

Dataset Translation Guideline


Our goal is to translate the RiSAWOZ dialogue dataset (including utterances and their annotations, ontology, value mappings and databases) into different languages, and **collect localized entities to achieve localization, which will make our translated dataset much closer to real-world scenarios.**

Please note that we have several scripts to speed up the whole process. However, they have not been tested in all languages, so there may be some bugs when running them. If you have any questions and suggestions, please feel free to contact Tianhao Shen (thshen@tju.edu.cn) or Mehrad Moradshahi (mehrad@cs.stanford.edu).

In general, the whole process can be divided into the following steps:

1. Translate utterances in dataset from the source language to the target language. Although we provide a tool (https://github.com/danovw/annotate_translation) to conduct translation and annotation at the same time, you can still translate the utterances first if translators cannot use our tool.
 - If you cannot use our tool for translation, please contact Tianhao Shen to determine the format of your translation file, and he will write a conversion script to import translations into the annotation tool.
 - For how to use this annotation tool, please check our brief guideline: [w Annotation tool guide.docx](#)
 - To avoid error propagation in translation, **we may update the dataset to fix errors. Please make sure you use the latest version of the source dataset on <https://github.com/stanford-oval/dialogues/tree/tianhao/dialogues/risawoz/data/original>.** For example, if you choose English as the source language, you should use `en_fewshot.json`, `en_valid.json`, and `en_test.json` in this folder to start the translation.
2. For each utterance, you should **annotate the translations of all the marked source entities in the translated utterance.** Our annotation tool can annotate the target entities by word-level or character-level spans.
 - Please note that we provide the word-level annotation tool by default. If you want to use character-level (e.g., it's necessary for you to make sub-word level annotations), please contact Tianhao Shen for more details.
 - The annotations will be saved in Tab-delimited csv format.
 - Here is a sample [output](#) for English data.
3. After translation and annotation, you should have a basic check. **We provide a check script (`check.py`) at this step to help you check the redundant/missing/incorrect annotations.** Please check the [output](#) Google Drive for this script.
 - Brief usage: `python check.py --new_data_path={Path of the folder containing the dataset, e.g. en_data} --annotation_path={Path of the folder containing the annotation csv file, e.g., output}`
 - If the dataset files in the source language (English for most teams) are revised, you should rerun this step to check if there are any changes on

translations and annotations. In this case, you should pass the path of the new dataset to this script.

- For the **redundant and incorrect annotations**, this script will directly delete them.
 - For the **missing annotations**, this script will show warnings with the values and locations of missing entities, which will be helpful for further correction.
 - Please note that **you should solve all warnings before moving on to the next step to stop error propagation**.
 - The **check script** will generate a new csv file, so it will not overwrite your original annotations in case there are any potential bugs. Because our annotation tools recognize your annotations by filename, you should manually back up your original annotation files and rename new generated files to the same filename as your original ones.
4. After solving all warnings, you can use our **post-processing script** (`convert.py` or `convert_char_level.py`) to **convert the csv annotation files into the json files which share the same format as RiSAWOZ dataset**.
- **Brief usage:** `python convert.py (or convert_char_level.py)`
`--csv_path={path to csv file (e.g.,`
 `english_dev_out_final_output.csv)}` `--data_folder={folder`
`containing few-shot train, dev, and test json files (e.g.,`
`en_data)}` `--output_folder={folder of output json file}`
`--target_lang={target language, e.g. "hindi"}`
 - **NOTE: Please leave slot names untranslated for now.**
 - After this step, you should get three splits of the translated dataset (few-shot train, valid and test set). This is an output [sample](#) for English.
 - If you use the character-level annotation tool, please also use the character-level post-processing script (i.e., `convert_char_level.py`).
5. We provide a **script** (`extract_alignment.py` in the [output](#) Google Drive) to **extract all alignment information from the converted datasets, and the alignment annotations will be organized by domain and slots**. Please use it to generate a json file (`preliminary_bilingual_alignment.json`). It looks like this:

```
{
  'domain1': {
    'slot1': {
      'src_value1': ['tgt_translation1', 'tgt_translation2', ...],
      ...
    },
    ...
  },
  ...
}
```

- Brief usage: `python extract_alignment.py --data_path={path of folder containing dataset files (e.g., en_data)} --output_path={output folder of preliminary bilingual value alignment file}`
- Here is a sample [output](#) for Chinese-English alignment file.
- If you find any incorrect translations in this file, please correct them in the annotation tool and rerun the step 3-5.

In the next steps, you will complete the interactions with databases, which is also necessary for task-oriented dialogue systems. In short, we need two mappings: a mapping between the source entities and canonicalized entities, and a mapping between the canonicalized and non-canonicalized forms for each target entity.

This is because an entity may have different forms of expressions depending on context, and we must select a canonicalized form to build the databases and mapping other forms to this canonicalized form to conduct searching in databases.

6. Since the source entities may also have different forms, we have finished the canonicalization in the source language first. Then we provide a script (`combine_translations.py` in the `db_utils` folder of [output](#) Google Drive) to read the json file generated in step 5 and combine all translations of source language entities which share the same canonicalized forms in the source language. You will get an `alignment_manual.json` at the end of this step. Here is a [sample](#) alignment file for chinese-english translation.

