AMYA CODERS

606202001

Rephrase Detection

Instruction Manual

Author:
Atul Sahay

June 16, 2020

Contents

1 Project's Directory Structure

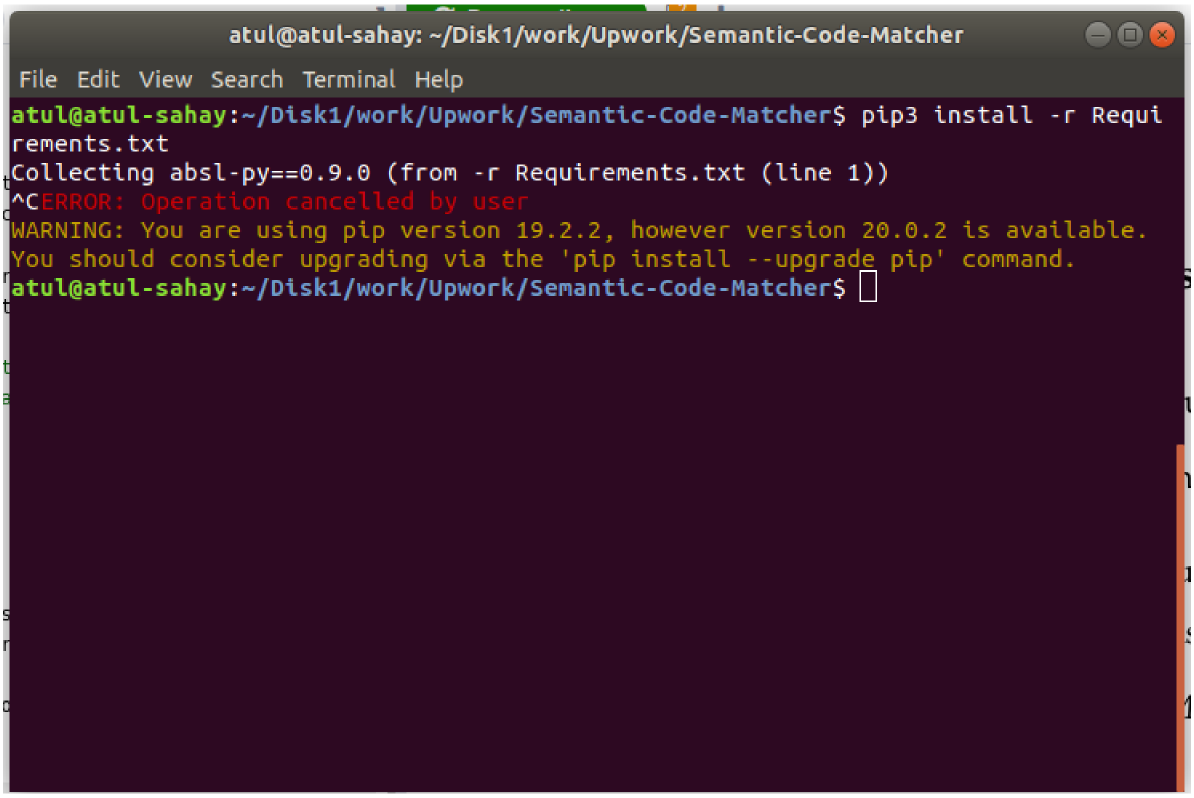
```
Explainable Latent Structures Using Attention
├── Model
│   ├── basic.py
│   ├── _init_.py
│   └── treelstm.py
├── pretrained
│   ├── model.pkl
│   └── vocab.pkl
├── snli
│   ├── Data
│   │   ├── train.pkl
│   │   ├── valid.pkl
│   │   └── test.pkl
│   ├── SaveModel
│   ├── utils
│   │   ├── dataset.py
│   │   └── _init_.py
│   ├── Vocab
│   │   └── vocab.pkl
│   ├── build_vocab.py
│   └── model.py
├── utils
│   ├── glove.py
│   └── vocab.py
├── dump_dataset.py
├── evaluate.py
├── train.py
├── README.md
├── Requirements.txt
└── Rephrase.ipynb
```

2 Requirements

2.1 How to install required modules for the project?

- Search for the **Requirements.txt** file in the project repository.
- Open the terminal and write (be sure you are in the project directory): See Fig: 1

user: Explainable-latent-structures$: **pip3 install -r Requirements.txt**



atul@atul-sahay: ~/Disk1/work/Upwork/Semantic-Code-Matcher
File Edit View Search Terminal Help

