BUILDING UNIVERSAL UNDERSTANDING

Quality Estimation in Practice:
from Implementation to State-of-the-Art

Fabio Kepler
Unbabel AI

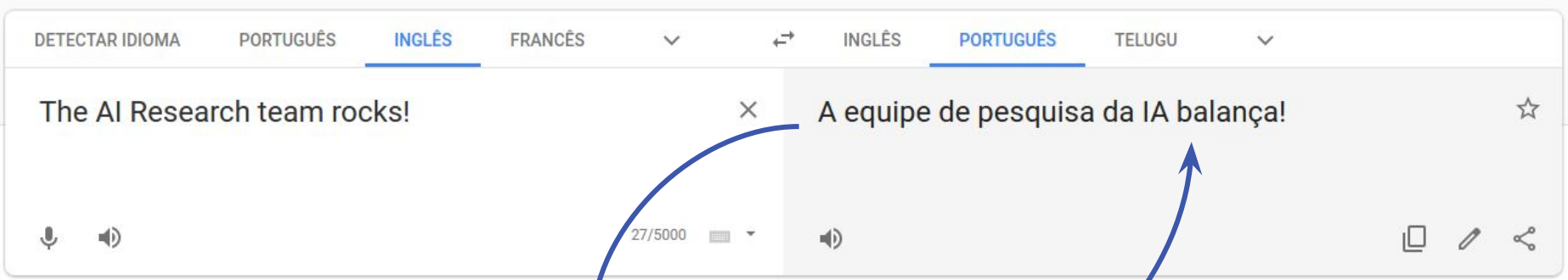
August 2019

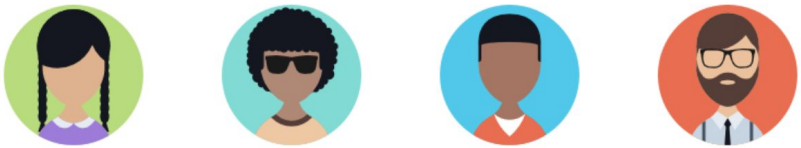
Roadmap

- **In Practice**
  - Unbabel's use case (pipeline)
- **Implementation**
  - WMT-QE winning systems from 2016 to 2018
  - Impact in production
- **State-of-the-Art**
  - WMT-QE 2019 participation
    - New models, much better numbers
    - Same observation as general consensus in the community: large pre-trained models helped a lot
    - Smart ensembling gave a boost

Why Quality Estimation?

Is Machine Translation Solved?





*"The AI Research team **wobbles**!"*

We still need humans in the loop

MT Quality

What could we do if we knew the **quality of a translation?**

- If it is good, we can skip the human **(+speed, -$)**
- Otherwise, we can at least highlight the parts that are wrong
- Ensures final quality in all cases **(higher MQM)**

Problem

- NMT models are not well calibrated
  - Therefore no reliable confidence score is provided with translations
- In fact, they are usually over-confident
  - Even when they hallucinate

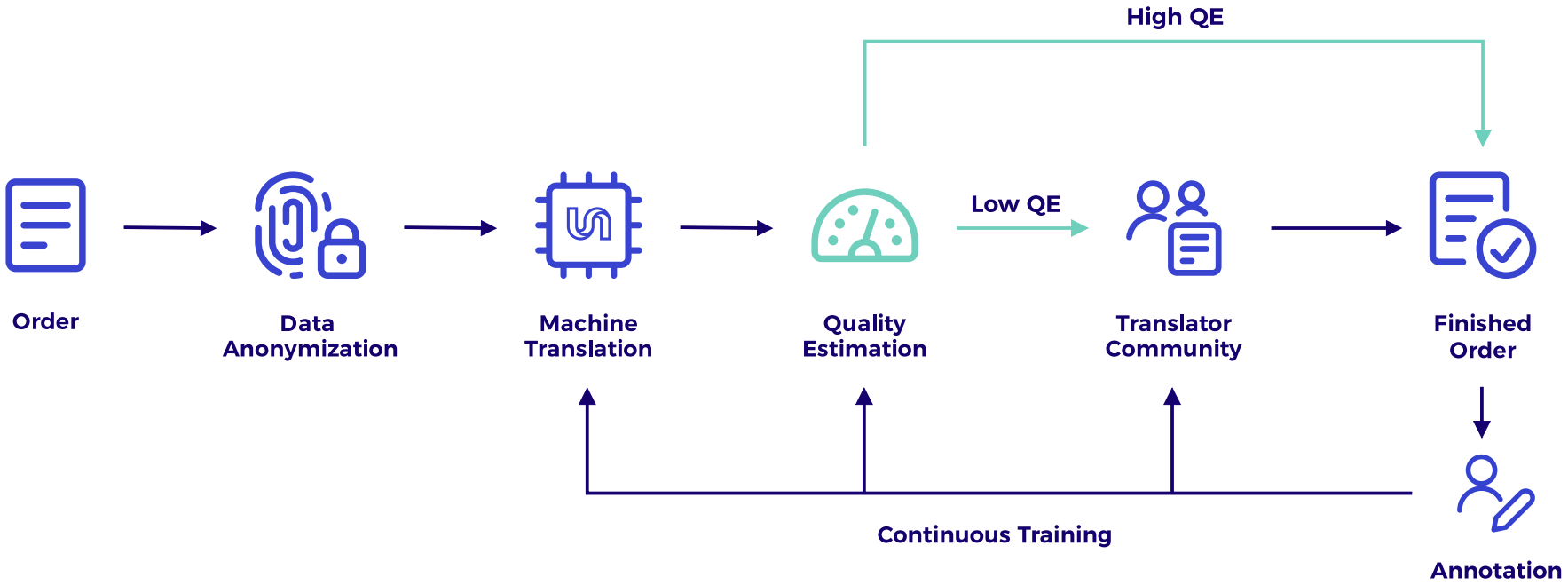
MT Quality Estimation

- Use a different system to estimate how good a translation is
- With no access to a reference translation
  - In other words, “not constrained by a (single) reference” (off- but hot-topic)
- Levels:
  - Word
  - Sentence
  - Document

In Practice

(use case)

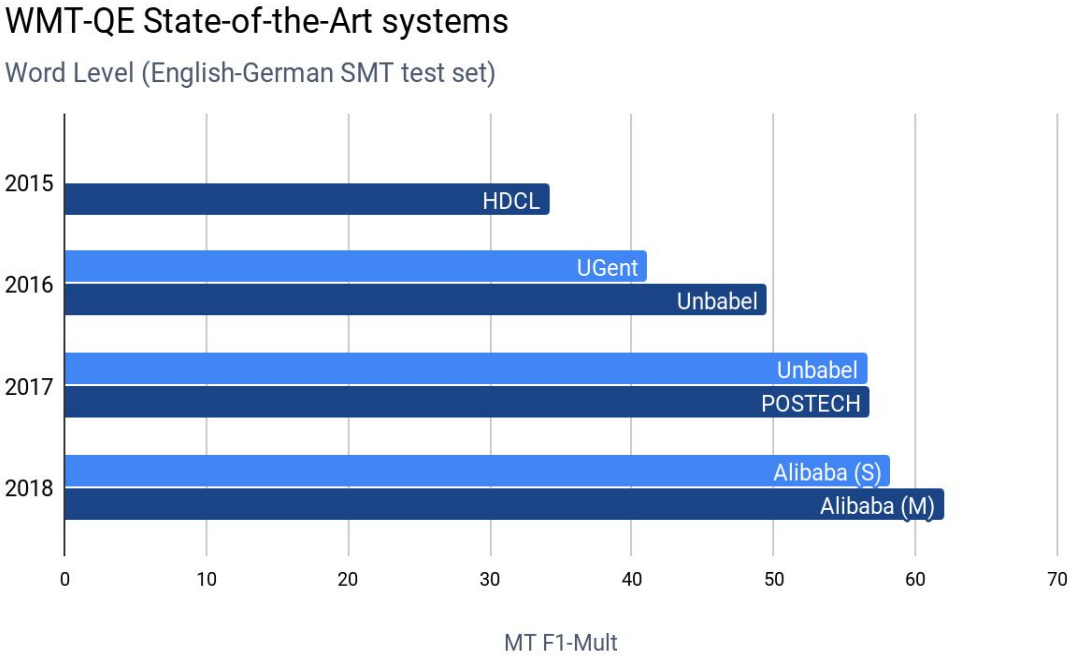
Unbabel's Pipeline



WMT QE Shared Task

WMT QE Competitions (2015-18)

