Translation Quality Estimation for Indian Languages

by

Nisarg Jhaveri, Manish Gupta, Vasudeva Varma

in

The 21st Annual Conference of the European Association for Machine Translation
(*EAMT-2018*)

ALACANT/ALICANTE, SPAIN

Report No: IIIT/TR/2018/-1

Centre for Search and Information Extraction Lab
International Institute of Information Technology
Hyderabad - 500 032, INDIA
May 2018

Translation Quality Estimation for Indian Languages

Nisarg Jhaveri **Manish Gupta*** **Vasudeva Varma**
International Institute of Information Technology,
Gachibowli, Hyderabad, Telangana - 500 032, India.
nisarg.jhaveri@research.iiit.ac.in,
{manish.gupta, vv}@iiit.ac.in

Abstract

Translation Quality Estimation (QE) aims to estimate the quality of an automated machine translation (MT) output without any human intervention or reference translation. With the increasing use of MT systems in various cross-lingual applications, the need and applicability of QE systems is increasing. We study existing approaches and propose multiple neural network approaches for sentence-level QE, with a focus on MT outputs in Indian languages. For this, we also introduce five new datasets for four language pairs: two for English–Gujarati, and one each for English–Hindi, English–Telugu and English–Bengali, which includes one manually post-edited dataset for English–Gujarati. These Indian languages are spoken by around 689M speakers world-wide. We compare results obtained using our proposed models with multiple state-of-the-art systems including the winning system in the WMT17 shared task on QE and show that our proposed neural model which combines the discriminative power of carefully chosen features with Siamese Convolutional Neural Networks (CNNs) works best for all Indian language datasets.

1 Introduction

In recent years, Machine Translation (MT) systems have seen significant improvements. However, the quality of the output obtained from these MT systems is neither perfect nor consistent across multiple test cases. The task of Translation Quality Estimation (QE) aims to estimate the quality of an MT output without any reference translation.

QE is now critically important with the increasing deployment of MT systems in practical environments. QE has been shown to be extremely useful and is widely used in Computer Aided Translation (CAT) environments (Escartín et al., 2017; Turchi et al., 2015). QE can also be useful in various applications and systems such as cross-lingual summarization, cross-lingual information retrieval, etc., which rely on high quality translations. With the help of QE, such systems can automatically pick the best translation out of several proposed translations by multiple MT systems. If the estimated quality is still unsatisfactory the system can alert the user about the poor quality, or fall-back to some alternate way to find a better translation.

Word, phrase, sentence or document level QE has been studied extensively by various researchers. WMT12-17 (the 7^{th} to 10^{th} workshops on statistical machine translation and the 1^{st} and 2^{nd} conferences on machine translation) held a shared task on QE (Callison-Burch et al., 2012; Bojar et al., 2013, 2014, 2015, 2016, 2017). The shared task has explored QE on several datasets and settings for English–Spanish and English–German language pairs over years.

Little work has been done to study QE for Indian languages. In this work, we focus on four Indian languages: Telugu, Hindi, Gujarati and Bengali. According to a 2007 estimate[1], there are 366

*The author is also a Principal Applied Scientist at Microsoft.

[1] https://web.archive.org/web/20071203134724/http://encarta.msn.com/media_701500404/Languages_Spoken_by_More_Than_10_Million_People.html

million Hindi speakers (across five countries), 207 million Bengali speakers (across four countries), 69.7 million Telugu speakers (across four countries), and 46.1 million Gujarati speakers (across eight countries) worldwide, denoting the importance of our choice of these datasets. While English is a West Germanic language that originated from Anglo-Frisian dialects, Hindi, Bengali and Gujarati are Indo-Aryan languages[2], and Telugu is a Dravidian language[3].

Indian languages are relatively free word order languages and morphologically richer when compared to English. Additionally, some Indian languages, for example Telugu, are highly agglutinative. In comparison with English, Hindi has approximately twice as many vowels and consonants. Although Hindi has tenses similar to those used in English, there is a lack of correspondence in their use to express various meanings. Gender and status relations between speakers causes morphological changes in Hindi words, unlike English. Compared to English, Bengali uses onomatopoeia extensively, and so one has to convey that through particular adjectives and adverbs. Besides these differences, there are some phrases, idioms and compound words in English which do not have equivalents in Indian languages due to significant cultural differences.

Because of the differences in the characteristics of the languages involved, existing methods for QE may or may not be effective for all language pairs. We experiment with multiple datasets in different language pairs, each involving English and an Indian language, to study the effectiveness of various models on these datasets.

In addition to the different characteristics of Indian languages, many of these languages are resource-scarce, from a Computational Linguistics perspective. Linguistic resources like dependency parsers or semantic role labelers are not available for most languages we use in this paper. Additionally, large amount of manually annotated data, such as parallel corpora are also difficult and costly to obtain. Hence, in this work, we try to minimize dependency on external large datasets, especially ones which require manual annotation. We hope that the QE accuracy can be further improved using such extra information, and plan to explore it as future work.

