# Dataprep

## Standardize data format. Retrieve source sentences.

### Raw data

```
all_annotations_v1.json contains littrans data from
https://github.com/marzenakrp/LiteraryTranslation

wmt23/ contains en-de and de-en datasets from the WMT2023 testsets
```

**json2csv_littrans.py** : all_annotations_v1.json -> all_csv/

Parses a json file with annotations and creates csv files for each book (language pair): para (gpt3, human) and sent formatted as para (gpt3, nmt). Additionaly, extracts human preferences into a csv file output/littrans_annotators_choices.csv
Removes new lines within text chunks.

**txt2csv_wmt23.py** : wmt23/ -> all_csv/

Converts the WMT23 txt files to csv files, merging the source and target languages into one file. Para (human, gpt4), sent (nmt). Removes new lines within text chunks.

**run_csv2json4Llama.sh** : dataprep/all_csv/{lang}.para.human.csv -> inputs/source_para_json/

csv2json4Llama.py extracts source paragraphs into json files formatted for Llama.

**split_source_sents.py** : dataprep/all_csv/{lang}.para.human.csv -> inputs/source_sent_json

needs GPU

Preprocesses source paragraphs, standardizes punctuation based on lang prior to segmentation. Splits source texts into sentences. Writes json files formatted for Llama. Also writes txt files.

## Create translations

**translate_gpt.sh** : inputs/source_${level}_json/*.json -> dataprep/translated/${level}-level

translate_with_openAI.py uses OpenAI API to produce translations with GPT-3 and GPT-4. Script needs to be manually adjusted depending on level and model. Read annotation.

### Translating with Llama

1. Translate (needs 4 GPUs) work is done on a cluster
2. **run_json2csv4Llama.sh** : dataprep/llama_translations/llama_{level}_json -> dataprep/llama_translations/llama_{level}_csv

3. **clean_Llama_with_gpt4.py** : dataprep/llama_translations/llama_{level}_csv -> dataprep/llama_translations/llama_{level}_gpt4_cleaned/

   > Flags missing transaltions with NO TRANSLATION FOUND. Flagged lines are sent back to the model for re-evaluation, which produces flags: `<<WRONG STATEMENT, TRANSLATION FOUND>>`, `<<INACCURATE TRANSLATION>>`, and `<<CORRECT STATEMENT, NO TRANSLATION FOUND, because>>` Make sure to indicate the "id" number of the line where to start processing file.

4. Feed flagged src-tgt pairs back to Llama for re-translation.

5. **remove_gpt4_flags.py** : dataprep/llama_translations/llama_{level}_llama_fixed -> dataprep/translated/{level}-level

### Merging sentences into paragraphs

**run_merge_sents2paras.sh** : dataprep/translated/sent-level -> inputs/sents

merge_sents2paras.py merges target sentences into pargraphs by aligning them with the source paragrasphs via source sentences from translated/sent-level. Sentences are already preprocessed, but source paragraphs from inputs/source_para_json/${langs}.para.source.json are not preprocessed.

The script preprocesses all texts equaly, removes remaining translation artifacts, normalizes punctuation and spaces.

Outputs csv files that are ready for the analysis.

### Copy all remaining files into inputs

cp dataprep/translated/para-level/* inputs/paras/
cp dataprep/all_csv/*para* inputs/paras
cp dataprep/all_csv/*sent* inputs/sents

# Analysis

needs GPU

cd analysis

**bash run_analysis.sh** :

**python3 align_sents.py -l ${level}**

writes csv files with aligned sentences to output/aligned_sentences_{level}
writes results to results/{level}_n2m_scores.csv with ["lang", "system", "total_src_sents", "n2m", "n2mR", "length_var", "merges", "splits", "mergesRatio", "splitsRatio"]

**python3 calculate_xwr.py -l ${level}**

Performs word alignment and calculates cross word ratio (XWR)
writes all alignment data to output/alignments_per_file/

writes results to results/{level}_alignment_scores.csv with ["lang", "system", "all_alignments", "cross_alignments", "xwr_mean", "xwr_std"]

**python3 merge_csv.py -l ${level}**

Final dataframe: results/{level}_syntax_scores.csv