README - Annotated Guideline for Finishing the German Translation of the RiSAWOZ dataset

Overview over the task + the files

A research group from Stanford works on the multilingual version of the RiSAWOZ dialogue dataset. This is a **task-oriented dialogue systems.** It was originally translated from Chinese to English, now the goal is to translate it from English to as many languages as possible.

Your task it to finish the translation from English to German.

**The current version of the project can be found in this repo: https://github.com/lena-schwert/dialogues/tree/hpi-de-risawoz/dialogues/risawoz

There are 3 files that need to be translated from English to German, creating dataset-specific mapping files in the progress. The maintainers of the projects supply a Guideline PDF to explain all steps.
You can find it here: `dialogues/risawoz/helper_files_translation_guide/Dataset Translation Guideline_received on 14032023.pdf`
This documents is a more detailed, additional help, since 1 of the 3 file and the 12 database files are already translated.

Here is an overview over what still needs to be done:

- Three splits (few-shot train, valid, test set) of translated RiSAWOZ dataset with "db_results" field
  --> **The "few-shot train" split of the dataset is already translated. The main task is to translate the valid and test split and make sure that they are consistent with the fewshot split, the mapping files and the translations in the databases.**
  - few-shot: 100 dialogues
  - valid + test: 600 dialogues each
- Two mapping files: `de2canonical.json` and `en2de_alignment.json`
  --> **These exist already but need to be extended with translations from the valid and test file**
- Translated databases in all 12 domains
  --> **these were already manually checked for translation quality and should not require any changes. However try to re-use these translations to stay consistent with the already translated dialogues!**

For questions beyond the guideline and these comments, you can contact Mehrad and Tianhao who are the maintainers for this project:
Mehrad Moradshahi: mehrad@stanford.edu
Tianhao Shen: thshen@tju.edu.cn

The existing e-mail correspondence between them and HPI people working on the project (Jim, Lena) is contained in this file: `dialogues/risawoz/helper_files_translation_guide/Emails with dialogues repo maintainers until June23.pdf`

Overview over the files in the Github repo:

- all files are contained in the `dialogues/dialogues/risawoz/` folder

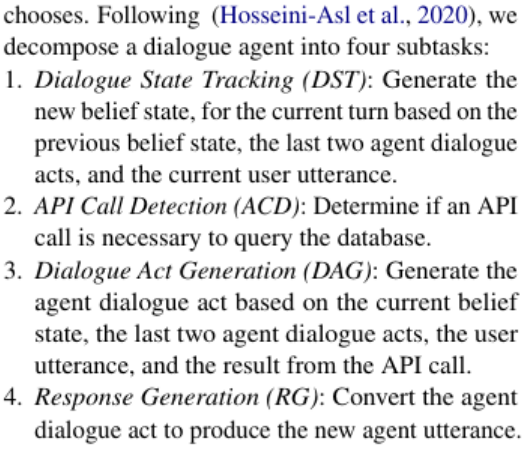
- `data`: at this level you can find the dataset files **that are already processed, i.e. the final version in the correct format**
  - `data/original`: this folder contains the dataset files in the original format before processing them according to the translation guideline
- `database` : contains one folder for each language with 12 files each, each file contains database entries for a different domain
- `helper_files_translation_guide`: this folder contains intermediary files necessary for following the steps in the guideline, the folders already contain the files that were used for processing the fewshot file. Change and extend them as needed.
  - `helper_files_translation_guide/db_utils`: contains additional scripts mainly for translating the database entries
  - `helper_files_translation_guide/official sample files`: these are files provided by the maintainers. Consult them if you are unsure how a file should look like.
- `scripts`: scripts used in the guideline.
- `src`
  - more scripts, these are used in the final "Validate your work" step
  - `src/knowledgebase/mappings`: contains the mapping files for all languages as mentioned above - **very helpful for comparing your work with that of the others**

If you are interested (not necessary for the task), you can get the associated paper from here:
https://arxiv.org/abs/2306.17674
(Published in the ACL Findings 2023)

Terminology

- **Task-Oriented Dialogue Agents/Systems** = " Task-oriented dialogue systems are designed to assist users in achieving specific goals. They can be divided into two categories by the implementation method: modularized and end-to-end" (Xiang et al. 2021)

- turn: the human and the dialogue agent have alternating turns. Each turn has 4 parts (source: X-Risawoz paper) :


chooses. Following (Hosseini-Asl et al., 2020), we decompose a dialogue agent into four subtasks:

1. *Dialogue State Tracking (DST)*: Generate the new belief state, for the current turn based on the previous belief state, the last two agent dialogue acts, and the current user utterance.
2. *API Call Detection (ACD)*: Determine if an API call is necessary to query the database.
3. *Dialogue Act Generation (DAG)*: Generate the agent dialogue act based on the current belief state, the last two agent dialogue acts, the user utterance, and the result from the API call.
4. *Response Generation (RG)*: Convert the agent dialogue act to produce the new agent utterance.

