The JHU-Microsoft Submission for WMT21 Quality Estimation Shared Task

Shuoyang Ding Marcin Junczys-Dowmunt
Matt Post Christian Federmann Philipp Koehn

Task: Word-level QE

Given the **source sentence** and the **MT output**, generate a binary quality tag for each **MT output word** as well as each **gap between every two output words**.

source	Impfungen	beenden	die	Pandemie							
target	Vaccinations	␣	put	␣	ends	␣	to	␣	the	␣	pandemic
labels	OK	OK	OK	BAD	BAD	OK	OK	OK	OK	OK	OK

Motivation

MT → Word-Level QE? Seem intuitive, but not very practical. **Challenges**:

- Unable to use bi-directional context
- Unable to predict gap tags
- Subword vs. Word

Hence, we use **Levenshtein Transformer**, which has an iterative decoding procedure that is very similar to what we need for word-level QE task.

prev. target	Vaccinations	␣	put	␣	ends	␣	to	␣	the	␣	pandemic
deletion	NO		NO		YES		NO		NO		NO
mask ins.		0		2				0		0	
word ins.	Vaccinations		put	an end			to		the		pandemic

- deletion head → word label prediction
- mask insertion head → gap label prediction

Like all non-autoregressive models, **knowledge distillation (KD)** from an autoregressive system is necessary to achieve the optimal performance.
To take advantage of **large-scale pre-training**, we also experimented with initializing the encoder & decoder of the model with those from the **M2M-100-small** multilingual translation model.

Finetuning

There are some differences in terms of the translation task vs. word-level QE task for a Levenshtein Transformer decoder:

- **Input**: Noised Human Translation vs. Machine Translation
- **Output**: Subword-level Labels vs. Word-level Labels

Hence, **finetuning** is still necessary after training for translation. For finetuning, we will need **translation triplets**, i.e. the triplet of source, MT output and (pseudo/human) post-edits.

Synthetic Triplet Finetuning

- Data Construction
  - **src-mt-ref** - **bt-rt-tgt** - **src-rt-ft** - **src-mt1-mt2** - **mvppe**
- Multiview Pseudo Post-Editing (MVPPE): two **views** of a multilingual translation model by passing different input and language code: **translation view** $p_t(\texttt{pe} \mid \texttt{src})$ and **paraphrase view** $p_p(\texttt{pe} \mid \texttt{tgt})$.
  We approximate the post-edited output distribution $p(\texttt{pe} \mid \texttt{src}, \texttt{tgt})$ by **interpolating the two views**, i.e. $p = \lambda_t p_t + \lambda_p p_p$.
- Subword: Levenshtein Transformer is trained to generate subwords, hence the labels used for finetuning need to be on subword-level as well.
  Naïve: run TER on subword-level segmented text – will cause 10 % errors when coverted back to word-level
  Heuristic: run TER on both subword and word-level text, then heuristically (see our paper appendix) interpolate two versions of labels

Human Post-Edit Triplet Finetuning

Same as above, but on human post-edited translation triplets.

Beyond non-autoregressive translation, **Levenshtein Transformer** is also an effective model for **word-level quality estimation**.

Take a picture to view the paper, data and code

Label Imbalance Mitigation

The test data often has far more `OK` labels than `BAD` labels. Like some previous work, we add a weight hyperparameter to the loss functions computed on `BAD` labels to mitigate this imbalance: $\mathcal{L} = \mathcal{L}_{OK} + \mu\mathcal{L}_{BAD}$

Ensemble

To ensemble predictions from k models $p_1(\texttt{OK}), p_2(\texttt{OK}), \ldots, p_k(\texttt{OK})$, we perform a linear combination of the scores for each label:

$$p_E(\texttt{OK}) = \lambda_1 p_1(\texttt{OK}) + \lambda_2 p_2(\texttt{OK}) + \cdots + \lambda_k p_k(\texttt{OK})$$

We use Nelder-Mead method to determine the interpolation weights by optimizing towards target-side MCC on the devset.

Experiments

Configuration	Stage 2	Stage 3	Target MCC
en-de OpenKiwi	N	default	0.337
en-de bilingual best	src-mt1-mt2	$\mu = 1{,}0$	0.500
en-de ensemble	N/A	N/A	0.504
en-zh OpenKiwi	N	default	0.421
en-zh bilingual best	mvppe	$\mu = 1{,}0$	0.459
en-zh ensemble	N/A	N/A	0.466
ro-en OpenKiwi	N	default	0.556
ro-en bilingual best	src-rt-ft	$\mu = 1{,}0$	0.604
ro-en multilingual best	N	$\mu = 1{,}0$	0.612
ro-en ensemble	N/A	N/A	0.633
ru-en OpenKiwi	N	default	0.279
ru-en bilingual best	src-rt-ft	$\mu = 3{,}0$	0.316
ru-en multilingual best	N	$\mu = 3{,}0$	0.339
ru-en ensemble	N/A	N/A	0.349
et-en OpenKiwi	N	default	0.503
et-en bilingual best	N	$\mu = 3{,}0$	0.556
et-en bilingual best (w/ aug)	N	$\mu = 3{,}0$	0.548
et-en multilingual best	N	$\mu = 3{,}0$	0.533
et-en ensemble	N/A	N/A	0.575
ne-en OpenKiwi	N	default	0.664
ne-en bilingual best	N	$\mu = 3{,}0$	0.677
ne-en multilingual best	N	$\mu = 3{,}0$	0.681
ne-en ensemble	N/A	N/A	0.688

Table 1: Target MCC results on `test20` dataset

	Target MCC	F1-OK	F1-BAD
N	0.489	0.955	0.533
src-mt-ref	0.493	0.955	0.537
src-mt1-mt2	**0.500**	0.956	**0.544**
bt-rt-tgt	0.490	0.956	0.534
src-rt-ft	0.494	0.956	0.538
mvppe	**0.500**	**0.960**	0.540

Table 2: Analysis of different data synthesis methods on en-de

Configuration	Target MCC	F1-OK	F1-BAD
ro-en $\mu = 1{,}0$	**0.612**	**0.949**	**0.659**
ro-en $\mu = 3{,}0$	0.577	0.930	0.619
ru-en $\mu = 1{,}0$	0.267	**0.960**	0.284
ru-en $\mu = 3{,}0$	**0.339**	0.943	**0.390**
et-en $\mu = 1{,}0$	0.478	**0.933**	0.511
et-en $\mu = 3{,}0$	**0.512**	0.925	**0.587**
ne-en $\mu = 1{,}0$	0.660	**0.885**	0.774
ne-en $\mu = 3{,}0$	**0.681**	0.855	**0.788**

Table 3: Analysis of label balancing factors on to-English language pairs

Johns Hopkins University Microsoft