[2]https://en.wikipedia.org/wiki/Indo-Aryan_languages

[3]https://en.wikipedia.org/wiki/Dravidian_languages

To study QE for Indian languages, we also introduce five datasets, for four different language pairs. One dataset, *news.gu*, described in Section 3.2 has been prepared by manually post-editing MT outputs. The other four datasets, described in Section 3.3 make use of existing parallel corpora to create datasets for QE. All datasets are prepared using Neural Machine Translation (NMT) API provided by Google Translate[4]. To the best of our knowledge, we are the first to study QE when using the NMT system.

In this paper, we evaluate the effectiveness of various state-of-the-art systems (proposed for other language pairs) including the winning system of the WMT17 shared task on various Indian language datasets. We also propose and evaluate multiple neural network models for QE. Finally we show that one of our proposed models *CNN.Combined*, described in Section 4.2.2, gives best results on most Indian language datasets. Our major contributions through this paper are as follows.

- Introduction of a manually post-edited QE dataset for English–Gujarati language pair and four other datasets prepared using parallel corpora.
- Proposal of multiple neural network architectures for QE, of which the *CNN.Combined* model is shown to work best for most Indian language datasets in our experiments.
- Evaluation and comparison of several methods of QE on multiple datasets including the WMT17 English–German dataset.

The rest of the paper is organized as follows. We describe related work in Section 2. Section 3 describes the datasets used for the experiments. Section 4 describes different methods and proposed models used for our experiments. Section 5 contains a few notes about the experimental settings. Section 6 provides analysis and related discussions. Finally, we conclude with a brief summary in Section 7.

2 Related Work

Related previous work on translation quality estimation can be organized into two broad kinds of

[4]https://translate.google.com/

approaches: manual feature engineering based approaches, and neural networks based approaches. WMT12-17 shared task on QE (Callison-Burch et al., 2012; Bojar et al., 2013, 2014, 2015, 2016, 2017) has recorded the overview and progress of the field over years.

2.1 Manual Feature Engineering based Approaches

Many previous studies on QE were predominantly based on feature engineering. Manual feature engineering can be costly, especially because it needs to be done for each language pair separately.

For Indian languages, few studies have been done, predominantly for English–Hindi language pair. Most of the approaches, most recently Joshi et al. (2016), are based on manual feature engineering, and traditional classification methods. We show in our experiments, that the neural network based models perform significantly better for all language pairs and datasets.

2.2 Neural Network based Approaches

In recent years, many deep learning methods have also been proposed for QE. Patel and Sasikumar (2016) proposed the use of Recurrent Neural Network Language Modeling (RNN-LM) to predict word-level quality labels using bilingual context window proposed by Kreutzer et al. (2015). Several other neural models also use the bilingual context window approach to compose the input layer, which takes the target word and the aligned source word and their contexts as input (Martins et al., 2016, 2017a, 2017b). These models, however, require word alignment information from the MT system or need to align the words using some external parallel corpora. Since our datasets are prepared using neural MT systems, we do not have alignment information from MT system. Additionally, we do not have enough resources to create external word-aligners for each language-pair. As a result, we do not include systems that need word alignment information in our experiments.

Kim and Lee (2016a), Kim and Lee (2016b), Kim et al. (2017a) and Kim et al. (2017b) have studied and proposed different end-to-end neural network based models, primarily based on predictor-estimator architecture. We compare with the architecture described by Kim et al. (2017a) in our experiments. The architecture is explained in Section 4.1.2.

Dataset	**Target Language**	**Train**	**Dev**	**Test**
wmt17	German (de)	23,000	1,000	2,000
news.gu	Gujarati (gu)	4,489	561	562
ilci.gu	Gujarati (gu)	40,000	5,000	5,000
ilci.hi	Hindi (hi)	40,000	5,000	5,000
ilci.te	Telugu (te)	40,000	5,000	5,000
ilci.bn	Bengali (bn)	40,000	5,000	5,000

Table 1: Target Languages and the Number of Sentence Pairs in each Dataset

Paetzold and Specia (2017) propose a character-level Convolutional Neural Network (CNN) architecture combined with engineered features. The system is comparable to our proposed work in two ways: 1) They do not use any external data or resources. 2) They also use a CNN-based architecture for QE. However, the final architectures are significantly different. Their best system, *SHEF/CNN-C+F*, is explained in Section 4.1.3.

3 Datasets

We used six different datasets for five different language pairs for our experiments. Source language is English for all the datasets. All datasets are split into the typical train, development and test sets. Table 1 shows the target languages and sizes of all the datasets. We describe these datasets in detail in this section.

3.1 WMT17: English-German Dataset

We use the English–German dataset released as part of the WMT17 QE Shared Task (Bojar et al., 2017). The dataset contains text from the Information Technology domain, translated from English to German using a statistical MT system and post-edited by professional translators. The dataset contains source sentences, MT sentences and post-edited sentences, along with Human-targeted Translation Edit Rate (HTER) scores (Snover et al., 2006) for each sentence pair.

