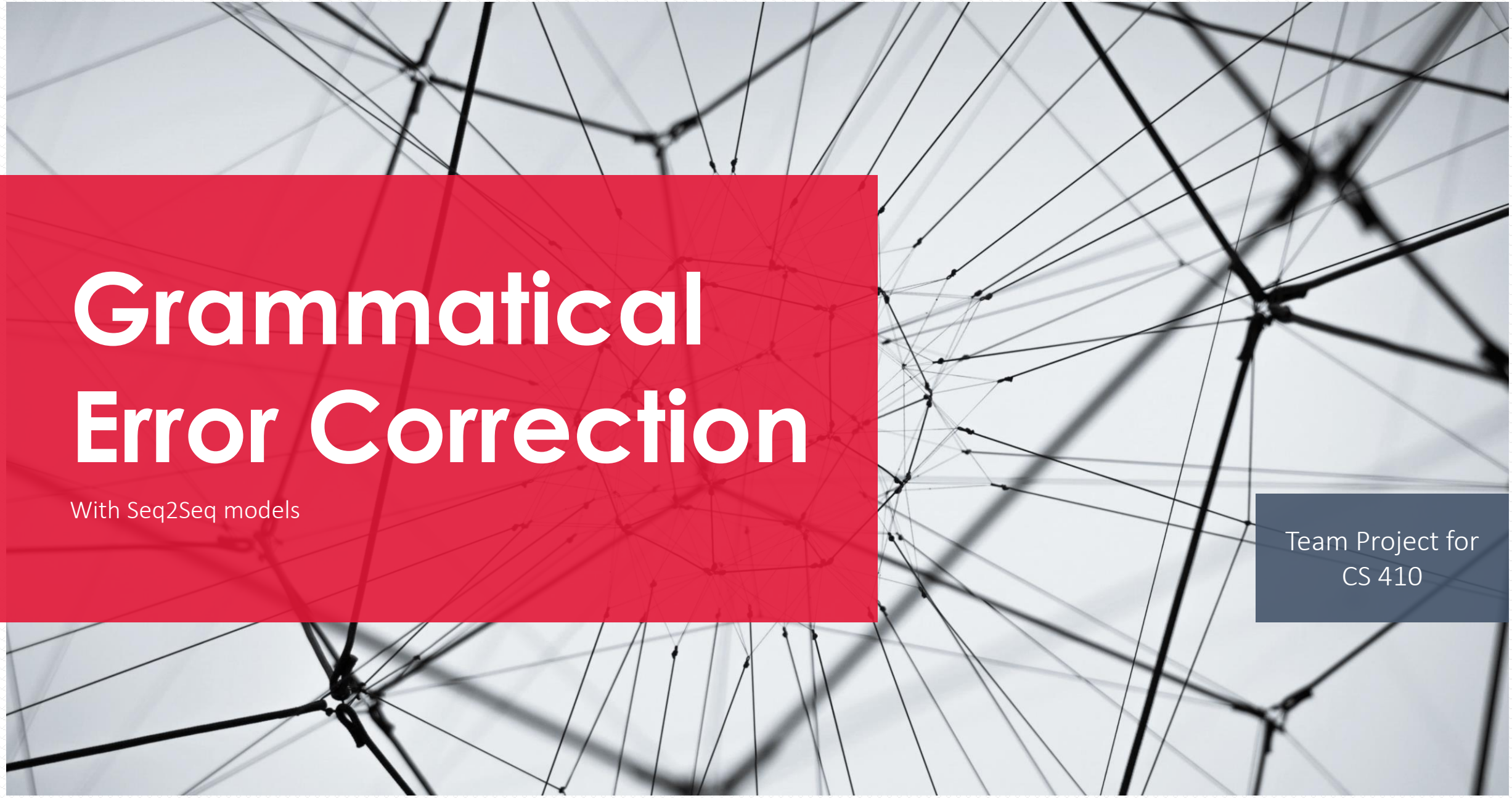
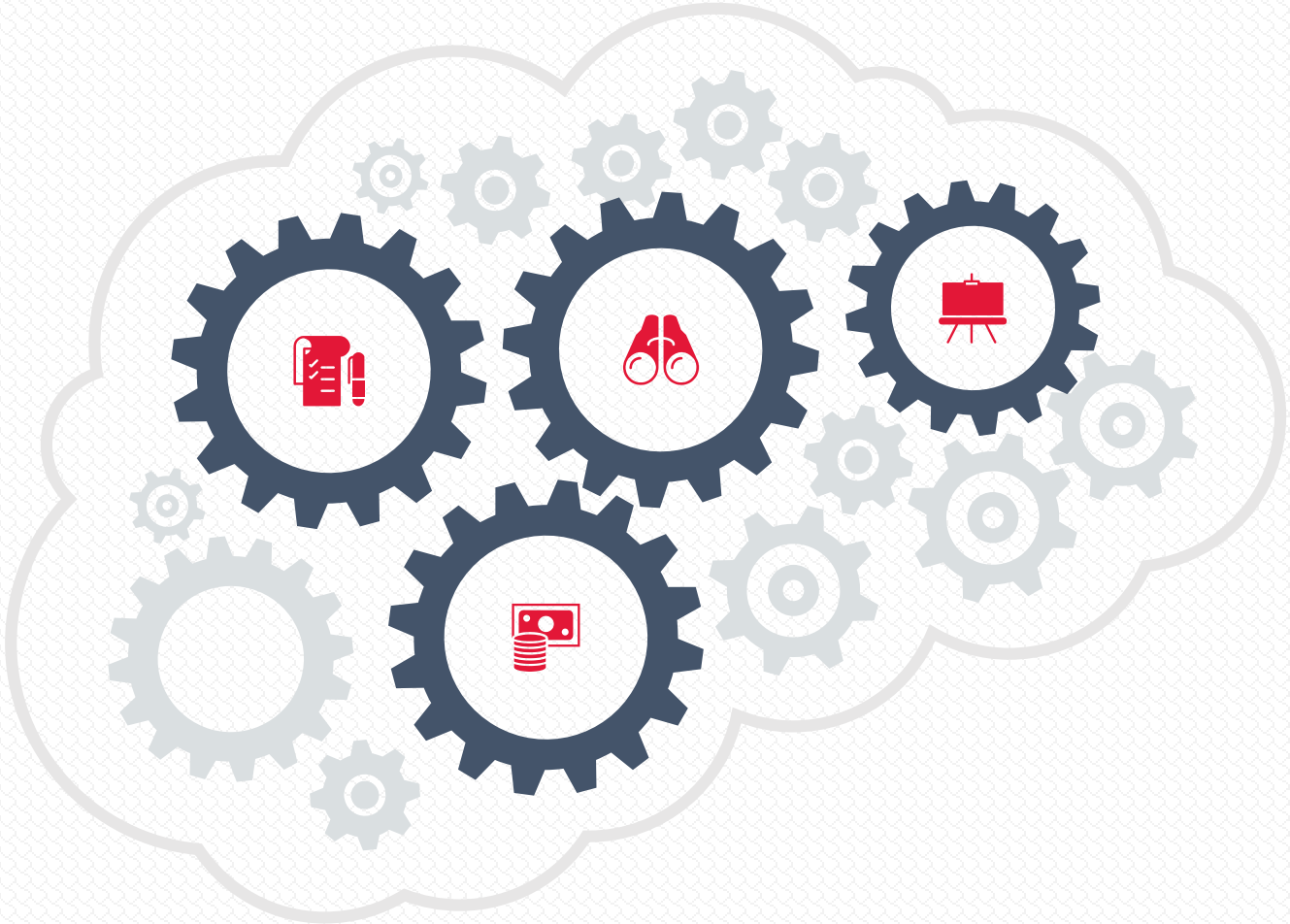


