Transfer Learning for Quality Estimation

Andrew Marmon
amarmon3@gatech.edu

Shouen Lee
shouenlee@gatech.edu

Rishi Gurnani
rgurnani96@gatech.edu

Haley Xue
hxue42@gatech.edu

Abstract

Many commercial translation workflows are missing accurate quality estimation which can help to improve translation results. For instance, predicting machine translation quality can help in post-editing tasks. We attempt a transfer learning task for quality estimation to score sentence-level translations between language pairs with limited training data. We explore multiple approaches from various submitted papers from WMT20 using pre-trained transformers. In addition, we adapted our own transformer architectures to be used for QE predictions. We assess our models in a multilingual setting and show that our results are in line with those of the state-of-the-art models submitted to WMT20. In fact, our models beat the baseline results obtained from OpenKiwi's Predictor-Estimator architecture.

1. Introduction/Background/Motivation

With the widespread use of machine translation systems, there is a growing need to evaluate translated results at low-cost. Machine translation quality estimation (QE) is the task of predicting the quality of a machine translated segment without access to a gold-standard translation.The quality of a translation is predicted only given the source sentence and its translation using QE. This also implies that successful quality estimation can be done without reference translations. The estimated quality can inform users about the reliability of the translation, or whether it needs to be post-edited. Accurately making this determination is critical for the task of removing poor translations that likely do not represent the intentions of the author.

For medium to high labeled languages, the level of fluency of translations from state of the art models tends to be very high, making translation seemingly high quality, though they may still contain meaning preservation errors. This makes it hard for models and even humans to identify mistakes in translations. The problem is exacerbated for low labeled languages.

Previous QE systems (Fomicheva *et al*. [4], Agarwal *et al*. [1]) generally include pretraining and fine tuning steps, where the former step involves masked language modeling (MLM) utilizing large datasets of multiple languages, with the expectation that the models will learn cross-lingual relationships. As with most supervised machine learning problems, an important bottleneck in QE is the need for labelled data. To alleviate the need for labelled data, we use pre-trained representations such as BERT and XLM in some of our approaches. These methods still require thousands of instances for fine-tuning language representations and QE scores.

We worked on the Transfer Learning for Quality Estimation Facebook project which aims to predict human direct assessment (DA) scores between sentence pairs over seven languages. We were provided with data over the language pairs, their DA scores, and the NMT model used for translation. The task of sentence-level QE for direct assessment (DA) involves predicting the perceived quality of the translation given the source and the translated sentences. Our models utilize the pretrained multilingual language models such as BERT, RoBERTa, and Cross-lingual Language Model (XLM).

In this paper we explore four different architectures and approaches, which are modified from existing models and architectures presented for Task 1 at WMT20, for quality estimation for each of the language pairs. We use six datasets which include translations from English to [Chinese, German] and [Estonian, Nepalese, Sinhala, Romanian] to English. Each translation is annotated with multiple DA scores which we averaged and z-normalized. In all of our approaches, the source and target sentences are fed to the model which generated a QE score for the translation pair.

2. Approach

All models used the same data sets (see Section 2.1). All models were built using the PyTorch Deep Learning framework.

2.1. Data Sets

We use six datasets which include translations from English to [Chinese, German] and [Estonian, Nepalese, Sinhala, Romanian] to English annotated with multiple DA scores which we averaged and z-normalized. This aggregate version of the DA scores is henceforth referred to as z-score. In all of our approaches, the source and target sentences are fed to the model which generated a QE score for the translation pair. This data was taken from `https://github.com/facebookresearch/mlqe`. An additional pre-processing step was taken to remove rows with missing QE scores.

Each language pair includes a `train`, `dev` (used for validation), and `test` data set. The `train` data set consists of 7,000 sentence pairs and the `dev` and `test` each consists of 1,000 sentence pairs.