  - each turn
- domain: each database covers a different domain, e.g. "movie" or "hospital"
- slot: each database item has multiple slots, e.g. "production country or area", "type", "decade", etc for the "movie" database

```
{
    "production_country_or_area": "Taiwan",
    "type": "romantisch",
    "decade": "2010er",
    "star": "Ko Chen Tung",
    "director": "Giddens Ko",
    "title": "Du bist mein Augapfel",
    "name_list": "Kai Ko, Michelle Chen, Owodog und Steven Hao",
    "release_date": "19.08.2011 in Taiwan / 06.01.2012 auf dem chinesischen Festland",
    "film_length": "100 Minuten",
    "douban_score": "9.1"
},
```


- slot values: These are the different options each slot can have. The `en2de_alignment.json` file collects these slot values and their German translations, e.g. listing all the different movie types


```
"type": {
    "romantic movie": "romantischer Film",
    "comedy": "Comedy",
    "action movies": "Action",
    "action movie": "Action",
    "romantic": "romantisch",
    "horror": "Horror",
    "sci-fi": "Science-Fiction",
    "action": "Action"
```


How to set up the Github repo: fork the `dialogues` repo

When all files are ready, the goal is to create a Pull Request to the original `dialogues` repo that is here: https://github.com/stanford-oval/dialogues. This means that the German files will then be officially added to the project.

This means that it is necessary to **fork** their repo and work on Github. Below is a brief explanation on how to do this and then add the existing files as a new branch.

- **resources**
  - https://docs.github.com/en/get-started/quickstart/fork-a-repo
  - https://docs.github.com/en/pull-requests/collaborating-with-pull-requests/working-with-forks

1. Go to the page of the project (https://github.com/stanford-oval/dialogues), click Fork
   - copy only the main branch or all?
   - rename?
2. Clone locally
3. add original repository as an upstream source?
   - **If you set this up, you will be able to pull newer commits of the original repo and integrate them into your branch. This makes sense since the authors may sometimes edit data files or scripts.**
   - follow step 7 here: https://docs.github.com/en/get-started/quickstart/fork-a-repo#configuring-git-to-sync-your-fork-with-the-upstream-repository
   - further instructions on syncing a fork
   - check this with `git remote -v`
4. Create a new branch, e.g. naming it `hpi-de-risawoz`
5. **Commit the git project files you received together with this document to this new branch.**
   --> You can download the project files from this repo: https://github.com/lena-schwert/dialogues/tree/hpi-de-risawoz/dialogues/risawoz
   --> The hpi-de-risawoz branch is the correct ones to clone/download from!

Syncing with the remote upstream repo: "Update origin with changes from upstream"

- **Under which conditions can I do this?**
  - the *working tree should be clean*, i.e. there are no uncommitted changes
- in CLI
  - `git fetch upstream`: gets new branches and changes to
  - then merge the changes with my local forked repo:
    - e.g. I want to merge upstream changes to `main` with my `main`: `git merge upstream/main main`
    - prompts you for a commit message

Detailed comments on the "Dataset Translation Guideline" PDF

As mentioned above, the maintainers of the `dialogues` repo created a detailed guideline to guide along the multiple step, quite specific translation and dataset formatting process.
You can find the version with some highlights and comments here:
`dialogues/risawoz/helper_files_translation_guide/Dataset Translation Guideline_received on 14032023.pdf`

General comments:

- **read each step of the guideline carefully and then the additional comments below before you start working on a step**! They are meant to make the process as clear as possible.
- **However, since I completed the process translating only a single file (fewshot), the steps might slightly vary when doing them simultaneously for all three splits! Adapt scripts + commands as needed to make them work for you :) It is likely that you will need to merge (copy contents, remove duplicates) my alignment files with the ones you create!**
  --> I worked on Linux so there will be slight differences between operating systems!
- all files mentioned here are contained in respective folders within the `helper_files_translation_guide` folder
-
  - **adapt the file path for all command line paths**: might be easiest to use absolute paths, so make sure you add your file paths in all the commands below, this is denoted by `ADD_YOUR_PATH_HERE`
  - e.g. `python check.py --new_data_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_guide/Step_3_checkpy_script/new_data_path --annotation_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_guide/Step_3_checkpy_script/annotation_path`
- always use the scripts from the HPI (or the Korean) repo since there were some smaller errors in the original script!
  --> There are small but important differences between the scripts in the `hpi-de-risawoz` branch and the `main` branch of the dialogues repo!
- always check whether the resulting file makes sense and whether it really has the expected formats! Compare with the existing files or the repos from the other translations:
  - **Korean** (Link to the corresponding folder on the "Korean" branch of their forked repo: https://github.com/SungBalance/stanford-dialogues/tree/korean/dialogues/korean_annotation
    --> They also debugged the scripts so it helps to look at how they did it.

- `src/knowledgebase/mappings`: mapping files in all other languages

Step 0: Recreating the Python environment

1. create a virtual environment, e.g. using conda or venv
2. install MongoDB locally (needed for the last step)
3. run `pip install -e .` to install all requirements using pip

- the `-e` flag makes it editable?

