

# 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 directory

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 prompt) 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`.

To **generate personas**:

```
bash call_generate_personas.sh calls generate_personas.py
```

`prompts.json` contains all the prompts and personas

## Feature extraction directory

### Sophistication

**Step 1** Concatenate all corpus files into one txt file in the data folder `bash concatenate_files.sh`:  
prompts\_n\_coherence/data/ --> feature\_extraction/data

**Step 2** Run the main script `bash sophistication.sh`: -->  
feature\_extraction/results/sophistication\_scores.csv

### Lexical richness

`bash lxr_scores.sh` prompts\_n\_coherence/data/

### Morphology

**Step 1** Extract vocabulary of most frequent words

`bash create_most_freq_vocs.sh` : prompts\_n\_coherence/data/ --> scripts/freq\_voc/, scripts/lemmas/

**Step 2** Run diversity analysis

#### for single file

```
python3 shannon_pairwise.py -f
~/switchdrive/IMAGINE_files/datasets/wmtnews21/wmtnews_test_de_A.txt -l de
-sys A_wmt -v freq_voc/wmtnews_test_de_A.freq_voc > test.txt
```

**for multiple files in a directory if considering the top 1000 most frequent lemmas with more than 1 morphological form**

lang = {"en", "de"}

```
bash shannon_1000_mostfrequent_script.sh
~/switchdrive/IMAGINE_files/chatGPT/project_2/final_files_simple_prompt/{c
orpus} lang
```

**if choosing all lemmas with more than one morphological form**

lang = {"en", "de"}

```
bash mrph_all.sh
~/switchdrive/IMAGINE_files/chatGPT/project_2/final_files_simple_prompt/{c
orpus} lang
```

## Extract Features with TextDescriptives

**features\_list.py** contains several dictionaries 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 dictionary 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:** `run_extract_features.sh`

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

```
bash run_extract_features.sh
```

- **Executes:**

1. **Script:** `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. **Script:** `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. **Script:** `transform_dataframe.py -f $feature_type`

- **Function:** Pools together and formats results for morphological and lexical features.
- **Sources:**
  - 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}  
}
```