Code review - Chart-to-Text: Generating Natural Language Explanations for Charts by Adapting the Transformer Model

Original repository: Chart2Text by Jason Obeid

Code for Chart-to-Text: Generating Natural Language Explanations for Charts by Adapting the Transformer Model

Dataset

The dataset is stored in the dataset directory `dataset/` and is split into train, valid and test sets in the `data/` directory.

Data is split up into directories:

- `dataset/captions/` contains the cleaned chart captions and `dataset/captions_old/` contains the uncleaned chart captions
- `dataset/titles/` contains the cleaned chart titles and `dataset/titles_old/` contains the uncleaned chart titles
- `dataset/data/` contains the chart data
- `dataset/multiColumn/` contains the chart data for charts with more than two columns i.e. grouped bar charts and multi-line charts

Chart type breakdown:

	Line	Bar	Total:
Simple	3564	3199	6763
Complex	902	640	1542
Total:	4466	3839	

Images available seperately at https://github.com/JasonObeid/Chart2TextImages due to large size ~1GB

Step 1: Cleaning dataset

Clean the text within the chart titles and summaries:

```
cd utils/
python refactorTitles.py
```

```
cd utils/
python refactorCaptions.py
```

Step 2: Preprocessing

```
cd etc/
python templatePreprocess.py
```

- Converts data tables into a sequence of records (taken as input by the model): `data/*split*/trainData.txt`
- Cleans summary tokens and substitutes any possible tokens with data variables(e.g., **2018 -> templateValue[0][0]**): `data/*split*/trainSummary.txt`
- Cleans the title tokens: `data/*split*/trainTitle.txt`
- Labels the occurrences of records mentioned within the summary: `data/*split*/trainDataLabel.txt`
- Labels the summary tokens which match a record: `data/*split*/trainSummaryLabel.txt`
- Saves the gold summaries: `data/*split*/testOriginalSummary.txt`

Step 3: Extract vocabulary for each split

```
cd etc
python extract_vocab.py --table ../data/valid/validData.txt --summary ../data/valid/validSummary.txt
python extract_vocab.py --table ../data/test/testData.txt --summary ../data/test/testSummary.txt
python extract_vocab.py --table ../data/train/trainData.txt --summary ../data/train/trainSummary.txt
```

It will generate vocabulary files for each of them:

- `data/*split*/trainData.txt_vocab`
- `data/*split*/trainSummary.txt_vocab`

Step 4: Binarize the data for each split

```
cd ../model
python preprocess_table_data.py --table ../data/valid/validData.txt --table_vocab ../data/valid/validData.txt_vocab --table_label ..
python preprocess_table_data.py --table ../data/test/testData.txt --table_vocab ../data/test/testData.txt_vocab --table_label ../dat
python preprocess_table_data.py --table ../data/train/trainData.txt --table_vocab ../data/train/trainData.txt_vocab --table_label ..

python preprocess_summary_data.py --summary ../data/valid/validSummary.txt --summary_vocab ../data/valid/validSummary.txt_vocab --su
python preprocess_summary_data.py --summary ../data/test/testSummary.txt --summary_vocab ../data/test/testSummary.txt_vocab --summar
python preprocess_summary_data.py --summary ../data/train/trainSummary.txt --summary_vocab ../data/train/trainSummary.txt_vocab --su
```

Outputs the training data:

- Data Records: `data/*split*/trainData.txt.pth`
- Summaries: `data/*split*/trainSummary.txt.pth`

Note: if you get a dictionary assertion error, then delete the old .pth files in data subfolders and try again

Step 5: Model Training

```
python model/train.py \
    --model_path "experiments" \
    --exp_name "chart2text" \
    --exp_id "run1" \
    --train_cs_table_path data/train/trainData.txt.pth \
    --train_sm_table_path data/train/trainData.txt.pth \
    --train_sm_summary_path data/train/trainSummary.txt.pth \
    --valid_table_path data/valid/validData.txt.pth \
    --valid_summary_path data/valid/validSummary.txt.pth \
    --cs_step True \
    --lambda_cs "1" \
    --sm_step True \
    --lambda_sm "1" \
    --label_smoothing 0.05 \
    --sm_step_with_cc_loss False \
    --sm_step_with_cs_proba False \
    --share_inout_emb True \
    --share_srctgt_emb False \
    --emb_dim 512 \
    --enc_n_layers 1 \
    --dec_n_layers 6 \
    --dropout 0.1 \
    --save_periodic 40 \
    --batch_size 6 \
    --beam_size 4 \
    --epoch_size 1000 \
    --max_epoch 81 \
    --eval_bleu True \
    --sinusoidal_embeddings True \
    --encoder_positional_emb True \
    --gelu_activation True \
    --validation_metrics valid_mt_bleu
```

Step 6: Generation

Pretrained models can be downloaded from the links below:

- Baseline model (trained with data variables)
- Baseline model adapted from Li et al. https://github.com/gongliym/data2text-transformer (trained without data variables)

```
python model/summarize.py --model_path aug17-80.pth --table_path data/test/testData.txt --output_path results/aug17/templateOutput-p
```

Step 6.1: Postprocessing after generation

Substitute any predicted data variables:

```
cd etc/
python summaryComparison.py
```

Step 7: Evaluation

Step 7.1: "Content Selection" evaluation

```
cd studyOutcome/
python automatedEvaluation.py
```

Step 7.2: BLEU evaluation

The BLEU evaluation script can be obtained from Moses:

```
perl model/src/evaluation/multi-bleu.perl data/test/testOriginalSummary.txt < results/aug17/generated-p80.txt
```