Grammatical Error Correction

With Seq2Seq models

Team Project for CS 410

Project Goals



The goal of this project is to implement a grammatical error correction model from paper "Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study“.

Learn how to use deep learning NLP toolkit such as PyTorch and fairseq.

While the original paper achieves human performance, this implementation is more about an empirical study on applying deep learning in NLP as a university team project.

What We Learned

Empirical Study

- Write code quickly

- Use a framework
  - Don't start from scratch. Use existing library, framework and toolkit.
  - Pre-processing, batching, checkpointing, common models are all included.
  - Tried both OpenNMT and fairseq NLP toolkits.
  - Settled with fairseq which is also what paper used.

- It is ok to just copy toolkit and components then modify them
  - This is officially recommended way to extending and playing with fairseq.
  - When you copy many times you will naturally learn how to refactor them later.

What We Learned

Empirical Study

- Run experiments
  - Run out of box samples to learn how toolkit works.
  - Run simple models to get a proof of concept and confidence.
- Keep track of what you ran
  - Source control constructed models and hyper-paramters.
  - Save the scripts so that you can repeat them again with other tweaks and datasets.
- Keep track of what you got
  - Many frameworks and toolkits will do this for you automatically.
  - Just need to keep folders clear and watch your hard drive free space.

What We Learned

Empirical Study

- Analyze model behaviour

- Feed model with some simplest tests
  - Look at generated values see if that's what you expected.
  - Look at scores and see if they make sense.

- Evaluate models with proper evaluation metric
  - Loss function for training.
  - Fluency score (cross entropy) and GLEU score etc.

What We Achieved

- A left-to-right 7-layer convolutional seq2seq model has been implemented using same architecture as the paper suggested.

```
FConvModel(
 (encoder): FConvEncoder(
  (embed_tokens): Embedding(137960, 500, padding_idx=1)
  (embed_positions): LearnedPositionalEmbedding(1024, 500, padding_idx=1)
  (fc1): Linear(in_features=500, out_features=1024, bias=True)
  (projections): ModuleList(
  )
  (convolutions): ModuleList(
   (0): ConvTBC(1024, 2048, kernel_size=(3,), padding=(1,))
   * 7
  )
  (fc2): Linear(in_features=1024, out_features=500, bias=True)
 )
 (decoder): FConvDecoder(
  (embed_tokens): Embedding(121816, 500, padding_idx=1)
  (embed_positions): LearnedPositionalEmbedding(1024, 500, padding_idx=1)
  (fc1): Linear(in_features=500, out_features=1024, bias=True)
  (projections): ModuleList(
  )
  (convolutions): ModuleList(
   (0): LinearizedConvolution(1024, 2048, kernel_size=(3,), padding=(2,))
   * 7
  )
  (attention): ModuleList(
   (0): AttentionLayer(
    (in_projection): Linear(in_features=1024, out_features=500, bias=True)
    (out_projection): Linear(in_features=500, out_features=1024, bias=True)
   )
   *7
  )
  (fc2): Linear(in_features=1024, out_features=500, bias=True)
  (fc3): Linear(in_features=500, out_features=121816, bias=True)
 )
)
```