2.2. Multilingual Bergamot + DeepQuest

We hypothesize that there are similar features and contextualized representations between languages which can be capitalized on by sharing model parameters between language pairs since no fundamental linguistic information is fed to the model. Therefore, to build a multilingual QE system, where a single model can be used to predict quality for multiple language pairs, we simply concatenate the available data for all languages and use it for training our transformer models, in our case, multilingual BERT. For a simple multilingual model, we stack all seven language pairs as one dataset and run it through the same multi-lingual transformer model and MLP (Figure 1). In this way, we train one model that can generalize over all the input languages for QE score prediction. This can be useful for multilingual translation systems where the user does not need to identify the input languages, and especially for zero-shot settings where a given language pair may not have been seen at training time.

We further explore a similar architecture to the BiRNN model from DeepQuest [5]. In this approach we use a transformer architecture with two pretrained multilingual BERT models, one is fed the source sentence and the other is fed the target sentence. Their outputs are normalized and simply concatenated together and fed through an MLP which includes a dropout layer, dense layer, rectified linear layer, and another dense layer which produces a QE score for the sentence pair (Figure 1).

For the sake of transfer learning for low-labeled language pairs, we apply our hypothesis that there are similar features and contextualized representations between languages which can be capitalized on by sharing model parameters between language pairs. We take another approach at language-specific QE by sharing parameters of our contextualized representations in the MLTMs between all language pairs during the entirety of training, but keeping

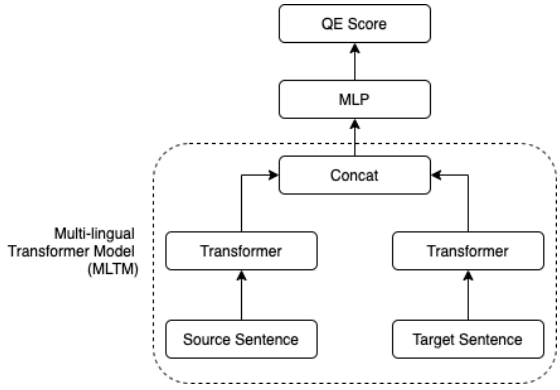

Figure 1. Multilingual Bergamot + DeepQuest architecture

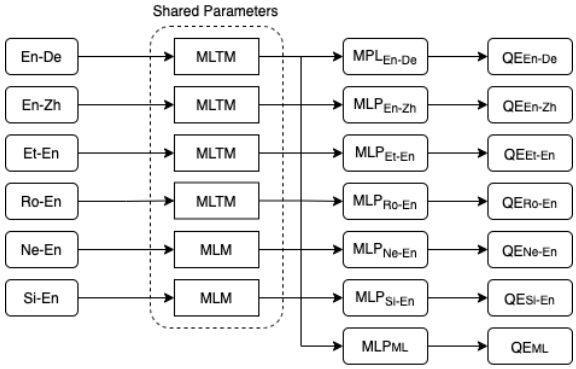

Figure 2. Multilingual Bergamot + DeepQuest model with two shared multilingual BERTs.

the parameters for the MLP layers language-specific (Figure 2). We can target low-labeled languages by first getting some representation of high-labeled language pairs and use their parameters for further training on low-labeled language pairs. We accomplish this by training our model with high-labeled language datasets followed by low-labeled language datasets. In this sequence of training, the model first learns representations of high-resource languages in the transformer pair and weights in the MLP before moving onto low-resource languages. This potentially allows low-resource language learning to use representations gained from high-resource languages in previous steps in the training process.

The source and target sentences are fed into the model and the output QE scores are compared to the z-normalized mean DA scores unchanged from the dataset. We use a combination of loss functions, specifically RMSE loss and 1 - Pearson Correlation to update the weights of our model and AdamW as our optimizer. We predicted using two different losses would increase the model's generalizing rate and be able to optimize for more than just RMSE or Pearson Correlation.

