Product Requirement Document

👋 Introduction

Create a repo for TextCNN model to do review sentiment classification…

⭐ Features & Functionalities

- Ability to define and manage model-related settings such as kernel sizes, embedding dimensions, max sequence length.
- Ability to configure model training settings

...

📋 Technical Constraints

The repository should satisfy the following requirements:

- Supports building modeling frameworks by PyTorch
- Should not use the trainer api of transformers

...

🔧 Usage

To train a model, run the following script:

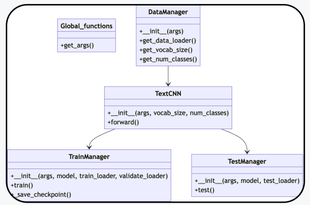
```
python main.py --learning_rate 0.01 --num_epochs 10 \
    --batch_size 16 --embedding_dim 300 \
    --kernel_sizes 3 4 5 --max_length 50 \
    --save-every_n_epoch 2 \
    ...
```

✅ Acceptance Criteria

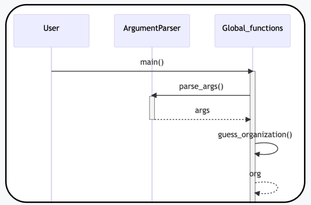
- For train, the mode training logs will be tested if the training loss decreases...
- For test, the terminal output will be tested... The accuracy should be above 0.6

...

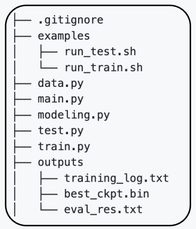
Software Design



UML Diagram



UML Sequence



Arch. Design

Env. Setup

Dependencies

reqs.txt

```
evaluate
datasets
scikit-learn
transformers
torch
```

setup_shell_script.sh

```
pip install -r reqs.txt
```

Implementation

data.py

```
class DataManager(object):
    def __init__(self, args): …
    def get_data_loader(self) -> Tuple[…]: …
    def get_vocab_size(self) -> int: …
    def get_num_classes(self) -> int: …
```

modeling.py

```
class TextCNN(object):
    def get_args(self, args, …): …
    def forward(self, inputs) -> Tensor: …
```

main.py

```
def get_args(self, args): …
def main(self) -> Tuple[…]: …
```

setup.py

test.py

train.py

Testing

Acceptance Testing

acceptance_tests/test_train.py

```
import unittest …

class TestTextCNN(unittest.TestCase):
    def setUp (self) -> None:
        …
    def test_training_metric (self) -> None:
        …
```

acceptance_tests/test_test.py

acceptance_tests/test_examples.py

acceptance_tests/test_args.py

Unit Testing

uni_test/test_building_blocks.py

```
from modeling import TextCNN

class TextCNNTester(unittest.TestCase):
    def prepare_config_args (self) ->
            SimpleNamespace:
        …
        args = SimpleNamespace( … )
        return args
```

uni_test/test_get_args.py

uni_test/test_output_shape.py

uni_test/test_param.py