Translation Edit Rate (TER) is computed as the minimum number of insertion, deletion, substitution and shift operations needed to be done on MT sentence to match a reference sentence, normalized by the length of the reference sentence. The way the HTER differs from TER is that for HTER, there is no pre-decided reference sentence. There is a human in the loop. The human expert generates the targeted reference by editing the system hypothesis, until it is fluent and has the same meaning as the original source sentence. We use the

HTER scores reported as quality scores for this dataset. The dataset contains 23,000, 1,000 and 2,000 sentences in the training, development and test sets respectively.

3.2 news.gu: English-Gujarati Dataset

We introduce a new QE dataset for the English–Gujarati language pair, prepared using the workbench published by Jhaveri et al. (2018). News articles from various sources were translated to Gujarati from English using the Neural Machine Translation (NMT) API provided by Google Translate and post-edited by one professional translator (different from the authors), who is also a native Gujarati speaker, over a duration of two months. The quality scores, HTER, were computed using the *tercom 0.7.2*[5] tool. The dataset contains a total of 5612 sentences, which was split randomly into training, development and test sets of sizes 4489, 561 and 562 sentences respectively.

3.3 ILCI Parallel Corpora

A parallel corpora for many Indian language pairs, including English has been released by the Indian Languages Corpora Initiative (ILCI)[6] (Choudhary and Jha, 2014). We use the parallel corpora of the health and the tourism domain, having 25,000 sentences for each of the domains for each language pair. We prepare the QE datasets using this for translation from English to four Indian languages, namely, English–Gujarati, English–Hindi, English–Telugu and English–Bengali.

To use the parallel corpora for the QE task, we obtain translations using Google Translate[7] from English to all the target languages. We computed the quality scores as the TER between the MT output and the reference sentences using *tercom 0.7.2*.

The datasets contain a total of 50,000 sentences each, which was divided randomly into training, development and test sets of sizes 40,000, 5000 and 5000 sentences respectively.

4 Models for Translation Quality Estimation

This section describes various models used for experiments and evaluation. We first discuss the baseline models in Section 4.1 and then the proposed models in Section 4.2.

[5]`http://www.cs.umd.edu/~snover/tercom/`
[6]`http://tdil-dc.in`
[7]`https://translate.google.com/`

4.1 Baseline Models

In this sub-section, we discuss previously proposed models for QE and their variations. Section 4.1.1 describes baseline model based on Support Vector Regression (SVR). Section 4.1.2 describes *POSTECH.two-step* and *POSTECH.multi-task* models. Section 4.1.3 describes the *SHEF/CNN-C+F* model.

4.1.1 SVR Baseline

The official baseline for WMT17 QE shared task is a Support Vector Regression (SVR) (Drucker et al., 1997) model trained with 17 features for the task. Some of these features use external data such as language models or word alignments trained on large parallel corpora. These features were adapted to use whatever scarce resources are available for our set of target languages as follows. Two features requiring word alignment tables were removed. No external data was used to compute the language models or n-gram counts. Additionally, a few features were added such as, average target token length and depth of parse tree of source sentence. The source parse tree were computed using Stanford CoreNLP toolkit (Manning et al., 2014), this was possible as all the datasets have English as the source language. We call this model *SVR.baseline*.

4.1.2 POSTECH Approaches

POSTECH's participation was the winning system at the WMT17 shared task, which uses a predictor-estimator architecture, many variations of which have been studied and proposed by Kim et al. (2017a), Kim et al. (2017b), Kim and Lee (2016a) and Kim and Lee (2016b). We follow the architecture described by Kim et al. (2017a) for this work.

Kim et al. (2017a) describe a two-step end-to-end neural QE architecture, called predictor-estimator architecture. The predictor-estimator architecture consists of two types of neural network models: 1) word predictor, which is trained on parallel corpora, i.e. using source and reference translations. 2) quality estimator, a neural regressor, trained on QE data.

The first model, word predictor, tries to predict each word in the target sentence using the source sentence and the remaining target sentence as context. They propose an RNN encoder-decoder (Cho et al., 2014; Bahdanau et al., 2014) model based word predictor, which uses bidirectional RNN in encoder as well as decoder to use the source sentence information as well as the entire left and right context of target sentence to predict each word.

The estimator part, then, extracts a *quality estimation feature vector* (QEFV) for each word in MT sentence using internal network connections of the word predictor network. For sentence-level QE, the QEFVs are then passed to bidirectional RNN to obtain a summary vector, which, then, is passed to regression layer which generates quality score for sentences.

We define two variations of the model for our experiments: *POSTECH.two-step* and *POSTECH.multi-task*.

POSTECH.two-step trains the two models, word predictor and quality estimator separately as described by Kim et al. (2017a). Input to the word prediction step is source and reference sentences, and the outputs are the predicted words. Whereas, the quality estimator takes source and MT sentence as input and outputs quality score for the sentence. No external parallel corpora have been used for pre-training the word predictor as it is not available for most of the language pairs we work with.