Other files in the repo:

- `setup.py` indicates that the repo will eventually become a Python library
- `pyproject.toml` is similar to `setup.py`: https://stackoverflow.com/questions/62983756/what-is-pyproject-toml-file-for
  - this is a new standard that will replace `setup.py`
  - installation can be done with this file when running `python -m pip install .`

Step 1 + 2: Translate and annotate the dataset files using the provided annotation tool.

- creates files
  - CSV file with the translation
  - this is the respective folder: `Step_x_German_translation_with_annotation_tool`
- The translation guideline gives some good help here.
- The two files you need to load into the tool (one by one) are these:
  - `dialogues/risawoz/data/original/en_valid.json`
  - `dialogues/risawoz/data/original/en_test.json`
- Be very accurate with the slot annotation and make sure you don't miss any slot values in the English sentences (they are highlighted bold + underlined)
- Sometimes two slots are directly next to each other ans might be confused for a single one!
- The better the quality of annotation and translation here, the less work you have later on correcting mistakes.

How to run the tool:
Windows: simply run the exe file
--> download it from here:
https://drive.google.com/drive/folders/1f3_WhW6YYBVmGSP3zBllPvY6JYnn74X2
Linux: clone this repo: https://github.com/danovw/annotate_translation

- install the Python environment as described
- running `python src/annotation.py` will start the program and launch the GUI
- NOTE: **The resulting CSV file with the translations will be stored here:**
  `annotate_translation/src/output/`
  --> you need to copy it to the `dialogues` repo after you're done

Step 3: Basic check of the CSV file using `check.py`

- creates files:

- new versions of the CSV file with annotation and translation
- output of the script: `new_en_fewshot_output.csv`

As I understand, this script checks whether all slots in the English datasets were annotated by you.
If this is not the case, the script will output a line like this: `WARNING: Please annotate missing target entity for source entity "Dushu Lake Church" at [[6, 9]] (char span: [[39, 56]]) in en_fewshot.json->attraction_restaurant_hotel_goal_1-28_v2###7842->1->system (#174) in annotation tool.`
What I recommend:

1. Open the annotation tool and load the CSV file that you already annotated and translated
2. Enter the respective dialogue umber of the warning, e.g. 174
3. Check whether you really forgot to annotate the slot.
4. If yes, add the annotation, if not ignore the warning. Many warnings were "false" warnings in my case which I found confusing.
   --> refer to the example file `example_output_checkpy_v24052023_astodolist.txt` to see how I approached this

Run this command in your Python environment to run the script:

- as always, doublecheck whether all paths are correct
- the `tee` command is optional, adding it simply stores the output of the command in a TXT file with the specified name, so you can easily browse it
- make sure you copy the csv file to the `annotation_path` folder and name it `en_fewshot_output.csv`

```
python check.py --
new_data_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_
guide/data_sources_used_by_multiple_steps/original_en_fewshot --
annotation_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translatio
n_guide/Step_3_checkpy_script/annotation_path | tee output_checkpy_ADD DATE ADN TIME
HERE.txt
```

Step 4: Create the first version of the JSON files out of the CSV files

- creates files:
  - `de_en_valid.json` (rename it to `de_valid.json`)
  - and the corresponding files for the other splits

`csv_path`: file name needs to correspond to one of [en_fewshot_output.csv, en_valid_output.csv, en_test_output.csv]

Note that the CSV file `en_fewshot_output.csv` corresponds to the copied and renamed `new_en_fewshot_output.csv` created in the previous step!

- you can ignore the character-level annotation script

Run this command in your Python environment to run the script:

- as always, doublecheck whether all paths are correct

```
python convert.py --
csv_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_guide
/Step_4_convertpy_script/csv_path/en_fewshot_output.csv --
data_folder=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_gu
ide/data_sources_used_by_multiple_steps/original_en_fewshot --
output_folder=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_
guide/Step_4_convertpy_script/output_folder/ --target_lang=de
```

Step 5: Start creating the mapping files with `extract_alignment.py` script + correct incorrect translations in the tool

- creates file: `preliminary_bilingual_alignment.json`
  - it needs to be checked for incorrect translations that need to be corrected by going back to the translation tool (Step 1+2)

Run this command in your Python environment to run the script:

- as always, doublecheck whether all paths are correct
- this uses the JSON file created in the previous step

```
python extract_alignment.py --
data_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_guid
e/Step_4_convertpy_script/output_folder --
output_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_gu
ide/Step_5_extract_alignmentpy_script/output_path/
```

Scroll through the JSON file after creating it to see whether any translations of slots are wrong or low quality. **If they are, you need to go back to step 1+2, find these terms in the dialogues (e.g. by searching the CSV file with the translations) and change them. You then redo all the steps until this one and continue with the improved `preliminary_bilingual_alignment.json` file.**

Step 6: use the `combine_translations.py` script to create a new alignment file

- creates files:
  - first version of alignment file that needs to be checked manually: `alignment_manual.json`
  - example output of the script with 44 "Cannot find canonical value" statements: `output_missing_canonicals_v26052023_1124.txt`

I do not have anything to add to the guideline. In my case I got 44 warnings like this `"Cannot find canonical value for science fiction TV shows in tv domain, type slot"`.
I was unsure how to resolve them maybe you find a way :)

Run this command in your Python environment to run the script:

- as always, doublecheck whether all paths are correct
- the `tee` command is optional, adding it simply stores the output of the command in a TXT file with the specified name, so you can easily browse it

- **Since there is likely a bug in the script**, you will likely get KeyErrors like this

```
Traceback (most recent call last):
  File "/home/lena/git/dialogues/dialogues/risawoz/helper_files_translation_guide/db_utils/combine_translations.py", line 46, in <module>
    if src_val in src_canonical[d][s].keys():
                  ~~~~~~~~~~~~~~~^^^
KeyError: 'TV'
```

  - to resolve this, you simply need to change all domain names in `preliminary_bilingual_alignment.json` to lowercase, e.g. from "TV" to "tv"
  - then the script should run without errors

```
python db_utils/combine_translations.py --
value_alignment_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_trans
lation_guide/Step_5_extract_alignmentpy_script/output_path/preliminary_bilingual_align
ment.json --
src_canonical_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/src/knowledgebase/ma
ppings/en2canonical.json --
output_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_gu
ide/Step_6_combinetranslationspy_script/output_path/ | tee
output_combinetranslations_missing_canonicals.txt
```

Step 7: manually mark canonical terms for each alignment slot

- creates files: a new version of `alignment_manual.json` where "#" and "@" marks where added to mark canonical terms and remove incorrect translations

In this step you need to make decisions on what you consider to be the best canonical translation of a given slot. Maybe it helps you to look at how I approached this before, I provide the file `alignment_manual_canonical_terms_marked_and_extended_v31052023.json` for this. Approach this how you see fit, but really make it consistent across the entire file.

I asked the authors for additional explanation about POS consistency:

Here is the e-mail where I added Tianhaos answers(also contained in the e-mail PDF)

My main question when working on step 7 is: Is it intended that I add additional translations here and select them as the canonical translation? In my case I think that that is very often necessary to adhere to the "POS consistency" that you request.

In the guideline you write "For each slot, all its slot values are consistent in POS tags as much as possible. For example, for the "production country/region" slot in TV series domain, we will use the unified noun form (i.e., "America"/"India"/"Japan"/"Korea") instead of mixing the noun and adjective form (i.e., "American"/"Indian"/"Japanese"/"Korean")"

In German, a lot of compound words and grammatical inflections exist, which influences the translations. For instance, compared to English and French, German has more cases which influence the specific word form, e.g. "American" can be translated as "amerikanisch, amerikanischer, amerikanisches, amerikanischen, amerikanischem", depending on the case :D

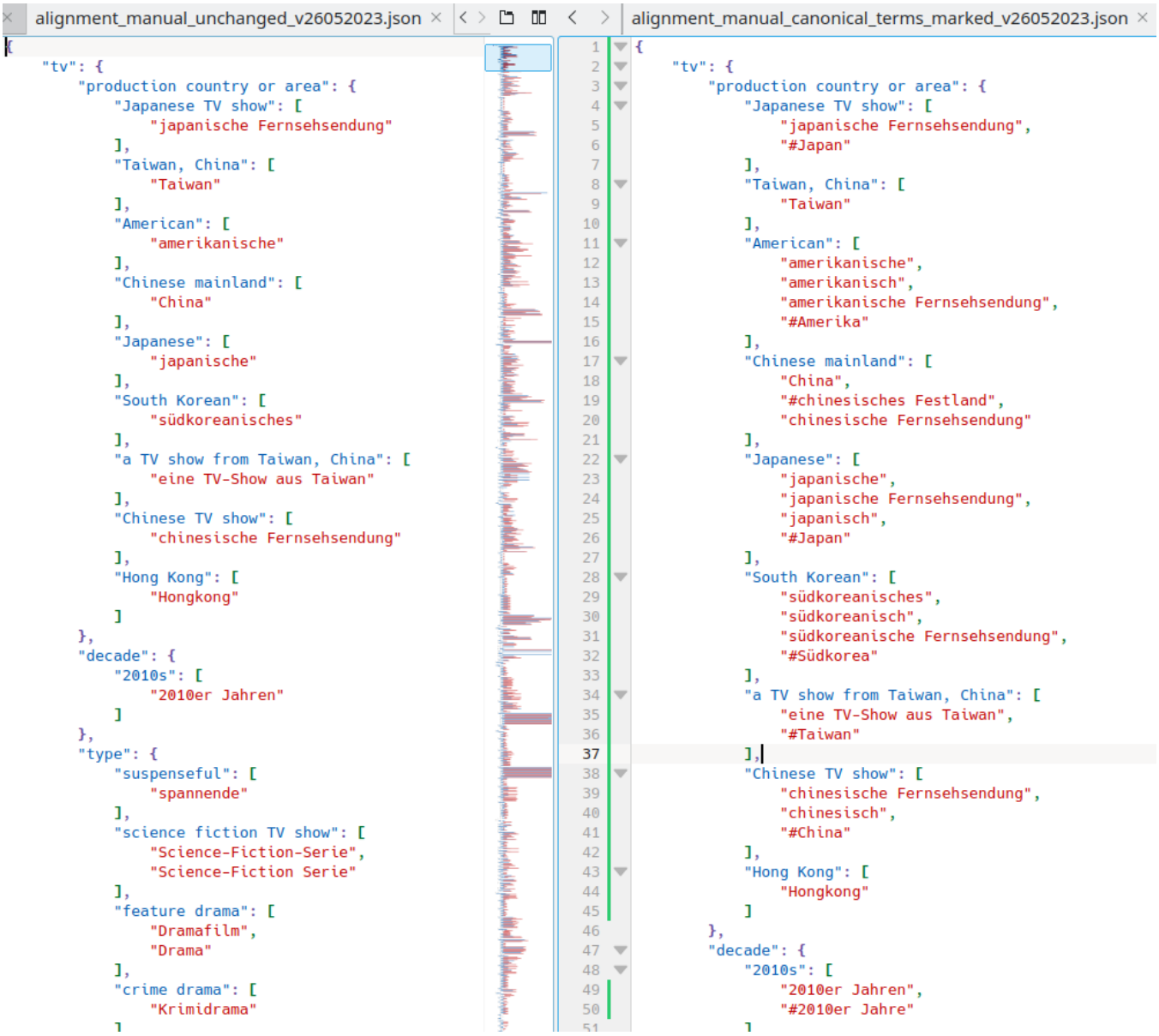
So I have a couple questions:

