Assignment 3

Files

data/

data_utils.py: Helper functions or classes used in data processing.

process.py: Process a raw dataset into a sample file.

back_translate.py：Translate a raw dataset into English and back to Chinese.

embed_replace.py: Replace low tfidf-scored tokens with another token that is similar in the embedding space.

semi-supervise.py: Use semi-supervised learning to generate new source from existing reference.

output/

Place new data generated by data augmentation.

(replaced.txt, back_translated.txt, and semi_beam.txt...)

saved/

Put tfidf files here.

(tfidf.dict, tfidf.model...)

word_vectors/

Put the word-vector file `merge_sgns_bigram_char300.txt` here

model/

config.py: Define configuration parameters.

dataset.py: Define the format of samples used in the model.

evaluate.py: Evaluate the loss in the dev set.

model.py: Define the model.

predict.py: Generate a summary.

rouge_eval.py: Evaluate the model with ROUGE score.

train.py: Train the model.

utils.py: Helper functions or classes used for the model.

vocal.py: Define the vocabulary object.

saved_model/

baseline/

Save baseline model.

pgn/

Save PGN model.

cov_pgn/

Save PGN model with coverage mechanism.

ft_pgn/

Save fine-tuned PGN model.

ss_pgn/

Save scheduled_sampling PGN model.

wt_pgn/

Save weight_tying PGN model.

pgn_big_samples/

Save PGN model training with all data (big_samples.txt) which includes original data set and the augmented ones.

files/

Place original data files, augmented data from data/output here for training. Rouge file is also placed here.

(HIT_stop_words.txt, train.txt, dev.txt, test.txt, rouge_result.txt, big_samples.txt...)

runs/

Save logs here for TensorboardX.

TO-DO list:

必备资料

为了完成以下任务，我们需要逐步熟悉、掌握Pytorch框架，所以请大家在完成每一个模块时先查阅一下Pytorch的API文档，弄清楚要使用的模块是做什么的以及如何使用。此外，模型的具体实现，需要参见论文 *Get To The Point: Summarization with Pointer-Generator Networks*。

作业3简介

这次我们要在作业2的PGN网络基础上，加入两种优化技巧：Weight tying，Scheduled sampling。和作业2的PGN模型进行比较查看优化效果。为了达到这个目的，我们只需要在model/config.py这个文件下面，定义两个变量`weight_tying`与`scheduled_sampling`，并且在model/model.py，model/train.py中实现代码后通过这个变量来控制我们的模型是Weight tying 还是 Scheduled sampling。

总结一下，第一部分作业我们需要训练三个模型：

1. PGN，令`pointer = True`即可。(默认`source = 'train'`，请看下面第二部分数据增强作业的说明)
2. PGN (with Weight tying)，令`pointer = True`以及`weight_tying = True`
3. PGN (with Scheduled sampling)，令`pointer = True`，`scheduled_sampling= True`

此外，作业3还要求大家完成数据增强的一些方法：单词替换，回译，半监督学习。为了达到这个目的，我们只需要在data/这个目录下面定义三个新的模块：embed_replace.py，back_translate.py，semi-supervised.py。分别运行这三个模块会形成三个新的样本replaced.txt,back_translated.txt和semi_beam.txt。我们将这三个样本和原始的train.txt文本合并成一个大样本big_samples.txt，拷贝到files/目录下。为了达到使用big_samples.txt样本训练的目的，我们只需要在model/config.py这个文件下面，定义一个变量`source`来控制`data_path`的路径取值，从而控制train.py模块训练数据的来源。

总结一下，第二部分作业我们需要训练一个模型：

1. PGN(training with big_samples.txt)，令`pointer = True`以及`source = 'big_samples'`即可。

   注：使用原始数据训练的PGN网络已经在第一部分作业中训练过了，即`pointer = True`以及`source = 'train'`

模块1: 单词替换

模块1任务目标： 通过替换reference中的词生成新的reference样本。这里存在一个问题，如果我们替换了样本中的核心词汇，比如将文案中的体现关键卖点的词给替换掉了，可能会导致核心语义丢失。所以我们可以通过 tfidf 权重对词表里的词
进行排序，然后替换排序靠后的词。选择在 embedding 的词向量空间中寻找语义最接近的词进行替换。TFIDF的计算以及相似词向量查找可以通过gensim实现，相关API请参看https://radimrehurek.com/gensim/auto_examples/index.html# 。中文Word2vec词向量可以在这个github的repo下载：Chinese Word Vectors 中文词向量。

