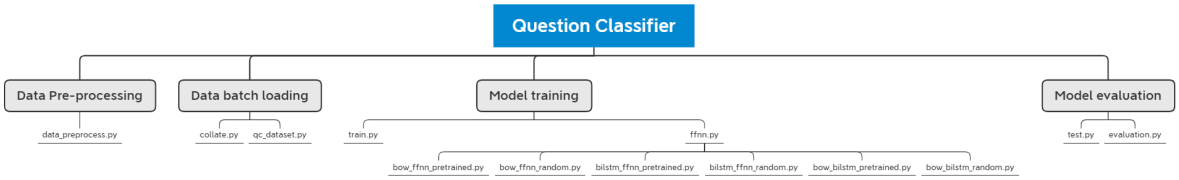
Structure



Main interface

question_classifier.py

Dependency

- `argparse`: create an argument parser to handle arguments passed from command line.

Class

None

Functions

The `question_classifier.py` does not have any functions. It uses the python build-in package `argparse` to create an argument parser to handle arguments passed from command line. It also reads the configuration file into a python dictionary for further use. It invokes methods in `train.py` or `test.py` depending on the `phase` argument passed from command line.

Data pre-processing

data_preprocess.py

Dependency

- `re` package used to remove punctuation by regular expressions.
- `numpy` package used to handle word embeddings.
- `torch` package used to handle sentence representations.

Class

- `Preprocess`

Functions

- `__init__(self, config)`: Receive configuration dictionary and store it for further use in other functions.
- `preprocess(self,path,phase)`:

    - `path`: the path of the file containing data to be preprocessed.
    - `phase`: either `"train"` or `"test"`
    - functionalities:
        - Remove stop words.
        - Remove punctuations.
        - Vocabulary:
            - `train`: generate vocabulary according to word frequency
            - `test`: load the saved vocabulary
        - Word embeddings:
            - `train`: extract word embeddings according to vocabulary from pretrained embeddings
            - `test`: load the saved word embeddings
        - Generate sentence representations using vocabulary.
        - Convert label to numeric representation.
            - `train`: generate the label to index conversion dictionary
            - `test`: load the generated label to index conversion dictionary other than generate another to keep the labels consistent
        - Generate labels' representations according to labels' index.
        - Save above variables for further use.
- `load_preprocessed(self)`: load all saved variables from previously saved files.
- `save(self,l,f_name)`: save the variable `l` into file `f_name`
- `load(self,f_name)`: load variable from file `f_name`

Data batch loading

qc_dataset.py

Dependency

- `torch.utils.data.Dataset`: implement `QCDataset` as a subclass of this for batch data loading

Class

- `QCDataset`: subclass of `torch.utils.data.Dataset`

Functions

- `__init__(self,xy)`: store data and labels into `QCDataset` object.
- `__getitem__(self, index)`: return data and label with `index` index.
- `__len__(self)`: return the number of data and labels

collate.py

Dependency

- `torch.nn.utils.rnn.pad_sequence`: pad the sentence representations with preset value

Class

None

Functions

- `qc_collate_fn_bow(QCDataset)`: the collate function for the `Dataloader` of bag of words
- `qc_collate_fn_bilstm(QCDataset)`: the collate function for the `Dataloader` of BiLSTM

Model training

train.py

Dependency

- `numpy`
- `torch`
- `data_preprocess.Preprocess`: the data pre-processing class of the project
- Models:
  - `bow_ffnn_pretrained`: model base on bag of words and feed forward neural network using pretrained word embeddings.
  - `bow_ffnn_random`: model base on bag of words and feed forward neural network using randomly generated word embeddings.
  - `bilstm_ffnn_pretrained`: model based on BiLSTM and feed forward neural network using pretrained word embeddings.
  - `bilstm_ffnn_pretrained`: model based on BiLSTM and feed forward neural network using randomly generated word embeddings.
  - `bow_bilstm_pretrained`: model based on bag of words, BiLSTM and feed forward neural network using pretrained word embeddings.
  - `bow_bilstm_pretrained`: model based on bag of words, BiLSTM and feed forward neural network using randomly generated word embeddings.
- `qc_dataset`: a subclass of `torch.utils.data.Dataset` for loading data in batch using `torch.utils.data.dataloader`
- `collate`: the collate functions used by `torch.utils.data.dataloader`
- `evaluation`: evaluate the performance of the model with accuracy, confusion matrix and F1-score

Class

- `Train`: handles the training phase of the question classifier

Functions

- `__init__(self,config)`: obtain and store the configuration for further usage
- `train(self)`:

1. Pre-process the data by using `data_preprocess.Preprocess` method.
2. Randomly split the data set into training set and development (validation) set using `torch.utils.data.random_split`.
3. Create data batch loader using `qc_dataset` and `collate` for training and development set.
4. Define and train different models according to configuration.
5. Print out the accuracy, confusion matrix, micro and macro F1-score and save them into the pre-configured file.

