OpenKiwi: An Open Source Framework for Quality Estimation

Fábio Kepler
Unbabel

Jonay Trénous
Unbabel

Marcos Treviso[*]
Instituto de Telecomunicações

Miguel Vera
Unbabel

André F. T. Martins
Unbabel

{kepler, sony, miguel.vera, andre.martins}@unbabel.com
marcosvtreviso@gmail.com

Abstract

We introduce OpenKiwi, a PyTorch-based open source framework for translation quality estimation. OpenKiwi supports training and testing of word-level and sentence-level quality estimation systems, implementing the winning systems of the WMT 2015–18 quality estimation campaigns. We benchmark OpenKiwi on two datasets from WMT 2018 (English-German SMT and NMT), yielding state-of-the-art performance on the word-level tasks and near state-of-the-art in the sentence-level tasks.

1 Introduction

Quality estimation (QE) provides the missing link between machine and human translation: its goal is to evaluate a translation system's quality without access to reference translations (Specia et al., 2018b). Among its potential usages are: informing an end user about the reliability of automatically translated content; deciding if a translation is ready for publishing or if it requires human post-editing; and highlighting the words that need to be post-edited.

While there has been tremendous progress in QE in the last years (Martins et al., 2016, 2017; Kim et al., 2017; Wang et al., 2018), the ability of researchers to reproduce state-of-the-art systems has been hampered by the fact that these are either based on complex ensemble systems, complicated architectures, or require not well-documented pretraining and fine-tuning of some components. Existing open-source frameworks such as WCE-LIG (Servan et al., 2015), QuEST++ (Specia et al., 2015), Marmot (Logacheva et al., 2016), or DeepQuest (Ive et al., 2018), while helpful, are currently behind the recent best systems in WMT QE shared tasks. To address the shortcoming above, this paper presents **OpenKiwi**,[1] a new open source framework for QE that implements the best QE systems from WMT 2015–18 shared tasks, making it easy to combine and modify their key components, while experimenting under the same framework.

The main features of OpenKiwi are:

- Implementation of four QE systems: QUETCH (Kreutzer et al., 2015), NuQE (Martins et al., 2016, 2017), Predictor-Estimator (Kim et al., 2017; Wang et al., 2018), and a stacked ensemble with a linear system (Martins et al., 2016, 2017);
- Easy to use API: can be imported as a package in other projects or run from the command line;
- Implementation in Python using PyTorch as the deep learning framework;
- Ability to train new QE models on new data;
- Ability to run pre-trained QE models on data from the WMT 2018 campaign;
- Easy to track and reproduce experiments via `YAML` configuration files and (optionally) MLflow;
- Open-source license (Affero GPL).

This project is hosted at `https://github.com/Unbabel/OpenKiwi`. We welcome and encourage contributions from the research community.[2]

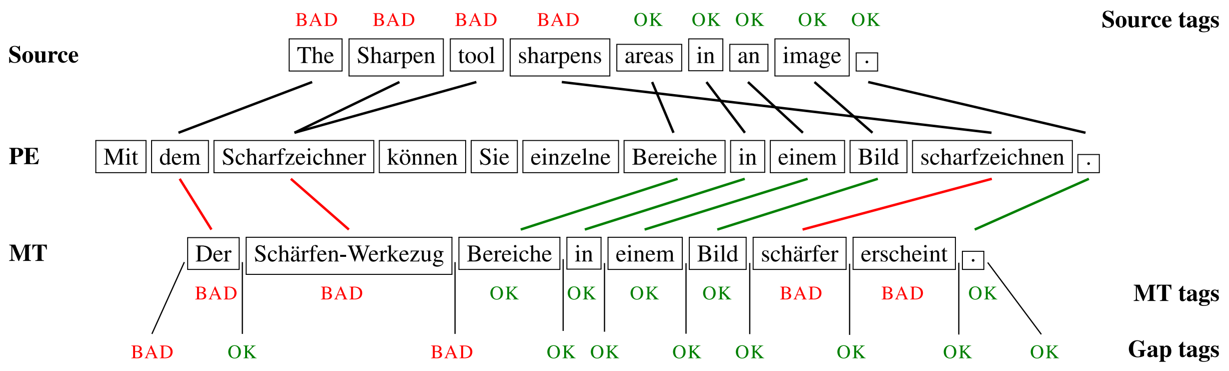
2 Quality Estimation

The goal of **word-level QE** (Figure 1) is to assign quality labels (OK or BAD) to each *machine-translated word*, as well as to *gaps* between words

[*] Work done during an internship at Unbabel in 2018.