The main idea of POSTECH system proposed by Kim et al. (2017a) is to take advantage of pre-training of word predictor using large external parallel corpora. Since we do not use any external corpora, we propose a variation of this model, which jointly learns both, word predictor and quality estimator, in a multi-task setting. We call this model *POSTECH.multi-task*. The inputs to this model are the source and MT sentence, and the outputs are predicted words and quality score.

Recently, Kim et al. (2017b) proposed single-level and multi-level stack propagation based learning for the two steps. We experimented with single-level stack propagation, as we do not have necessary training data for all sentence, word and phrase level QE, which the multi-level model requires. In our experiments, we did not see any significant improvement across datasets between single-level stack propagation (Kim et al., 2017b) and two-step learning (Kim et al., 2017a).

4.1.3 SHEF/CNN Approach

Paetzold and Specia (2017) propose an architecture that combines engineered features and character-level information using deep Convolutional Neural Networks (CNN) and Multi-Layered Perceptrons (MLP). The model *SHEF/CNN-C+F* has three parts, sentence encoders for source and MT sentence, MLP for engineered features and a final layer to combine both and generate quality scores.

The sentence encoder takes the sequence of characters as input, and converts it to a sequence of character embeddings. They stack four pairs of convolution and max-pooling for each window size. Each stack is applied to character embeddings in parallel, and later flattened and concatenated to get a sentence vector. Two different encoders, each for source and MT sentences are created. The encoded source and MT sentence are then concatenated with the encoded features, which are obtained by applying MLP on engineered features. A final layer is applied on the concatenated vectors, which predicts the quality scores.

4.2 Proposed Models

In this section, we discuss our proposed neural architectures for QE. Section 4.2.1 describes two proposed RNN-based models: *RNN* and *RNN.summary-attention*. Section 4.2.2 describes the proposed CNN-based models: *CNN.Siamese*, *CNN.Combined*, and *CNN.Combined.no-features*,

4.2.1 Recurrent Neural Network (RNN) Approaches

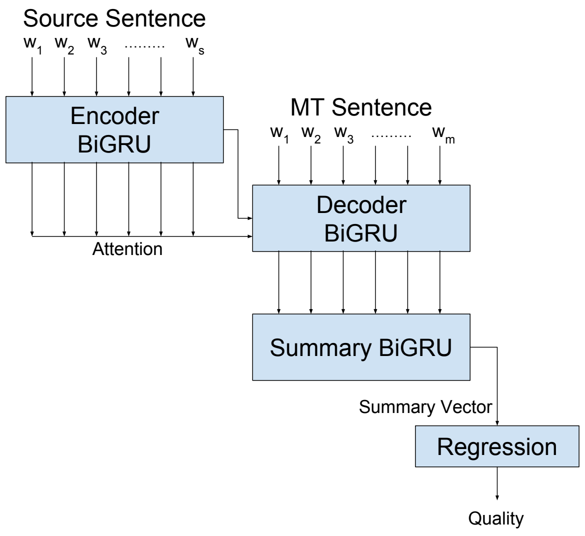

Figure 1: Architecture of the *RNN* model

The POSTECH architecture, described in Section 4.1.2, takes advantage of the pre-training of word predictor on large external parallel corpora. Since no such datasets are easily available for most language pairs in our case, we propose a simplified version of POSTECH removing the word prediction step, and simplifying the QEFV extraction.

The model takes source sentence and MT sentence as input. A bidirectional RNN encoder, is applied on the source sentence, which gives a fixed size representation, which in turn is used as the initial state for decoder. Decoder is also a bidirectional RNN, with attention over the encoder outputs for each word and predicts a QEFV for each word in MT sentence. The outputs of decoder, QEFVs, are then "summarised" by another bidirectional RNN, to generate a summary vector for the sentence pair. This summary vector is then passed to a regression layer, which outputs the predicted quality score. The predicted quality score is compared with the actual quality scores under the L2 loss function for training the network using backpropagation. Figure 1 shows the architecture of the *RNN* model.

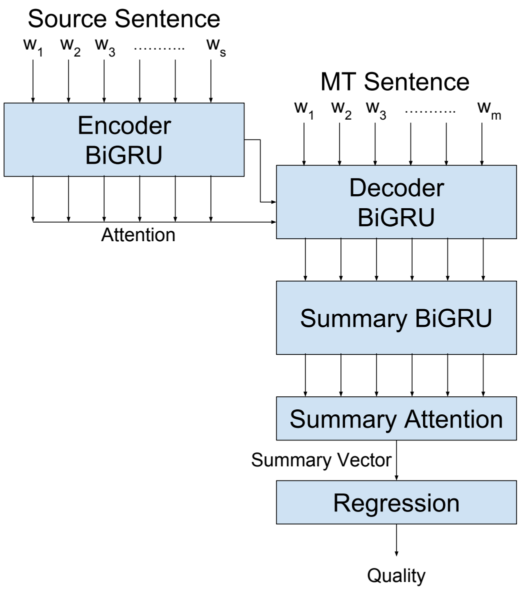

Figure 2: Architecture of the *RNN.summary-attention* model

We also propose a variation of this model, called *RNN.summary-attention*, in which the summary vectors are created using attention mechanism over bidirectional RNN outputs. The QEFVs obtained from decoder are passed to a bidirectional RNN, the outputs of which are then passed to a word attention mechanism, similar to Yang et al. (2016), to get a fixed length summary vector. Attention allows the model to give more importance to certain words in the context while ignoring the others, effectively learning the focus points to better predict the quality score. Figure 2 shows the architecture of the *RNN.summary-attention* model.

