CS7643 Project Report: Transfer Learning for Machine Translation Quality Estimation

Junyuan Tan
Georgia Institute of Technology
jtan319@gatech.edu

William Russ
Georgia Institute of Technology
wruss3@gatech.edu

Abstract

An important bottle neck for machine translation quality estimation is the need for labelled data. This requirement might not be an issue with high resource languages, but it would be more involved when estimating the quality of translation with lower resource languages. This project leverages multilingual models to encode high resource languages and use them to train the quality estimation engine, so that it can be applied to lower resource languages. A comparison of multilingual BERT model and LASER model is done, and each of them has advantages in areas such as accuracy and training time.

1. Introduction

1.1. Problem Statement

Machine translation quality estimation (QE) is a task to predict the quality of machine translation. There are certain languages that have high availability of training datasets, while some other languages do not. This project tries to explore a way of training the machine learning model using high resource languages, and adapt the model to estimate quality of machine translation for low resource languages.

1.2. Current Approach

There are ongoing research and development in the field of machine translation QE. A popular approach for this transfer learning is to leverage the famous Predictor-Estimator architecture developed by Kim, H., et al. [4] The Unbabel organization has proposed methods to replace the predictor part of architecture with Google's multilingual BERT embedding. [3] The output from multilingual BERT embedding is then passed to the original estimator, which is based on an RNN, to generate output Direct Assessment (DA) scores. The result of this setup produces great results. However, there are performance issues especially training and inference times as the multilingual BERT model is trained on 100+ languages, and it has various transformer based components as intermediate layers. This performance issue can be improved by using substitute pretrained representations.

1.3. Impact

Machine translation QE is mostly used on social media platforms to estimate the quality of machine translation before showing it to the end user. There are millions or even billions of requests to estimate machine translation quality each day on large social media platforms, and performance has become a critical part in user experience. If we can improve how long it takes to perform this QE task, social media platforms will be able to provide a better fine-tuned experience to users.

1.4. Dataset

The dataset we are using is provided by Facebook Research, which is adapted from WMT 2020 Quality Estimation Shared Task. [2] The dataset consists of Wikipedia data for languages with different resources. In this project, we are using medium-resource Romanian–English (Ro-En), Estonian–English (Et-En), and low-resource Nepalese–English (Ne-En) as training dataset input into the Predictor-Estimator model. Each of the language pairs contain 1,000 instances of source and target sentences.

Two sets of test instances of English–German (En-De) and English–Chinese (En-Zh) are used for evaluating the model performance. Each of these 2 testing data also contain 1,000 instances of source and target sentences.

2. Approach

Our work is inspired by Predictor-Estimator implementation found in Unbabel's OpenKiwi tool. [3] OpenKiwi has a very comprehensive implementation of various state-to-the-art models or architectures related to machine translation QE. We think that, in our project, it would be more effective if we reimplement the baseline Predictor-Estimator architecture with multilingual BERT.

2.1. Baseline

What we did first was to use HuggingFace's transformers to leverage multilingual BERT in PyTorch. [7] Input training data is actually pairs of source and target languages. The are formed as one input string using the following format: `[CLS] target [SEP] source [SEP]`, where `[CLS]` denotes the beginning of a sentence, and `[SEP]` denotes a sentence separator recognized by BERT. Input training data to the multilingual BERT model are treated as a replacement of the original predictor feature vectors.

This output is fed into an estimator, which contains an RNN and a linear layer as linear regression output. Specifically, a bi-directional LSTM is used to encode incoming data from the multilingual BERT output. Further processing is done according to what has been proposed by Kim, H., et al. [4] Formally, let $\boldsymbol{y}$ be the sentence, and $\boldsymbol{T_y}$ as the tokens in the sentence. We are taking the last hidden representations ($\overrightarrow{\boldsymbol{h}_{T_y}}$ and $\overleftarrow{\boldsymbol{h}_1}$) of the bi-directional LSTM from the left side to the right, and from the right side to the left, and concatenate them as a summary vector $\boldsymbol{s} = [\overrightarrow{\boldsymbol{h}_{T_y}}; \overleftarrow{\boldsymbol{h}_1}]$. See figure 1 for more details.