- English-German as the historical benchmark
- Based on a SMT model
- Big improvements each year



Previous Winning Models

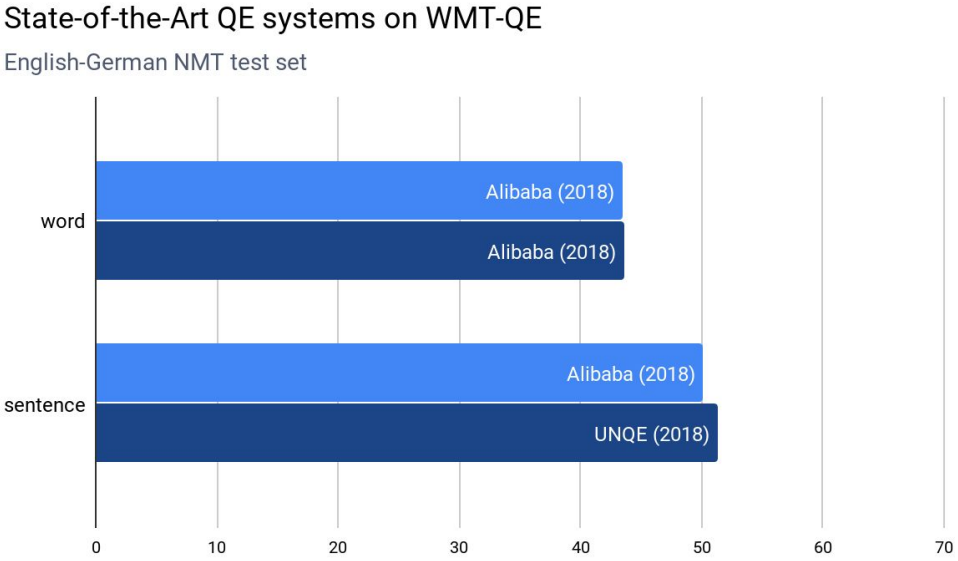
- 2016-2017: Unbabel's submissions were based on linear stackings of different models
  - A deep neural model with two "bi-directional" GRUs (later called *NuQE*)
  - A feature-rich linear model
  - Plus predictions from the winning system of the 2017 APE shared task (Marcin)

Previous Winning Models

- 2017: POSTECH (Korea) devised a neat 2-stages neural model
  - The **Predictor-Estimator**
  - It allowed pretraining with large parallel data in a “translation language-model” fashion (more than a year before BERT)
- 2018: Alibaba improved on it with a few tweaks on both submodels
  - E.g., a Predictor using Transformers

WMT QE Competitions (2018)

- Then data got neural
  - New NMT-based translations
  - Scores ballpark dropped by 10 points
- Alibaba's model still best
- UNQE's submission also uses a 2-stages architecture:
  - A "bi-directional" RNN encoder-decoder with attention
  - Then an HTER prediction block
- Also pretrained with large parallel data



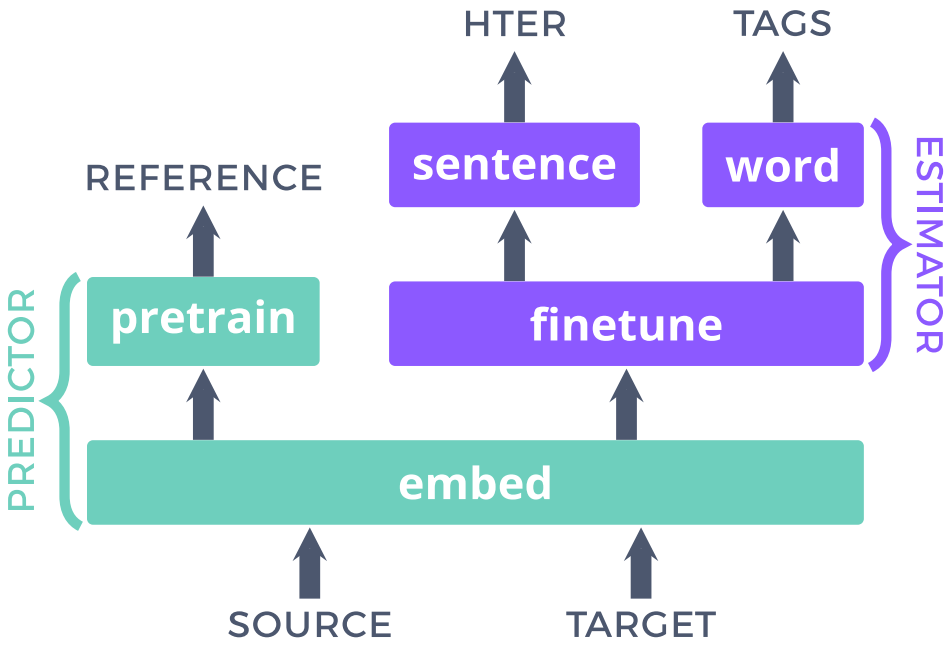
The Predictor-Estimator

Predictor-Estimator

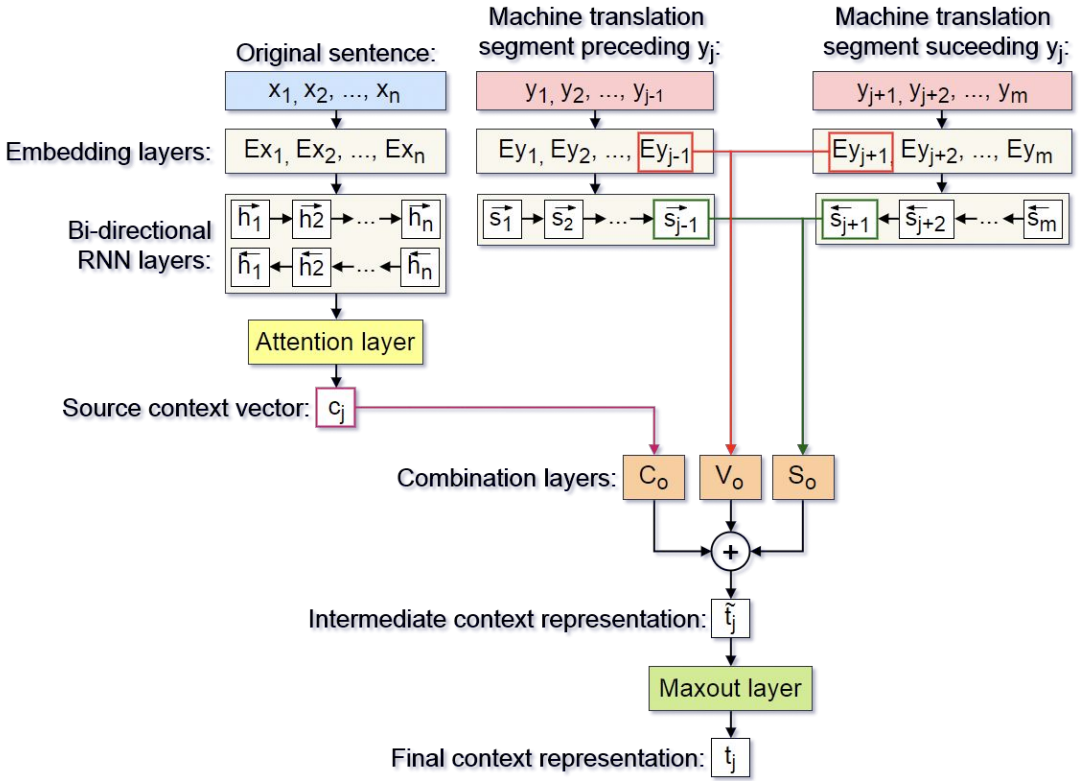
Hyun Kim, Jong-Hyeok Lee, and Seung-Hoon Na. "*Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation*." Proceedings of the Second Conference on Machine Translation. 2017.

