Social Computing Homework 1 Documentation

Liam Olausson

February 2025

Contents

1 Introduction

This project is designed to process text-based chat conversations, analyze participant behavior, and obtain insights into language patterns, particularly focusing on pronoun usage and named entity associations. Using natural language processing (NLP) techniques, the pipeline performs tasks such as tokenization, part-of-speech tagging, named entity recognition, and statistical analysis.

The pipeline consists of three main modules:

- `main.py`: Orchestrates the workflow, using other modules to process input files and generate outputs.
- `nlp_conversation_processor.py`: Handles preprocessing, tokenization, part-of-speech tagging, and entity recognition.
- `pronoun_entity_counter.py`: Analyzes tokenized data for pronoun usage and compares results by named entities or specified gender groups.

Output files are saved in a subdirectory called `OutputFiles`.

2 Project Structure

The project follows a modular design:

1. `main.py`: The entry point of the project. It coordinates input processing and calls functions from other modules to:
    - Tokenize text and extract linguistic features.
    - Analyze pronoun usage for individual entities and gender groups.
    - Save results to specified output files.
2. `nlp_conversation_processor.py`: Uses NLP libraries (e.g., SpaCy's transformer-based model) to process conversational text. It produces structured tokenized outputs, including part-of-speech tags and named entity annotations.
3. `pronoun_entity_counter.py`: Reads the structured output produced by `nlp_conversation_processor.py` and performs statistical analysis of pronoun usage for named entities. It also computes group averages (e.g., for males and females) based on user-defined name lists.

3 main.py

This is the main orchestrator of the pipeline. It interfaces with the user, takes in input filenames, and ensures proper coordination between the modules.

Workflow

1. Prompt the user to provide the input file name.
2. Check if the file exists and create the `OutputFiles` directory if necessary.
3. Call `process_conversation` (in `nlp_conversation_processor.py`) to preprocess and tokenize the text. Save the tokenized file as `output_tokenized_{filename}`.
4. Call functions in `pronoun_entity_counter.py` to analyze pronoun usage:
   - Extract preprocessed input using `preprocess_input_file`.
   - Analyze pronoun statistics per entity using `analyze_conversation`.
   - Compare pronoun averages by grouping participants using `compare_gender_pronouns`.
5. Generate and save the analysis results to `pronoun_analysis_{filename}`.

Functions

- `main()`: The entry point function. It manages file input, processing, and output generation in a coordinated manner.

4 nlp_conversation_processor.py

This module processes raw conversation text into a structured tokenized output using SpaCy's transformer-based NLP pipeline. It includes part-of-speech tagging, named entity recognition, and sentence segmentation.

Dependencies

- `re`: For regular expression operations to clean and process text.
- `spacy`: For transformer-based NLP processing (e.g., tokenization, POS tagging, and entity recognition).

Functions

`clean_turn_text(turn)`:

- Cleans a single turn by removing metadata (e.g., speaker name and timestamp) from the conversation line.
- **Input:** A string representing a conversation turn.
- **Output:** A cleaned string containing only the dialogue message.

`process_conversation(conversation_text, output_file)`:

- Processes each turn in a conversation text by:
  - Cleaning metadata (speaker and timestamp).
  - Tokenizing the cleaned text using SpaCy's NLP pipeline.
  - Extracting linguistic features: part-of-speech tags, lemmas, and named entities.
- **Input:** Raw conversation text and the output file path for saving results.
- **Output:** Writes a structured file with tokenized data, including entities and POS tags.

5 pronoun_entity_counter.py

This module analyzes the processed conversation text specifically for pronoun usage. It computes statistics for individual entities and groups.

Functions

`preprocess_input_file(file_path)`:

- Reads structured conversation data and extracts metadata for processing.
- **Input:** Path to the raw conversation file.
- **Output:** A list of tuples linking participants to their conversation turns.

`analyze_conversation(turn_data, tokenized_file)`:

- Analyzes tokenized input for total and average pronoun usage by named entities.
- **Input:**
  - `turn_data`: Preprocessed participant-turn mapping.
  - `tokenized_file`: Path to the tokenized output file.
- **Output:** A dictionary mapping entities to their pronoun statistics.

`compare_gender_pronouns(pronoun_stats, male_names, female_names)`:

- Computes the average pronoun usage for male and female participants based on predefined name lists.
- **Input:**
  - `pronoun_stats`: Results from `analyze_conversation`.
  - `male_names`: List of male names.
  - `female_names`: List of female names.
- **Output:** Two average values: pronoun usage per sentence for males and females.

6 Example Workflow

Input File (conversation.txt)

```
shelly (11:25:37 AM): Howdy, howdy.
mara (11:26:09 AM): hi shelly
bob (11:26:23 AM): I am here?
```

Intermediate File (OutputFiles/output_tokenized_conversation.txt)

Contains tokenized output:

```
turn 1
    sentence: Howdy, howdy.
    token: Howdy pos tag: INTJ lemma: howdy
...

turn 2
    sentence: Hey shelly
    token: hi pos tag: INTJ lemma: hi
    token: shelly pos tag: PROPN lemma: shelly
    named entities: shelly type: PERSON
...
```

Final Output File (OutputFiles/pronoun_analysis_conversation.txt)

```
Pronoun Usage Analysis by Entity:
Shelly: Total Pronouns = 3, Average Pronouns per Sentence = 0.00
Bob: Total Pronouns = 2, Average Pronouns per Sentence = 1.00
Mara: Total Pronouns = 1, Average Pronouns per Sentence = 0.00

Pronoun Usage Analysis by Gender:
Male Average Pronouns per Sentence: 1.00
Female Average Pronouns per Sentence: 0.00
```