1. Is it desirable that I add all these possible inflections? Or is this "completeness" not so important? In that case only the word shapes actually appearing in the translations would appear.
   --> Answer from Tianhao: "I think you can first focus on the word forms that appear in the translations since we did this in other languages."
2. Should I - if possible - select the word stem that all these versions of the same word have in common as the canoncial translation? For instance, then I would select the noun "Amerika" for American instead of "amerikanisch" which is the actual case-neutral translation of the adjective American.
   --> Answer from Tianhao: You should make sure all of the candidate values of a slot have the same part of speech (e.g., all nouns or adjectives), and you can decide the most suitable form of values for your language once they have consistent POS.
3. In general, do I understand correctly that the canonical translation should be the shortest version of the word, e.g. that it makes sense to always add the "basic" version of an adjective, e.g. "günstig" for cheap, when only the inflection "günstiges" appears in the original alignment file?
   --> Answer from Tianhao: Not exactly. The canonicalized value is not always the shortest version, and this depends on the slot. For example, when we talk about cuisines, we usually use adjectives in English (e.g., Japanese cuisine). And for product countries/regions of a movie, we use nouns (e.g., America).

Below in the picture I give some examples of how I could treat this and would appreciate a brief feedback whether the translations selected as canoncial with # correspond to your idea of POS consistency.

On the left you can see the unchanged alignment_manual.json file that I get as an output from Step 6, on the right you can see how I would modify it manually according to how I understand the instructions in step

7:



Thanks for your help!

Step 8: `build_mappings.py` script to create the two mapping files

- creates files
  - `de2canonical.json` which contains all alternative translations of the canonical terms
  - `en2de_alignment.json` which matches each English slot value to its canonical German translation

In this step, the first version of the two mapping files are created. This uses all the slot values you marked in the translation of the dialogues in Step 1+2.

The next step is meant to copy the slots of the databases to the `en2de_alignment.json` file and thereby extend it .

These files are later manually extended, since they only contain the slot translations contained in the dialogues so far.

The next step extends `en2de_alignment.json` by adding the translations from the database files. **Since this was already done, my recommendation would be to write a script that merges your**

alignment file with the one previously created after translating the fewshot file and the database files.

Run this command in your Python environment to run the script:

- as always, doublecheck whether all paths are correct
- make sure that the file name for `alignment_manual_path` is correct
- potential errors:
  - `json.decoder.JSONDecodeError: Expecting value: line 323 column 13 (char 8884)` --> you have to add/remove commas somewhere in the `alignment_manual.json` file to adhere to the JSON format
  - assertion errors in cases where the canoncial translation is not unambiguous, e.g. `AssertionError: only one standard translation for each source value is allowed, but got 0 for fair` --> you need to select only a single one

```
python db_utils/build_mappings.py --
alignment_manual_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_tran
slation_guide/Step_7_choose_canonical_forms/alignment_manual_canonical_terms_marked_an
d_extended_v31052023.json --
output_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_gu
ide/Step_8_build_two_alignment_files/output_path --lang=de --canonical_mark="#" --
incorrect_mark="@"
```

Step 9: translate the database entries with the `translate_db.py` and `translate_db_results.py` scripts

This step consists of two parts:

- first, extend the `en2de_alignment.json` by addding all the translations from the databases (maybe you can mostly skip it, since I already did it) The `translate_db.py` produces a warning for each slot translation missing
  --> If you skip it, that still requires you to merge your `en2de_alignment.json` file with mine --> remove duplicates, but copy all contents together, **do this either manually or by writing a custom script**
  - You can find my file (including the database translations) here: `Step_8_build_two_alignment_files/output_path/en2de_alignment_v29062023_afterStep9.json`
- You don't have to edit the database files since I already manually checked their translation quality!
- second, use the the `translate_db_results.py` script to translate the "db_results" field of the dataset files (de_fewshot.json, de_valid.json, de_test.json) from English to German using the translated database files in the `db_de` folder

Add all slot values to the en2de alignment file with the `translate_db.py` script

- creates file
  - `db_de` folder, i.e. the German version of the database files
  - heavily extends the `en2de_alignment.json` file that was first created in step 8

It is possible you can skip all of the below (meaning everything involving `translate_db.py`), since I already did add all the database translations to `en2de_alignment.json` using the process described below. Simply run the `translate_db.py` script and see whether it produces warnings.

