Analysis of Variable Clitics in Chat Transcripts

Introduction

This document explains the code used to analyze chat transcripts for sentences with variable clitics. Variable clitics are clitic pronouns that can appear before the first verb or attached to the second verb in a sequence of verbs. This script processes chat transcript files, identifies sentences with variable clitics, and outputs the results to a CSV file.

Code Overview

The script is divided into several functions, each responsible for a specific part of the processing:

1. **Importing Libraries and Setting Up Paths**
2. **Defining Helper Functions**
3. **Main Analysis Function**
4. **Running the Script**

1. Importing Libraries and Setting Up Paths

This section imports necessary libraries and sets up the directory path for the `tagged_cha_reader` module.

```
import sys
import os
import csv

# Add the directory containing tagged_cha_reader to the Python path
module_path = '/Users/danieltregubenko/Desktop/tagged-transcript-processing-mair
if module_path not in sys.path:
    sys.path.append(module_path)

# Now you can import the tagged_cha_reader module
import tagged_cha_reader
```

Screenshot: Directory Setup

Explanation:

- **sys, os, csv**: These libraries are standard Python libraries used for system operations, file handling, and CSV file operations.
- **module_path**: This variable contains the path to the directory where the `tagged_cha_reader` module is located. This path needs to be added to the Python path to import the module successfully.

2. Defining Helper Functions

This section defines helper functions to check for clitics and verbs, and to remove part-of-speech tags from sentences.

contains_clitic_pronoun(tokens)

```python
def contains_clitic_pronoun(tokens):
    """
    Checks if the given list of tokens contains clitic pronouns either before the
    or attached to the second verb.
    """
    clitic_pronouns = {"me", "te", "se", "lo", "la", "le", "les"}
    combined_clitics = {f'{cl1}{cl2}' for cl1 in clitic_pronouns for cl2 in clitic

    for token in tokens:
        word, pos = token.split('.')
        if pos == 'PRON' and word in clitic_pronouns:
            return True, word
        if pos == 'VERB':
            # Check if clitic is attached to the verb
            verb = word
            for clitic in clitic_pronouns.union(combined_clitics):
                if verb.endswith(clitic):
                    return True, verb
    return False, None
```

Screenshot: contains_clitic_pronoun Function

Explanation:

- **clitic_pronouns**: A set of clitic pronouns.
- **combined_clitics**: A set of combined clitics formed by concatenating two clitic pronouns.
- **Token Loop**: The function loops through the tokens to check for clitics before the first verb or attached to the second verb.

Customizable Part:

- **clitic_pronouns**: This set can be customized if the test subjects use different clitic pronouns.

has_two_verbs_with_gap(tokens)

```python
def has_two_verbs_with_gap(tokens):
    """
    Checks if there are two verbs in the given list of tokens.
    The verbs can be next to each other or have specific allowed words between th
    """
    allowed_gaps = {
        1: {"a", "que", "dando"},
        2: {"a a", "de que"}
    }

    verb_positions = [i for i, token in enumerate(tokens) if token.split('.')[-1]
    for i in range(len(verb_positions) - 1):
        gap_size = verb_positions[i + 1] - verb_positions[i] - 1
        if gap_size in allowed_gaps:
            gap_words = " ".join([tokens[j].split('.')[0] for j in range(verb_pos
            if gap_words in allowed_gaps[gap_size]:
                return True
    return False
```

Explanation:

- **allowed_gaps**: A dictionary defining the allowed words that can appear between two verbs.
- **Verb Loop**: The function identifies the positions of verbs and checks for allowed gaps.

Customizable Part:

- **allowed_gaps**: This dictionary can be customized based on the specific words used by test subjects.

untag_sentence(sentence)

```python
def untag_sentence(sentence):
    """
    Removes the part-of-speech tags from the sentence.
    """
    return " ".join([word.split('.')[0] for word in sentence.split()])
```

Explanation:

- **Sentence Processing: This function removes part-of-speech tags from the sentence, leaving only the words.**

3. Main Analysis Function This section defines the main function that processes chat files to identify sentences with variable clitics.

```
def analyze_variable_clitics(input_dir, output_file):
    """
    Analyzes chat files for sentences with variable clitics and generates a CSV wi
    """
    filenames = [f for f in os.listdir(input_dir) if f.endswith('.cha') or f.endswi
    print(f"Found {len(filenames)} files in the directory.")
    results = []

    for file_name in filenames:
        print(f"Processing file: {file_name}")
        file_prefix = os.path.splitext(file_name)[0]
        file_path = os.path.join(input_dir, file_name)

        with open(file_path, 'r', encoding='utf-8') as file:
            lines = file.readlines()

        sentences = [line[5:].strip() for line in lines if line.startswith('%pos:')
        for sentence in sentences:
            if sentence.strip() == "":
                continue
            tokens = sentence.split()
            while has_two_verbs_with_gap(tokens):
                contains_clitic, clitic_word = contains_clitic_pronoun(tokens)
                if contains_clitic:
                    untagged_sentence = untag_sentence(sentence)
                    print(f"Untagged Sentence: {untagged_sentence}\nClitic Word: {
                    results.append([file_prefix, untagged_sentence, clitic_word, ""

                # Remove the first verb pair to continue checking the rest of the s
                first_verb_idx = next(i for i, token in enumerate(tokens) if token.
                tokens = tokens[first_verb_idx + 1:]

    print(f"Found {len(results)} sentences with variable clitics.")

    # Write results to CSV in the same directory as input
    output_path = os.path.join(input_dir, output_file)
    with open(output_path, 'w', newline='', encoding='utf-8') as csvfile:
        writer = csv.writer(csvfile)
        writer.writerow(['ID', 'Sentence', 'Clitic', 'Clitic Position'])
        writer.writerows(results)
    print(f"Results written to {output_path}")
```

Explanation:

- **File Processing Loop**: The function processes each chat file in the specified directory.
- **Sentence Processing Loop**: The function processes each sentence, checking for variable clitics and recording the results.

Customizable Parts:

- **input_dir**: The directory containing the chat files.
- **output_file**: The name of the output CSV file.

4. Running the Script

This section sets the input directory and output file, and calls the main analysis function.

```
# Update the input directory path to the correct one
input_dir = '/Users/danieltregubenko/Desktop/Clitic'  # Directory
output_file = 'variable_clitic_results.csv'
analyze_variable_clitics(input_dir, output_file)
```

Screenshot: Running the Script

Explanation:

- **Directory Path**: This variable should be updated to point to the directory containing the chat files.
- **Output File**: This variable should be updated to specify the desired output CSV file name.

Conclusion

This document explains the functionality and customization options of the script used to analyze chat transcripts for variable clitics. By following the instructions and updating the necessary parts, users can adapt the script to their specific needs.