data/embed_replace.py

任务1，完成extract_keywords函数。

根据TFIDF确认需要排除的核心词汇。

任务2，完成replace函数。

embedding 的词向量空间中寻找语义最接近的词进行替换。

任务3，完成generate_samples函数。

替换全部的reference，和对应的source形成新样本

总结一下思路：

1. 计算TFIDF
2. 找到需要替换的词，在 embedding 的词向量空间找到相似的词进行替换。
3. 形成新的训练数据。

可能用到的方法：

gensim.models 中的 KeyedVectors, TfidfModel

KeyedVectors.most_similar

模块2: 回译

模块2任务目标：利用百度translate API 接口将source ， reference翻译成英语，再由英语翻译成汉语，形成新样本。具体请参看官网：https://api.fanyi.baidu.com/ 。另外不同的语言样本的训练效果会有所不同，建议多尝试几种中间语言，比如日语等。

data/back_translate.py

任务1：完成translate函数

建立http连接，发送翻译请求，以及接收翻译结果。关于这一部分可以参考百度接口translate API 的demo。

任务2：完成back_translate函数

对数据进行回译。

任务3：完成translate_continue函数

将全部数据进行翻译存储。

总结一下思路：

1. 建立http链接
2. 回译
3. 形成新文档

可能用到的参考：

百度translate API 的demo

模块3: 自助式样本生成

模块3任务目标：半监督学习是利用标签数据+无标签数据混合成的训练数据对模型进行训练。半监督学习利用有标签数据建立模型，对未标签样例进行标签。在本次作业中，就利用一种类似半监督学习的方式，我们训练出一个文本生成模型后，我们可以利用训练好的模型为我们原始样本中的 reference 生成新的 source，并作为新的样本继续训练我们的模型。

data/semi-supervise.py

任务1：完成semi_supervised函数

我们可以使用上次作业2中训练好的PGN (fine-tuned with coverage)模型将reference送入模型，生成新的source。

举个例子原始数据如下：

source：

帕莎 太阳镜 男士 太阳镜 偏光 太阳眼镜 墨镜 潮 典雅 灰 颜色 选择， 细节 特色 展示， 角度 展示， 风尚 演， 金属， 透光 量， 不易 变形， 坚固耐用， 清晰 柔和， 镜片 材质， 尼龙 高， 型号， 镜架 材质， 金属 镜腿， 镜布 镜盒， 鼻 间距， 轻盈 鼻托， 品牌 刻印， 无缝 拼接眼镜配件类型 镜盒 功能 偏光 类型 偏光太阳镜 镜片材质 树脂 风格 休闲风 上市时间 2016年夏季 镜框形状 方形 适用性别 男 镜架材质 合金

reference：

夏天 到 了， 在 刺眼 的 阳光 下 少不了 这 款 时尚 的 男士 太阳镜！ 时尚 的 版型， 适用 各种 脸型，突出 您 的 型 男 风范 。 其高 镍 金属 材质 的 镜架， 十分 的 轻盈， 带给 您 舒适 的 佩戴 体验 。

将reference送入模型后生成的数据new source：

版型 设计， 时尚 百搭， 适合 多种 场合 佩戴 。 时尚 金属 拉链， 经久耐用 。 带给 你 舒适 的 佩戴 体验 。 风范 与 铰链 的 镜架， 舒适 耐磨， 不易 变形 。 带给 你 意想不到 的 修身 效果， 让 你 的 夏季 充满 着 不 一样 的 魅力， 让 这个 夏天 格外 绚烂， 让 这个 夏天 格外 绚烂 。 时尚 的 版型， 让 你 穿 起来 更 有 男人味， 让 这个 夏天 格外 绚烂 绚烂， 让 这个 夏天 格外 绚烂 。 时尚 的 版型， 让 你 穿 起来 更 有 男人味， 让 这个 夏天 格外 绚烂 绚烂， 是 秋装 上 上 之选 。