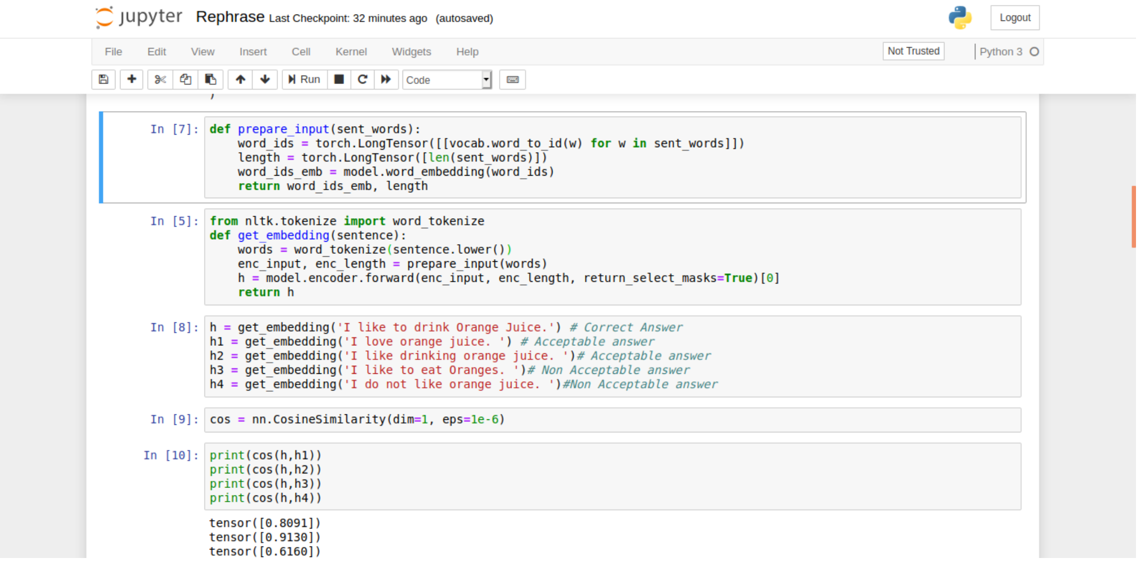
```
atul@atul-sahay:~/Disk1/work/Upwork/Semantic-Code-Matcher$ pip3 install -r Requi
rements.txt
Collecting absl-py==0.9.0 (from -r Requirements.txt (line 1))
^CERROR: Operation cancelled by user
WARNING: You are using pip version 19.2.2, however version 20.0.2 is available.
You should consider upgrading via the 'pip install --upgrade pip' command.
atul@atul-sahay:~/Disk1/work/Upwork/Semantic-Code-Matcher$ 
```

Figure 1: Installation of the required modules

3 Running the Pretrained Model

For the sake of running the pretrained model, all you need to is to open the jupyter notebook **Rephrase.ipynb**. See Fig : 3

user: Explainable-latent-structures$: **jupyter notebook Rephrase.ipynb**



```
In [7]: def prepare_input(sent_words):
            word_ids = torch.LongTensor([[vocab.word_to_id(w) for w in sent_words]])
            length = torch.LongTensor([len(sent_words)])
            word_ids_emb = model.word_embedding(word_ids)
            return word_ids_emb, length

In [5]: from nltk.tokenize import word_tokenize
        def get_embedding(sentence):
            words = word_tokenize(sentence.lower())
            enc_input, enc_length = prepare_input(words)
            h = model.encoder.forward(enc_input, enc_length, return_select_masks=True)[0]
            return h

In [8]: h = get_embedding('I like to drink Orange Juice.') # Correct Answer
        h1 = get_embedding('I love orange juice. ') # Acceptable answer
        h2 = get_embedding('I like drinking orange juice. ')# Acceptable answer
        h3 = get_embedding('I like to eat Oranges. ')# Non Acceptable answer
        h4 = get_embedding('I do not like orange juice. ')#Non Acceptable answer

In [9]: cos = nn.CosineSimilarity(dim=1, eps=1e-6)

In [10]: print(cos(h,h1))
         print(cos(h,h2))
         print(cos(h,h3))
         print(cos(h,h4))

         tensor([0.8091])
         tensor([0.9130])
         tensor([0.6160])
```

Figure 2: Jupyter Notebook