What We Achieved

- The base convolutional seq2seq model has been trained using mostly same hyper parameters.
- Fluency score function, which is used for both boost training / learning and boost inference, has been implemented. For example, nature sentences get higher score.

```
[0.1977] It is a truth universally acknowledged , that a single man in possession of a goo
[0.1937] I am going to a party tomorrow . </s>
[0.1902] I am going to the party tomorrow . </s>
[0.1864] It was the best of times , it was the worst of times , it was the age of wisdom ,
[0.1654] Yesterday I saw a car . </s>
[0.1630] Tomorrow I am going to a party . </s>
[0.1540] I saw the car yesterday . </s>
[0.1473] Tomorrow I am going to party . </s>
[0.1383] Tomorrow I go to party . </s>
[0.1296] Yesterday I see car . </s>
[0.1280] Yesterday I saw car . </s>
```

What We Achieved

- GLEU score function, which is used to evaluate JFLEG test set, has been implemented. For example, similar sentences have higher GLEU score.

```
There are several reason|There are several reasons
O       There are several reason
H       There are several reasons       -0.029993820935487747
P       -0.0028 -0.0339 -0.0015 -0.0464 -0.0653
GLEU 100.00
For not use car|Not for use with a car
O       For not use car
H       For not use car -0.06429481506347656
P       -0.1332 -0.0239 -0.1537 -0.0102 -0.0006
GLEU 27.40
Every knowledge is connected each other|All knowledge is connected
O       Every knowledge is connected each other
H       Every knowledge is connected to each other      -0.17184138298034668
P       -0.1573 -0.0054 -0.0348 -0.0004 -0.9934 -0.0301 -0.0026 -0.1507
GLEU 18.58
```

What We Achieved

- An error generation model has been implemented to generates more synthetic data from original training dataset, which will in turn boost training of the base model. For example.

```
S-3654  How many languages have you studied ?
H-3654  How many language have you studied ?    -0.19821381568908691
H-3654  How many languages do you study ?       -0.5254995822906494
H-3654  How much languages have you studied ?   -0.5455195903778076
H-3654  How many languages are you studied ?    -0.5917201042175293
```

What We Achieved

- Basic inference with the base model has been implemented. For example, entered sentence is corrected in multiple ways.

```
In the world oil price very high right now .

Iteration       0
O       In the world oil price very high right now .
H       In the world oil price very high right now .    0
Fluency Score: 0.1503

Iteration       1
O       In the world oil price very high right now .
H       In the world oil prices very high right now .   -0.2768438458442688
Fluency Score: 0.1539
Iteration       1
O       In the world oil price very high right now .
H       In the world oil prices are very high right now .       -0.31139659881591797
Fluency Score: 0.1831
Iteration       1
O       In the world oil price very high right now .
H       In the world oil price is very high right now . -0.3594667315483093
Fluency Score: 0.1731
Iteration       1
O       In the world oil price very high right now .
H       In the world oil price very expensive right now .       -0.4148099422454834
Fluency Score: 0.1434

Best inference  "In the world oil prices are very high right now ."     (0.1831)
```

What We Achieved

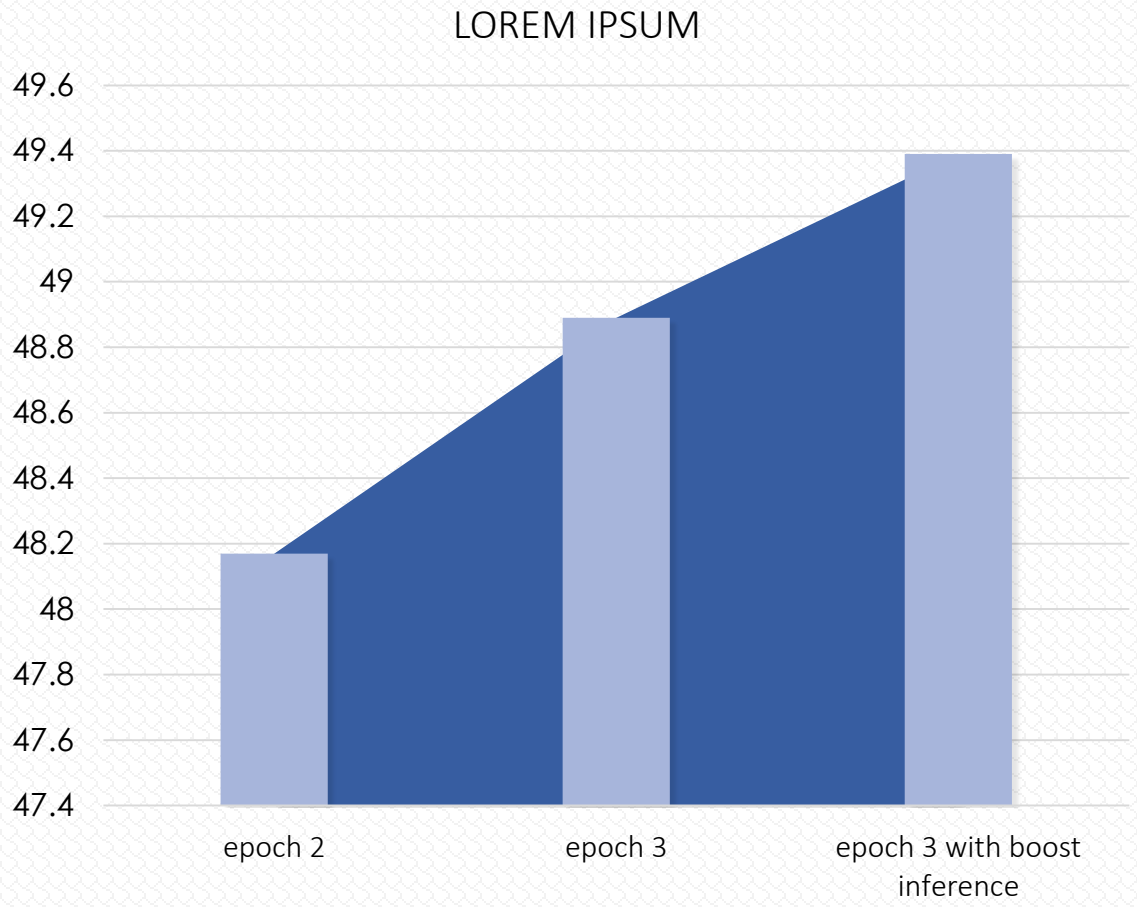
- Boost inference has been implemented to use both base model and language model. For example, entered sentence is corrected in multiple ways, the best scored one is chosen for multiple rounds of correction, until the score cannot be improved.

```
In the world oil price very high right now .

Iteration       0
O       In the world oil price very high right now .
H       In the world oil price very high right now .    0
Fluency Score: 0.1503

Iteration       1
O       In the world oil price very high right now .
H       In the world oil prices are very high right now .       -0.31139659881591797
Fluency Score: 0.1831

Boost inference from    "In the world oil prices are very high right now ."     (0.1831)

Iteration       2
O       In the world oil prices are very high right now .
H       In the world oil prices are very high now .     -0.4246770739555359
Fluency Score: 0.1883

Boost inference from    "In the world oil prices are very high now ."   (0.1883)

Iteration       3
No better scroed sentences.

Best inference  "In the world oil prices are very high now ."   (0.1883)
```