4.2.2 Convolutional Neural Network (CNN) Approaches

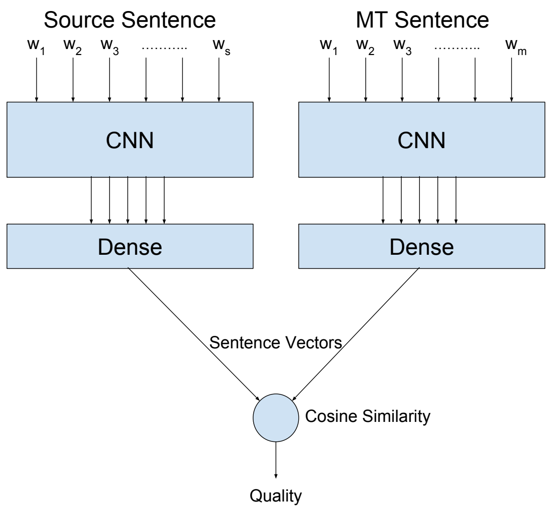

Figure 3: Architecture of the *CNN.Siamese* model

In the basic CNN model, we encode both the source and MT sentence, using CNN-based sentence encoders, similar to one proposed by Kim (2014) for the text classification task. The encoder takes a sentence as a list of word embeddings and applies multiple convolution filters with varying window sizes and applies max-over-time pooling (Collobert et al., 2011) operations for each filter, output of which is then passed to a dense layer, to obtain a sentence vector.

We create two independent encoders (weights are not shared), each for source and target language sentences. The source and MT sentences are encoded using encoder for their respective languages. Finally we take cosine similarity of the two encoded sentence vectors to obtain the quality score. We call this model *CNN.Siamese*. Figure 3 shows the architecture of this model.

We also propose an extension of *CNN.Siamese* model in which the model computes the quality scores in two different ways using the same encoded sentences. One path computes the cosine similarity between the two encoded sentences. The other path concatenates the sentence encodings, optionally along with feature embeddings, and applies a fully connected layer to produce quality scores, similar to *SHEF/CNN-C+F* model described in Section 4.1.3. The final quality score is computed by averaging the two quality scores given by different paths. The architecture of this model is shown in Figure 4. We include two variations, with and without engineered features in our experiments, called *CNN.Combined* and *CNN.Combined.no-features* respectively.

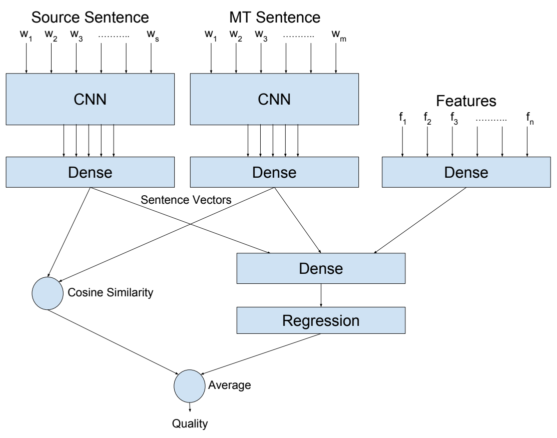

Figure 4: Architecture of the *CNN.Combined* model

For each CNN based model, we tried two initializations for word embeddings: 1) Random 2) Using the pre-trained models published by FastText[8] (Bojanowski et al., 2016), which are trained on Wikipedia[9] for corresponding languages. The experiments, which use the FastText embeddings are denoted by *+fastText* suffix.

5 Experimental Settings

The code used for experiments has been made publicly available at `https://goo.gl/gG9J6f`.

SVR.baseline model is trained using scikit-learn library (Pedregosa et al., 2011). Keras (Chollet and others, 2015), with Theano (Theano Development Team, 2016) is used to implement all the neural network models, including the baselines.

Development set was used for parameter tuning for *SVR.baseline* for each dataset. For neural models, development data was used as validation data while training models, to early stop the training to prevent overfitting.

GRU cells (Cho et al., 2014), with 500 hidden units, are used in RNNs in all the neural network models. Sentences are clipped to length of 100 words and padded with masking. Vocabulary size is limited to 40,000 words for all the experiments. Word embedding size is set to 300.

For all proposed CNN based models, 200 filters of sizes 3, 4 and 5 each were used in the sentence encoders. Sentence vector size was set to 500.

[8]`https://fasttext.cc/docs/en/pretrained-vectors.html`
[9]`https://www.wikipedia.org/`

6 Evaluation and Results

Two types of evaluation are performed for all experiments: 1) Using Pearson's correlation coefficient between the predicted quality scores and the actual quality scores, to evaluate scoring. 2) Using Spearman's correlation coefficient to evaluate the ranking of sentences according to quality.

