**Documentation:**

Unsupervised Quality Estimation for Neural Machine Translation

# 05-09-2021

****

This documentation includes instructions for running unsupervised quality estimation, as described in the paper [Unsupervised Quality Estimation for Neural Machine Translation (Fomicheva et al., 2020)](https://arxiv.org/abs/2005.10608)

**Requirements:**

* mosesdecoder: <https://github.com/moses-smt/mosesdecoder>
* subword-nmt: <https://github.com/rsennrich/subword-nmt>
* flores: <https://github.com/facebookresearch/flores>

**Download test data**

Download test data from [MLQE dataset repository](https://github.com/facebookresearch/mlqe).

**Set up:**

Given a test set consisting of source sentences and reference translations:

* SRC\_LANG: source language
* TGT\_LANG: target language
* INPUT: input prefix, such that the file $INPUT.$SRC\_LANG contains source sentences and $INPUT.$TGT\_LANG contains the reference sentences
* OUTPUT\_DIR: output path to store results
* MOSES\_DECODER: path to mosesdecoder installation
* BPE\_ROOT: path to subword-nmt installation
* BPE: path to BPE model
* MODEL\_DIR: directory containing the NMT model .pt file as well as the source and target vocabularies.
* TMP: directory for intermediate temporary files
* GPU: if translating with GPU, id of the GPU to use for inference
* DROPOUT\_N: number of stochastic forward passes

$DROPOUT\_N is set to 30 in the experiments reported in the paper. However, we observed that increasing it beyond 10 does not bring substantial improvements.

**Translate the data using standard decoding**

Preprocess the input data:

for LANG in $SRC\_LANG $TGT\_LANG; do

perl $MOSES\_DECODER/scripts/tokenizer/tokenizer.perl -threads 80 -a -l $LANG < $INPUT.$LANG > $TMP/preprocessed.tok.$LANG

python $BPE\_ROOT/apply\_bpe.py -c ${BPE} < $TMP/preprocessed.tok.$LANG > $TMP/preprocessed.tok.bpe.$LANG

done

Binarize the data for faster translation:

fairseq-preprocess --srcdict $MODEL\_DIR/dict.$SRC\_LANG.txt --tgtdict $MODEL\_DIR/dict.$TGT\_LANG.txt --source-lang ${SRC\_LANG} --target-lang ${TGT\_LANG} --testpref $TMP/preprocessed.tok.bpe --destdir $TMP/bin --workers 4

Translate:

CUDA\_VISIBLE\_DEVICES=$GPU fairseq-generate $TMP/bin --path ${MODEL\_DIR}/${SRC\_LANG}-${TGT\_LANG}.pt --beam 5 --source-lang $SRC\_LANG --target-lang $TGT\_LANG --no-progress-bar --unkpen 5 > $TMP/fairseq.out

grep ^H $TMP/fairseq.out | cut -d- -f2- | sort -n | cut -f3- > $TMP/mt.out

Post-process:

sed -r 's/(@@ )| (@@ ?$)//g' < $TMP/mt.out | perl $MOSES\_DECODER/scripts/tokenizer/detokenizer.perl

-l $TGT\_LANG > $OUTPUT\_DIR/mt.out

# Produce uncertainty estimates

**Scoring**

Make temporary files to store the translations repeated N times.

python ${SCRIPTS}/scripts/uncertainty/repeat\_lines.py -I $TMP/preprocessed.tok.bpe.$SRC\_LANG -n $DROPOUT\_N -o $TMP/repeated.$SRC\_LANG

python ${SCRIPTS}/scripts/uncertainty/repeat\_lines.py -i $TMP/mt.out -n $DROPOUT\_N -o $TMP/repeated.$TGT\_LANG

fairseq-preprocess --srcdict ${MODEL\_DIR}/dict.${SRC\_LANG}.txt $TGT\_DIC --source-lang ${SRC\_LANG} --target-lang ${TGT\_LANG} --testpref ${TMP}/repeated --destdir ${TMP}/bin-repeated

Produce model scores for the generated translations using --retain-dropout option to apply dropout at inference time:

CUDA\_VISIBLE\_DEVICES=${GPU} fairseq-generate ${TMP}/bin-repeated --path ${MODEL\_DIR}/${LP}.pt --beam 5 --source-lang $SRC\_LANG --target-lang $TGT\_LANG --no-progress-bar --unkpen 5 --score-reference --retain-dropout --retain-dropout-modules '["TransformerModel","TransformerEncoder","TransformerDecoder","TransformerEncoderLayer"]' TransformerDecoderLayer --seed 46 > $TMP/dropout.scoring.out

grep ^H $TMP/dropout.scoring.out | cut -d- -f2- | sort -n | cut -f2 > $TMP/dropout.scores

Use --retain-dropout-modules to specify the modules. By default, dropout is applied in the same places as for training.

Compute the mean of the resulting output distribution:

python $SCRIPTS/scripts/uncertainty/aggregate\_scores.py -i $TMP/dropout.scores -o $OUTPUT\_DIR/dropout.scores.mean

-n $DROPOUT\_N

**Generation**

Produce multiple translation hypotheses for the same source using --retain-dropout option:

CUDA\_VISIBLE\_DEVICES=${GPU} fairseq-generate ${TMP}/bin-repeated --path ${MODEL\_DIR}/${LP}.pt --beam 5 --source-lang $SRC\_LANG --target-lang $TGT\_LANG --no-progress-bar --retain-dropout --unkpen 5 --retain-dropout-modules TransformerModel TransformerEncoder TransformerDecoder TransformerEncoderLayer TransformerDecoderLayer --seed 46 > $TMP/dropout.generation.out

grep ^H $TMP/dropout.generation.out | cut -d- -f2- | sort -n | cut -f3- > $TMP/dropout.hypotheses\_

sed -r 's/(@@ )| (@@ ?$)//g' < $TMP/dropout.hypotheses\_ | perl $MOSES\_DECODER/scripts/tokenizer/detokenizer.perl -l $TGT\_LANG > $TMP/dropout.hypotheses

Compute similarity between multiple hypotheses corresponding to the same source sentence using Meteor evaluation metric:

python meteor.py -i $TMP/dropout.hypotheses -m <path\_to\_meteor\_installation> -n $DROPOUT\_N -o $OUTPUT\_DIR/dropout.gen.sim.meteor