Machine Translation Quality Estimation and the Impact of Data Volume on Performance

Hai Ly

School of Science

Thesis submitted for examination for the degree of Master of Science in Technology.

Espoo 30.05.2022

Supervisor

Prof. Mikko Kurimo

Advisor

Dr. Sebastian Andersson

Aalto University
School of Science

Aalto University
School of Science

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi
Abstract of the master's thesis

Author Hai Ly

Title Machine Translation Quality Estimation and the Impact of Data Volume on Performance

Degree programme Computer, Communication and Information Sciences

Major Machine Learning, Data Science, Artificial Intelligence **Code of major** SCI3044

Supervisor Prof. Mikko Kurimo

Advisor Dr. Sebastian Andersson

Date 30.05.2022 **Number of pages** 65 **Language** English

Abstract

Machine Translation Quality Estimation (MTQE) is a growing research topic that aims to predict human post-editing efforts without relying on references. This can save time and costs in the post-editing process in the translation industry. Most of the recent research has focused on building MTQE systems to improve model performance with very limited data volumes. This thesis investigates the impact of data volumes on the MTQE performance of the four language pairs of interest: Finnish-English, English-Finnish, Finnish-Swedish, and English-Swedish. The goals are to: 1) inspect data volume impact, 2) inspect source segment length impact, and 3) investigate whether it is possible to reliably detect near-perfect machine translations. OpenKiwi and TransQuest MTQE frameworks were selected for the experiments. To investigate data volume impacts, MTQE models were trained on different sizes of data volume to predict HTER scores. They were then utilized to evaluate on the corresponding held-out dataset with the Pearson and Mean Absolute Error metrics. After that, prediction results from the best model in each language pair were utilized. Source segment length impacts were investigated by grouping different samples based on the number of word tokens in the source segment and analyzing the Pearson scores in these groups. To identify if it is feasible to detect near-perfect machine translations, different threshold values were set on the prediction results to turn them into classification results. The results obtained from TransQuest demonstrated that MTQE models trained with large data volumes yield better and more stable metrics. The models seemed to better predict Pearson scores at short (1-3 word tokens) source segments than other source segment lengths. In addition, depending on the threshold values of HTER, trained MTQE models could predict near-perfect machine translations with high precision and small to medium recall. OpenKiwi was not robust to the chosen data and required additional data filtering. The framework seemed to be less sensitive to data volume changes and more sensitive to data quality than in TransQuest.

Keywords Machine translation quality estimation, data volume impact

Preface

The last time I wrote my bachelor's thesis was already 3 years ago, it was a long process with many challenges but ended with a happy feeling, so this Master's thesis is. I prioritize this thesis higher than many other things because I know at the end, I can be proud of this work. I could not have finished this thesis without a lot of support from my advisor-Dr.Sebastian Andersson, my family (Trang & Tony), and my supervisor-prof.Mikko Kurimo. I owe them big time and cannot thank them enough. Espoo, 30.05.2022 Hai Ly

Espoo, 30.05.2022

Hai Ly

Contents

Abbreviations

BLEU	Bilingual Evaluation Understudy
BERT	Bidirectional Encoder Representations from Transformers
CAT	Computer-Aided Translation
DNN	Deep Neural Network
HTER	Human Target Translation Error Rate
mBERT	Multilingual BERT
NMT	Neural Machine Translation
SMT	Statistical Machine Translation
TER	Translation Error Rate
WER	Word Error Rate
WMT	Yearly Conference on Machine Translation since 2006
XLM-R	Unsupervised Cross-lingual Language Model Pretraining at a very large scale

1 Introduction

The word "translation" has the Latin root word "translatio" that is made up by two subwords "trans" means "across" and "latio" means to "carry". As a very simple definition, language translation is the process of transferring meanings of words, sentences, documents in one language to another language [1]. In the past, translation used to be carried out solely by the human translators but recently Machine Translation (MT) has been introduced to save time and costs, thus increasing translation efficiency. The two main purposes of using MT are to create quickly understandable translation where translation quality is not highly prioritized (assimilation) and to assist professional translators with "good enough" draft translations for editing before publishing (dissemination) [2]. One example of assimilation is the automatic translation feature integrated in modern web browsers such as Google Chrome and Microsoft Edge that instantly delivers translated web pages for users. In this situation, users mainly need to know the main idea of an article written in a foreign language without caring much about grammatical details of that language. An example of dissemination can be found in translation companies where they first translate required documents with MT systems and then let translators edit the MT version documents. However, would it be beneficial to know in advance which MTs to pass to translators? This is an interesting question because when a system can predict the quality of a translation/MT with high accuracy, then good MT may be published as it is, thus improving translation throughput and potentially saving costs for producing translations.