The output of this summary vector is then fed into a linear layer for linear regression calculation. The linear regression result is used to compare with the gold-standard Direct Assessment (DA) score annotated by human translators.

2.2. Model Specifications

In addition to the information mentioned above for the Predictor-Estimator model with BERT feature vectors as a predictor, Mean Squared Error loss is used to compute the difference between our predicted DA score and the gold-standard DA score. We are using Adam optimizer [5] for adjusting the model parameters with a learning rate $lr = 0.001$. We are doing the experiment with 10 total epochs at a time.

2.3. Model with LASER Embeddings

The BERT model we are trying to use has great prediction performance. However, due to model complexity, it takes longer time to execute. Improvements on training or inference time have become crucial. The reason is that we are living with these fast growing online social media platforms, like Facebook, Instagrams, etc., and there are millions or billions of user requests to run machine translation on posts or comments. Any improvements on training or inference time may have actual business values.

We first try to explore different pre-trained representations to replace BERT. The first thing we came up with is the LASER model developed by Facebook. [1] The model is based on bi-directional LSTM whereas BERT model is based on transformer architecture proposed by Vaswani, A., et al. [6] The LASER model has effectively 2 bi-directional LSTMs, one for encoder, and one for decoder, whereas the BERT model has multiple multi-head attention components, which have multiple layer in each of the multi-head attention component. Due to the nature of model complexity in the transformer architecture, training time and inference time will take longer than the simpler LASER model. And hence, we decided to give LASER model a try.

The original LASER implementation provided by Facebook in the repository (https://github.com/facebookresearch/LASER) [1] has a Docker image for consuming the sentence embedding out of LASER model. This approach is suitable for a production micro-service. However, in out experimentation, we would like a simpler approach to manage getting sentence embeddings. We did some investigation, and there is a ready-made PyTorch approach to consume sentence embeddings as PyTorch tensors. This effort is made available by Github user yannvgn in his repository (https://github.com/yannvgn/laserembeddings). [8]

Implementation provided in the repository returns only the sentence embeddings of input sentences, which are resulting from the max pooling operation of BiLSTM output. In our experimentation, we need the actual BiLSTM output vectors as an input to our estimator RNN. In this case, additional work is required to override a few classes including `Laser`, `BPESentenceEmbedding` and `SentenceEncoder` in Python package `laserembeddings` to return the full BiLSTM layer output to us.

The source and target languages BiLSTM representations from LASER model have 1024 feature vectors for each token. We concatenate both representations in the dimension of token. Formally, let the output from LASER model have the shape $\boldsymbol{N} \times \boldsymbol{T_{src}} \times \boldsymbol{H}$ and $\boldsymbol{N} \times \boldsymbol{T_{trg}} \times \boldsymbol{H}$, respectively, where $\boldsymbol{N}$ denotes how many sentence pairs in a batch, $\boldsymbol{T_{src}}$ and $\boldsymbol{T_{trg}}$ denotes how many tokens are there in the source and target languages after the built-in tokenization process in LASER model, and $\boldsymbol{H}$ denotes the number of hidden representations for each token, which is 1024 in this case. We concatenate these output so that it has a shape of $\boldsymbol{N} \times (\boldsymbol{T_{src}} + \boldsymbol{T_{trg}}) \times \boldsymbol{H}$. Note that $(\boldsymbol{T_{src}} + \boldsymbol{T_{trg}})$ may have variable lengths depending on the input source and target language pairs. In this case, we leverage the provided methods `pack_padded_sequence` and `pad_packed_sequence` from PyTorch to handle this issue.

