Readme

用CVAE实现Image Caption的代码框架

文件结构

```
├── scripts  # 数据预处理，整理为指定格式并保存到data中
│   ├── build_vocab.py  # 构建词表vocab.pkl
│   ├── generate_cocoeval.py  # 生成符合pycoco计算指标的格式pycocoref_val/test.json
│   ├── generate_dataset.py  # 生成最终使用的dataset_train/val/test.json
│   ├── generate_resnet_feat.py  # 预先生成coco数据集所用所有图像的resnet152特征以便后续直接使用
│   └── prepro_ann.py  # 预处理instances_train/val2014.json，得到object categories的标注ann_dict.json
├── data  # 原始数据和预处理后的数据
│   ├── ann_dict.json
│   ├── dataset_coco.json  # karpathsplit给出的原始coco数据集
│   ├── dataset_test.json
│   ├── dataset_train.json
│   ├── dataset_val.json
│   ├── instances_train2014.json  # coco2014目标检测原始标注
│   ├── instances_val2014.json  # coco2014目标检测原始标注
│   ├── pycocoref_test.json
│   ├── pycocoref_val.json
│   └── vocab.pkl
├── models  # CVAE模型
│   └── CVAE
│       ├── __init__.py
│       ├── p_decoder.py
│       ├── q_encoder.py
│       └── vae_framework.py
├── utils  # 一些辅助代码
│   ├── log_print.py  # 训练中print一些信息
│   ├── tensorboard_writer.py  # 将训练以及val的一些重要信息保存在tensorboard中
│   └── vocab.py  # 自定义的词表类
├── config.py  # 命令行参数
├── data_load.py  # 数据加载
├── eval_metrics.py  # val和test中用到一些计算指标的函数
├── loss.py  # 损失函数
├── train_cvae.py  # 训练
└── test_cvae.py # 测试
```

训练&测试

1. 将项目拷贝到服务器上，由于项目已经预处理完成，因此可直接进行训练
2. 修改config.py中的log_dir参数为自己存储的位置`'/home/username/checkpoints/CVAE_Caption/log/{}`
3. 在服务器上开始训练，例如`CUDA_VISIBLE_DEVICES=0 python train_cvae.py --id cvae_u0.5_k0.05 --unk_rate 0.5 --kl_rate 0.05`，其中`id`参数为模型的名称必须指定，其余参数可选，参见config.py
4. 训练过程中，可通过端口映射登录服务器，例如`ssh -L 16038:127.0.0.1:6038 chengkz@210.28.132.173`，将服务器上的6038映射到本地的16038，之后在服务器激活虚拟环境后启动tensorboard`tensorboard logdir=/home/chengkz/checkpoints/CVAE_Caption/log --port=6038`，之后就可以在本地`http://localhost:16038`查看
5. 训练完成后，使用`CUDA_VISIBLE_DEVICES=0 python test_cvae.py --id cvae_u0.5_k0.05 --step 120000`进行测试，通过`id`和`step`参数指定要测试的模型名称和训练步数

说明

由于时间原因，本次代码的结构和之前褚有刚学长给出的并没有完全统一，但大致结构是类似的；因此同学们目前可跳过其中的数据预处理等繁琐的部分，掌握核心部分（模型代码、训练和测试代码），学会使用该框架训练，并在其基础上进行修改增添；代码中重要部分基本带有注释。

下面是该框架中相比之前代码的一些可能变化：

- 数据预处理上，将原始karpathsplit给出的coco数据又按照自己想要的格式进行了处理；
- 为了加速训练，预先生成了coco中所有图片的resnet152特征，训练测试时直接加载该特征，也就意味着训练测试过程中不包含编码端
- 训练时使用tensorboard以便随时查看分析训练情况，tensorboard是一种重要的训练辅助工具，可参考https://pytorch.org/docs/master/tensorboard.html并查找资料学习使用
- 为了满足oracle evaluation等需要，手动改写了pycoco的部分源码，使得能够直接计算自己生成的一组句子和给出的一组ref情况下的指标，如果需要可在`/home/data_ti4_c/chengkz/anaconda3/envs/nlp_caption/lib/python3.7/site-packages/pycocoevalcap`中查看和修改
- 除了oracle evaluation外，还添加了一些衡量多样性的指标，详见`test_cvae.py`