1.1 Problem Statement

The conventional method of estimating Machine Translation (MT) quality depends on human reference translation which is compared with MT to compute some conventional quality metrics such as BLEU. Machine Translation Quality Estimation (MTQE) is the field of studying methods that estimate MT quality without the need of human reference translations. MTQE research has been recently gaining more interest; for example, it has been a research challenge in the yearly WMT conference since 2012 [3].

In the translation companies, MT often serves as a basic, pre-translation reference for the human translator to review and correct since MT quality is not always satisfactory for direct use. This process is called machine translation post-editing (MTPE) or just post-editing. Post-editing is usually expensive and time-consuming, compared to using MT directly, but unavoidable in most professional translations where quality is important. Lucia and Atefeh [4] pointed out that post-editing time of bad quality machine translations can even exceed the time for the human translator to translate a source sentence from scratch. In contrast, acceptable MT may not need any post-editing and can be directly published. Having prior knowledge of

MT quality before the translation could therefore be beneficial in the translation industry [5]. However, it is impossible to estimate the MT quality using e.g. BLEU in translation production since there is no access to a human reference translation before the translation. One possible solution that can address this issue is using MTQE before the post-editing process, which ideally avoids unnecessary post-editing efforts, both when MT is of good quality and when it is of bad quality, hence increasing production throughput and lowering costs [5].

In the WMT20 [3], one shared task is to predict MT quality of the output of MT systems at three different levels: word, sentence, and document which are reflected in three subtasks. The first subtask aims at predicting Direct Assessment (DA) scores at sentence level: scoring is done by professional translators against the source segment but not the reference. The second subtask targets post-editing effort scores (HTER) estimation at both sentence and word levels of two language pairs English-German and English-Chinese with each language pair consisting of 9000 sentences. HTER score is automatically calculated based on the minimum edit distance between the MT and its post-edited version. In the last subtask, participants need to detect document-level translation errors and predict MT scores generated from the word-level errors with a predefined formula.

This thesis focuses on the second subtask of the WMT20's shared task which is to predict HTER at sentence level because the current research on estimating MT quality at document level is still premature and sentence-level MT quality prediction is a good step towards the document-level MT quality prediction. Moreover, in industry settings, using DA is not feasible because it is always expensive and practically slow while HTER can be automatically collected during the normal post-editing production process.

According to the WMT20 [3], MTQE systems are largely language-dependent and the WMT20 data does not cover the four language pairs of interest in this study. Therefore, industry production data in these four language pairs were utilized to train and evaluate MTQE models. In addition, the results of the second subtask of the WMT20's shared task shows that with very little data, the current best models clearly do not yield any (near) perfect correlation, 0.758 Pearson score for English-German pair and 0.664 for English-Chinese pair [3]. Hence, it is worth trying larger data (about 50 to 100 times more than WMT20 data) from industry to investigate whether more data have a positive impact on MTQE performance because using more good data often achieve better performance in data-driven Natural Language Processing (NLP) tasks [6], [7]. According to Bentivogli et al [8], NMT performance in terms of translation quality drops noticeably when long sentences are fed into the system. Therefore, it is also worth investigating the impact of the source segment lengths on MTQE performance. Furthermore, the thesis studies the possibility of detecting (near) perfect MT in the four language pairs since (near) perfect MT can be used as-is, without going through post-editing, in some particular cases. For instance, when validating MTQE in computer-aid translation workflow, Alva-Manchego et al.

[5] show that setting proper threshold values of their QE scores (similar to HTER) can lower costs and save time.

Hence, this thesis studies three research questions:

1. What is the impact of data volumes on MTQE performance in the four language pairs: Finnish-English (fi-en), English-Finnish (en-fi), Finnish-Swedish (fi-sv) and English-Swedish (en-sv)?

2. What is the impact of source segment length on MTQE performance in the four language pairs?

3. Can near-perfect MT be reliably detected in the four language pairs?

1.2 Structure of The Thesis

This thesis is structured as follows: Chapter 2 provides the background. Chapter 3 presents the methodology to solve the problem statements in four main points: data volumes, chosen MTQE framework, MTQE performance metrics, and MTQE results analysis. After that, Chapter 4 shows the detailed implementation of the experiments that were utilized to answer the research questions. Chapter 5 presents the experiment results. Finally, Chapter 6 concludes the thesis.

2 Background

This Chapter reviews the background literature in MT, post-editing, and MTQE with the purpose to construct understanding of how an MT system works, the importance of post-editing, and how to evaluate and estimate MT quality.