We also report statistical significance of the results considering *POSTECH.two-step* as baseline, over ten different runs.

Table 2 shows comparison of different models for the scoring task using Pearson's correlation. Table 3 shows comparison of different models for the ranking task using Spearman's correlation.

We find that *POSTECH.two-step* model works best for *WMT17* en–de dataset for both the tasks, but fails to give best results for any other dataset, in the low-resource settings explored in this paper. We also find that the proposed CNN-based models generally work better for Indian language datasets. The better performance of CNN-based models over RNN-based models for Indian languages might be because of the free word order property of Indian languages. CNN does not directly rely on entire sequence and order of words, rather it picks best phrases depending on filter sizes from the sentence without explicitly looking at the order.

Our final model *CNN.Combined*, with or without the use of FastText embeddings works best for four out of five Indian language datasets for the scoring task. For *news.gu* dataset, our combined CNN model, without engineered features, *CNN.Combined.no-features+fastText*, gives the best results. On investigating the relatively low results of the two variants of *CNN.Combined* model on *news.gu*, we found that due to some engineered features and the relatively small size of train set, the combined CNN model with features was rapidly overfitting. The similar model without engineered features, *CNN.Combined.no-features* works as expected and yields the best results on *news.gu*.

Similarly, for the ranking task, the two variants of *CNN.Combined* model outperform all the models for four out five datasets. For the remaining Indian language dataset, *news.gu*, *CNN.Siamese+fastText* model yields the best result.

We also notice that using *fastText* embeddings in CNN based models generally works better com-

Model	**wmt17**	**news.gu**	**ilci.gu**	**ilci.hi**	**ilci.te**	**ilci.bn**
SVR.baseline (original features)	39.98	-	-	-	-	-
SVR.baseline	38.26	20.12	44.67	39.58	44.20	33.65
POSTECH.multi-task	42.44	38.85	45.63	46.51	45.21	38.66
POSTECH.two-step	**50.40**	30.14	49.47	50.23	46.18	44.43
SHEF/CNN-C+F (original features)	40.34†	-	-	-	-	-
SHEF/CNN-C+F	34.22†	29.05	44.32†	39.73†	46.60	34.93†
RNN	41.71†	37.74*	48.56	50.58	49.07*	45.14*
RNN.summary-attention	39.68†	37.30*	48.85	52.59*	49.42*	44.85
CNN.Siamese	44.22†	43.75*	49.29	52.71*	49.56*	44.83
CNN.Siamese+fastText	47.39†	48.60*	51.85*	53.06*	49.69*	45.40
CNN.Combined.no-features	45.83†	43.43*	48.88	52.01*	49.31*	44.68
CNN.Combined.no-features+fastText	48.14†	**49.06***	52.12*	53.17*	49.35*	45.00
CNN.Combined	46.98†	41.51*	52.46*	53.00*	**51.14***	**46.62***
CNN.Combined+fastText	48.96†	46.11*	**52.71***	**53.51***	50.06*	46.08*

Table 2: Results for the Scoring Task, Pearson's Correlation ($*$ and $\dagger$ indicate statistically significantly better or worse ($p < 0.05$) compared to *POSTECH.two-step* respectively)

Model	**wmt17**	**news.gu**	**ilci.gu**	**ilci.hi**	**ilci.te**	**ilci.bn**
SVR.baseline (original features)	43.16	-	-	-	-	-
SVR.baseline	40.65	7.06	42.15	38.44	41.20	31.62
POSTECH.multi-task	44.52	22.46	43.15	44.43	42.03	35.69
POSTECH.two-step	**52.06**	19.61	46.85	48.23	42.83	40.94
SHEF/CNN-C+F (original features)	43.37†	-	-	-	-	-
SHEF/CNN-C+F	37.98†	14.89†	42.97†	39.09†	44.39*	32.61†
RNN	43.42†	27.42*	46.00	48.77	46.33*	42.11*
RNN.summary-attention	41.74†	23.21	46.07	50.48*	46.34*	41.90*
CNN.Siamese	46.20†	31.98*	46.48	51.16*	46.05*	41.43
CNN.Siamese+fastText	49.49†	**41.87***	48.34*	51.67*	45.13*	41.27
CNN.Combined.no-features	47.90†	29.81*	46.03	50.37*	45.77*	41.23
CNN.Combined.no-features+fastText	50.10†	41.13*	49.08*	51.78*	45.13*	40.88
CNN.Combined	48.79†	30.70*	**50.21***	51.32*	**47.58***	**44.19***
CNN.Combined+fastText	51.06	38.20*	49.77*	**52.28***	45.90*	42.39*

Table 3: Results for the Ranking Task, Spearman's Correlation ($*$ and $\dagger$ indicate statistically significantly better or worse ($p < 0.05$) compared to *POSTECH.two-step* respectively)

pared to using random embeddings. However, in some cases, especially for Telugu and Bengali datasets, random initialization of embeddings performs better.