[1] `https://unbabel.github.io/OpenKiwi`.

[2] See `https://unbabel.github.io/OpenKiwi/contributing.html` for instructions for contributors.



Source tags
BAD BAD BAD BAD OK OK OK OK OK
Source
The Sharpen tool sharpens areas in an image .
PE
Mit dem Scharfzeichner können Sie einzelne Bereiche in einem Bild scharfzeichnen .
MT
Der Schärfen-Werkezug Bereiche in einem Bild schärfer erscheint .
MT tags
BAD BAD OK OK OK OK BAD BAD OK
Gap tags
BAD OK BAD OK OK OK OK OK OK OK

Figure 1: Example from the WMT 2018 word-level QE training set. Shown are the English source sentence (top), the German machine translated text (bottom), and its manual post-edition (middle). We show also the three types of word-level quality tags: MT (or target) tags account for words that are replaced or deleted, gap tags account for words that need to be inserted, and source tags indicate what are the source words that were omitted or mistranslated. For this example, the HTER sentence-level score (number of edit operations to produce PE from MT normalized by the length of PE) is $8/12 = 66.7\%$, corresponding to 4 insertions, 1 deletion, and 3 replacements out of 12 reference words.

(to account for context that needs to be inserted), and *source words* (to denote words in the original sentence that have been mistranslated or omitted in the target). In the last years, the most accurate systems that have been developed for this task combine linear and neural models (Kreutzer et al., 2015; Martins et al., 2016), use automatic post-editing as an intermediate step (Martins et al., 2017), or develop specialized neural architectures (Kim et al., 2017; Wang et al., 2018).

Sentence-level QE, on the other hand, aims to predict the quality of the whole translated sentence, for example based on the time it takes for a human to post-edit it, or on how many edit operations are required to fix it, in terms of HTER (Human Translation Error Rate) (Specia et al., 2018b). The most successful approaches to sentence-level QE to date are based on conversions from word-level predictions (Martins et al., 2017) or joint training with multi-task learning (Kim et al., 2017; Wang et al., 2018).

3 Implemented Systems

OpenKiwi implements four popular systems that have been proposed in the last years, which we now describe briefly.

QUETCH. The "QUality Estimation from scraTCH" system (Kreutzer et al., 2015) is designed as a multilayer perceptron with one hidden layer, non-linear *tanh* activation functions and a lookup-table layer mapping words to continuous dense vectors. For each position in the MT, a window of fixed size surrounding that position, as well as a windowed representation of aligned words from the source text, are concatenated as model input.[3] The output layer scores OK/BAD probabilities for each word with a softmax activation. The model is trained independently to predict source tags, gap tags, and target tags. QUETCH is a very simple model and does not rely on any kind of external auxiliary data for training, only the shared task datasets.

NuQE. OpenKiwi also implements the NeUral Quality Estimation system proposed by Martins et al. (2016). Its architecture consists of a lookup layer containing embeddings for target words and their source-aligned words, in the same fashion as QUETCH. These embeddings are concatenated and fed into two consecutive sets of two feed-forward layers and a bi-directional GRU layer. The output contains a softmax layer that produces the final OK/BAD decisions. Like QUETCH, training is also carried independently for source tags, gap tags, and target tags. NuQE is also a blackbox system, meaning it is trained with the shared task data only (i.e., no auxiliary parallel or roundtrip data).

Predictor-Estimator. Our implementation follows closely the architecture proposed by Kim et al. (2017), which consists of two modules:

- a *predictor*, which is trained to predict each token of the target sentence given the source and

[3]The alignments are provided by the shared task organizers, which are computed with `fast_align` (Dyer et al., 2013).

the left and right context of the target sentence;

- an *estimator*, which takes features produced by the *predictor* and uses them to classify each word as OK or BAD.

Our predictor uses a bidirectional LSTM to encode the source, and two unidirectional LSTMs processing the target in left-to-right (LSTM-L2R) and right-to-left (LSTM-R2L) order. For each target token t_i, the representations of its left and right context are concatenated and used as query to an attention module before a final softmax layer. It is trained on the large parallel corpora provided as additional data by the WMT shared task organizers. The estimator takes as input a sequence of features: for each target token t_i, the final layer before the softmax (before processing t_i), and the concatenation of the i-th hidden state of LSTM-L2R and LSTM-R2L (after processing t_i). In addition, we train this system with a multi-task architecture that allows us to predict sentence-level HTER scores. Overall, this system is capable to predict sentence-level scores and all word-level labels (for MT words, gaps, and source words)—the source word labels are produced by training a predictor in the reverse direction.