所以新生成的数据如下：

版型 设计， 时尚 百搭， 适合 多种 场合 佩戴 。 时尚 金属 拉链， 经久耐用 。 带给 你 舒适 的 佩戴 体验 。 风范 与 铰链 的 镜架， 舒适 耐磨， 不易 变形 。 带给 你 意想不到 的 修身 效果， 让 你 的 夏季 充满 着 不 一样 的 魅力， 让 这个 夏天 格外 绚烂， 让 这个 夏天 格外 绚烂 。 时尚 的 版型， 让 你 穿 起来 更 有 男人味， 让 这个 夏天 格外 绚烂 绚烂， 让 这个 夏天 格外 绚烂 。 时尚 的 版型， 让 你 穿 起来 更 有 男人味， 让 这个 夏天 格外 绚烂 绚烂， 是 秋装 上 上 之选 。 夏天 到 了， 在 刺眼 的 阳光 下 少不了 这 款 时尚 的 男士 太阳镜！ 时尚 的 版型， 适用 各种 脸型， 突出 您 的 型 男 风范 。 其高 镍 金属 材质 的 镜架， 十分 的 轻盈， 带给 您 舒适 的 佩戴 体验 。

总结一下思路：

1. 将reference送入模型进行预测
2. 生成新的source 和原来的reference组成新样本存入文件。

(注意model/config.py下这两个参数gamma，max_dec_steps，会影响生成数据的长度)

模块4: Weight tying

模块4任务目标：共享Decoder的input embedding 和 output embedding 权重矩阵，使得其输入的词向量表达具有一致性。具体请参看论文：https://arxiv.org/abs/1608.05859 。除此之外共享权重还可以尝试加入Encoder的input embedding。（three-way tying）

model/model.py

任务1：完成Encoder类，Decoder类，PGN类

本次作业中使用的是three-way tying，即Encoder的input embedding，Decoder的input emdedding和Decoder的output embedding之间的权重共享。

model/predict.py

任务2：完成Predict类

如果做上文提到的three-way tying，那么对Encoder类的除了在PGN类中调用外，也在Predict类中进行了调用。因此也需要在Predict类中做相应的修改。

总结一下思路：

1. `weight_tying = Ture` Encoder的input embedding，Decoder的input embedding 和 output embedding 共享权重
2. `weight_tying = False` Encoder的input embedding，Decoder的input embedding 和 output embedding 权重分开

可能用到的方法：

torch.t 矩阵的转置，torch.mm矩阵的乘法，可以通过`nn.Embedding.weight` weight属性获得权重

模块5: Scheduled sampling

模块5任务目标：每个 time step 以一个 p 的概率进行 Teacher forcing，以 1-p的概率不进行 Teacher forcing。p 的大小可以随着 batch 或者 epoch衰减，即开始训练的阶段完全使用 ground truth 以加快模型收敛，到后面逐渐将 ground truth 替换成模型自己的输出，到训练后期就与预测阶段的输出一致。详见Scheduled Sampling for Sequence Prediction with Recurrent Neural Networks。

model/utils.py

任务1：完成ScheduledSampler类

可以通过epoch或者batch数控制按照一定的概率给出是否需要进行teacher forcing的指示。

model/model.py

任务2：完成PGN类

在PGN模型的前向传播过程中执行，Teacher forcing或者使用上一个step的预测结果作为输入

model/train.py

任务3：完成train函数

调用ScheduledSampler类，确定是否使用Teacher forcing，以及调用PGN模型的前向传播过程

model/evaluate.py

任务4：完成evaluate函数

这里也用到了PGN模型的前向传播过程，所以也需要动手修改。

总结一下思路：

1. 每个batch 或者 epoch确认是否进行teacher forcing
2. 在前向传导中执行

测试:

作业3 **embedding_size =512**	**rouge1**	**rouge2**	**rougeL**
PGN	'f': 0.245650	'f': 0.042663	'f': 0.154976
PGN coverage	'f': 0.267554	'f': 0.044646	'f': 0.162138
PGN scheduled_sampling	'f': 0.244096	'f': 0.043349	'f': 0.152910
PGN weight_tying (decoder input, output)	'f': 0.185832	'f': 0.031970	'f': 0.159547
PGN weight_tying three-way tying	'f': 0.188746	'f': 0.033649	'f': 0.163048
PGN big_samples	'f': 0.245189	'f': 0.049502	'f': 0.162425