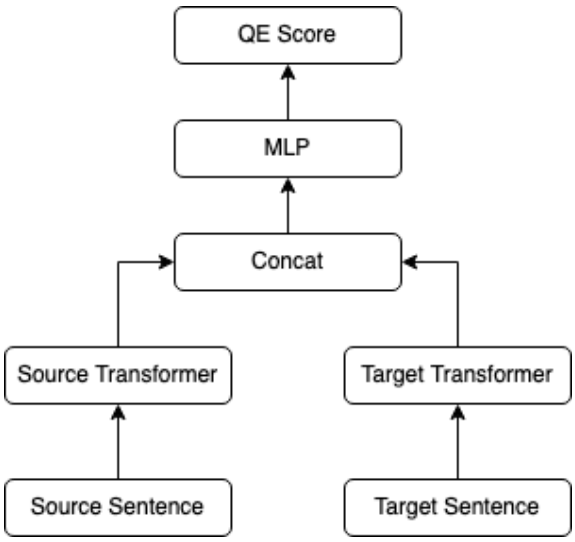

Figure 3. Bilingual DeepQuest with transformer sentence embeddings architecture. We use transformers trained on the source and target pairs individually in order to see if that improves performance over the multilingual transformer variants. Because of this, weight sharing is not applicable and the Bergamot approach cannot be applied.

2.3. Bilingual DeepQuest with Transformer Sentence Embeddings

We also improve upon DeepQuest paradigm of concatenating sentence embeddings for a source and translation pair with non-shared parameter transformers. As stated previously, DeepQuest leverages BiRNN to encode sentences, whereas in our approach we propose using a transformer-based model to accomplish this task. As opposed to the above Bergamot + DeepQuest architecture, we must train two different transformers which are trained on the source and target languages individually instead of being a shared multilingual model as seen in Figure 3. We then use these transformers as feature encodings for these source and translation sentences, feed them through a feedforward layer and batch normalization, concatenate them, and then predict the quality estimation z-score to determine the translation quality.

We also compare the above Multilingual Bergamot + DeepQuest model against this single-language counterpart trained only on the source and target languages in question. Specifically, in the above multilingual BERT model we would tokenize any language (English, German, Chinese, etc.) into a model which can handle every language at the same time. In this approach, we use each trained model separately to determine how this affects performance. For English to German translation, this means we will use BERT trained on English to feed into the source sentence embedding and BERT trained on German to be leveraged in the translation sentence embedding.

2.4. TransQuest

In addition to implementing our BERT based models, we also chose to experiment with the proposed architecture for the TransQuest framework; specifically we recreated the MonoTransQuest model [7]. Instead of using multilingual BERT to embed the source and target in the quality estimation task, TransQuest leverages the XLM-Roberta transformer model. Like the researchers who developed TransQuest, we hypothesize that crosslingual embeddings may greatly improve direct assessment score predictions as XLM-R has been shown to outperform mBERT on crosslingual classification tasks [2]. While in our other models, we faced challenges in representing inputs for low-resource languages, we believe that XLM-R should be more robust as it was trained on over 100 languages. Moreover, MonoTransQuest's relatively simple architecture makes it less computationally and time intensive to train, making it an attractive architecture which balances accuracy and efficiency.

MonoTransQuest utilizes the pre-trained XLM-R-large model framework available in the HuggingFace Transformer's library [8]. The intuition behind the MonoTransQuest model is to use a sequence classification architecture and adapt it to a regression problem to predict quality scores (Figure 4).

First, the source sentence and translation sentence are embedded using the pre-trained tokenizer. The two sentences are concatenated together with a `</s>` separator token (the XLM-R equivalent of the [SEP] token) and fed into the pre-trained XLM-R model. The resulting `<s>` token (the XLM-R equivalent of [CLS]) of the output is then fed into a classification head consisting of a dropout layer, fully connected layer, tanh activation, another dropout layer, followed by a final fully connected layer. The hidden size of the fully connected layers is 1024, which matches that of the pre-trained model. To adapt the model to a regression problem, the number of classes is set to one, and instead of applying a softmax activation function to the final output from the classification head, we used a sigmoid layer to predict the final quality score (This is a slight modification to the original MonoTransQuest architecture). We note that since the sigmoid function has a range of [0, 1], we first pre-processed the z-scores so that they were also scaled to be between 0 and 1 when training the model. We used mean-squared error as our loss function.