What We Achieved

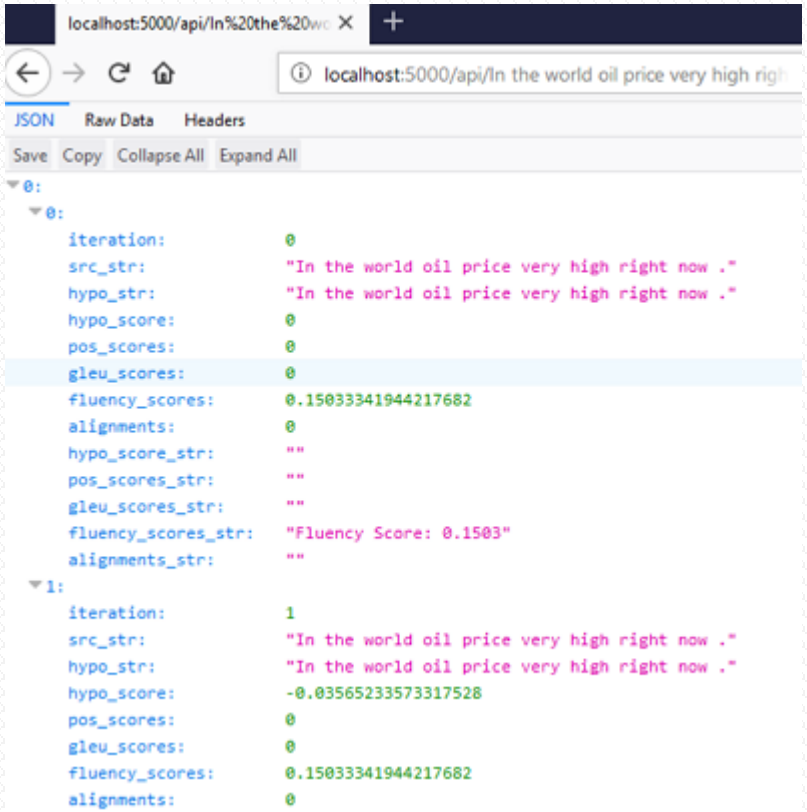
JFLEG test set GLEU

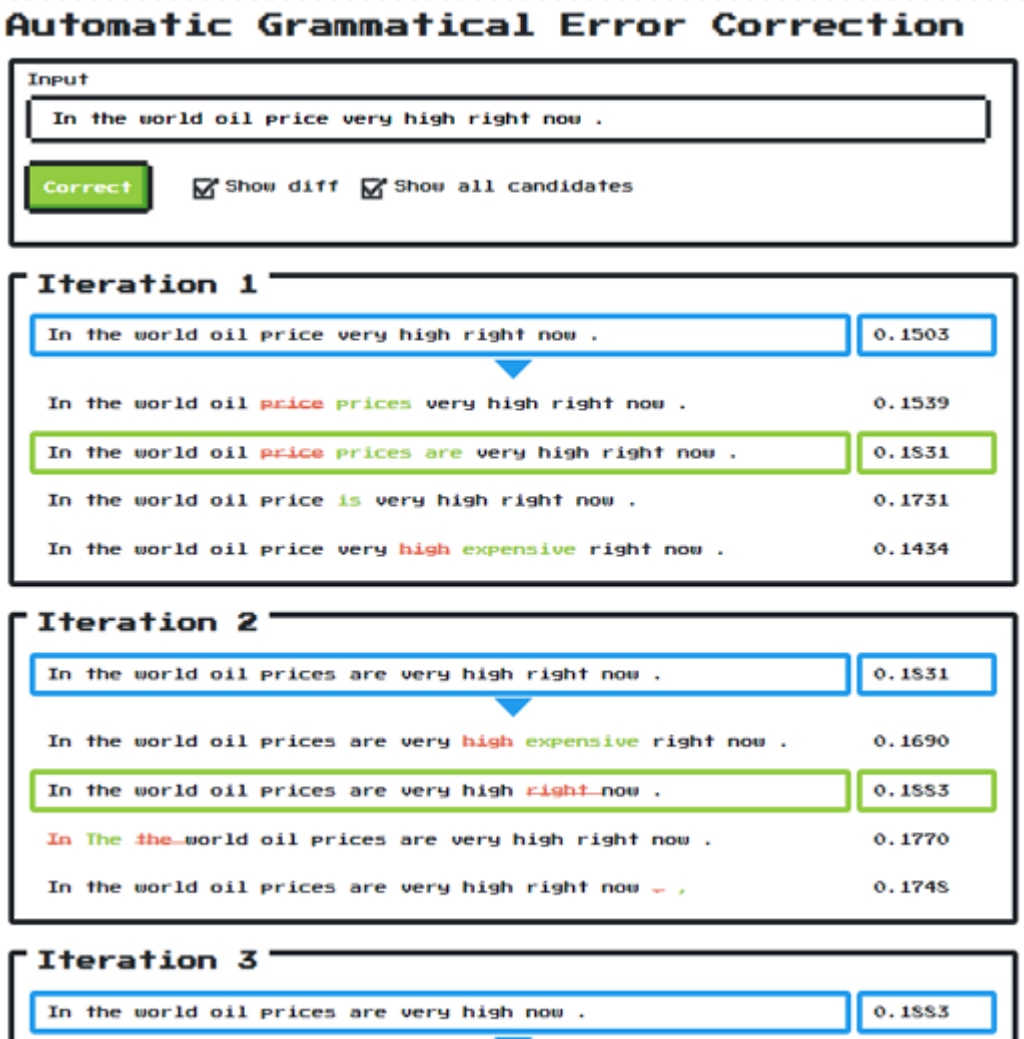
- The base model has a GLEU score 48.17 on JFLEG test set when it was trained for 2 epochs.
- The base model has a GLEU score 48.89 on JFLEG test set when it was trained for 3 epochs.
- The introduction of boost inference increased GLEU from 48.89 to 49.39. The percentage of increase is consistent with the paper (≈ 1%).