If you skip it, that still requires you to merge your `en2de_alignment.json` file with mine --> remove duplicates, but copy all contents together, **do this either manually or by writing a custom script**

- You can find my file (including the database translations) here: `Step_8_build_two_alignment_files/output_path/en2de_alignment_v29062023_afterStep9.json`

--> If there are no warnings, you can simply continue with the second part of this script using the `translate_db_results.py` script.

The idea of the `translate_db.py` script is to check for all database files whether all the slots have a German translation of the English slot values. The goal is that the script runs without warnings. This requires you to copy many of the translations from the database files to `en2de_alignment.json` file.

To help with the copying, the script `extract_translations_for_en2de_alignment.py` helps.

Semi-automatic workflow using the `extract_translations_for_en2de_alignment.py` script.

I did the copying to the alignment value separately for each database file to separate the process into individual steps. This is how I did it:

1. In the `translate_db.py` script, add the name of the database you want to work on in line 16 in the `domain_list` variable. The variable `domain_list` is now a list with a single entry.
   - e.g. `domain_list = ["movie"]`
2. Run the following command after adding your one base file path. The first part of the path should be specific to your computer, the rest of the path should correspond to the structure of the repo on the `hpi-de-risawoz` branch.
   - the `tee` command is optional, adding it simply stores the output of the command in a TXT file with the specified name, so you can easily browse it
   - NOTE: **It is important that `tgt_db_path` is not the `database/db_de` folder because the script saves its results in this folder so your files might be overwritten!**

```
python db_utils/translate_db.py --src_lang=en --tgt_lang=de --
src_db_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/database/db_en --
tgt_db_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translatio
n_guide/Step_9_translate_database_files/original_version_by_Jim --
slot_alignment_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_tr
anslation_guide/Step_9_translate_database_files/slot_alignment_path/slot_alignment
.json --
value_alignment_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_t
ranslation_guide/Step_8_build_two_alignment_files/output_path/en2de_alignment.json
| tee Step_9_translate_database_files/output_translate_db_py_warning.txt
```

3. Look at the outputs of the script. It prints one line for each slot value that is missing in the alignment file

- e.g. `Warning: missing value "120 minutes" of slot "film_length" in source language "en" not found in the bilingual value alignment file! Please add the corresponding translation to the alignment file and try again.`

4. Now use the `extract_translations_for_en2de_alignment.py` script to add these missing values. Use the script once for each slot value of the database, where e.g. `production_country_or_area` would be the first slot for the `movie` database

```
{
    "production_country_or_area": "Taiwan",
    "type": "romantisch",
    "decade": "2010er",
    "star": "Ko Chen Tung",
    "director": "Giddens Ko",
    "title": "Du bist mein Augapfel",
    "name_list": "Kai Ko, Michelle Chen, Owodog und Steven Hao",
    "release_date": "19.08.2011 in Taiwan / 06.01.2012 auf dem chinesischen Festland",
    "film_length": "100 Minuten",
    "douban_score": "9.1"
},
```

5. For each slot, the script extracts a unique list of translations used in the database file. You have to compare these to the ones that are already contained in the `en2de_alignment.json` file. There are multiple options:
    1. *All entries* extracted by the script *match* the respective entries in the alignment file. You can move on to the next slot then.
    2. The translation in the extracted file
    3. Make sure you have a translation that is correct for each English slot, otherwise the script will still produce a warning!
6. NOTE: **You do not need to change the original database files, only the alignment files! The script will edit the database files and replace all slot values using the translations you copied to the `en2de_alignment.json` file.**
7. If all errors are resolved, the output will be very short without any warnings and look like this: `Load 100 items in movie domain from en database Finished translation from en to de in movie domain! successful: 100, failed: 0`
8. The script then saves the corrected database file in the `risawoz/database/db_de/` folder.

Translate the database results part of the German dataset file with the `translate_db_results.py` script

- creates file:
    - updated version of the JSON files, e.g. `de_fewshot.json` where now all of the "db_results" fields are translated to German as well

After running this script, the database entries in the German data files (de_fewshot.json, de_valid.json, de_test.json) will have been translated from English to German. Check that this is the case!