We pass these representations to the estimator, which is also built by a bi-directional LSTM. The output from the estimator is then got further processed to generate summary vectors. This process is the same as what is described in section 2.1. The LASER model implementation also uses the same model specifications specified in section 2.2.

During training of the 2 models, we have freezed the pa-

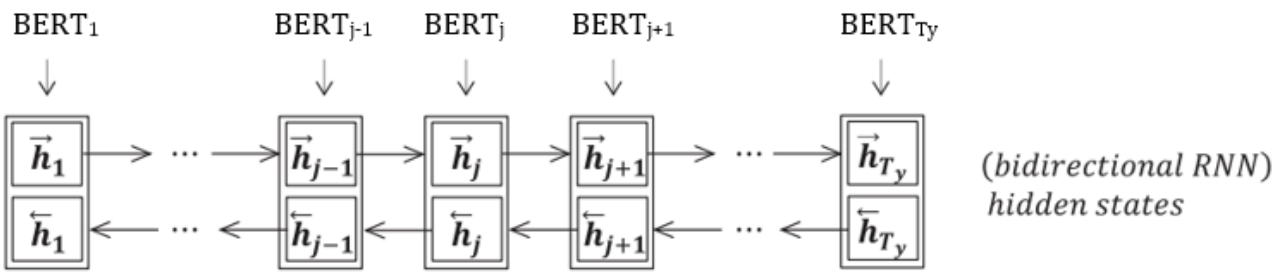

Figure 1. Hidden representation of estimator RNN layer.

