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`.

Feature extraction

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/{corpus} lang
```

if chosing 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/{corpus} lang
```

Extract Features with TextDescriptives

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

`bash run_extract_features.sh` executes 3 python scripts:

1. `python3 extract_features.py --corpus $corpus` iterates through all corpora and extracts features with TextDescriptives, includes custom formula for German Flesch Reading Ease

2. `python3 combine_results_per_lang_domain.py` iterates through ../results/per_corpus/{corpus}.
   The script restructures data to ../results/per_feature/{feature_to_extract}/{corpus}.csv, ../results/per_language/{language}/{feature}.csv, and ../results/per_domain/news/{language}/{feature}.csv

3. `python3 transform_dataframe.py -f $feature_type` adds morphological features from ../results/morphology/{corpus}.csv and lexical features from ../results/lexical_richness/{corpus}.csv
   The results are written to ../results/per_feature/{feature_to_extract}/{corpus}.csv, ../results/per_language/{language}/{feature}.csv, and ../results/per_domain/news/{language}/{feature}.csv

create data