Predictor-Estimator

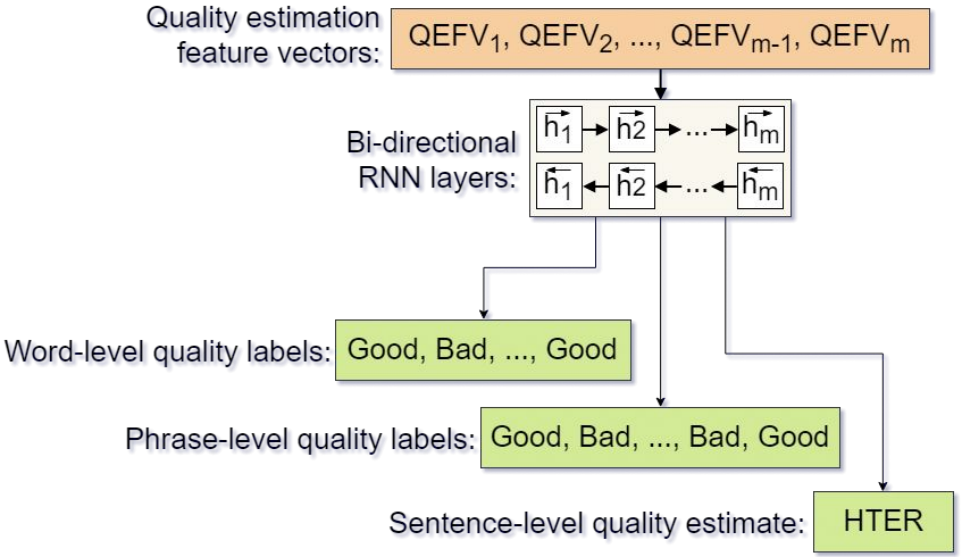
- The **predictor** module is **pretrained** on parallel corpora
  - Predicting every token on the TARGET side given its left and right context produced by two uni-directional LSTMs
- The **estimator** module is trained in a **finetuning** step
  - Estimates word- and sentence-level scores from the input **embedded** by the **predictor** module



Predictor


Original sentence:
Machine translation segment preceding y_j:
Machine translation segment suceeding y_j:
$x_1, x_2, ..., x_n$
$y_1, y_2, ..., y_{j-1}$
$y_{j+1}, y_{j+2}, ..., y_m$
Embedding layers:
$Ex_1, Ex_2, ..., Ex_n$
$Ey_1, Ey_2, ..., Ey_{j-1}$
$Ey_{j+1}, Ey_{j+2}, ..., Ey_m$
Bi-directional RNN layers:
Attention layer
Source context vector: c_j
Combination layers: C_o V_o S_o
+
Intermediate context representation: $\tilde{t}_j$
Maxout layer
Final context representation: t_j

Estimator

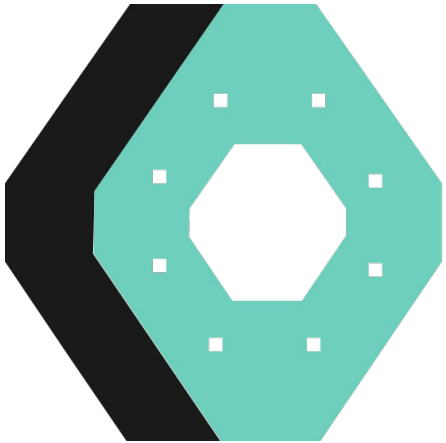

Quality estimation feature vectors:
QEFV1, QEFV2, ..., QEFVm-1, QEFVm
Bi-directional RNN layers:
h1
h2
...
hm
h1
h2
...
hm
Word-level quality labels:
Good, Bad, ..., Good
Phrase-level quality labels:
Good, Bad, ..., Bad, Good
Sentence-level quality estimate:
HTER

Predictor-Estimator

- Unfortunately, no reference implementation made available
- Neither from Alibaba’s variant

Implementation

Introducing



OpenKiwi

By Unbabel

OpenKiwi toolkit

Goals

- Facilitate the **research** 🔄 **production** feedback loop
- Serve as foundation for **future research**
- Make Unbabel an **Industry Leader** in QE

In a Nutshell

- Implementation of **several WMT winning systems** in one single framework
- **State-of-the-art** results
- Easy-to-use API for **training** and **inference**
- Extensive **documentation**
- **Modular design** for easy extensibility
    - Bulletproofed in this year's shared task submission

OpenKiwi production-easy

- Tested (**76% coverage**)
- Documented

predict(*examples*, *batch_size=1*) [source]

Create Predictions for a list of examples.

Parameters:
- **examples** – A dict mapping field names to the list of raw examples (strings).
- **batch_size** – Batch Size to use. Default 1.

Returns: A dict mapping prediction levels (word, sentence ..) to the model predictions for each example.

Raises: `Exception` – If an example has an empty string as *source* or *target* field.

Example

```
>>> import kiwi
>>> predictor = kiwi.load_model('tests/toy-data/models/nuqe.torch')
>>> src = ['a b c', 'd e f g']
>>> tgt = ['q w e r', 't y']
>>> align = ['0-0 1-1 1-2', '1-1 3-0']
>>> examples = [kiwi.constants.SOURCE: src,
                kiwi.constants.TARGET: tgt,
                kiwi.constants.ALIGNMENTS: align]
>>> predictor.predict(examples)
{'tags': [[0.4760947525501251,
   0.47569847106933594,
   0.4948718547821045,
   0.5305878520011902],
  [0.5105430483818054, 0.5252899527549744]]}
```

OpenKiwi production-easy

- Simple usage as a Python package
- Easy training of models with any data
  - Like Unbabel's internal
- Easy usage of any pre-trained model

Run via API:

```
import kiwi

nuqe_config = 'experiments/train_nuqe.yaml'
kiwi.train(nuqe_config)
```

Or via CLI:

```
kiwi train --config experiments/train_nuqe.yaml
```

```
>>> import kiwi
>>> predictor = kiwi.load_model('tests/toy-data/models/nuqe.torch')
>>> src = ['a b c', 'd e f g']
>>> tgt = ['q w e r', 't y']
>>> align = ['0-0 1-1 1-2', '1-1 3-0']
>>> examples = [kiwi.constants.SOURCE: src,
                kiwi.constants.TARGET: tgt,
                kiwi.constants.ALIGNMENTS: align]
>>> predictor.predict(examples)
{'tags': [[0.4760947525501251,
   0.47569847106933594,
   0.4948718547821045,
   0.5305878520011902],
  [0.5105430483818054, 0.5252899527549744]]}
```