For model training, we used an Adam optimizer with a learning rate of 1e-05 and a linear learning schedule with a warmup of 10% of the training data set to reduce volatility in the beginning of training. Each model was trained with 3 epochs as we noticed that Pearson correlations and losses tended to plateau after 3 epochs. During training, the pre-trained XLM-R model is fine-tuned, and the weights are

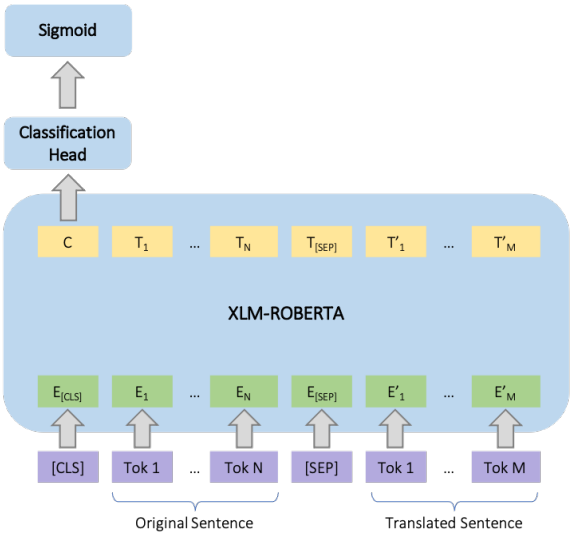

Figure 4. MonoTransQuest architecture

updated for our data sets. Thus the XLM-R and classification head have learnable parameters; the tokenization and post-processing steps do not (i.e. inversely transforming the predicted z-scores to allow for negative values). More details regarding hyper parameters can be found in section 3 below.

2.5. TransEncoder + MLP

To predict z-score of translations we trained three separate models on the same data set (top gray box in Fig. 5). The first two models were translation models for a target language pair (A, B). Our two translation models use the same architecture. This architecture, shown in Fig. 6, is a one-layer transformer encoder (TransEncoder). One model translated A to B and the other translated B to A. Once trained, each model was used to generate an embedding of its target language (e.g., the A to B translation model produced an embedding for language B).

The third model is a multi-layer perceptron (MLP) trained to predict the z-score of a (A, B) translation. The input to this z-score model is the concatenated embeddings of A and B. The MLP architecture consisted of the following: dense layer, relu, dropout, dense layer.

We used an Adam optimizer to minimize the mean-square error loss of the z-score for the MLP. We used an Adam optimizer to optimize the cross-entropy loss of the target language sentences for the two TransEncoders. The learned parameters in the TransEncoder include: the input embeddings, positional encodings, query matrices, value matrices, key matrices, batch normalization parameters, and the weights and biases of the feed forward and linear layers. The learnable parameters in the MLP include the weights and biases of the dense layers.

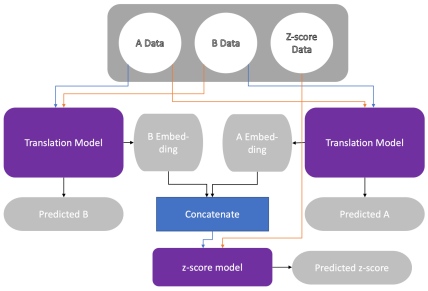

Figure 5. The workflow of z-score prediction for a target language pair (A, B). Three models are trained, each are shown in purple. The inputs to these models are shown by blue arrows. The targets of these models are shown by orange arrows.

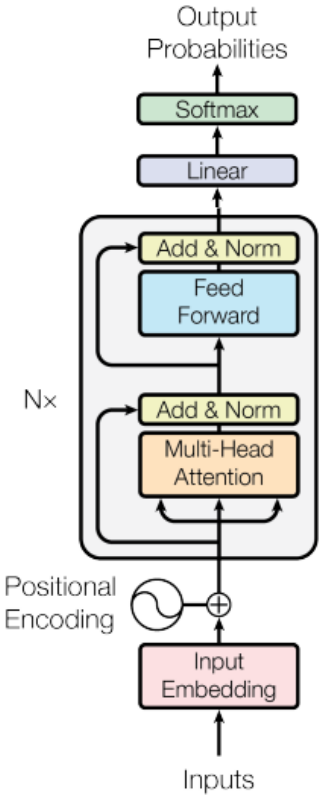

Figure 6. This is the architecture of the translation model. The output of the linear layer is used as the embedding.