What We Achieved

- An enhanced interactive mode with RESTful API and Web GUI.
  - RESTful API
  - Web GUI





Demo

- Either clone the repository and follow the instructions (the hard way)
  - All source code and scripts can be found at
  - https://github.com/rgcottrell/pytorch-human-performance-gec
  - Pre-trained model can be downloaded at
  - https://usgpudisks915.blob.core.windows.net/gec/cnn/checkpoint_best.zip

- Or access the Web GUI to test pre-trained model (the easy way)
  - http://gec.eastus.cloudapp.azure.com/

Instructions

Initialize Submodules

- After checking out the repository, be sure to initialize the included git submodules:

```
git submodule update --init –recursive
```

- The reasons of using them as submodules rather than Python package are:
  - some scripts and functions might need be patched in order to work properly.
  - a few scripts are modified based on the original scripts, which is the officially recommended way of using fairseq.

Instructions

Install Required Dependencies

- The environment used for the development is Windows 10 64bit + Python 3.6 + CUDA 9.2 + pytorch 0.4.1.
- PyTorch can be installed by following the directions on its project page. Conda is recommended as it will install CUDA dependencies automatically.

```
conda install pytorch cuda92 -c pytorch
pip3 install torchvision
```

Instructions

Install Required Dependencies

- This project also uses the fairseq NLP toolkit, which is included as a submodule in this repository. To prepare the library for use, make sure that it is installed along with its dependencies.

```
cd fairseq
pip install -r requirements.txt
python setup.py build develop
```

- Other project dependencies are placed under fairseq-scripts folder, which can be installed by running

```
cd fairseq-scripts
pip install -r requirements.txt
```

Instructions

Folder Structures

```
.
|-- OpenNMT-py                  The other NLP toolkit we tried early (legacy)
|-- checkpoints                 Trained models and checkpoints
|   |-- errorgen-fairseq-cnn        An error generation model that takes corrected sentences as input,
                                    uncorrected sentences as output
|   |-- lang-8-fairseq              A simple single layer LSTM model for error correction
|   `-- lang-8-fairseq-cnn          A 7-layer convolutional seq2seq model for error correction
|-- corpus                      Raw and prepared corpus
|   |-- errorgen-fairseq            Corpus generated by the error generation model - the result of boost learning.
|   |-- lang-8-en-1.0               Raw Lang-8 corpus
|   |-- lang-8-fairseq              Corpus format required by fairseq
|   `-- lang-8-opennmt              Corpus format required by OpenNMT
|-- data-bin                    Pre-processed and binarized data
|   |-- errorgen-fairseq            Binarized synthetic data and dictionaries
|   |-- lang-8-fairseq              Binarized Lang-8 data and dictionaries
|   `-- wiki103                     Pre-trained WikiText-103 language model and dictionaries
|-- fairseq                     fairseq submodule
|-- fairseq-scripts             fairseq scripts used to implement the model and process proposed by the paper
|-- opennmt                     OpenNMT data and model folder (legacy)
|-- opennmt-scripts             OpenNMT scripts folder (legacy)
`-- test                        Random test text files can be thrown to here
```

Fairseq Custom Scripts

All fairseq scripts have been grouped under fairseq-scripts folder. The whole process is:

1

Preparing data
Pre-process data

4

Interactive mode
Boosting

2

Train the model

3

Testing the model
Evaluate the model

Instructions

Preparing Data

- The first step is to prepare the source and target pairs of training and validation data. Extract original lang-8-en-1.0.zip under corpus folder. Then create another folder lang-8-fairseq under corpus folder to store re-formatted corpus.