- `ens_bow(self,train_set,dev_set,voca_embs,vocabulary,labels_index)`: train bag of words models in an ensembled way
- `ens_bilstm(self,train_set,dev_set,voca_embs,vocabulary,labels_index)`: train BiLSTM models in an ensembled way
- `ens_bow_bilstm(self,train_set,dev_set,voca_embs,vocabulary,labels_index)`: train bag of words models with BiLSTM layer in an ensembled way

ffnn.py

Dependency

- `torch`

Class

- `FFNN`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self,input_size,hidden_size,output_size)`: set up the structure of the feed forward neural network.
- `forward(self, inp)`: defined the computation performed at each call

bow_ffnn_pretrained.py

Dependency

- `torch`
- `ffnn.FFNN`: use the feed forward neural network in hierarchical way

Class

- `BOW_FFNN_PRE`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self, embeddings, hidden_size, output_size, freeze = True)`: set up the structure of the feed forward neural network with bag of words and pretrained embeddings.
- `forward(self, input)`: defined the computation performed at each call

bow_ffnn_random.py

Dependency

- `torch`
- `ffnn.FFNN`: use the feed forward neural network in hierarchical way

Class

- `BOW_FFNN_RANDOM`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self, voca_size, input_size, hidden_size, output_size, freeze = True)`: set up the structure of the feed forward neural network with bag of words and randomly generated embeddings.
- `forward(self, input)`: defined the computation performed at each call

bilstm_ffnn_pretrained.py

Dependency

- `torch`
- `ffnn.FFNN`: use the feed forward neural network in hierarchical way

Class

- `BiLSTM_FFNN_PRE`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self, embeddings, bilstm_hidden_size, ffnn_hidden_size, ffnn_output_size, freeze = True)`: set up the structure of the feed forward neural network with BiLSTM layer and pretrained embeddings.
- `forward(self, input, lengths)`: defined the computation performed at each call

bilstm_ffnn_random.py

Dependency

- `torch`
- `ffnn.FFNN`: use the feed forward neural network in hierarchical way

Class

- `BiLSTM_FFNN_RANDOM`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self, voca_size, input_size, bilstm_hidden_size, ffnn_hidden_size, ffnn_output_size, freeze = True)`: set up the structure of the feed forward neural network with BiLSTM layer and randomly generated embeddings.
- `forward(self, input, lengths)`: defined the computation performed at each call

bow_bilstm_pretrained.py

Dependency

- `torch`
- `ffnn.FFNN`: use the feed forward neural network in hierarchical way

Class

- `BOW_BiLSTM_PRE`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self, embeddings, bilstm_hidden_size, ffnn_hidden_size, ffnn_output_size, freeze = True)`: set up the structure of the feed forward neural network with bag of words and BiLSTM layer and pretrained embeddings.
- `forward(self, input, lengths)`: defined the computation performed at each call

bow_bilstm_random.py

Dependency

- `torch`
- `ffnn.FFNN`: use the feed forward neural network in hierarchical way

Class

- `BOW_BiLSTM_RANDOM`: subclass of `torch.nn.Module` to enable the model to be hierarchical

Functions

- `__init__(self, voca_size, input_size, bilstm_hidden_size, ffnn_hidden_size, ffnn_output_size, freeze = True)`: set up the structure of the feed forward neural network with bag of words and BiLSTM layer and pretrained embeddings.
- `forward(self, input, lengths)`: defined the computation performed at each call

Model evaluation

test.py

Dependency

- `torch`
- `data_preprocess.Preprocess`: the data pre-processing class of the project
- `evaluation`: evaluate the performance of the model with accuracy, confusion matrix and F1-score

Class

- `Test`: handles the testing phase of the question classifier

Functions

- `__init__(self,config)`: obtain and store the configuration for further usage
- `test(self)`:
    - Pre-process the test set
    - Load the model from configured location
    - Use the model to predict the labels of sentences in testing set
    - Print out the accuracy, confusion matrix, micro and macro F1-score and save them into the pre-configured file.

evaluation.py

Dependency

- `torch`
- `numpy`: used to help calculate the accuracy
- `pandas`: this ONLY helps to format the confusion matrix from numpy's ndarray. We are aware of that pandas is not included in the three allowed packages, but spending time thinking how to re-implement a formatting solution which pandas already provided is meaningless.

Class

None

Functions

- `get_accuracy_bow(model,loader)`: get the accuracy of bow model in a batched way
- `get_accuracy_bilstm(model,loader)`: get the accuracy of BiLSTM model in a batched way
- `get_accuracy_test(model,model_type,x,y,lengths)`: get the accuracy of models in a non-batched way
- `get_accuracy_ens_bow(models,x,y)`: get the accuracy of ensembled bag of words models
- `get_accuracy_ens_bilstm(models,x,y,lengths)`:get the accuracy of ensembled BiLSTM models
- `get_confusion_matrix(y_real,y_preds,size)`: calculate the confusion matrix from the actual and predicted labels
- `get_micro_f1(conf_mat)`: calculate the micro f1-score based on confusion matrix
- `get_macro_f1(conf_mat)`: calculate the macro f1-score based on confusion matrix