2.1 Machine Translation (MT)

Machine Translation or automatic translation already has a long history since the 9th-century when a cryptographer, named Al-Kindi, developed many techniques that utilized cryptanalysis, frequency analysis, probability, and statistics to solve systemic language translation [9]. However, it was not until the proposal of using computer-based MT in 1947 that Andrew Booth and Warren Weaver met and discussed the idea of using computers to perform some natural languages translation. A few years later, the first MT conference (at MIT) was organized by YehoShua Bar-Hillel in 1952. As a result, everyone believed that human intervention before or after the automatic translation was essential to creating high-quality translation outputs.

Two years after that, there were some experiments carried out in Georgetown-IBM which showed the very first human attempt in MT with only 250 words and six grammar rules. This marked an initial success in the field of MT and hence gave the green light to continue funding research because of high expectations in this research field. Even though there was a high expectation in the 1950s, researchers soon realized the complexity of linguistic problems and MT research progress did not meet the expectations. Consequently, this led to the formation of the Automatic Language Processing Advisory Committee (ALPAC) to monitor the research situation, resulting in funding reduction for MT research [10]. However, alongside with the ALPAC report, many tools were invented such as Systran and METEO, which are both rule-based MT systems.

The following decades (the 1980s and 1990s) saw a rising trend in MT quality with many newly invented better toolkits (Trados, EUROTRA, and CAT tools) and more research in Statistical Machine Translation (SMT). Google launched Google Translate in 2016, a free phrase-based translation service that can instantly translate words, phrases, and even web pages between English and more than 100 other languages. It immediately gained popularity, and it currently has over 500 million daily users. One year later, Koehn et al. [11] introduced another phrase-based SMT toolkit, Moses, which is open-source and supports training, tuning statistical models.

It is worth mentioning that already in 2013, Nal Kalchbrenner and Phil Blunsom had an idea of using Recurrent Continuous Translation Models (a class of probabilistic continuous Translation models) for MT which utilizes the Convolutional

Sentence Model to encode the given input sentence (source language) into a continuous vector and obtains the output sentence (target language) by decoding the state vector using Recurrent Language Model [12]. However, using RNN when parsing long sentence inputs has one drawback which is the exploding/vanishing gradient issue.

One year later (2014), Sutskever et al.[13] and Cho et al.[14] tackled the problem by using sequence to sequence (seq2seq) learning in the Long Short-Term Memory (LSTM) architecture. Especially Cho et al. also introduced the gate mechanism to allow more control on memory. Since then, more improvements have been made in NMT; and the current state-of-the-art architecture is Transformer which first appeared in the paper "Attention Is All You Need" [15] in 2017. In 2018, Marcin et al. [16] introduced Marian NMT, an NMT system written in C++ which has later gained popularity. The MT data used for experiments in this thesis also comes primarily from Marian NMT system.

2.2 Neural Network

An Artificial Neural Network (ANN) refers to a network of artificial neurons (nodes) that is inspired by powerful human brains. Each neuron serves as a computing unit that is connected to other neurons via edges (synapses) and there are weight values that are associated with each edge. These weights are known to reflect how much each neuron affects others[17]. Modern Neural Network nodes and edges are already complex and large but neurons and synapses in human brains have have far more complex structures and are larger in size: 10^5 and 10^{11} respectively.

A simple ANN consists of three main layers: input layer, hidden layer(s), and output layer, depicted in Figure 1 below. The input layer is responsible for receiving external data in some form such as image, audio, or text. Each neuron in the network is capable of accepting many inputs and generating a single output which again serves as input for other neurons. Under the hood, each node computes the output signal given the input signals by the weighted sum of all inputs plus a bias term. The obtained weighted sum is then "activated" by some activation functions [18].

$$\sigma(w^T x) = tanh((w^T x + b) \tag{1}$$

Equation 1 shows the tanh (non-linear) function on the weighted sum. w is weighted matrix, x is input vector, b is bias vector.

The information flow in Figure 1 is just a forward pass that gives the neural network information to study but does not yet instruct it how to learn new things. Speaking at the machine level, "not yet learning" means weight matrices are not optimized and it is quite common that their values are randomly chosen from a uniform distribution [19].