Stacked Ensemble. The systems above can be ensembled by using a stacked architecture with a feature-based linear system, as described by Martins et al. (2017). The features are the ones described there, including lexical and part-of-speech tags from words, their contexts, and their aligned words and contexts, as well as syntactic features and features provided by a language model (as provided by the shared task organizers). This system is only used to produce word-level labels for MT words.

4 Design, Implementation and Usage

OpenKiwi is designed and implemented in a way that allows new models to be easily added and run, without requiring much concern about input data processing and output generation and evaluation. That means the focus can be almost exclusively put in adding or changing a `torch.nn.Module` based class. If new flags or options are required, all that is needed is to add them to the CLI parsing module.

Design. As a general architecture example, the training pipeline follows these steps:

- Each input data, like source text and MT text, is defined as a `Field`, which holds information about how data should be tokenized, how the inner vocabulary is built, how the mapping to IDs is done, and how a list of samples is padded into a tensor;
- A `Dataset` holds a set of input and output fields, and builds minibatches of samples, each containing their respective input and output data;
- A training loop iterates over epochs and steps, calling the model with each minibatch, computing the loss, backpropagating, evaluating on the validation set, and saving snapshots as requested;
- By default, the best model is kept and predictions on the validation set are saved as probabilities.

The flow rarely needs to be changed for the QE task, so all that is needed for quick experimentation is changing configuration parameters (check the **Usage** part below) or the model class.

Implementation. OpenKiwi supports Python 3.5 and later. Since reproducibility is important, it uses Poetry[4] for deterministic dependency management. To decrease the risk of introducing breaking changes with new code, a set of tests are also implemented and currently provide a code coverage close to 80%.

OpenKiwi offers support for tracking experiments with MLflow,[5] which allows comparing different runs and searching for specific metrics and parameters.

Usage. Training an OpenKiwi model is as simple as running the following command:

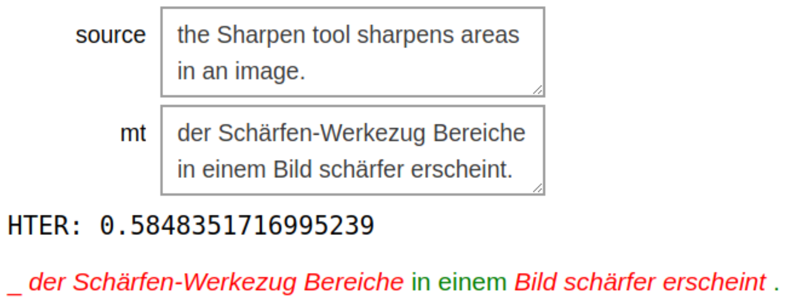
```
$ python kiwi train --config
 ↪ config.yml
```

where `config.yml` is a configuration file with training and model options.

OpenKiwi can also be installed as a Python package by running `pip install openkiwi`. In this case, the above command can be switched by

```
$ kiwi train --config config.yml
```

[4]https://poetry.eustace.io/
[5]https://mlflow.org/



```
source  the Sharpen tool sharpens areas
        in an image.

mt      der Schärfen-Werkezug Bereiche
        in einem Bild schärfer erscheint.

HTER: 0.5848351716995239

_ der Schärfen-Werkezug Bereiche in einem Bild schärfer erscheint .
```

Figure 2: Interactive visualization of the system output. Words tagged as BAD as shown in *red*, and BAD gaps are denoted as red underscores ("_"). The Jupyter Notebook producing this output is available at `https://github.com/Unbabel/OpenKiwi/blob/master/demo/KiwiViz.ipynb`.

If used inside another Python project, OpenKiwi can be easily used like the following:

```
import kiwi

config = 'config.yml'
run_info = kiwi.train(config)
```

After training, predicting on new data can be performed by simply calling

```
model = kiwi.load_model(
    run_info.model_path
)
source = [
    'the Sharpen tool sharpens '
    'areas in an image .'
]
target = [
    'der Schärfen-Werkezug '
    'Bereiche in einem Bild '
    'schärfer erscheint .'
]
examples = [{
    'source': source,
    'target': target
}]
out = model.predict(examples)
```

Figure 2 shows an example of QE predictions using the framework.

