服务器

ssh -p 22 user@10.201.156.73

password: 321kgCODE

路径

/home/user/hotpot/hotpot-master

环境

```
source activate hotpot
```

下载数据

```
./download.sh
```

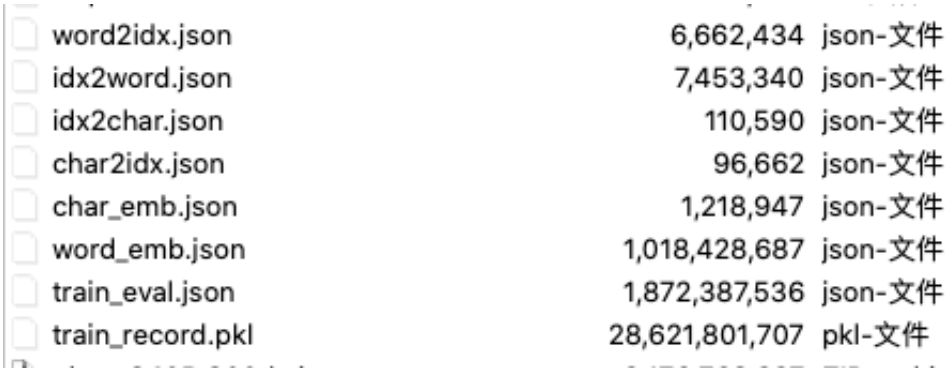
glove


```
unzip glove.840B.300d.zip
```

预处理

```
python main.py --mode prepro --data_file hotpot_train_v1.1.json --para_limit
2250 --data_split train
```

生成




```
python main.py --mode prepro --data_file hotpot_dev_distractor_v1.json --
para_limit 2250 --data_split dev
```

生成


```
python main.py --mode prepro --data_file hotpot_dev_fullwiki_v1.json --
data_split dev --fullwiki --para_limit 2250
```

生成

预处理流程

1. 通过`process_file`得到`example`和`eval_example`

```
process_file(config.data_file, config) #处理一个json文件
_process_article() #处理文件中的一个item
_process(sent, is_sup_fact, is_title=False) # 对每句话进行处理
```

```
# 对一条数据，解析成以下格式

example = {'context_tokens': context_tokens,
           'context_chars': context_chars,
           'ques_tokens': ques_tokens,
```

```
            'ques_chars': ques_chars,
            'y1s': [best_indices[0]],
            'y2s': [best_indices[1]],
            'id': article['_id'],
            'start_end_facts': start_end_facts # 由多个(start_token_id,
end_token_id, is_sup_fact=True/False)组成，每句话的起始token和终止token在
context_tokens中的位置，以及这句话是否是支撑事实。
          }

eval_example = {'context': text_context, # 合并成一个str的context
                'spans': flat_offsets,  # context中每个token的span
                'answer': [answer],
                'id': article['_id'],
                'sent2title_ids': sent2title_ids  # list，由多个[para_title,
idx]组成，表示context中每句话对应的标题以及在标题下的idx，title本身对应的idx为-1;
               }  # 汇总后写入{}_eval.json文件中
```

`best_indices`:

1. Yes, [-1, -1]
2. No, [-2, -2]
3. Yes or no 之外的答案，且不在上下文中 (0, 1)
4. Yes or no 之外的答案，且在上下文中，对应的位置

```
        answer = article['answer'].strip()
        if answer.lower() == 'yes':
            best_indices = [-1, -1]
        elif answer.lower() == 'no':
            best_indices = [-2, -2]
        else:
            if article['answer'].strip() not in ''.join(text_context):
                # in the fullwiki setting, the answer might not have been
retrieved
                # use (0, 1) so that we can proceed
                best_indices = (0, 1)
            else:
                _, best_indices, _ = fix_span(text_context, offsets,
article['answer'])
                answer_span = []
                for idx, span in enumerate(flat_offsets):
                    if not (best_indices[1] <= span[0] or best_indices[0] >=
span[1]):
                        answer_span.append(idx)
                best_indices = (answer_span[0], answer_span[-1])
```

2. 得到字词嵌入矩阵

若word2idx_file文件存在，载入此文件作为word2idx_dict，否则通过get_embedding得到word_emb_mat, word2idx_dict, idx2word_dict；

字嵌入同理；

3. build_features构建特征

Para_limit用来限制context_tokens长度，在训练阶段只选择len(context_tokens) > Para_limit的数据进行训练

```
datapoints.append({'context_idxs': torch.from_numpy(context_idxs), # 一维张量，句子中每个token的idx，长度为para_limit
    'context_char_idxs': torch.from_numpy(context_char_idxs), # 二维张量，[para_limit, charlimit]，字id矩阵，[i, j]第i个token中第j个字符的id
    'ques_idxs': torch.from_numpy(ques_idxs),  # 类似context_idxs
    'ques_char_idxs': torch.from_numpy(ques_char_idxs), # 类似context_char_idxs
    'y1': y1, # 见`best_indices`
    'y2': y2,
    'id': example['id'],
    'start_end_facts': example['start_end_facts']})  # 由多个(start_token_id, end_token_id, is_sup_fact=True/False)组成，每句话的起始token和终止token在context_tokens中的位置，以及这句话是否是支撑事实。
```

```
torch.save(datapoints, out_file)
```

存入{}.record.pkl文件中