We anticipated that this approach would be successful due to the success of Transformers in the field of Natural Language Processing. One problem encountered was that Spacy did not have specific tokenizers for each of the languages present in our data set. Because of this, we used the multi-lingual Spacy tokenizer for Sinhala, Estonian, and

Nepali.

3. Experiments and Results

3.1. Multilingual Bergamot + DeepQuest

Transfer learning from high-labeled language pairs to language pairs between high-labeled and low-labeled languages for quality estimation shows to benefit model performance when predicting translation quality. We hypothesized that there are similar features and contextualized representations between languages which we attempt to capitalize on by sharing model parameters for high-labeled languages with low-labeled languages during training.

We are able to achieve significantly better Pearson correlation scores on the low-labeled language pairs with shared parameters (Table 3) with the exception of Si-En, whose lower score may be attributed to a lack of available tokenizers for Sinhala. In this case, its shared and unshared scores would be invalid.

Multilingual Bergamot + DeepQuest model was trained on Google Cloud with a Nvidia P100 GPU for 3 - 4 hours on average over all the language pair training sets (batch size=16). We found that a learning rate of 2e-5 with 0.3 dropout. Higher dropouts reduce the model overfitting but decrease the Pearson Correlation scores. The model generalizes well at the start ($\leq$ 10 epochs) but then quickly overfits as the training loss for RMSE drops below 0.2 and the training Pearson Correlation rises over 0.9.

3.2. Bilingual DeepQuest with Transformer Sentence Embeddings

Another key finding is that using the DeepQuest model with transformers trained only on the language that is being embedded does outperform the multilingual shared parameter model on the languages that are publicly available (i.e. English, German, and Chinese BERT models). As can be seen in Table 2, the Bilingual DeepQuest model achieves significantly higher pearson correlation score on both the En-De translation as well as En-Zh test datasets, improving scores from 0.25 to 0.45 and 0.2 to 0.36 respectively.

This result does have some interesting implications in terms of the efficacy of multilingual transformers as opposed to their single-language counterparts. Ideally, multilingual models would be able to leverage the shared learnings from all the languages it is trained on to reinforce its ability to create useful sentence embeddings any given language. However, this result suggests that for this quality estimation task, the fact that this model is trained on many languages can actually be a hinderence in predicting the quality of very common translations like English to German and English to Chinese.

This model architecture was trained over 25 epochs with a batch size of 16 on a P100 GPU in Google Colab, resulting in a training time of roughly 5 hours. An Adam optimizer was used with a learning rate of 0.0001 to train this model. To reduce overfitting, we employed a dropout of 0.3 in the fully-connected output layer. Many different configurations were tried, however no grid search hyperparameter tuning was done due to the contraints applied on Google Colab GPUs running a 5 hour training run for each hyperparamter configuration.

3.3. TransQuest

Our MonoTransQuest model was trained separately on the training dataset for each language pair and evaluated using the test dataset for the same pair. Even though our implementation was done from scratch, we were able to closely replicate the results that the original researchers achieved [7]. It seems that the strength of XLM-R in crosslingual tasks helped in achieving better Pearson correlation scores. Although we note that, interestingly, the MonoTransQuest model demonstrated better performance on low-resource language pairs.

We conducted hyperparameter tuning (Table 4) using the En-De data sets, and our best set of parameters (dropout = 0.1, learning rate = 1e-05, batch size = 8) was used across all language pairs to keep our result comparisons consistent. Although increasing dropout reduced overfitting, it also decreased the Pearson score. We noticed that with learning rates larger than 1e-05, our model would not learn and thus predict the same z-score for all sentence pairs. We suspect that this is because of the vanishing gradients problem stemming from our use of the tanh and/or sigmoid activation layer(s). We used tanh as it was the activation function used in the original BERT code [3]. We also experimented by replacing the tanh layer with ReLU; however, while training was slightly faster, there was no significant improvement in Pearson scores; in fact, scores were worse for several language pairs. Future improvements to this model could include changing the activation layers and other layers in the classification head.