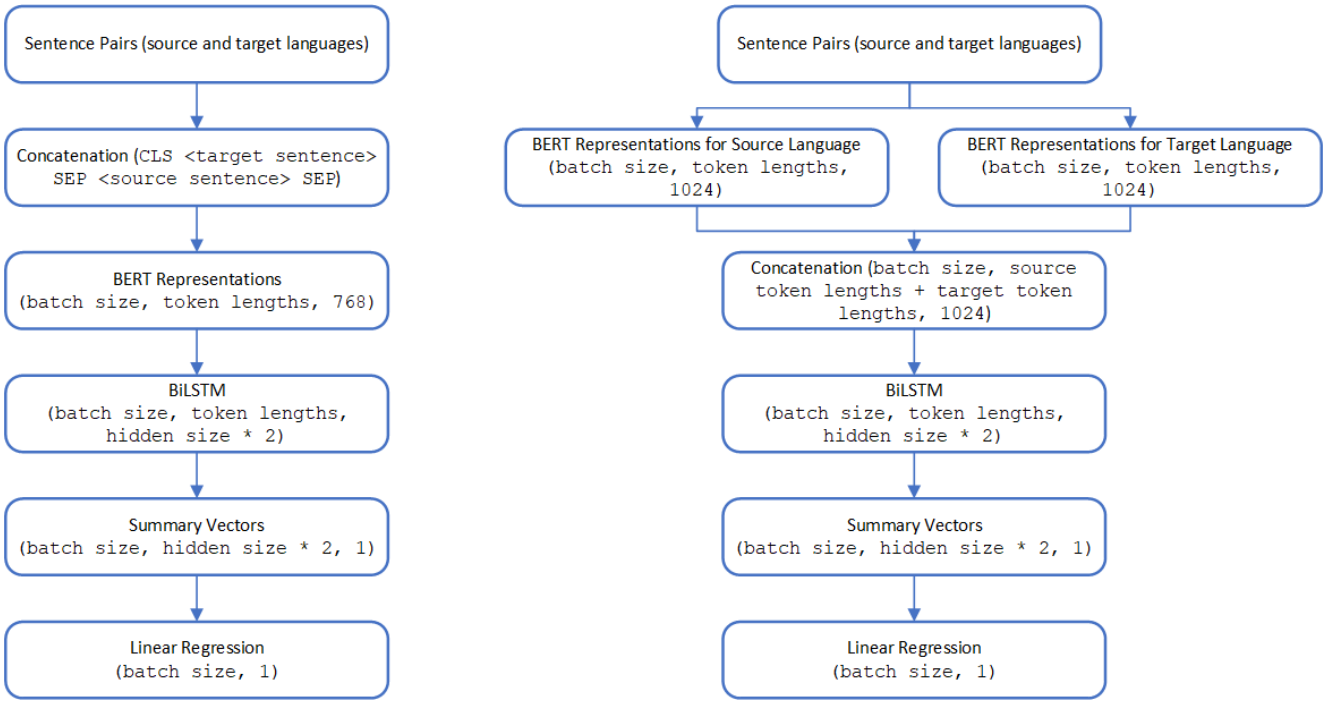

Figure 2. Architecture comparison of Predictor-Estimator model with BERT representations and LASER representations.

rameters found in the BERT and LASER embeddings so that they are not adjusted. Trainable parameters in these 2 architectures only include the parameters in estimator module, and the ones in linear regression layer. We have also included an architectural comparison of the baseline implementation and our experiment with LASER embeddings. Please refer to figure 2 for details.

3. Experiments and Results

3.1. Results

We define "success" in this project if the training or inference time of the improved model is less than the original baseline model, and also the accuracy of the improved model on testing data is not dropped by more than 20%.

We run our trained model using the testing data stated in section 1.4, which contains 2 sets of sentences in language pairs English-German and English-Chinese. Each set of training data contains 1,000 pairs of source and target sentences.

The environment that we produce these experiments is Google Colab with Python 3.7. An NVIDIA Tesla T4 GPU is used to produce the following results. We found that, under this environment, the average training time of the model with LASER embeddings (44.44 seconds) using the dataset specified in 1.4 is 3-times faster than the one with BERT embeddings (118.56 seconds). We have recorded the training time for 10 epochs, and an average training time is calculated in table 1.

In the case of performance comparison, the mean squared error loss has been decreasing most of the time during training as shown in figure 3, so overfitting should not be an issue during the training phase. We also evaluated the difference between the predictions and the gold-labels on

the test set using 2 metrics: Pearson Correlation and Root Mean Squared Error (RMSE). The results are shown in table 2. From what is shown in the data, RMSE ranges from 0.7263 and 0.7452, while Pearson Correlation ranges from 0.2352 and 0.2497. As a result, The 2 models are performing similarly in that the differences in both metrics are not large.

4. Further Research

As machine translation QE is a field that is actively being researched, newer ideas or proposals have come up. Possible future directions around improving upon our work would be to test different pre-trained representations. Another direction would be to explore ways to fine tune the models, like adding dropout layers or using different normalization techniques to improve model performance.

5. Conclusion

In this project, we have done a comparison between the baseline model based on Predictor-Estimator architecture with BERT embeddings and the improved model using LASER embeddings. Average time to train the model and model performance have been compared.

We conclude that the improvement we have worked on in this project is a "success" because the model with LASER embeddings has greatly reduced training and inference time up to 3 times, and the model performance has almost not changed.

6. Work Division

Please refer to table 3 for contribution information.

References

[1] Mikel Artetxe and Holger Schwenk. Massively multilingual sentence embeddings for zero-shot cross-lingual transfer and beyond, 2019. 2

[2] Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Francisco Guzmán, Mark Fishel, Nikolaos Aletras, Vishrav Chaudhary, and Lucia Specia. Unsupervised quality estimation for neural machine translation. *Transactions of the Association for Computational Linguistics*, 8:539–555, 2020. 1

[3] Fabio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, António Góis, M. Amin Farajian, António V. Lopes, and André F. T. Martins. Unbabel's participation in the wmt19 translation quality estimation shared task, 2019. 1

