CS4248 Natural Language Processing
AY 2022/23 Semester 1

Default final project topic: Grammatical error correction

Introduction

Grammatical error correction (GEC) deals with correcting writing errors in a written text in English. These errors include spelling errors, punctuation errors, grammar errors, word choice errors, etc. In this final project, your goal is to implement a state-of-the-art grammatical error correction system. Evaluation of your GEC system is to be carried out on the CoNLL-2014 and BEA-2019 shared tasks. Below are the papers describing the shared tasks:

https://aclanthology.org/W14-1701.pdf
https://aclanthology.org/W19-4406.pdf

Training data

You can obtain the training data from:
https://www.cl.cam.ac.uk/research/nl/bea2019st/

The training data comes from four sources:

FCE v2.1:
https://www.cl.cam.ac.uk/research/nl/bea2019st/data/fce_v2.1.bea19.tar.gz

Lang-8 Corpus of Learner English:
Request this corpus from:
https://docs.google.com/forms/d/e/1FAIpQLSflRX3h5QYxegivjHN7SJ194OxZ4XN_7Rt0cNpR2YbmNV-7Ag/viewform
An email with a download link will be sent to you.

NUCLE (NUS Corpus of Learner English):
You can obtain this corpus from the LumiNUS Project folder.

W&I+LOCNESS v2.1:
https://www.cl.cam.ac.uk/research/nl/bea2019st/data/wi+locness_v2.1.bea19.tar.gz

Pre-processing

Among the downloaded files, you can find the files that together constitute the training data:

```
fce/m2/fce.train.gold.bea19.m2
fce/m2/fce.dev.gold.bea19.m2
lang8/lang8.train.auto.bea19.m2
nucle/bea2019/nucle.train.gold.bea19.m2
wi+locness/m2/A.train.gold.bea19.m2
```

```
wi+locness/m2/B.train.gold.bea19.m2
wi+locness/m2/C.train.gold.bea19.m2
```

The above files are in the so-called M2 format. Essentially, each `.m2` file consists of blocks separated by a blank line between two blocks. Each block consists of one tokenized sentence, followed by annotations of the errors found in the sentence. Each annotated error includes the start token position, end token position, error type, and the replacement string. You can find detailed description of M2 format in the M2 scorer readme file.

You can use the Python script `m2_to_parallel.py` to convert an M2 file into two files: a source file and a target file, consisting of the source sentences (`.src`) written by English learners and target sentences (`.tgt`) which are the target or corrected sentences. The command to generate the corresponding source and target files from an M2 file is as follows:

```
python m2_to_parallel.py --data filename.m2 --erroneous_only
```

Using the `--erroneous_only` argument, only the erroneous portion of the data (i.e., where a corrected sentence differs from its source sentence) is used to generate the `.src` and `.tgt` files.

Development data

The file that contains the development data is:

```
wi+locness/m2/ABCN.dev.gold.bea19.m2
```

Source and target files are generated similarly from `ABCN.dev.gold.bea19.m2` using `m2_to_parallel.py`, but **without** the argument `--erroneous_only`.

Test data

The test data (including the source sentences and the gold-standard error annotations) of CoNLL-2014 shared task can be downloaded from:

https://www.comp.nus.edu.sg/~nlp/conll14st/conll14st-test-data.tar.gz

The test data of BEA-2019 shared task (only the source sentences but without the gold-standard error annotations) is the following file:

```
wi+locness/test/ABCN.test.bea19.orig
```

Scorers

The M2 scorer is used to score the output in the CoNLL-2014 shared task. The M2 scorer can be downloaded from:

https://www.comp.nus.edu.sg/~nlp/sw/m2scorer.tar.gz

The ERRANT scorer is used to score the output in the BEA-2019 shared task. The ERRANT scorer can be downloaded from:
https://github.com/chrisjbryant/errant

Since only the source sentences in the test set of BEA-2019 shared task are available, in order to score your output, you must upload the corrected sentences output by your system to CodaLab's server:

https://codalab.lisn.upsaclay.fr/competitions/4057

Baseline system

Your goal in this final project is to build a GEC system that achieves high accuracy. You can find the performance of state-of-the-art (SOTA) GEC systems at:

https://nlpprogress.com/english/grammatical_error_correction.html

You are encouraged to come up with a novel approach that performs as well as or even outperforms SOTA GEC systems.

However, to get you started, you may like to build a baseline system following the approach described in the following AACL 2020 paper:

Stronger Baselines for Grammatical Error Correction Using a Pretrained Encoder–Decoder Model
Satoru Katsumata and Mamoru Komachi
https://aclanthology.org/2020.aacl-main.83.pdf

The source code of this paper is available at:
https://github.com/Katsumata420/generic-pretrained-GEC

This paper follows the pre-train then fine-tune paradigm. It makes use of a pre-trained language model (BART) and fine-tunes the pre-trained language model on GEC training data. On the reserved GPU node xgpg0, the proposed training method in this paper takes about 40 minutes to train for 3 epochs. Testing on the test set of CoNLL-2014 or BEA-2019 shared task takes less than 2 minutes. This paper reports $F_{0.5}$ score of 62.6% on the CoNLL-2014 test set and 65.6% on the BEA-2019 test set.

Other final project topics

Other topics are also possible (subject to approval of the lecturer). Some sample final project topics proposed in the past year include:

Aspect-based sentiment analysis
Opinion summarization
Question answering
Rumor detection and analysis
Information extraction