- To split the Lang-8 learner data training set, use the following command:

```
python transform-lang8.py -src_dir <dataset-src> -out_dir <corpus-dir>
```

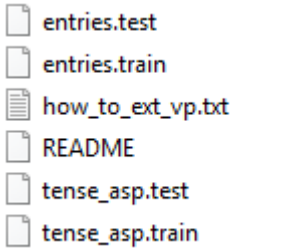
- e.g.

```
python transform-lang8.py -src_dir ../corpus/lang-8-en-1.0 -out_dir ../corpus/lang-8-fairseq
```

Instructions

Preparing Data

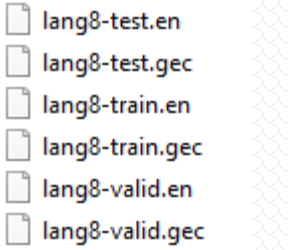
- This will turn original lang-8 data from



entries.train

1	0	1179536	http://lang-8.com/184400/journals/734998	0	Good luck on your new start !
2	0	1179537	http://lang-8.com/184400/journals/734998	1	My teacher is going to move to change his job .
3	0	1179538	http://lang-8.com/184400/journals/734998	2	He is a so nice guy and taught me English very kindly and was willing to accept my getting of
4	1	1179539	http://lang-8.com/184400/journals/734998	3	And he took in my favorite subject like soccer . And he took in my favorite subjects like s
5	1	1179540	http://lang-8.com/184400/journals/734998	4	Actually , who let me know about Lang - 8 was him . Actually , he was the one who let me know
6	0	1179541	http://lang-8.com/184400/journals/734998	5	He is also good at Japanese and studies ' Kanji ' .
7	1	1179542	http://lang-8.com/184400/journals/734998	6	His Kanji 's ability is much better than me . His Kanji ability is much better than mine .
8	1	1179543	http://lang-8.com/184400/journals/734998	7	We 've known each other for only half a year , but his lesson was a lot of fun . We 've kn
9	0	1179544	http://lang-8.com/184400/journals/734998	8	I 'm going to miss him but I really wish him the best of luck with his new life .
10	0	1179545	http://lang-8.com/184400/journals/734998	9	I 'm looking forward to seeing him again through here .

- To a format that fairseq understands



Instructions

Pre-process Data

- Once the data has been extracted from the dataset, use fairseq to prepare the training and validation data and create the vocabulary:

```
preprocess-lang8.bat
```

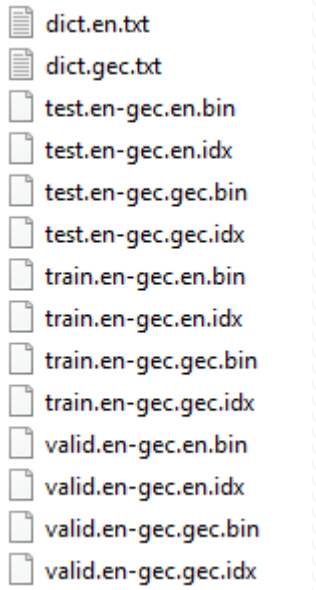
- Or

```
python ..\fairseq\preprocess.py
    --source-lang en
    --target-lang gec
    --trainpref ..\corpus\lang-8-fairseq\lang8-train
    --validpref ..\corpus\lang-8-fairseq\lang8-valid
    --testpref ..\corpus\lang-8-fairseq\lang8-test
    --destdir ..\data-bin\lang-8-fairseq
    --workers 4
```

Instructions

Pre-process Data

- The result is a set of dictionaries and indexes



Instructions

Train the Model

- To train the error-correcting model, run the following command:

```
train-lang8-cnn.bat
```

- Note that this script may need to be adjusted based on the GPU and memory resources available for training.

