学习笔记

关于环境配置

该 `repo` 使用的是 `TensorFlow` 作为学习框架，我们使用的是 Anaconda 对 TensorFlow 进行安装，可以参考这个教程：

https://blog.csdn.net/u010858605/article/details/64128466

mac 和 Ubuntu 等 Linux 系统可以直接通过源码安装，下载源码：

```
$ git clone https://github.com/tensorflow/tensorflow.git
```

参照具体教程：https://www.tensorflow.org/install/source?hl=zh-cn

PostScript:

作者还使用了许多 TensorFlow 之外的 python 库，应该可以直接：

```
$ pip install <package-name>
```

关于运行

运行以下命令进行预处理：

```
$ python run.py --prepare
```

运行以下命令进行训练：

```
$ python run.py --train
```

运行以下命令测试模型效果：

```
$ python run.py --evaluate
```

运行以下命令进行预测：

```
$ python run.py --predict
```

几点问题

数据文件存放位置

在 `capsule-mrc/` 目录下新建 `data/` 文件夹（与 `README.md` 同级），内部文件结构如下：

```
./devset:
dev.json  protocol.txt  README

./testset:
protocol.txt  README  test.json

./trainset:
protocol.txt  README  train.json
```

`dataset.py`

这个文件似乎有点小问题：

函数 `_load_dataset` 更改如下：

```
def _load_dataset(self, data_path, sampling=False):
    '''
    加载数据集
    '''
    with open(data_path, encoding='utf-8') as fin:
        data_set = []
        filter_long_para, filter_long_query, filter_zero_query = 0, 0, 0
        for lidx, line in enumerate(fin):
            if sampling:
                if random.randint(1, 10) > 1:
                    continue
            sample = json.loads(line.strip())

            if len(sample['passage']) > self.max_p_len:
                filter_long_para += 1
                continue
            if len(sample['query']) > self.max_q_len:
                filter_long_query += 1
                continue
            if len(sample['query']) == 0:
                filter_zero_query += 1
                continue
            scores = []
            if 'answer' in sample:
                fake_label = sample['answer']
                alternatives = sample['alternatives'].split('|')
                for alternative in alternatives:
                    score = 0
                    if '无法确定' in alternative or '无法确认' in alternative or '无法确的'
in alternative:
                        score += 3
                    elif '不' in alternative or '没' in alternative:
                        score += 2
                    scores.append(score)
                if sum(scores) < 5:
                    sample['choose_type'] = 1.0
                else:
```

```
                    sample['choose_type'] = 0.0
                f_index = scores.index(min(scores))  # 积极答案
                scores[f_index] = 10
                s_index = scores.index(min(scores))  # 消极答案
                scores[s_index] = 10
                t_index = scores.index(min(scores))  # 无法确定答案
                segmented_alternatives = [sample['alternatives'][f_index],
                                          sample['alternatives'][s_index],
                                          sample['alternatives'][t_index]]

                sample['segmented_alternatives'] = segmented_alternatives
                # EDIT: @shesl-meow: these lines wasn't called in any where else
                # pos_alternatives = [sample['pos_alternatives'][f_index],
sample['pos_alternatives'][s_index],
                #                     sample['pos_alternatives'][t_index]]
                # sample['pos_alternatives'] = pos_alternatives
                if f_index == fake_label:
                    sample['label_answer'] = 0
                elif s_index == fake_label:
                    sample['label_answer'] = 1
                else:
                    sample['label_answer'] = 2
            else:
                alternatives = sample['alternatives'].split('|')
                for alternative in alternatives:
                    score = 0
                    if '无法确定' in alternative or '无法确认' in alternative or '无法确的'
in alternative:
                        score += 3
                    elif '不' in alternative or '没' in alternative:
                        score += 2
                    scores.append(score)
                if sum(scores) < 5:
                    sample['choose_type'] = 1.0
                else:
                    sample['choose_type'] = 0.0
            data_set.append(sample)

    print('passage超长过滤:', filter_long_para, 'query问题过滤:', filter_long_query +
filter_zero_query)
    return data_set
```

函数 `word_iter` 更改如下:

```
def word_iter(self, set_name=None):
    '''
    遍历数据集里的所有词语
    Args:
        set_name: if it is set, then the specific set will be used
    Returns:
        a generator
    '''
    if set_name is None:
```

```
            data_set = self.train_set + self.dev_set + self.test_set
        elif set_name == 'train':
            data_set = self.train_set
        elif set_name == 'dev':
            data_set = self.dev_set
        elif set_name == 'test':
            data_set = self.test_set
        else:
            raise NotImplementedError('No data set named as {}'.format(set_name))
        if data_set is not None:
            for sample in data_set:
                for token in sample['passage']:
                    yield token
                for token in sample['query']:
                    yield token
                for tokens in sample['alternatives']:
                    for token in tokens:
                        yield token
```

一个大问题

在运行预加载时，该项目加载了这样一个文件 `data/vocab/word2vec.model`。

经过我的潜心阅读这篇文章：https://my.oschina.net/magicly007/blog/851583

猜测这个是一个作者预先训练好的词向量模型，加载的函数定义在了 `capsule-mrc/capsuleNet-mrc/vocab.py` 这个文件的 `load_pretrained_embeddings` 函数中。

该函数中使用 `word2vec.load` 进行加载，这是 python 库 `Gensim` 的一个函数，可以使用 `model.save` 这个方法创建这个文件。

而我们并没有这样预训练好的模型，作者也没有传到 github 上，所以我们需要：

1. 放弃；
2. 找一个用 `Gensim` 训练好的开源的 `word2vec` 模型；
3. 自己用 `Gensim` 训练一个 `word2vec` 模型。