Run this command in your Python environment to run the script:

- as always, doublecheck whether all paths are correct
- uses files:
    - `src_data_path`: the original English dataset files (copied from `dialogues/dialogues/risawoz/data/original/`)
    - `tgt_data_path`: the preliminary version of the German dataset JSON files, i.e. **the output of Step 4**

- `src_db_path + tgt_db_path`: English and German database folders
- `value_alignment_path`: the value alignment file `en2en_alignment.json` that was finished in Step 8

```
python db_utils/translate_db_results.py --src_lang=en --tgt_lang=de --
src_data_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_
guide/data_sources_used_by_multiple_steps/original_en_fewshot --
tgt_data_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_
guide/Step_4_convertpy_script/output_folder --
src_db_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/database/db_en --
tgt_db_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/database/db_de --
output_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_translation_gu
ide/Step_9_translate_database_files/output_path --
value_alignment_path=ADD_YOUR_PATH_HERE/dialogues/dialogues/risawoz/helper_files_trans
lation_guide/Step_8_build_two_alignment_files/output_path/en2de_alignment.json --debug
```

The script should run without errors, the output looks like this:

```
Load 40 items in attraction domain from de database
Load 665 items in flight domain from en database
Load 665 items in flight domain from de database
Load 17 items in hospital domain from en database
Load 17 items in hospital domain from de database
Load 105 items in tv domain from en database
Load 105 items in tv domain from de database
100%|
| 100/100 [00:00<00:00, 15
5.81it/s]
100%|
| 1/1 [00:00<00:00,
1.56it/s]
set()
(dialogues) lena@powerhorse:~/git/dialogues/dialogues/risawoz/helper_files_translation_guide$
```

Last step: Validate the work using the steps described in the repo

- creates files
  - converted files for each split here: `risawoz/data/de_{split}.json`
  - preprocessed data for each split in this folder: `data/preprocessed/de_v1`
  - a file for each split summarizing the remaining issues/warnings: `{split}_entity_check_1.tsv`
  - Example file for the `convert.py` script: `example_output_convertpy_06072023.txt`

Link: https://github.com/stanford-oval/dialogues/tree/camera-ready-acl2023/README.md

In the following, you can see the copied instructions from the repo, where I added my comments after the arrow -->

1. "Once you've created the dataset in the target language, put the content in the following file `risawoz/data/original/{language}_{split}.json`"
   --> copy the file `de_fewshot.json` from step 9 to this folder
2. "Add the new database files under `risawoz/database/db_{lang}/`. Follow the same formatting as English. The slot names don't need to be translated, only slot values."
   --> this is slightly confusing, this was already done in Step 9
3. "Run `python3 dialogues/risawoz/src/convert.py --setting {language} --splits {split}` to convert your data into a format suitable for preprocessing"

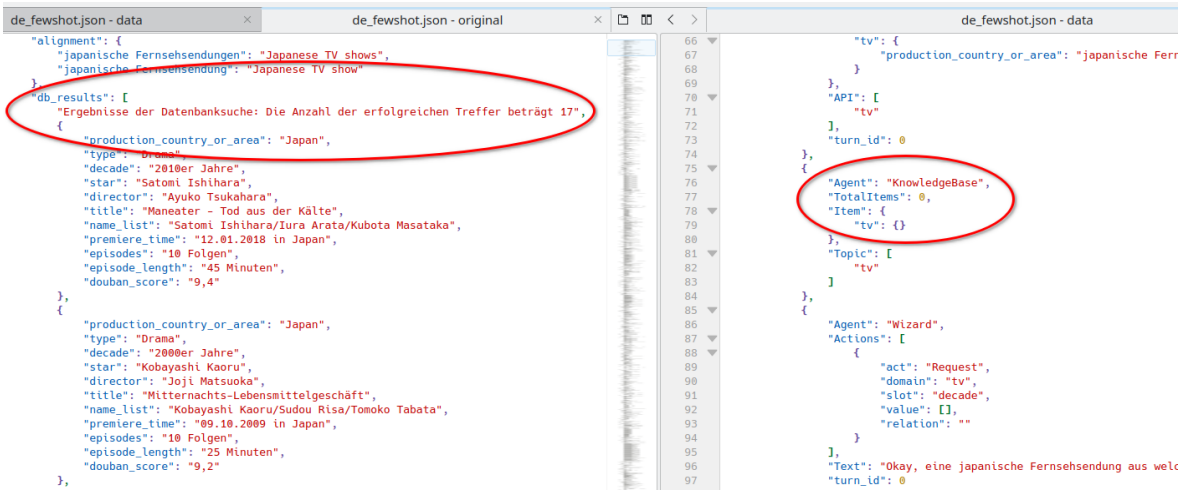
--> For German you run: `python3 dialogues/risawoz/src/convert.py --setting de --splits fewshot` (do it for each split)