Our word-level CNN encoder based Siamese architecture, *CNN.Siamese* model outperforms the *SHEF/CNN-C+F* model, which is a character based deep CNN model, combined with engineered features. We also show that combining the Siamese architecture with MLP based architecture in *SHEF/CNN-C+F*, *CNN.Combined* model, further improves the results.

The RNN based models work comparably or better for all Indian language datasets, but are much simpler and have much lower number of trainable parameters compared to *POSTECH* models. However, the difference between the two RNN based models, *RNN* and *RNN.summary-attention*, across datasets is inconclusive.

In Table 4, we show some examples of scores predicted by our proposed system *CNN.Combined+fastText* and the baseline (*POSTECH.two-step*) system, along with source, MT and reference sentences and actual quality scores. Note that across examples with low to high quality scores, our method can accurately predict the quality score much better than the baseline.

Dataset	Source sentence	MT sentence	Correct sentence	Base-line	Our model	Actual TER
news.gu	Every year , loud sound from firecrackers causes stress , terror and even death in strays and birds .	દર વર્ષે , ફટાકડાથી ઘોંઘાટવાળા અવાજ તણાવ , આતંક અને ભટકતા અને પક્ષીઓમાં મૃત્યુ પણ થાય છે .	દર વર્ષે , ફટાકડાથી સર્જાતો ઘોંઘાટ પ્રાણીઓ અને પક્ષીઓમાં તણાવ , આતંક અને મૃત્યુ પણ સર્જે છે .	0.03	0.31	0.33
ilci.gu	The total distance of this route is 163 kilometers from Pathankot to Jogindernagar .	આ માર્ગની કુલ અંતર પઠાણકોટથી જોગીન્દ્રનગરથી 163 કિ.મી . છે .	પઠાનકોટથી જોગિન્દરનગર સુધીના આ રૂટનું કુલ અંતર ૧૬૩ કિલોમીટર છે .	0.31	0.75	0.73
ilci.hi	The tombs of Shahjahan and Mumtaz are surrounded by fine meshes .	शाहजहां और मुमताज की मकबरे परिश्रम से घिरे हैं ।	शाहजहाँ और मुमताज के मकबरे चारों तरफ से महीन जालियों से घिरे हैं ।	0.89	0.53	0.50
ilci.te	People of Hindustan , Pakistan , Bangladesh , Egypt do business in Manama Souk .	హిందూస్తాన్ , పాకిస్థాన్ , బంగ్లాదేశ్ , మనామ సౌక్ లోని ఈజిప్టు ప్రజలు .	భారతదేశం , పాకిస్థాన్ , బాంగ్లాదేశ్ , మిశ్ర్ ప్రజలు మానామా సూక్లో వ్యాపారం చేస్తారు .	0.98	0.62	0.62
ilci.bn	There are eight - ten houses of wood in Gejam village .	গাজাম গ্রামের আটটি কাঠের কাঠামো রয়েছে ।	গেজম বসতিতে আট - দশটি কাঠের বাড়ি আছে।	0.58	0.85	0.89

Table 4: Example of output by baseline (*POSTECH.two-step*), compared with our proposed model (*CNN.Combined+fastText*), across all datasets.

7 Conclusions

In this paper, we study the effectiveness of different neural network architectures for QE for Indian languages. We also introduce multiple datasets for the task, which can be used as benchmark for future work in the area. We observe that our proposed *CNN.Combined* model beats the state-of-the-art methods by a significant margin.

References

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural mt by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bojanowski, Piotr, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprint arXiv:1607.04606*.

Bojar, Ondřej, Christian Buck, Chris Callison-Burch, Christian Federmann, Barry Haddow, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2013. Findings of the 2013 Workshop on SMT. In *Proc. of the Eighth Workshop on SMT*, pages 1–44, Aug.

Bojar, Ondřej, Christian Buck, Christian Federmann, Barry Haddow, Philipp Koehn, Johannes Leveling, Christof Monz, Pavel Pecina, Matt Post, Herve Saint-Amand, Radu Soricut, Lucia Specia, and Aleš Tamchyna. 2014. Findings of the 2014 workshop on smt. In *Proc. of the Ninth Workshop on SMT*, pages 12–58, Jun.

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Barry Haddow, Matthias Huck, Chris Hokamp, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Carolina Scarton, Lucia Specia, and Marco Turchi. 2015. Findings of the 2015 workshop on smt. In *Proc. of the Tenth Workshop on SMT*, pages 1–46, Sep.

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Matthias Huck, Antonio Jimeno Yepes, Philipp Koehn, Varvara Logacheva, Christof Monz, Matteo Negri, Aurelie Neveol, Mariana Neves, Martin Popel, Matt Post, Raphael Rubino, Carolina Scarton, Lucia Specia, Marco Turchi, Karin Verspoor, and Marcos Zampieri. 2016. Findings of the 2016 conf. on mt. In *Proc. of the First Conf. on MT*, pages 131–198, Aug.