- Brief usage: `python combine_translations.py --value_alignment_path={path to the bilingual value alignment json file generated in the step 5 (e.g., preliminary_bilingual_alignment.json)} --src_canonical_path={path of canonical value mapping file in the source language (e.g., en2canonical.json)} --output_path={output folder of reorganized bilingual alignment file}`
- The output will share the same format as the one in step 5, but the translations will be reorganized according to canonicalized source values.

7. After obtaining the merged mapping, you should manually select one canonicalized translation for each source entity in `alignment_manual.json`. There are two basic principles in this process.

- **Part-of-Speech (POS) consistency:** 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").
- **Value consistency:** We will use the same translation for the same value in different domains and slots. For example, "中等" is a possible value in the "pricerange" slot of both the restaurant and hotel domains, so we will use "moderate" as the canonicalized translation instead of choosing different translations (e.g., "medium" or "mid-priced") for each slot.

- In my practice, I use two marks to annotate the translations in alignment_manual.json: "#" to annotate the canonical translations, and "@" to annotate the incorrect translations or falsely categorized entities. These two marks are added to the beginning of entities (e.g. "#America" or "@2010s").
 - Here is a sample [file](#) for how to use these two marks.
- 8. Then you can construct the two mapping mentioned above. We will also provide a script (build_mappings.py) to help you finish this process. You can find it in the db_utils folder of [output](#) Google Drive.
 - Brief usage: python build_mappings.py


```
--alignment_manual_path={path to alignment_manual.json}
--output_path={output folder of two mapping files}
--lang={target language (e.g., hi)} --canonical_mark="#"
--incorrect_mark="@"
```
 - You can change the canonical mark and incorrect mark as long as these two marks don't appear in all entities.
 - You should get a {tgt}2canonical.json and a {tgt}2en_alignment.json in this step, where {tgt} is the target language.
 - Take a look [here](#) to see the content of these files generated for English.
- 9. With the two mappings, you can start translating databases and the "db_results" field (the ground-truth search results in each turn of dialogues) in the dataset. We provide two scripts (translate_db.py and translate_db_results.py in the db_utils folder of [output](#) Google Drive) for you to finish them respectively.
 - You should prepare a slot_alignment.json before running the scripts, which can be generated by copying the content of slot_alignment dictionary from the post-processing script mentioned in step 4.
 - **NOTE: Please leave slot names untranslated for now. So just provide an identical mapping.**
 - Brief usage:
 - translate_db.py: python translate_db.py


```
--src_lang={source language (e.g., en)}
--tgt_lang={target language (e.g., hi)}
--src_db_path={folder containing databases in the
source language (e.g., en_db)}
--tgt_db_path={folder containing databases in the
target language (e.g., hi_db)}
--slot_alignment_path={path to
slot_alignment.json} --value_alignment_path={path
to {tgt}2en_alignment.json file}
```
 - translate_db_results.py: python translate_db_results.py


```
--src_lang={source language (e.g., en)}
--tgt_lang={target language (e.g., hi)}
--src_data_path={folder containing dataset in the
source language (e.g., en_data)}
--tgt_data_path={folder containing dataset in the
target language (e.g., hi_data)}
```

```
--src_db_path={folder containing databases in the
source language (e.g., en_db)}
--tgt_db_path={folder containing databases in the
target language (e.g., hi_db)}
--output_path={folder of output files (dataset in
target language with db_results)}
--value_alignment_path={path to
{tgt}2en_alignment.json file} --debug (optional,
add this option to see verbose debug information)
```

- Here is a sample English [dataset](#) with db_results and generated sample [databases](#).
 - Please note that translating the "db_results" field requires translated databases, **so you should translate databases first.**
 - **Since some entities only exist in databases and may never appear in the dialogue utterances, they may not have translations at this moment. They will be shown as warnings when translating databases, and you should also translate them to keep the integrity of databases. Please solve all warnings before the next step.**

After step 9, you should get the below things in the target language:

- Three splits (few-shot train, valid, test set) of translated RiSAWOZ dataset with "db_results" field
- Two mappings: {tgt}2canonical.json and {src}2{tgt}_alignment.json
- Translated databases in all 12 domains

Now you are ready to validate your work. For the following steps you would use the [dialogues library](#). Please follow the steps outlined in section "Validating your work" in the README file. These steps check your dataset format, content, and agreement with database results to make sure everything is consistent. Only after these tests pass, can we successfully process your dataset for training.

After this step, you've finished the dataset translation process. In parallel with translation, the last thing you should do is to collect localized entities in all 12 domains according to the attributes listed in the databases.

10. You should collect localized entities to fill in all the attributes as much as possible. This process is mainly manual as you need to identify websites that contain relevant entities.
11. For each domain, you only need to collect the same number of entries that are in the translated database. We build a one-to-one mapping between translated entities and localized entities, and use that to create the localized version of your dataset by replacing translated slots with local ones.
12. If you cannot find a proper value for some attributes, you can choose translated values to fill them instead.
13. The database files should be in the same format as the translated ones.

Congratulations! We'll be all set for releasing our dataset after this step! Thanks so much for the hard work of all teams!

