Montag, 12. Oktober 2020

Developer documentation

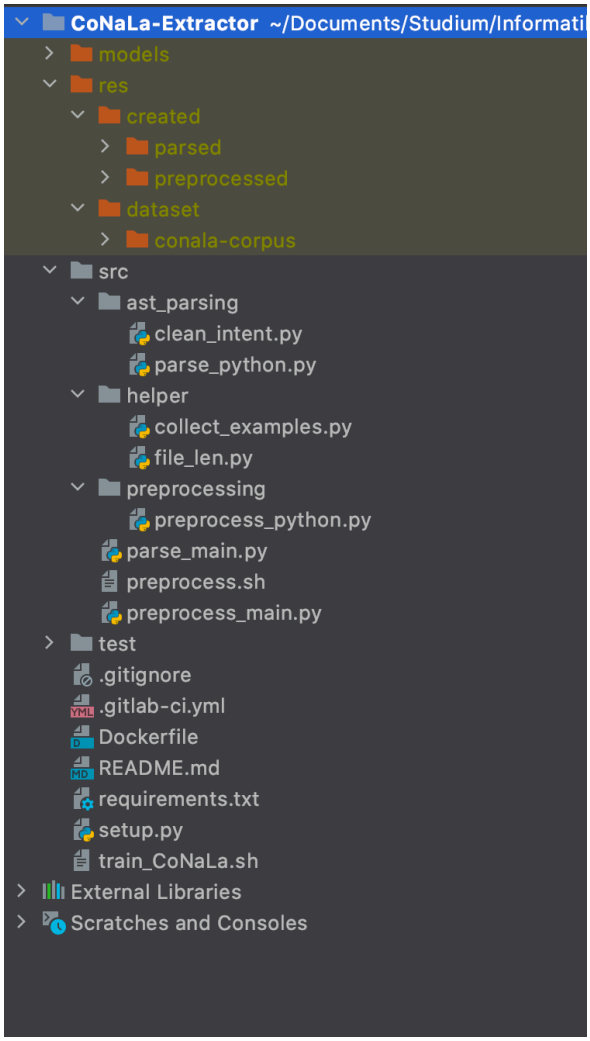
Author: Joshua Kraft

Summary:

- Name: CoNaLa-Extractor
- Description: creates and preprocesses ast's and target intents out of the mined CoNaLa dataset for training of the code2seq model
- Loc: 634 (src/test, 365/147, 70/30%)
- Testcoverage: 72%
- Style: 0 found PIP8 warnings

Repository structure:

- Source/Test/Resources split
- Source and Test share the same structure
- Source includes all runnable code on the highest level including all main() files
- Source/core/ast_ parsing or preprocessor includes the main files for parsing/preprocessing which are used in the main functions
- Res contains the used and created datasets
- The conala should be placed under dataset/conalacorpus (.jsonl)
- The parsing and preprocessing stage will save the results under /created
- Some of the paths in the mains are hardcoded and therefore need to be changed if the location an name of the data is updated on the computer
- Preview ——————————>



Code description:

The Project contains a parser and preprocessor with the associated main file.

The data is found under the following link: https://conala-corpus.github.io/#dataset-information and is structured with an intent and code snippet per example.

In the first stage the parse_main will create ast's out of the code snippets and cleans the intents. Which includes removing no information words and transforming the questions into statements.

The intents and ast's are saved in one big .json file which will be further processed

In the second stage the preprocess_main will create a trainable example cutoff the asts and intents. This means looking for all possible paths between two terminals in the tree. Each path is further processed into an example which form is described in the code2seq documentation. The intents will be split into subtokens and also processed the fit the code2seq model.

After all parsing and preprocessing the main will split the set into train/eval/test dataset, which can be used to train the code2seq model.

To train the model you will have to process the created data a second time with preprocess.sh and then run train_conala.sh in the root dir. Make sure to have code2seq in the right place for accessing its files.

Main classes/structures:

ast: the ast is a python specific data structure but can be broken down into a python dict which can be easily accessed by the preprocessor

To be continued:

This project is working and tested on correctness.

But the training results are not as expected. The model overfits the data. This could mainly be a consequence of the fact, that the code snippets are rather short and incomplete.

We already could fix a mistake in the intent creation but suggesting to change the parameters of the network. Maybe try to modify the dropout rate of the network to prevent overfitting.