OpenKiwi production-easy

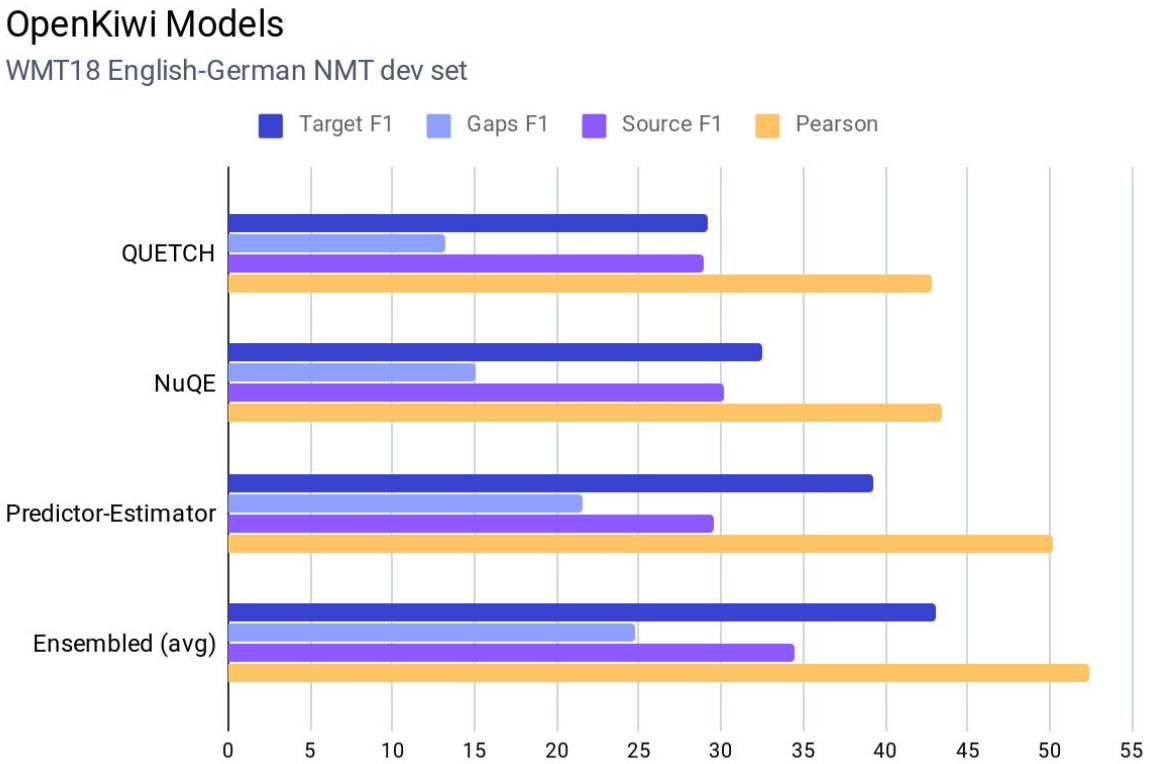
Or train and predict in one go

```
$ pip install openkiwi

import kiwi

config = 'config.yml'
run_info = kiwi.train(config)
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
}]
predictions = model.predict(examples)
```

OpenKiwi research-easy

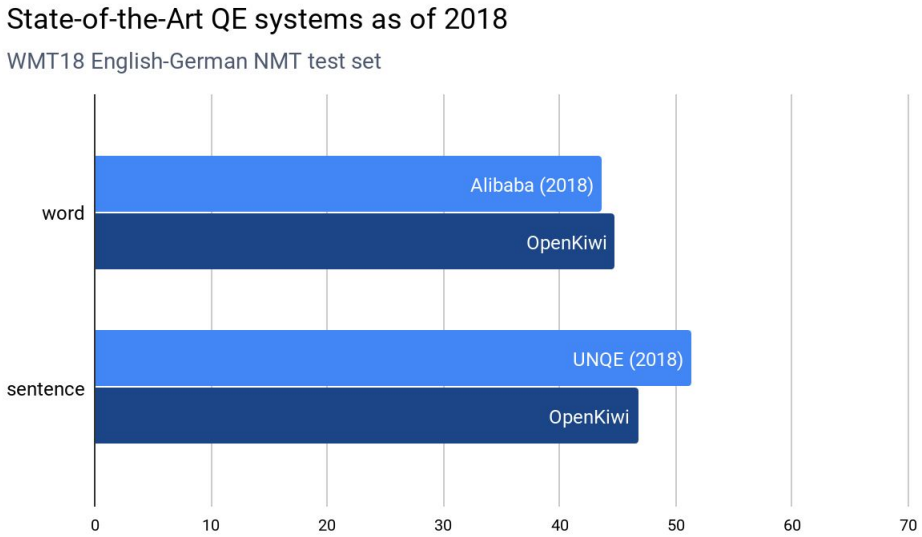
- Quick experimentation
- Modular Design ⟶ Easily Extensible
- Automatic tracking of results through MLFlow

Implemented Models



OpenKiwi scoreboard

- Better at word-level
- Trailing a bit on sentence-level
- Using order of magnitude **less compute**
  - Predictor-Estimator is only pre-trained on only 3M in-domain data



Open Source

- Contributions are welcome!

- **https://github.com/Unbabel/OpenKiwi**

Simple example

Source

This is a simple sentence .

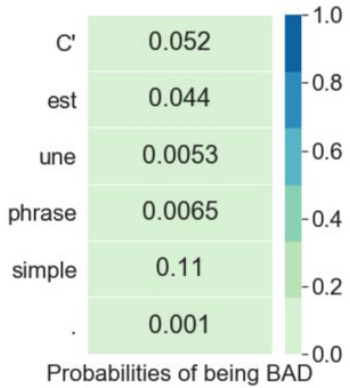
MT

C' est une phrase simple .

```
['OK', 'OK', 'OK', 'OK', 'OK', 'OK']

MACHINE_TRANSLATION: C' est une phrase simple .
```

```
import kiwi

predest = kiwi.load_model('qe_model.torch')

source_english = ['This is a simple sentence .']
mt_french = ["C' est une phrase simple ."]

probs = predest.predict({kiwi.constants.SOURCE: source_english,
                         kiwi.constants.TARGET: mt_french})
print(probs)
```

```
{'tags': [[0.051981423050165176,
   0.043979279696941376,
   0.005278203636407852,
   0.006495318375527859,
   0.109250508248806,
   0.0010430893162265421]],
 'sentence_scores': [0.028975505381822586]}
```



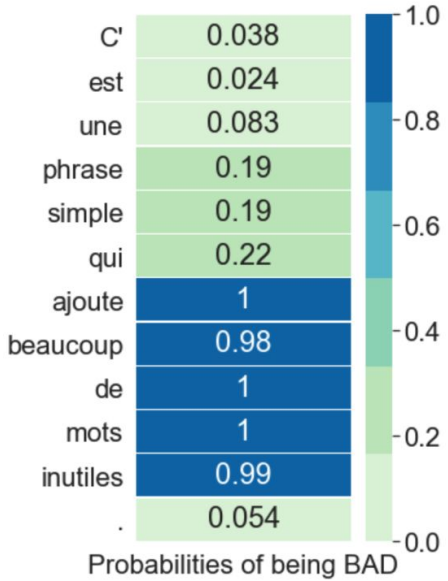
BAD Example

Source

This is a simple sentence .