Bojar, Ondřej, Rajen Chatterjee, Christian Federmann, Yvette Graham, Barry Haddow, Shujian Huang, Matthias Huck, Philipp Koehn, Qun Liu, Varvara Logacheva, Christof Monz, Matteo Negri, Matt Post, Raphael Rubino, Lucia Specia, and Marco Turchi. 2017. Findings of the 2017 conf. on mt (wmt17). In *Proc. of the Second Conf. on MT, Volume 2: Shared Task Papers*, pages 169–214, Sep.

Callison-Burch, Chris, Philipp Koehn, Christof Monz, Matt Post, Radu Soricut, and Lucia Specia. 2012. Findings of the 2012 workshop on smt. In *Proc. of the Seventh Workshop on SMT*, pages 10–51, Jun.

Cho, Kyunghyun, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for smt. In *EMNLP*, pages 1724–1734.

Chollet, François et al. 2015. Keras. `https://keras.io`.

Choudhary, Narayan and Girish Nath Jha. 2014. Creating multilingual parallel corpora in indian languages. In *Human Language Technology Challenges for Computer Science and Linguistics*, pages 527–537.

Collobert, Ronan, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. 2011. Natural language processing (almost) from scratch. *JMLR*, 12(Aug):2493–2537.

Drucker, Harris, Christopher JC Burges, Linda Kaufman, Alex J Smola, and Vladimir Vapnik. 1997. Support vector regression machines. In *NIPS*, pages 155–161.

Escartín, Carla Parra, Hanna Béchara, and Constantin Orăsan. 2017. Questing for quality estimation a user study. *The Prague Bulletin of Mathematical Linguistics*, 108(1):343–354.

Jhaveri, Nisarg, Manish Gupta, and Vasudeva Varma. 2018. A workbench for rapid generation of cross-lingual summaries. In *LREC*, page to appear.

Joshi, Nisheeth, Iti Mathur, Hemant Darbari, and Ajai Kumar. 2016. Quality estimation of english-hindi mt systems. In *Proc. of the Second Intl. Conf. on Information and Communication Technology for Competitive Strategies*, page 53.

Kim, Hyun and Jong-Hyeok Lee. 2016a. Recurrent neural network based translation quality estimation. In *Proc. of the First Conf. on MT*, pages 787–792, Aug.

Kim, Hyun and Jong-Hyeok Lee. 2016b. A recurrent neural networks approach for estimating the quality of mt output. In *NAACL-HLT*, pages 494–498.

Kim, Hyun, Hun-Young Jung, Hongseok Kwon, Jong-Hyeok Lee, and Seung-Hoon Na. 2017a. Predictor-estimator: Neural quality estimation based on target word prediction for mt. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 17(1):3.

Kim, Hyun, Jong-Hyeok Lee, and Seung-Hoon Na. 2017b. Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proc. of the Second Conf. on MT, Volume 2: Shared Task Papers*, pages 562–568, Sep.

Kim, Yoon. 2014. Convolutional neural networks for sentence classification. In *EMNLP*, pages 1746–1751.

Kreutzer, Julia, Shigehiko Schamoni, and Stefan Riezler. 2015. Quality estimation from scratch (quetch): Deep learning for word-level translation quality estimation. In *Proc. of the Tenth Workshop on SMT*, pages 316–322.

Manning, Christopher D., Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*, pages 55–60.

Martins, André F. T., Ramón Astudillo, Chris Hokamp, and Fabio Kepler. 2016. Unbabel's participation in the wmt16 word-level translation quality estimation shared task. In *Proc. of the First Conf. on MT*, pages 806–811, August.

Martins, André F. T., Fabio Kepler, and Jose Monteiro. 2017a. Unbabel's participation in the wmt17 translation quality estimation shared task. In *Proc. of the Second Conf. on MT, Volume 2: Shared Task Papers*, pages 569–574, Sep.

Martins, André FT, Marcin Junczys-Dowmunt, Fabio N Kepler, Ramón Astudillo, Chris Hokamp, and Roman Grundkiewicz. 2017b. Pushing the limits of translation quality estimation. *TACL*, 5:205–218.

Paetzold, Gustavo and Lucia Specia. 2017. Feature-enriched character-level convolutions for text regression. In *Proc. of the Second Conf. on MT, Volume 2: Shared Task Papers*, pages 575–581, Sep.

Patel, Raj Nath and M Sasikumar. 2016. Translation quality estimation using recurrent neural network. In *Proc. of the First Conf. on MT*, pages 819–824, Aug.

Pedregosa, Fabian, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. 2011. Scikit-learn: Machine learning in python. *JMLR*, 12:2825–2830.

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

Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May.

Turchi, Marco, Matteo Negri, and Marcello Federico. 2015. Mt quality estimation for computer-assisted translation: Does it really help? In *Proc. of the 53rd Annual Meeting of the ACL and the 7th IJCNLP (Volume 2: Short Papers)*, volume 2, pages 530–535.

Yang, Zichao, Diyi Yang, Chris Dyer, Xiaodong He, Alex Smola, and Eduard Hovy. 2016. Hierarchical attention networks for document classification. In *NAACL-HLT*, pages 1480–1489.