

Manual of Py2ONTO-Edit

1. Introduction

Py2ONTO-Edit is a Python script to extract and translation ontology terms.

Version: V1.0

Version of Manual: V1.0

Update date: August. 2025

2. Getting Started

2.1 Requirements:

Python `>= 3.10`

2.2 Package dependency:

Note: Py2ONTO-Edit supports running in command-line interface (CLI) and code

`pip install -r requirement.txt`

2.3 download local translation model

Note: Local translation functions require downloading the local Argos-Translate model. The model file must be downloaded into the *models* folder in this project.

English to Chinese (en2zh):

- en_zh.argosmodel
- translate-en_zh-1_1.argosmodel
- translate-en_zh-1_9.argosmodel (suggestion)

English to German (en2de):

- translate-en_de-1_5.argosmodel (suggestion)

English to French (en2fr):

- en_fr.argosmodel(suggestion)

weblink: https://drive.google.com/drive/folders/11wxM3Ze7NCgOk_tdtRjwet10DmtvFu3i

or in Argos Official web page

- translate-en_zh-1_9.argosmodel (en2zh:suggestion)
 - translate-en_fr-1_9.argosmodel (en2fr:suggestion)
 - translate-en_de-1_0.argosmodel (en2de:suggestion)
- weblink: <https://www.argosopentech.com/argospm/index/>

2.4 Other translation services

Note: You must enter your DeepL auth key, ChatGLM-130B auth key, and Gemini auth key in the file 'translation_api_key_setting.yaml' to translate terms via Py2ONTO-Edit.

2.5 Usage

We built two Jupyter Notebook-based examples for Py2ONTO-Edit, please visit Usage-FOLDER in our project in GitHub (<https://github.com/MedportalProject/Py2ONTO-Edit/tree/main/Usage>)

Use case in CIL:

Example in CLI.ipynb: <https://github.com/MedportalProject/Py2ONTO-Edit/blob/main/Usage/Example%20in%20CLI.ipynb>

Use case in Python code: <https://github.com/MedportalProject/Py2ONTO-Edit/blob/main/Usage/Example%20in%20code.ipynb>

2.6 Help of Py2ONTO-Edit

```
python editonto.py -h
```

2.7 Usage of PyONTO-Edit in programming environment (Python)

```
# import all function of py2onto-edit
from editonto import *
# load HumanDO.owl
humanDO = EDIT_ONTO("./HumanDO.owl")

# 1.1 Segmentation method 1: Global extraction method
# Get all data under a single root node and store to new_onto.owl
humanDO.cut_part_onto('orofacial cleft')

# 2.1 Export all class data from ontology into csv file
humanDO.owl_to_csv("./new_onto.owl")

# 2.2 Translation with DeepL
humanDO.translate_terms_with_deepl("./part_onto.csv", "en2zh", "your-deepl-api")

# 2.3 Saving translated label data to the ontology
# zh:Chinese label; fr:French label; de:German label
humanDO.add_Chinese_label('./new_onto.owl', './all_classes_with_deepl.csv', 'zh')
```

2.8 Usage of PyONTO-Edit in CLI

Task 1: only terms extraction

```
python editonto.py -o ./HumanDO.owl -m all -s "orofacial cleft"
```

Task 2: only terms extraction (selective depth extraction)

*Use EFO ontology

```
python editonto.py -o ./efo.owl -m select -s "cell type" -e "endothelial cell,kidney cell, stem cell"
```

Task 3: only terms translation

```
python editonto.py -o ./result/cut_onto.owl -m none -l "en2de" -t d
```

Task 4: extraction and translation of ontology terms

```
python editonto.py -o ./HumanDO.owl -m all -s "orofacial cleft" -l "en2zh" -t d
```

Note:

-l: select translation mode

en2zh: English to Chinese

en2fr: English to French

en2de: English to German

-t: select translation server

d: DeepL

l: argos translate

g: gemini

c: chatglm4

Limitation

1. Use of Well-Structured Ontologies is Required

The tool assumes a well-structured ontology as input; poorly structured or inconsistent models may lead to errors during processing.

2. Manual Review Still Needed for Accurate Translation

Even under strict translation requirements, the output may require manual review and correction to ensure semantic precision and domain accuracy.

3. Limited Language and API Support

Currently, the tool supports only a limited set of languages (English to Chinese, English to German, English to French) and translation APIs (DeepL, Argos translate, Gemini, ChatGLM).