# DAG transducer

This project is the two-part system for our ACL2018 submission.

## 1 Dataset

There are three direcotry in directory `data/export`.

- `wsj00a` is training data sample.
- `wsj20a` is dev dataset sample
- `wsj21a` is test dataset sample

In directory `output`, there are four files. They are the generation result of our complete experiment.

- `output/test-pred.txt` is the final sentences
- `output/test-output.txt` is the final lemma sequences
- `output/test-sentences.txt` is the standard sentences
- `output/test-lemmas.txt` is the standard lemma sequences

## 2 Requirements

- Python3 3.5 or newer
- CMake 2.8 or newer
- Modern C++ compiler that supports C++14
- C++ Boost Library 1.58 or newer
- Git
- Torch (use for OpenNMT)

Run following command in project root to install python packages

```
pip install -r requirements.txt > /dev/null
```

# 3 Rule Extraction

Run following command in project root:

```
bash scripts/extract_rules
```

Or run following commands step by step:

```
# Dump gzip data files to binary format
python3 rule_extraction/reader.py
# Generate intermediate graphs for train data
python3 rule_extraction/data_stream.py
# Extract induced-rules
python3 rule_extraction/extract_compiled_rules.py
# Generate extended-rules
python3 rule_extraction/extend_rules.py
```

Serveral files will be generated in directory `data`. `data/rules.txt` is the rules file for reading and `data/compiled-rules.txt` is the rules file for transducer.

Every rule in `rules.txt` is like:

```
{NP_VP:ARG1(a0, a1)} => _v_1 => {NP:ARG2(b0), NP:ARG1(c0)}
a0 = c0
a1 = "to" + L + b0
```

- Variables are directly put after state labels.
- Direction of state is not displayed since we can induce it from statements.

# 4 DAG transducer

Run following command in project root to build and train DAG transducer:

```
bash scripts/train_transducer
```

Or run following commands step by step:

```
mkdir build && cd build
cmake ../transducer -DCMAKE_BUILD_TYPE=Release && make
cd ..
# Train the main perception for selecting rules
build/Release/transducer -c scripts/train.config -m data_driven
# Train the perception for generator dynamic-rules
```

```
build/Release/transducer -c scripts/train.config -m data_driven.generator
# Transduce test dataset
build/Release/transducer -c scripts/transduce.config -m data_driven
```

Three files will be generated in directory `data`.

- Two feature files: `data/train.feat` and `data/dynamic.feat`
- Transduction result for test dataset: `data/test-output.txt`

# 5 Morphology Generator

First follw link Install Torch to install Torch. Make sure executable 'th' is in `PATH`. Then install OpenNMT by running following commands in project root:

```
luarocks install tds
luarocks install bit32 # if using LuaJIT
git clone https://github.com/OpenNMT/OpenNMT
```

Then run following command in project root:

```
bash scripts/opennmt_run
```

Or run following commands step by step:

```
# Generate standard sequence of lemmas for training
python3 rule_extraction/process_sentence.py
bash scripts/opennmt_prepare_data
bash scripts/opennmt_train
```

The final model file will is `data/final-model_epoch*.t7`. Run following command to generated final sentences:

```
bash scripts/opennmt_infer data/final-model_epoch*.t7
```

- `data/test-pred.txt` is the output for test dataset
- `data/dev-pred.txt` is the output for dev dataset