Tensorflow文本分析框架介绍及源码解读

洪侠
xia.hong@baifendian.com

大纲

- 本次主要分享http://git.baifendian.com/Text_Modeling/tf_text_modeling上一个文本建模框架，主要包括以下部分：
- 一、设计的动机、理念、过程
- 二、框架基本流程介绍；文本建模基本介绍
- 三、框架详解、代码解读
- 四、例子与演示
- 五、答疑
- **目标：帮助大家掌握该框架，能使用以及自行阅读源码**

- 一、设计的动机、理念、过程

动机

- 动机：在不断的开发中，想把有用代码积累下来，减少碎片代码，提高代码复用的能力，减少重复开发，形成社区

- 与机器学习类似，这是由人参与的学习系统：
    - 复用率高的代码应该模块化沉淀下来，并形成良好的结构，不断更新支持后续不同的人与不同的项目的需求
    - 任务代码共享，反向传播。
    - ML社区实际上在不同层面自动做了这个沉淀与迭代。

- 关键概念：版本控制；代码质量；结构设计

代码沉淀举例

- 功能：
    - Bash：…
    - python (标准库与常用扩展库的使用)
        - 多进程：如何设计成更简洁的函数 MpWork(reader, process_func,writer, workers=20, batch_num=10000)
        - 常用的wraps。@compute_time @cache_func @try_func
        - Nlp 常用util:
            - 各种文件的读写函数、工作常见的nlp功能的API与脚本：分词，繁简体，翻译，词向量相似等等
- 结构举例：
    - Util
        - Basic
        - Nlp
        - tf_utils
- 这是一个关于团队协作编程的话题。我自己一个人的话，就是积累自己一个人的各种UTIL； 团队会更复杂。我觉得是有价值的。

关于这个框架

- 优点：
    - 代码表达即知识。
    - 它尽量地在不同任务之间共享代码。它可以帮助我们清晰地认识不同任务之间的差别/联系，即它能让我们以一种最佳的方式存储这些知识。
    - 利于新手学习
    - 因为不同任务被一套代码实现，天然适合多任务，不管是同时训练还是分开训练
- 缺点：
    - 相对单单一个任务更为臃肿。如果要新增一个干净的单任务的repo，可以在它上面修改，把没用的都删了。
    - 逻辑分支有点多，不保证没Bug，因为比较轻量，看着改吧 =、=

二、本框架代码基本流程介绍、文本建模介绍

本框架基本功能

- 一个正常的TF项目：数据；模型；训练；评估；预测
- 模块化的数据读取；以及利用这些函数构成一些data_loaders
- 支持不同情况下tag/seq_tag模型定义代码
- 主训练代码，支持各种tensorboard展示，模型存储
- 一个Config文件控制所有行为
- 预测代码
- 简单演示

本框架基本流程

- 数据
    - 我的方法是用python生成一个迭代器
        - class data(): __iter__()，返回每个batch模型需要的那些Numpy向量。python的可读性、自由度比tf的那些数据接口舒服多。
        - for batch in train_data: fd=zip(model.inputs, batch)
- 如何用python生成，进行了模块化/函数式的设计，增加了代码复用
- 如果需要用一个读线程去读，然后用Queue存储，进行乱序等操作，可以调tf.data的Dataset和相关操作封装这个迭代器，比以前的data_queue接口干净多。【目前我还没用…】

本框架基本流程

- 模型：
  - 常用的基础代码放在tf_utils
  - **with tf.name_scope(self.name_scope):**
    self.build_inputs()
    self.build_embeddings()
    self.build_text_repr()
    self.build_outputs()
    self.build_others()
  - 所有Tensor都会在这里定义好

本框架基本流程

- 训练
  - Instance model, data
  - for batch in range(batchs):
    - For k,batch_data in enumerate(train_data):
      - If k%10==0:
        - Do step_summary and train
        - Else: train
      - Do eval on dev/test, summary, model_save
  - Start tenor board
- 预测
  - 方法1：存在源代码，重新生成一个模型，加载参数
    - 优点：可以动态生成计算图(修改batch_size，seq_len等)
    - 缺点：暴露了源码，需求对应版本的源码
  - 方法2：直接根据模型文件生成模型
    - 优点：不需要源码
    - 缺点：不容易修改计算图(比如训练是batch_size=100,预测希望=5，计算图中把batch_size定死了)；可以专门生成一个供预测的动态图。

常见的文本任务

Tag	Exclusive(small tags)=classification	情感分析；粗互斥话题分类；
	No_exclusive(small tags)=multilabel	非互斥交叉话题标签体系；
	Exclusive(Large tags)	大量互斥图谱实体分类体系
	No_exclusive(Large tags)	w2v；多对多图关系建模
seq_tag	与上完全相同，除了标签打在每个tok上。	分词、词性、NER、SRL；(10-400) Entity linking； 词义/实体 消歧；
struct prediction	不仅要打标签，还要输出一个结构	文法；语义分析；关系识别(知识图谱)；
Sequence generate	它不是对seq打tag，而是encoder-decoder最终产生seq of tags	语言模型；翻译

三、框架详解及源码解读

数据详解

- 模块化的loader
- 演示
- 讲解clf_data, 以及用到的tf_utils函数

模型部分

```
with tf.name_scope(self.name_
    self.build_inputs()
    self.build_embeddings()
    self.build_text_repr()
    self.build_outputs()
    self.build_others()
```

- inputs：定义网络输入、输出的tensor
- embeddings:定义embedding layer, 包括可能的字词联合
- text_repr: 通过各种模型编码表示序列，按任务输出seq of vec, or sent_vec
- Outputs:构建最后几层，scores，Loss， metrics
- others：初始化；优化算法；summary；图的说明

模型text_repr详解

- Embedings: char/word/word&char
- text_repr:
    - cnn: text-cnn, gated-text-cnn(Language Modeling with Gated Convolutional Networks),
    - Rnn:cell=(lstm/gru/sru/) bi=(True/False)
    - cnn_rnn, rnn_cnn
    - Get sent_vec :
        - Add/add_idf/pooling/[-1]/attn_add
    - wide&deep, dependency
- Loss：见下图

Loss详解

LOSS		small_tag_num	big_tag_num	
tag	exclusive	Softmax_with_energy_loss	**sampled_softmax/hs**	**https://zhuanlan.zhihu.com/p/34044634**
	no_exclusive	sigmoid_with-energy_loss，Nbinary softmax classifiar	**sampled_sigmoid=nce/hs**	
seq_tag	exclusive	* 直接的loss部分与上相同，不同的地方在于训练时LOSS的其他部分，预测时的行为 * 如果NO CRF，直接通过softmax判断，与Tag逻辑相同 * 如果做了CRF，影响了loss，也影响了序列的概率判断，详情见tf crf实现 * 如果做了seq generator，训练是与上相同，预测使用beam_search(前一个时间点的输出作为后一一个时间点的输入) * 如果是做结构预测，TODO		
	no_exclusive			

训练部分与预测部分

- 直接看代码吧，不多

summary,profile,评估

- Summary:
  - step_summary
  - eval_summary
- profile:
  - time Line

四、例子与演示

- 1、分类任务
- 2、多语言迁移学习多标签
- 3、序列标注
- 4、词向量
- [5、wikidata 多语言超大量标签任务]
- [6、语言模型]