[1] One person should focus on sentiment polarity (positive, negative, neutral), and the other on basic emotions.

To ensure that manual annotation is unbiased, hide the columns containing the `flair` and GPT annotations during your evaluation. Manual labels must be made independently, without consulting machine-generated outputs. Add additional columns with your manual analysis and use the same labels as the systems in order to easily compare agreement. We recommend doing this on a copy of the Excel file to prevent accidental overwriting by scripts.

After completing the manual annotations, compare them quantitatively with the system outputs. Compute agreement and disagreement scores for both sentiment and emotion labels. You may do this directly in Excel or by writing a separate Python script that takes the manually annotated subsample as input. Report these numbers in your submission.

2.1.3 Numerical Evaluation of Automatic Sentiment/Emotion Results

Compute concise summary statistics based on the `flair` and OpenAI annotations. At a minimum, report:

- the number of positive, negative, and neutral sentences per character,
- sentiment distribution across acts or scenes,
- the number of highly emotional sentences (`flair` confidence ≥ 0.90),
- any additional counts that support your interpretation of sentiment dynamics.

These quantitative measures will form the basis for your discussion of the emotional trajectories of the selected characters.

[1] See this post on StackOverflow on how to randomize rows in Excel: `https://stackoverflow.com/questions/11477546/how-to-randomize-excel-rows`

2.1.4 Visualization of Per-Character Sentiment Development over Acts

Create at least three meaningful plots to summarize the sentiment data. Make sure to use all data points of your selected characters, not only the manually annotated ones.

You can, for example, visualize:

- positive vs. negative sentence counts per act,
- sentiment distribution across scenes,
- comparison of overall sentiment trends between the two characters.

In addition, generate one sentiment-development plot *per character*. Use the script `part2_plots.py` for this step. Each plot should illustrate how sentiment evolves over the course of the play (e.g., across acts or scenes) and be saved as `play_charactername_sentiment_development.png`.

In total, submit five figures: three summary plots and two character-specific progression plots.

2.2 Reporting (0.5 Points)

Document your observations in a short report (`discussion.txt`). Include:

- the criteria used to evaluate sentiment accuracy,
- challenges encountered during manual annotation and comparison to system outputs (`flair` and GPT),
- examples of sentences with clear sentiment annotation (where the manual annotation matched the system output),
- examples where sentiment detection was ambiguous or incorrect (compared to your manual annotation),
- interpretation of the created plots, and
- suggestions on how to improve automatic sentiment analysis for Shakespearean texts, based on system or data preprocessing weaknesses.

If you choose to implement an advanced improvement that goes beyond the required steps, it may be used to compensate for any missing points in the project grading (maximum of 3 points in total).

Feedback and Evaluation

- Did ChatGPT perform as expected? Summarize your experience.
- Did you find anything surprising? What are the strengths and weaknesses of the approach?
- What part was the most difficult for you? Where did you struggle the most?
- How long did you work on the project?
- Feel free to share any feedback regarding the project.

Deliverables

Submit the following files in a `.zip` folder:

- Python scripts: `part1_sentence_extractor.py`, `part1_speaker_selector.py`, `part1_flair_sentiment.py`, `part2_call_API.py`, `part2_plots.py`

- JSON files: `all_sentences_{your_play}.json`, `selected_speakers_{your_play}.json`
- Excel file: `sentiment_analysis_{your_play}.xlsx` with automatic and manual annotations.
- Visualizations: `plot_(1|2|3).png`, and two `play_charactername_sentiment_development.png`
- Report: `discussion.txt`

Have fun exploring sentiment and emotion analysis of good old Shakespeare with generative AI!