- NOTE: This is not the same script as in Step 4 of the guideline!
- NOTE: **I ran `convert_fromkoreanrepo.py` instead to try and resolve a bug I got, see comments in step 6 below.**
    1. To run the script, you need to run a local instance of MongoDB since that program handles access to the databases.
    2. First, make sure that you have `pymongo` installed in your Python environment. As far as I know the project uses version 3.11.2, maybe later versions will also work without issues.
    3. Start a local instance of MongoDB with a command line command (Linux): `sudo systemctl start mongod`
    4. The output will look like this (refer to my output file `output_convertpy_05072023_1026.txt`):

```
processing fewshot data...
API call likely failed for dial_id: movie_tv_goal_4-21###6617, turn_id: 0
API call likely failed for dial_id: movie_tv_goal_4-21###6617, turn_id: 1
API call likely failed for dial_id: movie_tv_goal_4-21###6617, turn_id: 2
API call likely failed for dial_id: movie_tv_goal_4-21###6617, turn_id: 6
API call likely failed for dial_id: attraction_restaurant_goal_2-18###5165, turn_id: 0
API call likely failed for dial_id: attraction_restaurant_goal_2-18###5165, turn_id: 1
API call likely failed for dial_id: restaurant_hotel_goal_2-63_v2###7740, turn_id: 0
API call likely failed for dial_id: restaurant_hotel_goal_2-63_v2###7740, turn_id: 3
API call likely failed for dial_id: attraction_restaurant_hotel_goal_3-46###6201, turn_id: 4
API call likely failed for dial_id: attraction_restaurant_hotel_goal_3-46###6201, turn_id: 7
API call likely failed for dial_id: Hospital_goal_4-49###4239, turn_id: 0
API call likely failed for dial_id: attraction_restaurant_goal_3-66###6081, turn_id: 0
API call likely failed for dial_id: attraction_restaurant_goal_3-66###6081, turn_id: 3
API call likely failed for dial_id: attraction_restaurant_goal_3-66###6081, turn_id: 4
API call likely failed for dial_id: attraction_goal_4-39###4879, turn_id: 0
```

    Quote from the instructions about interpreting this output: "You'll likely see the following in your output logs "API call likely failed for...". This could mean many things ranging from wrong alignment during translation, mismatch between entities in the translated sentence and database values, etc. To have a more accurate check, we restore the API calls from the existing search results. You should also try your best to solve the failed API calls by correcting belief states annotations and the two mappings. Our script will show some clues for you to solve these issues (e.g., the mismatches between the belief states and ground-truth search results which make an API call fail). If you get only a few of these errors, it means that the translated dataset is already of relatively high quality."
    5. **The authors communicated that about 100 errors here are okay, since errors stem from the English source dataset.** The output of my ruin is contained in `example_output_convertpy_06072023.txt`
    --> Still try to resolve the errors, as the message like `API call likely failed for dial_id: movie_tv_goal_4-21###6617, turn_id: 0` will result in the problem described below (missing database entries in the resulting dataset file)
    6. **I had some quality issues with my resulting `de_fewshot.json` file because it was missing database entries. As you can see in the screenshot below (first turn of `movie_tv_goal_4-21###6617`), on the left hand side, the `original/de_fewshot.json` file contains multiple database results that are no longer contained in the file created**

by `convert.py`. Hopefully you find the bug here :)



4. "If conversions is successful, you will see the converted file: `risawoz/data/{language}_{split}.json`. You can check the file to make sure it looks good." --> Make sure that you now have all 3 files in the respective folder. Scroll through the files to see whether they look as expected. If unsure, compare with the respective files in the other languages.
5. "Run `python3 dialogues/risawoz/src/preprocess.py --max_history 2 --last_two_agent_turns --gen_full_state --only_user_rg --sampling balanced --setting {lang} --fewshot_percent 0 --version 1 --splits {split}` to preprocess the data for training. --> Run `python3 dialogues/risawoz/src/preprocess.py --max_history 2 --last_two_agent_turns --gen_full_state --only_user_rg --sampling balanced --setting de --fewshot_percent 0 --version 1 --splits fewshot` (do it for each split separately or for all at once, your choice). Check that the folder `data/preprocessed/de_v1` was created with the files. The output simply looks like this:

```
(dialogues) lena@powerhorse:~/git/dialogues$ python3 dialogues/risawoz/src/preprocess.py --max_history 2 --last_two_agent_turn
s --gen_full_state --only_user_rg --sampling balanced --setting de --fewshot_percent 0 --version 1 --splits fewshot
Reading all files from ['dialogues/risawoz/data/de_fewshot.json']
100%|████████████████████████████████████████████████████████████████████████████████████████████████████| 100/100
[00:00<00:00, 1119.23it/s]
fewshot 2636
```

6. "Run `python3 dialogues/risawoz/scripts/check_entity.py --directory dialogues/risawoz/data/preprocessed/ --version 1 --splits {split}` to sanity check the data. This script ensures that entities in the output are present in the input. This is necessary since our models are trained to copy entities from the input. This script will create a file `dialogues/risawoz/data/preprocessed/{split}_entity_check_1.tsv` including erroneous turns." --> Run `python3 dialogues/risawoz/scripts/check_entity.py --directory dialogues/risawoz/data/preprocessed/ --version 1 --splits fewshot --setting de`. Now check the resulting `fewshot_entity_check_1.tsv` file.
7. **Communicate with the authors about whether the quality is sufficient. As stated before, they communicated that about ~100 errors stem from annotation errors in the original dataset. If you have substantially more errors you might need to backtrack them as mentioned in the guideline.**

Finally share your results: create a pull request from your forked repo to the `dialogues` repo

Many guidelines exist online to explain how the pull request feature works on Github :)

Discuss with the maintainers of the `dialogues` repo what quality standards you need to reach. There are some automated tests that they will ask you to run.

You can check out the accepted pull requests by the other language teams for some guidance.