MT

C' est une phrase simple qui ajoute beaucoup de mots inutiles .




```
['OK', 'OK', 'OK', 'OK', 'OK', 'OK', 'BAD', 'BAD', 'BAD', 'BAD', 'BAD', 'OK']

MACHINE_TRANSLATION: C' est une phrase simple qui ajoute beaucoup de mots inutiles .

'sentence_scores': [0.5956864953041077]
```

Demonstration

Impact in Production Unbabel

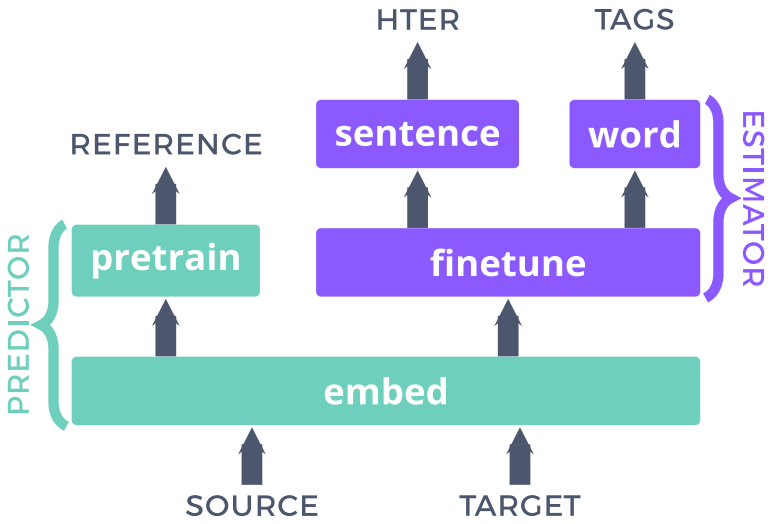
- Skipping **5%** of the jobs
- Average MQM of skipped jobs close to **80**
- For **28+** Language Pairs

State-of-the-Art

WMT19 QE Shared Task

Surfing the wave

- Given the great modularity provided in **OpenKiwi**
- We exploited the similarity of the current wave of **Muppet Models®** to the **Predictor**-**Estimator** 2-stages approach



Surfing the wave

- We created several variants replacing the **predictor** by various pretrained models:
  - **PredEst-RNN**: the original bi-LSTM Predictor-Estimator as implemented in OpenKiwi
  - **PredEst-Trans:** a Transformer-based version like implemented by Alibaba (2018)
  - **PredEst-BERT**: the pretrained multilingual BERT as the Predictor
  - **PredEst-XLM**: the pretrained XLM as the Predictor

Other models

Linear

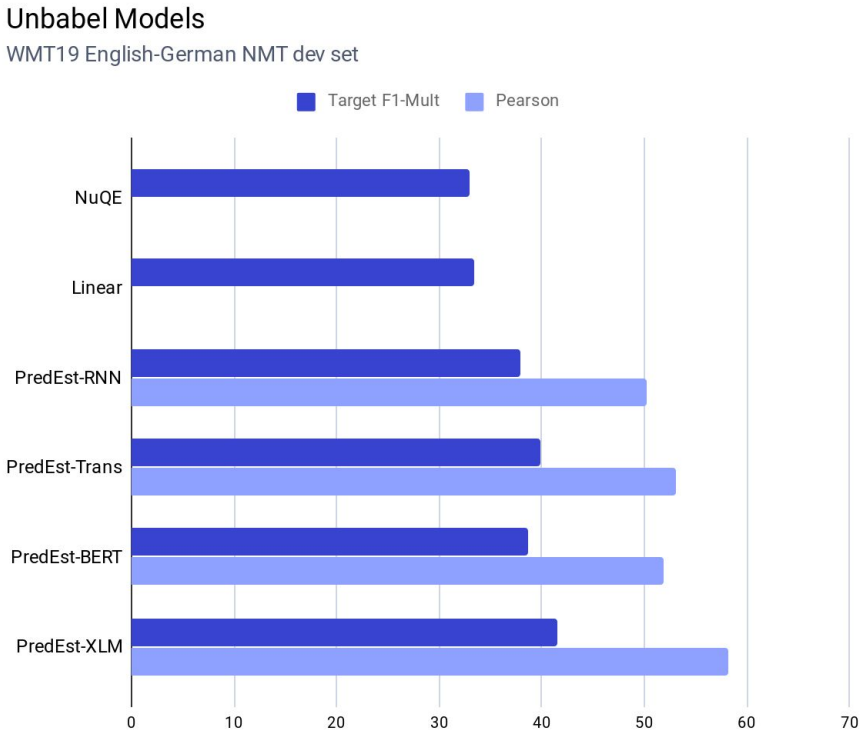
- First-order sequential model incorporating rich features (ngrams, POS tags, dependencies)
- As open-sourced in OpenKiwi

NuQE

- Same deep neural model as open-sourced in OpenKiwi
- Just a few hyper-params tweaked
- No pre-training

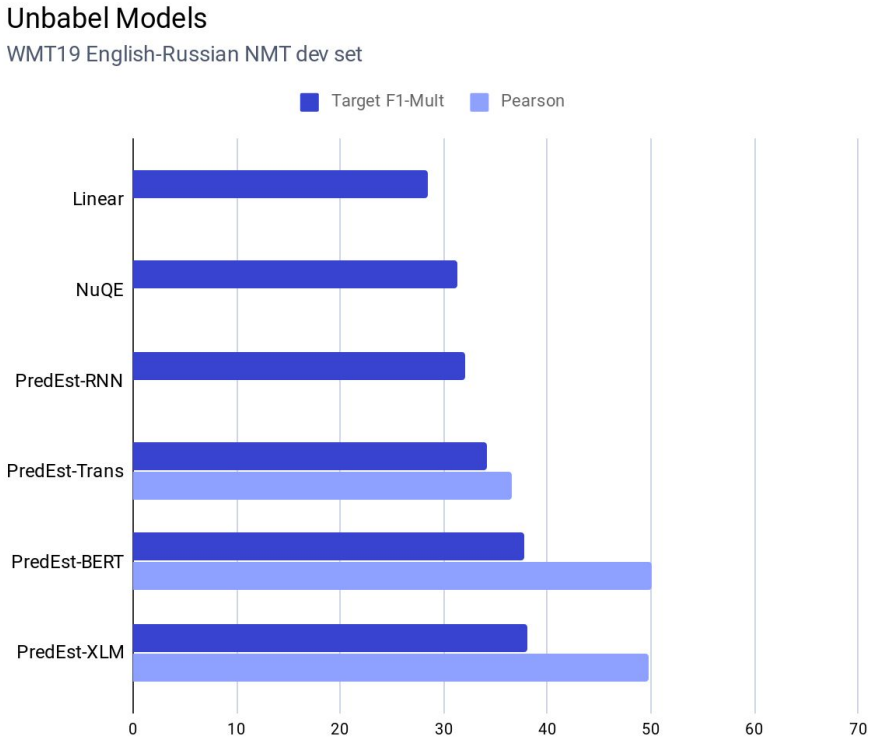
Validation Results English-German

- XLM provided the best single model
- Not that much improvement over plain pre-trained Predictor with Transformers
  - In-domain parallel data of about 3M
- NuQE and linear are the only models that use no extra data (despite POS tags)
  - They lack behind the weakest Predictor-Estimator by almost 5 points



Validation Results English-Russian

- BERT and XLM performed considerably better
- Most probably because English-Russian in-domain parallel corpus for pre-training the Predictor was very noisy
- QE data is also much more skewed that for English-German
  - Large majority of sentences with HTER 0
  - Then a bunch with HTER 1
  - And very few in between



Other models

APE-QE

- A translation or an APE system can be used for QE by treating its output as a surrogate reference and computing quality labels for the MT given that reference
    - **PSEUDO-APE**: Off-the-shelf translation system (OpenNMT)
    - **APE-BERT**: APE system built on BERT; described in detail in Unbabel's APE task paper

