Introduction of the Project

The project can be roughly divided into 3 parts: data preprocessing, building classifier, and training classifier. In the following, I'll introduce these three phases in detail.

In addition, to clarify the detail of config file, the `Config Setting` sectino would introduce each hyperparameters in the config file.

Config Setting

- `[GENERAL]`
  - raw_path: path to original dataset
  - stop_word_path: path to the stop word list
  - vocabulary_path: path to the vocabulary
  - label_path: path to the label vocabulary
  - test_path: path to the test dataset
  - train_path: path to the dev training dataset
  - dev_path: path to the validation dataset
  - model_path: path to save the model
  - output_path: path to write down the result of testing
- `[WORD_EMBED]`
  - pretrain_embedding_path: path to the pre-trained embedding
- `[MODEL]`
  - ensemble_size: size of ensemble model
  - bow: using BOW or not
  - bilstm: using BiLSTM or not
  - freeze: frozen word embedding or fine-tuned embedding. Only available for `from_pretrain=True`
  - from_pretrain: pre-trained word embedding or random embedding
  - embedding_dim: dimension of word embedding. only available for `from_pretrain=False`
  - bilstm_hidden_dim: dimension of hidden layer of BiLSTM
  - input_dim: dimension of input of neural classifier
  - hidden_dim: dimension of hidden layer of the classifier
- `[TRAIN]`
  - padding: padding the sentence or not
  - padding_len: length of sentence after padding
  - lr: learning rate
  - batch_size: size of batch
  - epochs: total number of epochs

Data Proprocessing

This part is responsible for preprocessing the raw training dataset.

Preprocessing

Located in `src/utils/preprocess.py`.

Training data and Dev training data:

```
def create_training_data(raw_data_path, training_data_path, dev_data_path):
"""
description:
  split raw training dataset into training dataset for dev training and leave
10 percent for validationg and materialize them.
parameters:
  raw_data_path: path to the original dataset.
  training_data_path: path to the dev training dataset.
  dev_data_path: path to the validation set.
return:
  raw_data: the original training data
"""
...
```

Building label vocabulary

```
def collect_labels(label_file_path, raw_data):
"""
description:
  collect all labels of the dataset and materialize them.
parameters:
  label_file_path: path to label vocabulary
  raw_data: original training dataset
"""
```

Get stop words list:

```
def get_stop_words(stop_word_path):
"""
description:
  get stop words list
parameters:
  stop_word_path: path to the stop words list
return:
  stopwortds: list of stop words
"""
```

Building vocabulary

```
def build_vocabulary(raw_data, vocabulary_path, stop_words):
"""
description:
  build vocabulary for training dataset
parameters:
  raw_data: original training dataset
  vocabulary_path: path to materialize the vocabulary
  stop_words: stop words list consisting of words that would not be considered
in the vocabulary.
"""
```

Preprocessing pipeline

```
def preprocess(
    raw_data_path, \
    training_data_path, \
    dev_data_path, \
    label_file_path, \
    stop_word_path, \
    vocabulary_path
  ):
"""
description:
  ensemble all the function mentioned above
parameters:
  raw_data_path: path to original training data
  training_data_path: path to the dev training dataset.
  dev_data_path: path to the validation set.
  label_file_path: path to label vocabulary
  stop_word_path: path to the stop words list
  vocabulary_path: path to materialize the vocabulary
"""
```

Loading preprocessed data

Located in `src/utils/file_preload.py`

Loading dataset:

```
def load_data(path):
"""
description:
  loading dataset from the given path
parameters:
  path: path to the dataset
return:
  data: dataset
"""
```

Loading label vocabulary

```
def load_labels(path):
"""
description:
  loading vocabulary from the given path
parameters:
  path: path to the vocabulary
return:
  data: vocabulary
"""
```

Loading stop words

```
def load_stop_words(path):
"""
description:
  loading stop words
parameters:
  path: path to stop words file
return:
  stopwords: list of stop words
"""
```

Loading pre-trained embedding

```
def load_pre_train(path):
"""
description:
  loading pre-trained word embedding
parameters:
  path: path to word embedding file
return:
  pre_train_words: mapping from words to embedding
"""
```

Loading vocabulary

```
def load_vocabulary(path):
"""
description:
  loading the vocabulary
parameters:
  path: path to the vocabulary
return:
  voc: vocabulary list
"""
```

Creating weight of pre-trained embedding

```
def create_word_embedding(pretrain_words_dict, vocabulary):
"""
description:
  creating weight of word embedding
parameters:
  pretrain_words_dict: dictionary of pre-trained word embedding
  vocabulary: list of vocabulary
return:
  pretrain_weight: weight of word embedding
"""
```

Building Classifier

This stage is about building an end-to-end question classifier.

Sentence Vector

This part is responsible for translating sentences into tensors that can be used to train the classifier.

