Tracing Linguistic Footprints of ChatGPT Across Tasks, Domains, and Personas

Brief Description

This repository contains the code and data supporting the research paper "Tracing Linguistic Footprints of ChatGPT Across Tasks, Domains and Personas in English and German." The project explores how the output of large language models like ChatGPT differs from human-generated text and analyzes the impact of task-specific prompting on linguistic features in both English and German texts.

Usage

Text generation: text_generation/

The script `generate.py` sends requests to the OpenAI-API. As input it takes a JSON file in the following form:

```
{
  "file1": {
    "title": "very interesting and engaging topic",
    "prompt": "part of the text to use for the prompt",
    "text": "the rest of the text"
  },
  "file2": {

  }
}
```

The `make_json.py` script can be used to create such a file from a collection of txt files (see below). From the input for every file in the JSON the API is called to generate a text of more than 500 tokens. After making sure that both the remainder of the human text (without the pompt) and the machine generated text are of the same length by truncating the shorter one, it saves them in two separate folders called `human` and `machine`. An example call looks like this:

```
generate.py gpt-3.5-turbo path_to_input_file.json de
```

In this example the model gpt-3.5-turbo is used. Additional positional arguments are the path to the input and the language of the document (needed for the tokenizer). This will create an output folder in the directory from which the script is run with two subfolders `human` and `machine`. Optionally the output directory can be specified with the flag `--outfolder`, for more info on the optional arguments see `generate.py --help`.

Generate personas:

```
bash call_generate_personas.sh
```

- **Parameters**:

    - model: `gpt-4, gpt-3.5-turbo-16k`
    - outfolder: `../generated_data` in this folder subfolders task/corpus/system will be created
    - infolder: `../data_collection/100_files_json/` all the JSON for generation
- **Calls python script**: `generate_personas.py`
- **Output file structure**: `f"{outputdir}/{corpus}/{task}/{file_counter}.txt"`

`prompts.json` contains all the prompts and personas

Feature extraction: feature_extraction/scripts/

The metrics for Sophistication, Lexical and Morphological richness are calcualted using BiasMT tool.

Sophistication

Step 1 `bash concatenate_files.sh`

- Input: `../../generated_data/`
- Output: `../concatenated_data/`
- Function: Concatenates all corpus files into one txt file in the data folder

Step 2 `bash sophistication.sh`

- Output: `../results/sophistication/sophistication_scores.csv`

Lexical richness

`bash lxr_scores.sh`

- Input: `../../generated_data/`
- Output: `../results/lexical_richness`

Morphology for the German corpora

Step 1 `bash create_most_freq_vocs.sh`

- Input: `../../generated_data/`
- Output: `freq_voc/`, `lemmas/`
- Function: Extracts vocabulary of most frequent words

Step 2 `bash mrph_all.sh`

- Language: `de`
- Input: `../../generated_data/`
- Output: `../results/morphology/${corpus_name}`
- Function: Measure the surprisal levels within the inflectional paradigms of the German lemmas and Produces Shannon entropy and Simpson diversity metrics

Extract Features with TextDescriptives

Link to the TextDescriptives library

features_list.py contains several dictionnaries with feature names:

- features_list is a list of TextDescriptives features
- features_custom is a list of custom-added feature names
- features_to_visualize_dict is a dictionnary with feature names used by textDescriptives and throughout the project as keys and modified feature names as values
- features_raw_counts is a list of features that are measured in raw counts

Extract features and sort results by feature, language and domain

Main Script: `bash run_extract_features.sh`

- **Description**: Executes three Python scripts to extract linguistic features, reorganize results, and transform dataframes for further analysis.
- **Executes**:

  1. `extract_features.py --corpus $corpus`

     - Function: Iterates through all specified corpora to extract features using the TextDescriptives library, including a custom formula for German Flesch Reading Ease.

  2. `combine_results_per_lang_domain.py`

     - Function: Restructures data into a more accessible format, sorting by individual features, language, and domain.
     - Iterates through: `../results/per_corpus/{corpus}`
     - Output Directories:
       - Per Feature: `../results/per_feature/{feature_to_extract}/{corpus}.csv`
       - Per Language: `../results/per_language/{language}/{feature}.csv`
       - Per Domain: `../results/per_domain/news/{language}/{feature}.csv`

  3. `transform_dataframe.py -f $feature_type`

     - Function: Pools together and formats results for morphological and lexical features.
     - Inputs:
       - Morphological Features: `../results/morphology/{corpus}.csv`
       - Lexical Features: `../results/lexical_richness/{corpus}.csv`
     - Output Directories:
       - Per Feature: `../results/per_feature/{feature_to_extract}/{corpus}.csv`
       - Per Language: `../results/per_language/{language}/{feature}.csv`
       - Per Domain: `../results/per_domain/news/{language}/{feature}.csv`

Citation

```
@article{YourLastName2024,
  title={Tracing Linguistic Footprints of ChatGPT Across Tasks, Domains
and Personas in English and German},
  author={Anastassia Shaitarova, Nikolaj Bauer, Jannis Vamvas, Martin
```

```
Volk},
  journal={Journal Name},
  year={2024},
  volume={xx},
  pages={xxx-xxx}
}
```