Ensemble methods

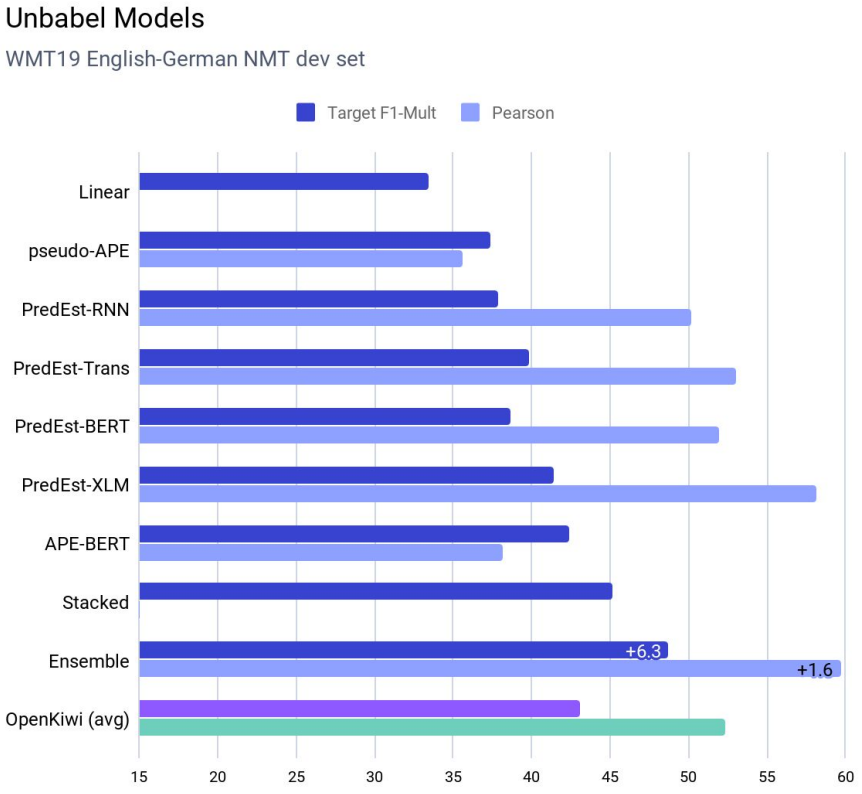
Word level

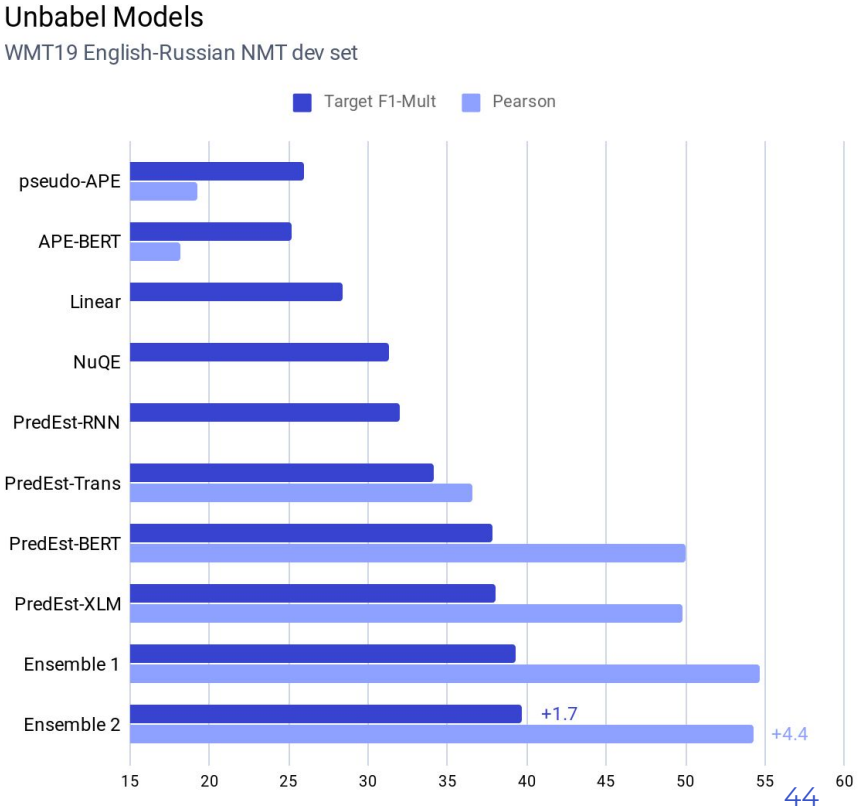
- Learn convex combination of model predictions via **Powell's conjugate direction method** (a variant of coordinate descent) to optimize F1-Mult score on dev set

Sentence level

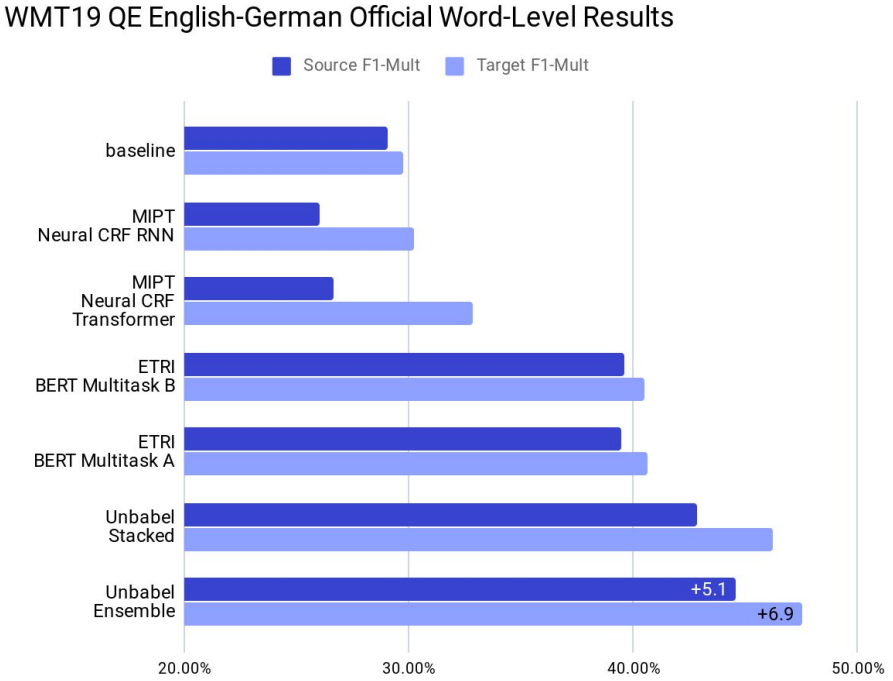
- Perform **L2 regularized regression** on the dev set with model outputs as features
- Choose best regularization constant via 20-fold **cross validation**