Model training and experiments were conducted using Tesla T4 and P100 GPUs on Google Colab. Training took on average 30-40 minutes and about 30 seconds to predict scores for 1,000 instances on each test dataset.

3.4. TransEncoder + MLP

Our Transformer Encoder + MLP model was trained separately on the training data set for each of the six language pairs studied in this work. The number of epochs was set to 10 and batch size was set to 128. The learning rate and dropout percentage were set to 0.01 and 0.2 respectively based on a hyper-parameter search, the results of which are shown in Table 5. This hyper-parameter search was conducted on the En-De language pair. The values shown in Table 5 are best mean average error on the validation set

	En-De	Et-En	En-Zh	Ro-En	Si-En	Ne-En	Avg
Multilingual Bergamot	0.59	0.67	0.65	0.66	0.67	0.78	0.62
Bilingual DeepQuest	0.55	N/A	0.53	N/A	N/A	N/A	0.54
MonoTransQuest	0.55	0.69	1.26	0.32	0.51	0.42	0.63
TransEncoder + MLP	0.60	0.74	0.59	0.75	0.67	0.67	0.67

Table 1. MAE losses between human DA scores and predicted values for given test sets. Avg is the average MAE across language pairs. For Bilingual DeepQuest, the only non-multilingual pre-trained BERT models publicly available are English, German, and Chinese.

	En-De	Et-En	En-Zh	Ro-En	Si-En	Ne-En	Avg
Multilingual Bergamot	0.25	0.71	0.20	0.79	0.24	0.89	0.51
Bilingual DeepQuest	0.45	N/A	0.36	N/A	N/A	N/A	0.41
MonoTransQuest	0.47	0.72	0.47	0.88	0.61	0.78	0.65
TransEncoder + MLP	0.002	0.032	-0.09	0.14	-0.019	-0.008	0.001
OpenKiwi	0.15	0.48	0.19	0.68	0.37	0.39	0.38

Table 2. Pearson correlation scores between human DA scores and predicted values for the given test sets. Avg is the average Pearson correlation scores across language pairs. Similar to Table 1, the only non-multilingual pre-trained BERT models publicly available are English, German, and Chinese.

	Et-En	Ne-En	Ro-En	Si-En
No shared parameters	0.44	0.38	0.55	0.32
Shared ML BERT parameters	0.71	0.89	0.79	0.24

Table 3. Shared vs No Shared transformer parameters.

	0.1	0.3	0.5
1e-5	RMSE: 0.68 MAE: 0.55 Pearson: 0.47	RMSE: 0.68 MAE: 0.55 Pearson: 0.41	RMSE 0.70 MAE: 0.56 Pearson: 0.10
1e-6	RMSE: 0.76 MAE: 0.63 Pearson: 0.23	RMSE: 0.70 MAE: 0.56 Pearson: 0.22	RMSE 0.69 MAE: 0.53 Pearson: 0.17

Table 4. Hyperparameter tuning for MonoTransQuest on En-De pair. Rows are dropout values and columns are learning rates.

over 10 epochs.

Six models were trained, each using the hyper-parameters specified above. The performance of each model on the test set are shown in Tables 1 and 2. None of the models over-fit since the number of training epochs was set to a small value, 10. However, the low R^2 values of the TransEncoder + MLP models (shown in Table 2) coupled with the reasonable MAE values (shown in Table 1) indicate that the model is not able to generalize well and just predicts an average z-score regardless of the input. This could also be because the number of epochs was just 10 and that this model did not utilize pre-training.