Bag-Of-Word

Building a Bag-Of-Word sentence vector

Located in `src/sentVect/bow.py`.

```
class BOW(torch.nn.Module):
"""
description:
  Bag-Of-Word Model
parameters:
  vocab_size: size of the vocabulary
  embedding_dim: dimension of the word embedding
  from_pretrain: random initialize or pre-trained embedding
  freeze: fine-tune or freeze
"""
```

```
  def __init__(
    self,
    vocab_size,
    embedding_dim,
    from_pretrain=False,
    pre_train_weight=None,
    freeze=False
  ):
```

BiLSTM

Located in `src/sentVect/mybilstm.py`.

```
class Bilstm(torch.nn.Module):
"""
description:
  BiLSTM model
parameters:
  vocab_size: size of the vocabulary
  input_dim: dimension of the input vector
  hidden_dim: dimension of the hidden layer of BiLSTM model
"""
  def __init__(self,  vocab_size, input_dim, hidden_dim):
```

Ensemble of BOW and BiLSTM

Located in `src/sentVect/bow_bilstm.py`

```
class BowBilstm(torch.nn.Module):
"""
description:
  ensemble of BiLSTM and BOW
parameters:
  vocab_size: size of the vocabulary
  embedding_dim: dimension of the word embedding
  from_pretrain: random initialize or pre-trained embedding
  freeze: fine-tune or freeze
  bilstm_hidden_dim: dimension of the hidden layer of BiLSTM model
"""
  def __init__(self,
        vocab_size,
        embedding_dim,
        from_pretrain,
        pre_train_weight,
        freeze,
        bilstm_hidden_dim
  ):
```

Classifier

Located in `src/classifier/network.py`

Neural network classifier

```
class NeuralNetwork(torch.nn.Module):
"""
description:
  classifier
parameters:
  input_dim: dimension of the input
  hidden_dim: dimension of the hidden layer
  output_dim: dimension of the output, usually set to the size of label list
"""
  def __init__(self, input_dim, hidden_dim, output_dim):
```

Model

This part is assembling the components mentioned above.

```
class QuestionClassifier(torch.nn.Module):
"""
description:
  question classifier that implements pipeline
parameters:
  bow: using BOW or not
  bilstm: using BiLSTM or not
  vocab_size: size of the vocabulary
  from_pretrain: buid from pre-trained embedding or random embedding
  pre_train_weight: pre-trained embedding
  embedding_dim: dimension of the embedding
  bilstm_hidden_dim: dimension of the hidden layer of BiLSTM
  input_dim: dimension of the input dimension of the neural network classifier
  hidden_dim: dimnesion of the hidden layer of the neural network classifier
  output_dim: output dimension of the classifer
"""
  def __init__(
    self,
    bow,
    bilstm,
    vocab_size,
    from_pretrain=False,
    pre_train_weight=None,
    freeze=False,
    embedding_dim=300,
    bilstm_hidden_dim=150,
```

```
    input_dim=300,
    hidden_dim=128,
    output_dim=50
  ):
```

Training and Testing

The final phase is training the models of different settings and conducting some experimental test on them.

Data Loading

Data loader to store the original dataset for training or testing

Located in `src/dataloader.py`

```
class DataLoader:
"""
description:
  data loader for training or testing
parameters:
  vocabulary: vocabulary of dataset
  labels: label vocabulary
  stop_words: list of stop words
  training_set: dataset
  batch_size: size of batch
  padding: padding the sentence or not
  padding_len: sentence length after padding
"""
  def __init__(self, vocabulary, labels, stop_words, train_set, batch_size=1,
padding=False, padding_len=25):

  ...

  def get_sent_offset(self, feature, label, labels, vocabulary, stop_words):
  """
  description:
    get offset from vocabulary for specific sentence and label
  parameters:
    feature: the question
    label: the label
    labels: vocabulary of label
    vocabulary: vocabulary of dataset
    stop_words: list of stop words
  return:
    feat: list of offset
    label: index of a label
  """
  ...
```

```
  def get_batch(self):
  """
  description:
    get one batch for training
  return:
    feat: lists of offset
    label: lists of indexes of labels
  """
  ...
```

Config loading

loading config file

Located in `src/config.py`

```
def get_config(path):
"""
description:
 loading config file from a specific path
return:
  config: dictionary of configuration
"""
```

Training and Testing

Located in `src/question_classifier.py`

Test on dataset

```
def test_trec(model, dataloader):
"""
description:
  test model on a dataset
parameters:
  model: the model
  dataloader: dataloader of a dataset
return:
  acc_rate: accuracy
  reals: real labels
  preds: prediction
"""
```

Train the model

```
def train(config, vocabulary, labels, stop_words, save_path='', mode='dev'):
"""
description:
```

```
  train a model with specific settings
parameters:
  config: config
  vocabulary: vocabulary of the dataset
  labels: vocabulary of the labels
  stop_words: list of stop words
  save_path: path to save the model
  mode: 'dev train' mode or 'train' mode
return:
  model: the trained model
"""
```

Test a model

```
def test(config, vocabulary, labels, stop_words, save_path):
"""
description:
  test on a model with some dataset
parameters:
  config: config
  vocabulary: vocabulary of the dataset
  labels: vocabulary of the labels
  stop_words: list of stop words
  save_path: path of the model
"""
```