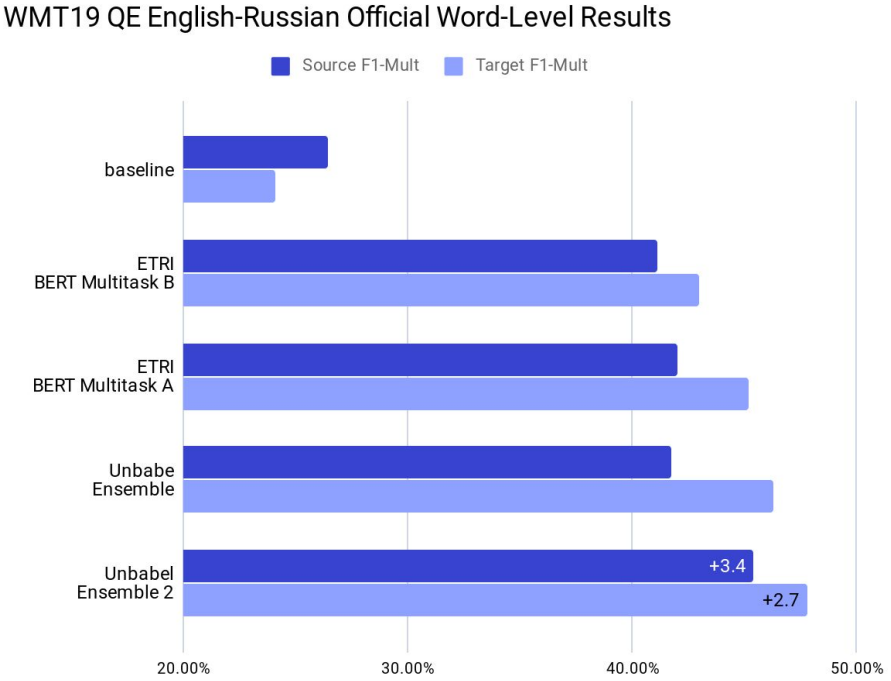
Dev set results English-{German,Russian}





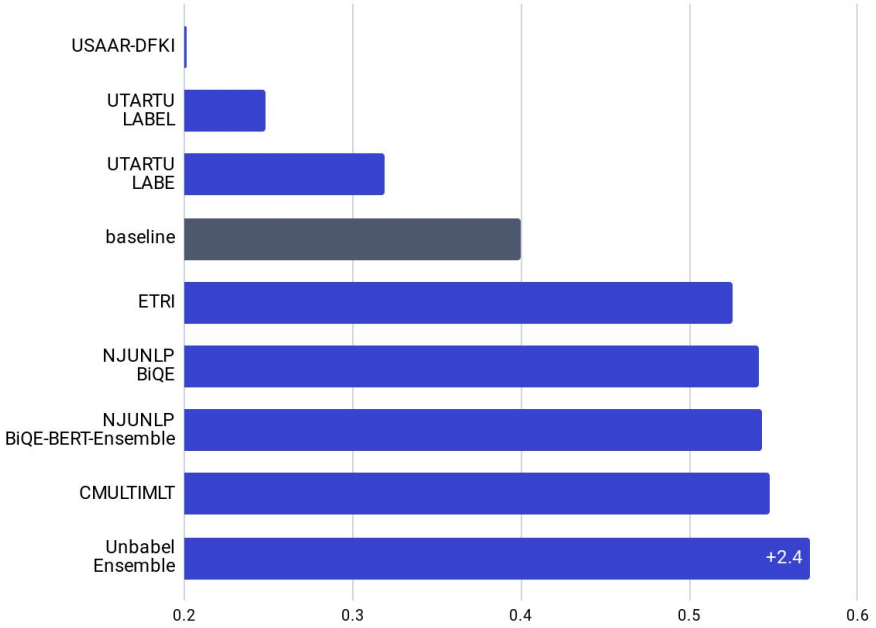
Official Results Word-Level



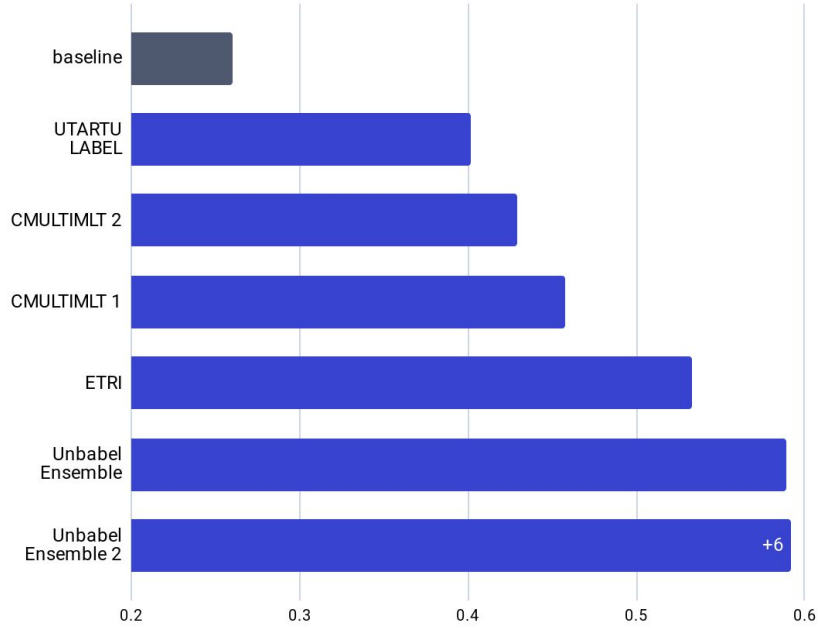


Official Results Sentence-Level

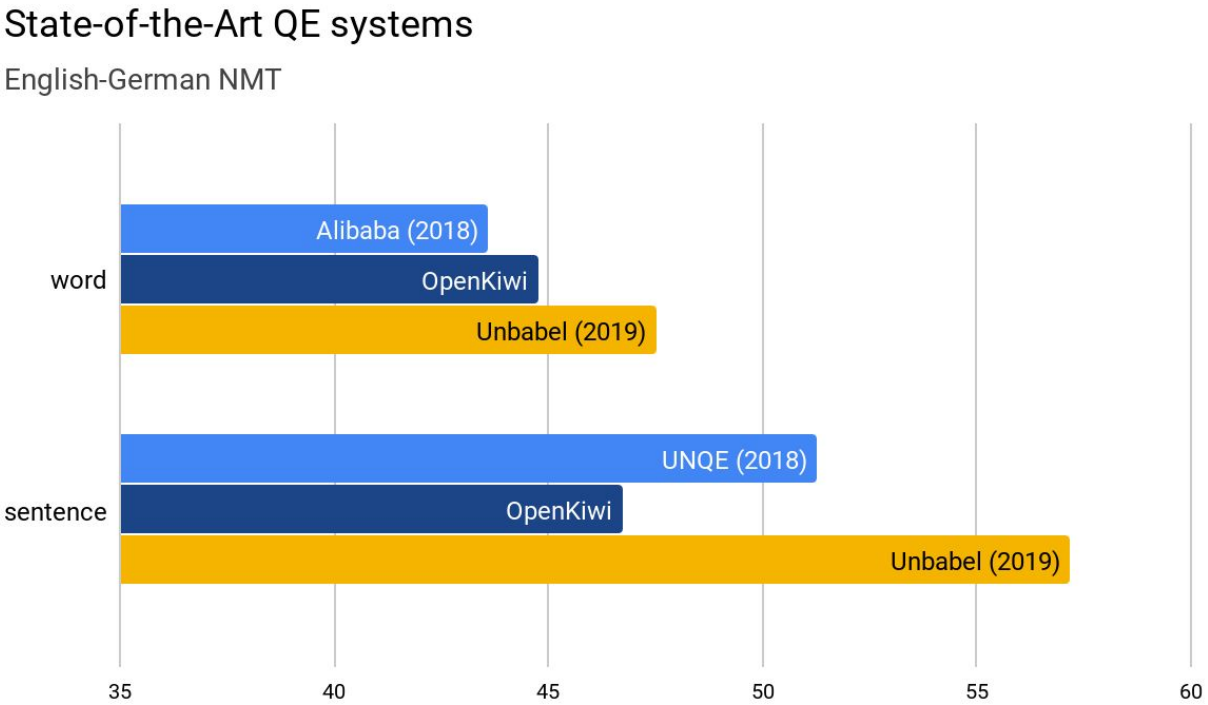
WMT19 QE English-German Official Sentence-Level Results