4. Conclusion

We found that using the state-of-the-art pre-trained transformer models did perform better than the OpenKiwi [6] baseline on every single translation task that they were used in, including Multilingual Bergamot, Bilingual DeepQuest, and MonoTransQuest. Interestingly, the transformer-based TransEncoder + MLP model that did not leverage one of these pre-trained large language models (i.e. BERT, XLM-R, etc.) did not outperform the baseline, showing that these models trained on millions of sentences do translate well to the quality estimation task. In addition, we note that the XLM-R based TransQuest model produced predictions with the highest Pearson correlation for all language pairs except for Ne-En. This is in line with our expectations given that XLM-R is better suited for cross-lingual tasks.

We proposed new ways to solve the quality estimation problem that outperformed the baseline in every quality estimation task. However, due to limited resources we were not able to match the batch sizes, number of epochs, etc. that other transformer-based architectures were able to leverage that could yield similar or better results. It is a subject of future work to scale these models to multiple GPUs/TPUs to obtain the highest possible performing results.

5. Work Division

Work was divided equally among the team as each group member implemented, trained, experimented, and analyzed results of a different deep learning model. All members also contributed to the writing of the final report.

The specific contributions of each member are detailed in Table 6.

	.05	.01	.001	.0001
.2	.5948	.5774	.5837	.5856
.3	.6308	.5895	.5869	.5888
.5	.6273	.5937	.5824	.5931

Table 5. Results of hyper-parameter sweep for our Transformer Encoder + MLP model. Values shown are best mean average error on the validation set over 10 epochs. Row headers are drop-out percentage, column headers are learning rate.

Student Name	Contributed Aspects	Details
Andrew Marmon	Data Loader and Bilingual DeepQuest	Formulated and implemented the bi-lingual transformer-based approach that leverages the underlying architecture of DeepQuest. I also implemented a generalized PyTorch DataLoader for the quality estimation dataset that can be used across different language pairs.
Shouen Lee	Implementation, Multilingual Bergamot + DeepQuest	Applied Bergamot shared parameter architecture and DeepQuest concatenate over transformer outputs. I attempted the multi-lingual challenge with shared and non-shared parameters to transfer learn for low-labeled languages. Wrote sections relating to Multilingual Bergamot + DeepQuest model.
Rishi Gurnani	Implementation, Analysis, Writing	Trained the TransEncoder + MLP model for all language pairs. Performed hyper-parameter search. Wrote Sections 2 and 3 parts related to TransEncoder + MLP. Also wrote the Section 2 pre-amble and data set section.
Haley Xue	Implementation and Analysis of TransQuest	Replicated the MonoTransQuest architecture using the HuggingFace Transformers library. Performed hyperparameter tuning and analyzed results. Experimented with changing proposed architecture in original research paper.

Table 6. Contributions of team members.

References

[1] Mayank Agarwal, Kartik Talamadupula, Stephanie Houde, Fernando Martinez, Michael Muller, John Richards, Steven Ross, and Justin D. Weisz. Quality estimation interpretability for code translation, 2020. 1

[2] Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*, 2019. 3

[3] Jacob Devlin. Google research: Bert. https://github.com/google-research/bert/blob/bee6030e31e42a9394ac567da170a89a98d2062f/modeling.py#L231, 2019. 5

[4] Marina Fomicheva, Shuo Sun, Lisa Yankovskaya, Frédéric Blain, Vishrav Chaudhary, Mark Fishel, Francisco Guzmán, and Lucia Specia. BERGAMOT-LATTE submissions for the WMT20 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1010–1017, Online, Nov. 2020. Association for Computational Linguistics. 1

[5] Julia Ive, Frédéric Blain, and Lucia Specia. deepQuest: A framework for neural-based quality estimation. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 3146–3157, Santa Fe, New Mexico, USA, Aug. 2018. Association for Computational Linguistics. 2

[6] Fábio Kepler, Jonay Trénous, Marcos V. Treviso, Miguel Vera, and André F. T. Martins. Openkiwi: An open source framework for quality estimation. *CoRR*, abs/1902.08646, 2019. 6

[7] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. Transquest: Translation quality estimation with cross-lingual transformers. 2020. 3, 5

[8] 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. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45. Association for Computational Linguistics, 10 2020. 3