Dataprep

Standardize data format. Retrieve source sentences.

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

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

json2csv_littrans.py

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). Extracts human preferences into a csv file output/littrans_annotators_choices.csv Removes new lines within text chunks.

txt2csv_wmt23.py

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 -> csv2json4Llama.py

Iterates through all_csv/{lang}.para.human.csv and extracts source paragraphs into json files formatted for Llama.

split_source_sents.py needs GPU

Iterates through all_csv/{lang}.para.human.csv files, preprocesses source texts and standardizes punctuation based on lang prior to segmentation. Splits source texts into sentences. Writes json files formatted for Llama, writes txt files.

Create translations

translate_gpt.sh -> translate_with_openAI.py

Iterates through /inputs/source_${level}_json/*.json. Uses OpenAI API to produce translations with GPT-3 and GPT-4. Saves files to translated/${level}-level. 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** -> json2csv.py

   > Converts json files from llama_translations/llama_{level}_json to llama_translations/llama_{level}_csv

3. **clean_Llama_with_gpt4.py**

   > Iterates through llama_translations/llama_{level}_csv and 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>>` Writes files to llama_translations/llama_{level}_gpt4_cleaned/ 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**

> input_dir llama_translations/llama_{level}_llama_fixed output_dir = translated/{level}-level

Merging sentences into paragraphs

Merge the target sentences into pargraphs by aligning them with the source paragrasphs via source sentences Source sentences from translated/sent-level come 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.

bash run_merge_sents2paras.sh -> merge_sents2paras.py inputdir="translated/sent-level" outputdir="../inputs/sents"

Copy all remaining files into inputs

cp translated/para-level/* ../inputs/paras/ cp all_csv/*para* ../inputs/paras cp 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