WMT19 QE English-Russian Official Sentence-Level Results



History


State-of-the-Art QE systems
English-German NMT
word
Alibaba (2018)
OpenKiwi
Unbabel (2019)
sentence
UNQE (2018)
OpenKiwi
Unbabel (2019)
35
40
45
50
55
60

Key Takeaways

- **Strong translation models** are hard to beat by dedicated QE systems
- **Diversity of models** can be just as important as individual model performance for **ensembling**
- A **smart ensembling strategy** is key to be able to scale to **many models** that have **high variance** in their individual performance

Key Takeaways

- Having a **modular QE framework** to build upon was key to quick experimentation with:
  - Submodels
  - Varying architectures
  - Hyper-parameters searching
- **The Muppet models** are yet again successful in a transfer learning task
  - But are very brittle in how they are fine-tuned and used

Unbabel

BUILDING UNIVERSAL UNDERSTANDING

Quality Estimation

AI Research

Unbabel Open Day 2019

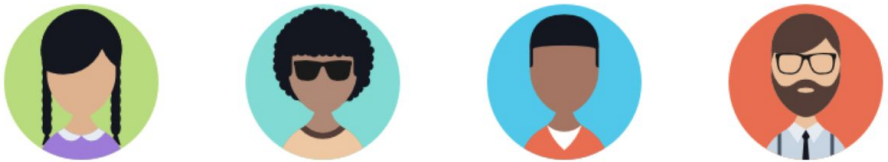
Roadmap

- Why QE?
  - Unbabel's use case (pipeline)
  - NMT is not calibrated (so no confidence along translations)
    - Neural models are over-confident
  - Current evaluation metrics (BLEU) are restricted to a single (good?) reference translation
    - QE is not constrained by a reference
- OpenKiwi:
  - Implementation of WMT-QE winning systems from 2016 to 2018
  - Better word-level numbers, close sentence-level ones
- Some impact in Unbabel's pipeline
- WMT-QE 2019 participation:
  - New models, much better numbers

Quality Estimation

Goal: estimate the **quality of translation**

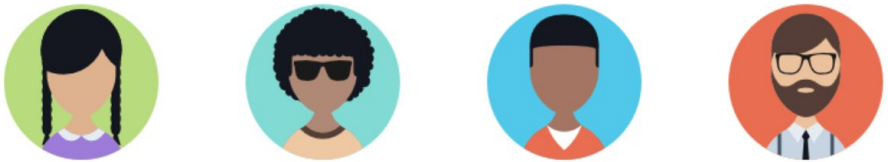
- If MT is good, we can skip the human **(+speed, -$)**
- Otherwise, we can at least highlight the parts that are wrong
- Ensures final quality **(higher MQM)**
- **Infinite data supply from our post-editors!**



MT Quality

What could we do if we knew the **quality of a translation?**

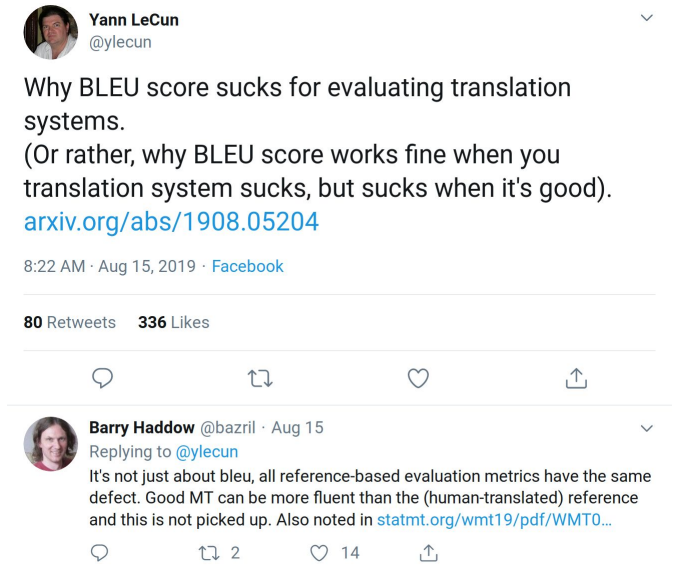
- If it is good, we can skip the human **(+speed, -$)**
- Otherwise, we can at least highlight the parts that are wrong
- Ensures final quality **(higher MQM)**
- **Constant data supply from our post-editors!**



Problems

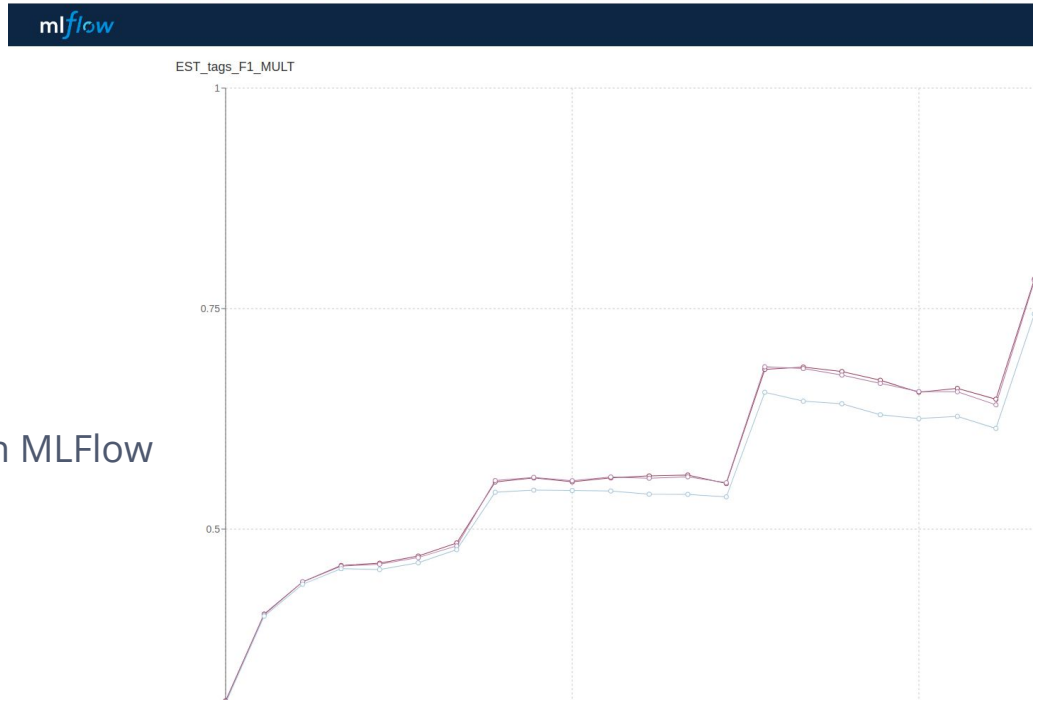
- Current evaluation metrics (BLEU) are restricted to a single (good?) reference translation
  - A major consensus at ACL and WMT just three weeks ago:
- NMT models are not well calibrated
  - So no reliable confidence provided with translations
- In fact, they are usually over-confident
  - Even when they hallucinate

- ACL and WMT were held just less than three weeks ago
- A major consensus:



OpenKiwi research-easy

- Quick experimentation
- Modular Design ⟶ Easily Extensible
- Automatic tracking of results through MLFlow



Key Takeaways

- **Strong translation models** are hard to beat by dedicated QE systems
- **The Muppet models** are yet again successful in a transfer learning task
- **Diversity of models** can be just as important as individual model performance for **ensembling**
- A **smart ensembling strategy** is key to be able to scale to **many models** that have **high variance** in their individual performance
- **Predicting a Gaussian** over HTER scores and training with **Maximum Likelihood** instead of Squared Loss was crucial for good performance in the sentence level task