[4] Hyun Kim, Hun-Young Jung, Hongseok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. Predictor-estimator: Neural quality estimation based on target word prediction for machine translation. *ACM Trans. Asian Low-Resour. Lang. Inf. Process.*, 17(1), Sept. 2017. 1, 2

[5] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017. 2

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017. 2

[7] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020. 2

[8] yannvgn. Laser embeddings. https://github.com/yannvgn/laserembeddings, 2019. 2

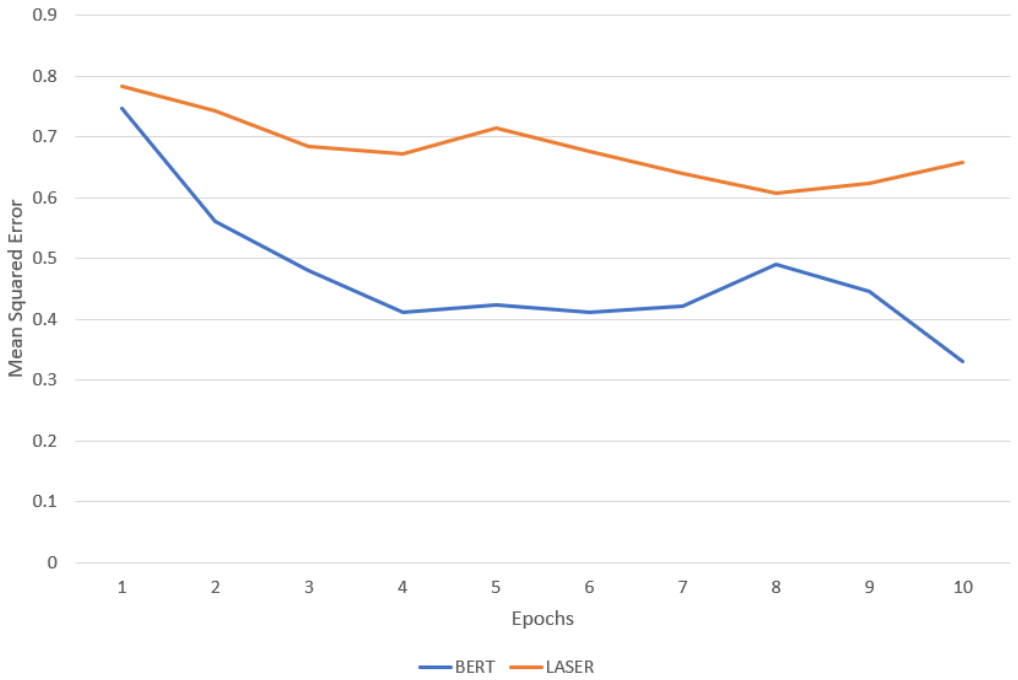

Figure 3. Comparison of mean squared error loss during training phase between model with BERT embeddings and LASER embeddings

	Epoch 1	Epoch 2	Epoch 3	Epoch 4	Epoch 5	Epoch 6	Epoch 7	Epoch 8	Epoch 9	Epoch 10	Avg. Time
BERT	109.84s	116.06s	120.05s	120.18s	119.87s	120.29s	119.94s	119.77s	119.80s	119.81s	**118.56s**
LASER	44.85s	44.14s	44.44s	44.14s	44.62s	44.62s	44.46s	44.31s	44.36s	44.46s	**44.44s**

Table 1. Comparison of training time for models with BERT and LASER embeddings for 1,000 pairs of source and target languages training data.

	RMSE	Pearson Correlation
BERT	0.7452	0.2352
LASER	0.7263	0.2497

Table 2. Comparison of model performance on testing data using Root Mean Squared Error and Pearson Correlation

Student Name	Contributed Aspects	Details
Junyuan Tan	Baseline Implementation	Scraped the dataset for this project and implemented and adapted the baseline using BERT embeddings for this project.
Junyuan Tan	LASER Embeddings Implementation	Implemented the comparison using the LASER model.

Table 3. Contributions of team members.