5 Benchmark Experiments

Datasets. To benchmark OpenKiwi, we use the following datasets from the WMT 2018 quality estimation shared task, all English-German (En-De):

- Two quality estimation datasets of sentence triplets, each consisting of a source sentence (SRC), its machine translation (MT) and a human post-edition (PE) of the machine translation: a larger dataset of 26,273 training and 1,000 development triplets, where the MT is generated by a phrase-based statistical machine translation (SMT); and a smaller dataset of 13,442 training and 1,000 development triplets, where the MT is generated by a neural machine translation system (NMT). The data also contains word-level quality labels and sentence-level scores that are obtained from the post-editions using TERCOM (Snover et al., 2006).
- A corpus of 526,368 artificially generated sentence triplets, obtained by first cross-entropy filtering a much larger monolingual corpus for in-domain sentences, then using round-trip translation and a final stratified sampling step.
- A parallel dataset of 3,396,364 in-domain sentences used for pre-training of the predictor-estimator model.

Systems. In addition to the models that are part of OpenKiwi, in the experiments below, we also use Automatic Post-Editing (APE) adapted for QE (APE-QE). APE-QE has been used by Martins et al. (2017) as an intermediate step for quality estimation, where an APE system is trained on the human post-edits and its outputs are used as pseudo-post-editions to generate word-level quality labels and sentence-level scores in the same way that the original labels were created. Since OpenKiwi's focus is not on implementing a sequence-to-sequence model, we used an external software, OpenNMT-py (Klein et al., 2017), to train two separate translation models:

- SRC $\rightarrow$ PE: trained first on the in-domain corpus provided, then fine-tuned on the shared task data.

Model	En-De SMT					En-De NMT				
	MT	gaps	source	r	ρ	MT	gaps	source	r	ρ
QUETCH	39.90	17.10	36.10	48.32	51.31	29.18	13.26	28.91	42.84	49.59
NuQE	50.04	35.53	42.08	59.62	60.89	32.49	15.01	30.19	43.41	50.87
PRED-EST	57.29	43.68	33.02	70.95	74.49	39.25	21.54	29.52	50.18	55.66
APE-QE	55.12	47.04	51.11	58.01	60.58	37.60	21.78	**34.46**	35.23	38.88
ENSEMBLED	61.33	**53.05**	**51.11**	**72.89**	**76.37**	43.04	**24.74**	**34.46**	**52.34**	**56.98**
STACKED	**62.40**	–	–	–	–	**43.88**	–	–	–	–

Table 1: Benchmarking of the different models implemented in OpenKiwi on the WMT 2018 development set, along with an ensembled system (ENSEMBLED) that averages the predictions of the NuQE, APE-QE, and PRED-EST systems, as well as a stacked architecture (STACKED) which stacks their predictions into a linear feature-based model, as described by Martins et al. (2017). For each system, we report the five official scores used in WMT 2018: word-level F_1^{mult} for MT, gaps, and source tokens, and sentence-level Pearson's r and Spearman's ρ rank correlations.

Model	En-De SMT					En-De NMT				
	MT	gaps	source	r	ρ	MT	gaps	source	r	ρ
deepQUEST	42.98	28.24	<u>33.97</u>	48.72	50.97	30.31	<u>11.93</u>	<u>28.59</u>	38.08	48.00
UNQE	–	–	–	70.00	72.44	–	–	–	**<u>51.29</u>**	**<u>60.52</u>**
QE Brain	<u>62.46</u>	<u>49.99</u>	–	**<u>73.97</u>**	**<u>75.43</u>**	<u>43.61</u>	–	–	50.12	60.49
OpenKiwi	**62.70**	**52.14**	**48.88**	71.08	72.70	**44.77**	**22.89**	**36.53**	46.72	58.51

Table 2: Final results on the WMT 2018 test set. The first three systems are the official WMT18-QE winners (underlined): deepQUEST is the open source system developed by Ive et al. (2018), UNQE is the unpublished system from Jiangxi Normal University, described by Specia et al. (2018a), and QE Brain is the system from Alibaba described by Wang et al. (2018). Reported numbers for the OpenKiwi system correspond to best models in the development set: the STACKED model for prediction of MT tags, and the ENSEMBLED model for the rest.

- MT $\rightarrow$ PE: trained on the concatenation of the corpus of artificially created sentence triplets and the shared task data oversampled by a factor of 20.

These predictions are then combined in the ensemble and stacked systems as explained below.