Instructions

Train the Model

```
(fastai) S:\pytorch-human-performance-gec\fairseq-scripts>python ..\fairseq\train.py    ..\data-bin\lang-8-fairseq
--save-dir ..\checkpoints\lang-8-fairseq-cnn    --arch fconv    --encoder-embed-dim 500    --decoder-embed-dim 500
--decoder-out-embed-dim 500    --encoder-layers "[(1024, 3)] * 7"    --decoder-layers "[(1024, 3)] * 7"    --optimizer
 nag    --momentum 0.99    --lr 0.25    --dropout 0.2    --max-tokens 1000    --max-sentences 10    --clip-norm 0.1
 --fp16
Namespace(arch='fconv', bucket_cap_mb=150, clip_norm=0.1, criterion='cross_entropy', data=['..\\data-bin\\lang-8-fairs
eq'], ddp_backend='c10d', decoder_attention='True', decoder_embed_dim=500, decoder_embed_path=None, decoder_layers='[(
1024, 3)] * 7', decoder_out_embed_dim=500, device_id=0, distributed_backend='nccl', distributed_init_method=None, dist
ributed_port=-1, distributed_rank=0, distributed_world_size=1, dropout=0.2, encoder_embed_dim=500, encoder_embed_path=
None, encoder_layers='[(1024, 3)] * 7', fp16=True, fp16_init_scale=128, keep_interval_updates=-1, left_pad_source='Tru
e', left_pad_target='False', log_format=None, log_interval=1000, lr=[0.25], lr_scheduler='reduce_lr_on_plateau', lr_sh
rink=0.1, max_epoch=0, max_sentences=10, max_sentences_valid=10, max_source_positions=1024, max_target_positions=1024,
 max_tokens=1000, max_update=0, min_loss_scale=0.0001, min_lr=1e-05, momentum=0.99, no_epoch_checkpoints=False, no_pro
gress_bar=False, no_save=False, optimizer='nag', optimizer_overrides='{}', raw_text=False, reset_lr_scheduler=False, r
eset_optimizer=False, restore_file='checkpoint_last.pt', save_dir='..\\checkpoints\\lang-8-fairseq-cnn', save_interval
=1, save_interval_updates=0, seed=1, sentence_avg=False, share_input_output_embed=False, skip_invalid_size_inputs_vali
d_test=False, source_lang=None, target_lang=None, task='translation', train_subset='train', update_freq=[1], upsample_
primary=1, valid_subset='valid', validate_interval=1, weight_decay=0.0)
| [en] dictionary: 137960 types
| [gec] dictionary: 121816 types
| ..\data-bin\lang-8-fairseq train 1114139 examples
| ..\data-bin\lang-8-fairseq valid 5257 examples
| model fconv, criterion CrossEntropyCriterion
| num. model params: 289444792
| training on 1 GPUs
| max tokens per GPU = 1000 and max sentences per GPU = 10
| loaded checkpoint ..\checkpoints\lang-8-fairseq-cnn\checkpoint_last.pt (epoch 3 @ 416794 updates)
| epoch 004:   0%|                                                                                | 0/139271 [00:00<?, ?it/s]D
:\Python\Anaconda3_py36\envs\fastai\lib\site-packages\torch\nn\functional.py:52: UserWarning: size_average and reduce
args will be deprecated, please use reduction='sum' instead.
  warnings.warn(warning.format(ret))
| epoch 004:   0%| | 1/139271 [00:00<38:13:00,  1.01it/s, loss=1.860, ppl=3.63, wps=67, ups=0.8, wpb=80, bsz=8, num_up
| epoch 004:   0%| | 2/139271 [00:01<29:30:02,  1.31it/s, loss=1.358, ppl=2.56, wps=169, ups=1.4, wpb=60, bsz=8, num_u
| epoch 004:   0%| | 3/139271 [00:01<22:57:08,  1.69it/s, loss=1.586, ppl=3.00, wps=276, ups=1.8, wpb=67, bsz=8, num_u
| epoch 004:   0%| | 4/139271 [00:01<18:35:25,  2.08it/s, loss=1.368, ppl=2.58, wps=233, ups=2.2, wpb=58, bsz=8, num_u
| epoch 004:   0%| | 5/139271 [00:01<15:20:57,  2.52it/s, loss=1.260, ppl=2.40, wps=337, ups=2.4, wpb=74, bsz=8, num_u
| epoch 004:   0%| | 6/139271 [00:02<13:18:08,  2.91it/s, loss=1.147, ppl=2.21, wps=343, ups=2.6, wpb=75, bsz=8, num_u
pdates=416800, lr=0.025, gnorm=0.453, clip=83%, oom=0, loss_scale=16.000, wall=2, train_wall=58167]Traceback (most rec
```

Instructions

Testing the Model

- To test the model, run the following command to try to correct a list of sentences:

  ```
  generate-lang8-cnn.bat
  ```

- This command will try to correct all sentences in a file with probabilities and scores in the output. It is a convenient way to peed that the model behaves as expected against lots of test data.

```
S-2052  Baby has come !
T-2052  The baby has arrived !
H-2052  Baby has come ! -0.035532381385564804
P-2052  -0.1289 -0.0451 -0.0015 -0.0010 -0.0011
F-2052  0.13193245232105255
H-2052  A baby has come !       -0.47248873114585876
P-2052  -2.6618 -0.0920 -0.0643 -0.0095 -0.0048 -0.0025
F-2052  0.1371646523475647
H-2052  Baby have come !        -0.8105476498603821
P-2052  -0.1289 -3.9195 -0.0022 -0.0010 -0.0011
F-2052  0.12460633367300034
H-2052  Baby came !     -1.3400437831878662
P-2052  -0.1289 -5.1932 -0.0349 -0.0032
F-2052  0.10877018421888351
```

Instructions

Evaluate the Model

- Evaluate scripts are used to score model using text or pre-processed files in batch.
- Evaluate can be done against lang-8 test dataset using

  `generate-lang8-cnn-rawtext.bat`

- The paper evaluates against JFLEG test dataset, which can be done using

  `generate-jfleg-cnn-rawtext.bat`

- Above scripts can be modified to test other test dataset easily as they use plain text.
- Other scripts such as `generate-lang8.bat` or `generate-lang8-cnn.bat` can only deal with pre-processed data so it is less convenient.

Instructions

Evaluate the Model

- JFLEG test set GLEU score without boost inference

```
| Translated 747 sentences (15126 tokens) in 14.6s (51.02 sentences/s, 1033.01 tokens/s)
| Generate test with beam=5: BLEU4 = 65.88, 84.0/70.5/60.6/52.5 (BP=1.000, ratio=1.011, syslen=14379, reflen=14226)
| Generate test with beam=5: GLEU = 48.89
```

Instructions

Interactive Mode

- While evaluate scripts are good at batch processing, two interactive scripts are provided to see details of generation / correction.
- Below script will run in console mode:

  ```
  interactive-lang8-cnn-console.bat
  ```

- Below script will boot a local server to provide a web GUI and RESTful API interface:

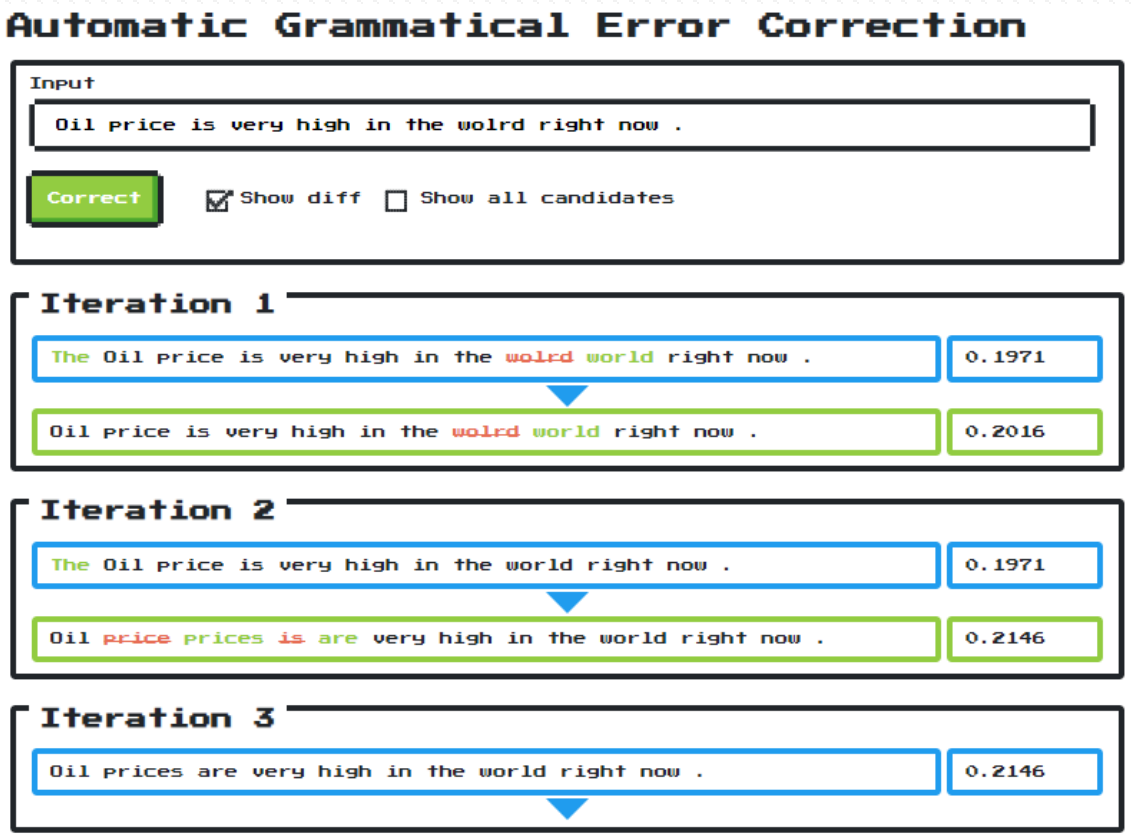
  ```
  interactive-lang8-cnn-web.bat
  ```

Instructions

Interactive Mode

- Interactive mode allows users to enter a sentence in console, or Web GUI, to see how subtle difference in input are corrected.

```
| [en] dictionary: 137960 types
| [gec] dictionary: 121816 types
| loading model(s) from ..\checkpoints\lang-8-fairseq-cnn\che
| dictionary: 267744 types
| Type the input sentence and press return:
In the world oil price very high right now .
O       In the world oil price very high right now .
H       In the world oil price very high right now .    -0.0
P       -0.0632 -0.0120 -0.0011 -0.0325 -0.1453 -0.0646 -0.0
GLEU N/A (no target was provided. use format "source sentence
Fluency Score: 0.1503
Oil price is very high in the wolrd right now .
O       Oil price is very high in the wolrd right now .
H       The Oil price is very high in the world right now .
P       -0.3303 -0.6238 -0.2892 -0.0621 -0.0287 -0.0590 -0.1
GLEU N/A (no target was provided. use format "source sentence
Fluency Score: 0.1885
```



Instructions

Boosting

- To augment training data to provide more examples of common errors, this project builds an error-generating model that can produce additional lower quality sentences for correct sentences. This uses the same training data as the regular model, but reverses the source and target sentences.
- Once the data has been extracted from the dataset, use fairseq to prepare the training and validation data and create the vocabulary:

  ```
  preprocess-errorgen.bat
  ```

- To train the error-correcting model, run the following command:

  ```
  train-errorgen-cnn.bat
  ```

- Note that this script may need to be adjusted based on the GPU and memory resources available for training.

Instructions

Boosting

- Now the error-generating model can be use to generate additional training data. The generating script will only consider sentences longer than four words that are at least 5% less fluent (as measured by the fluency scorer) than the corrected sentences. This ensures that the new sentences are more likely to showcase interesting corrections while avoiding trivial edits. Notice that in this case we use the latest model checkpoint rather than the most generalized, because in this particular case overfitting to the training data is an advantage!

  generate-errorgen-cnn.bat

- The sentences generated in the corpus\errorgen directory can then be used as additional data to train or fine tune the error-correcting model.

Instructions

Boosting

- Samples of new generated dataset

284377	Is you kidding ? ? ?
284378	Do you kidding ? ? ?
284379	are you kidding ? ? ?
284380	Is it kidding ? ? ?
284381	How should I do that .
284382	How should I do ?
284383	How do I do that ?
284384	My classes has just started .
284385	My classes just started .
284386	how are you guys doing ?
284387	How are u guys doing ?
284388	Glee have many charming characters .
284389	Glee has many charming character .
284390	But I never forget .
284391	but I will never forget .
284392	But I never forgot .

284377	Are you kidding ? ? ?
284378	Are you kidding ? ? ?
284379	Are you kidding ? ? ?
284380	Are you kidding ? ? ?
284381	How should I do that ?
284382	How should I do that ?
284383	How should I do that ?
284384	My classes have just started .
284385	My classes have just started .
284386	How are you guys doing ?
284387	How are you guys doing ?
284388	Glee has many charming characters .
284389	Glee has many charming characters .
284390	But I will never forget .
284391	But I will never forget .
284392	But I will never forget .

Demo

- If you failed or don’t have time to setup the dev environment, simply access the Web GUI to test pre-trained model (limited time offer)
  - http://gec.eastus.cloudapp.azure.com/

THANK YOU

Presentation by
Tianfei Chen
Bob Cottrell