Experiments. We show benchmark numbers on the two English-German WMT 2018 datasets. In Table 1, we compare different configurations of OpenKiwi on the development datasets. For the single systems, we can see that the predictor-estimator has the best performance, except for predicting the source and the gap word-level tags, where APE-QE is superior. Overall, ensembled versions of these systems perform the best, with a stacked architecture being very effective for predicting word-level MT labels, confirming the findings of Martins et al. (2017).

Finally, in Table 2, we report numbers on the official test set. We compare OpenKiwi against the best systems in WMT 2018 (Specia et al., 2018a) and another existing open-source tool, deepQuest (Ive et al., 2018). Overall, OpenKiwi outperforms deepQuest for all word-level and sentence-level tasks, and attains the best results for all the word-level tasks.

6 Conclusions

We presented OpenKiwi, a new open source framework for QE. OpenKiwi is implemented in PyTorch and supports training of word-level and sentence-level QE systems on new data. It outperforms other open source toolkits on both word-level and sentence-level, and yields new state-of-the-art word-level QE results.

Since its release, OpenKiwi was adopted as the baseline system for the WMT 2019 QE shared task[6], Moreover, all the winning systems of the word-, sentence- and document-level tasks of the

[6]More specifically, the NuQE model: http://www.statmt.org/wmt19/qe-task.html

WMT 2019 QE shared task[7] (Kepler et al., 2019) used OpenKiwi as their building foundation.

Acknowledgments

The authors would like to thank Eduardo Fierro, Thomas Reynaud, and the Unbabel AI and Engineering teams for their invaluable contributions to OpenKiwi.

They would also like to thank the support provided by the European Union in the context of the PT2020 projects 027767 and 038510.

References

Chris Dyer, Victor Chahuneau, and Noah A Smith. 2013. A simple, fast, and effective reparameterization of IBM model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648.

Julia Ive, Frédéric Blain, and Lucia Specia. 2018. DeepQuest: a framework for neural-based Quality Estimation. In *International Conference on Computational Linguistics (COLING)*.

Fábio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, António Góis, M. Amin Farajian, António V. Lopes, and André F. T. Martins. 2019. Unbabel's Participation in the WMT19 Translation Quality Estimation Shared Task. In *Conference on Machine Translation (WMT)*.

Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. 2017. Predictor-Estimator using Multilevel Task Learning with Stack Propagation for Neural Quality Estimation. In *Conference on Machine Translation (WMT)*.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander M Rush. 2017. Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Julia Kreutzer, Shigehiko Schamoni, and Stefan Riezler. 2015. QUality Estimation from ScraTCH (QUETCH): Deep Learning for Word-level Translation Quality Estimation. In *Workshop on Machine Translation (WMT)*.

Varvara Logacheva, Chris Hokamp, and Lucia Specia. 2016. Marmot: A toolkit for translation quality estimation at the word level. In *LREC*.

André F. T. Martins, Ramon Astudillo, Chris Hokamp, and Fábio Kepler. 2016. Unbabel's Participation in the WMT16 Word-Level Translation Quality Estimation Shared Task. In *Conference on Machine Translation (WMT)*.

André F. T. Martins, Marcin Junczys-Dowmunt, Fabio Kepler, Ramon Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017. Pushing the limits of translation quality estimation. *Transactions of the Association for Computational Linguistics (to appear)*.

Christophe Servan, Ngoc-Tien Le, Ngoc Quang Luong, Benjamin Lecouteux, and Laurent Besacier. 2015. An Open Source Toolkit for Word-level Confidence Estimation in Machine Translation. In *The 12th International Workshop on Spoken Language Translation (IWSLT'15)*, Da Nang, Vietnam.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200.

Lucia Specia, Frédéric Blain, Varvara Logacheva, Ramón Astudillo, and André F. T. Martins. 2018a. Findings of the wmt 2018 shared task on quality estimation. In *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, pages 702–722, Belgium, Brussels. Association for Computational Linguistics.

Lucia Specia, Gustavo Paetzold, and Carolina Scarton. 2015. Multi-level translation quality prediction with quest++. In *ACL-IJCNLP 2015 System Demonstrations*, pages 115–120, Beijing, China.

Lucia Specia, Carolina Scarton, and Gustavo Henrique Paetzold. 2018b. Quality estimation for machine translation. *Synthesis Lectures on Human Language Technologies*, 11(1):1–162.

Jiayi Wang, Kai Fan, Bo Li, Fengming Zhou, Boxing Chen, Yangbin Shi, and Luo Si. 2018. Alibaba Submission for WMT18 Quality Estimation Task. In *Conference on Machine Translation (WMT)*.

[7] `http://www.statmt.org/wmt19/qe-results.html`