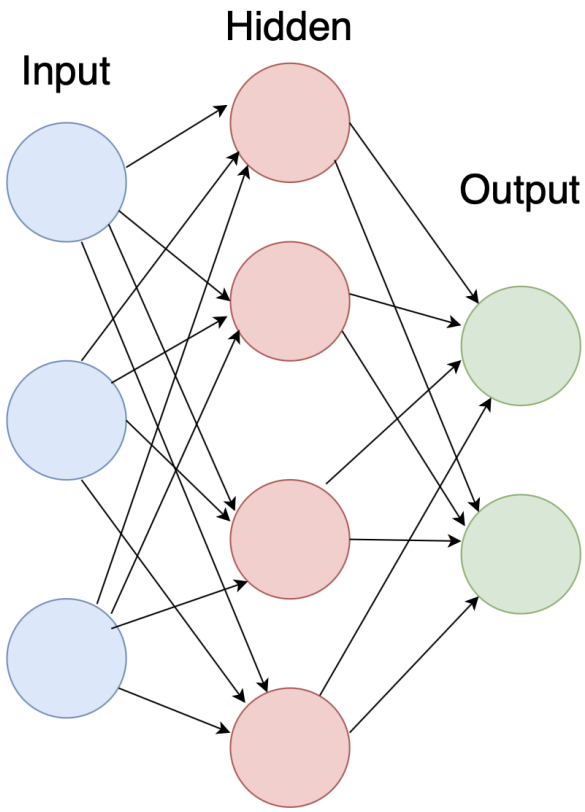

Figure 1. A Simple Artificial Neural Network

To find the optimal weights, a loss function is used to compute errors/losses between predicted and true values of a network. Then through training a neural network, weights are repeatedly updated. Although many learning/optimization algorithms exist, back-propagation with the Gradient Descent algorithm is the most straightforward and simple one, which was proposed by Linnainmaa in his Master thesis in 1970 [20]. In back-propagation training, training data are iteratively fed into the network.

The data pass through many layers until they get transformed into predicted values. Gradient Descent algorithm computes gradients/slopes of the loss function with respect to each weight in the hidden layer to know in which opposite direction the weights should be updated. There is also a learning rate scalar to define how fast to update the weights. The errors keep "propagating" back from the middle layers to the initial input layer by using the chain rule to approximate gradients.

Figure 2 illustrates back-propagation algorithm, the red arrows depict losses that are propagating back from the output layer to the input layer. Let's denote $\mathcal{L}$ as Loss function, $f1$ and $f2$ as learned functions, h as hidden node output. To back-propagate loss, it is necessary to compute the derivatives with respect to the model parameters (θ, w). The learning process moves on until some prior given

conditions satisfy. For example, when training loss converges or training reaches maximum allowed epochs (number of times that data is fed into the network).

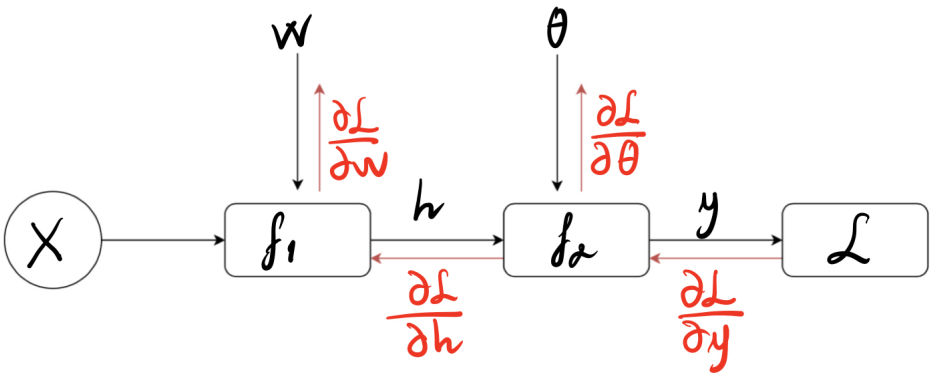

Figure 2. Back-propagation in a simple multi-layer neural network

2.3 Neural Language Models

The statistical approach in language model such as tri-gram assumes Markov chain of short window slide among the sequence of words to learn the joint probability. Nevertheless, this approach suffers from the curse of dimensionality problem because when testing a model with unseen data, there is a high chance that the learned word sequence is different from test data [21]. To tackle this problem, Bengio et al. [21] published a paper in which they mentioned distributed representation (in some applications, it is often called word embedding) approach. This constructed the language model by representing word sequences into a vector space where similar words are positioned "near" together and then processing to train these word embedding vectors with a neural network to obtain the probability distribution of word sequences. Besides that, the approach obtained word generalization since even an unseen sequence of words could be generalized based on similar words that have a "close" represented value in a continuous vector space. Based on the research results, distributed representation learning later takes advantage of long contexts. For example, there was the implementation of Recurrent Neural Networks By Mikolov et al [22] in 2010.

Figure 3 illustrates the idea of training a sequence of words to predict the next word by using the neural network. Even though the vocabulary size of a corpus may be very large, there is no need to select a similarly large number of features when initializing each word with a vector of real values in continuous vector space. In the figure, function $C()$ refers to the mapping of every word with index i to corresponding

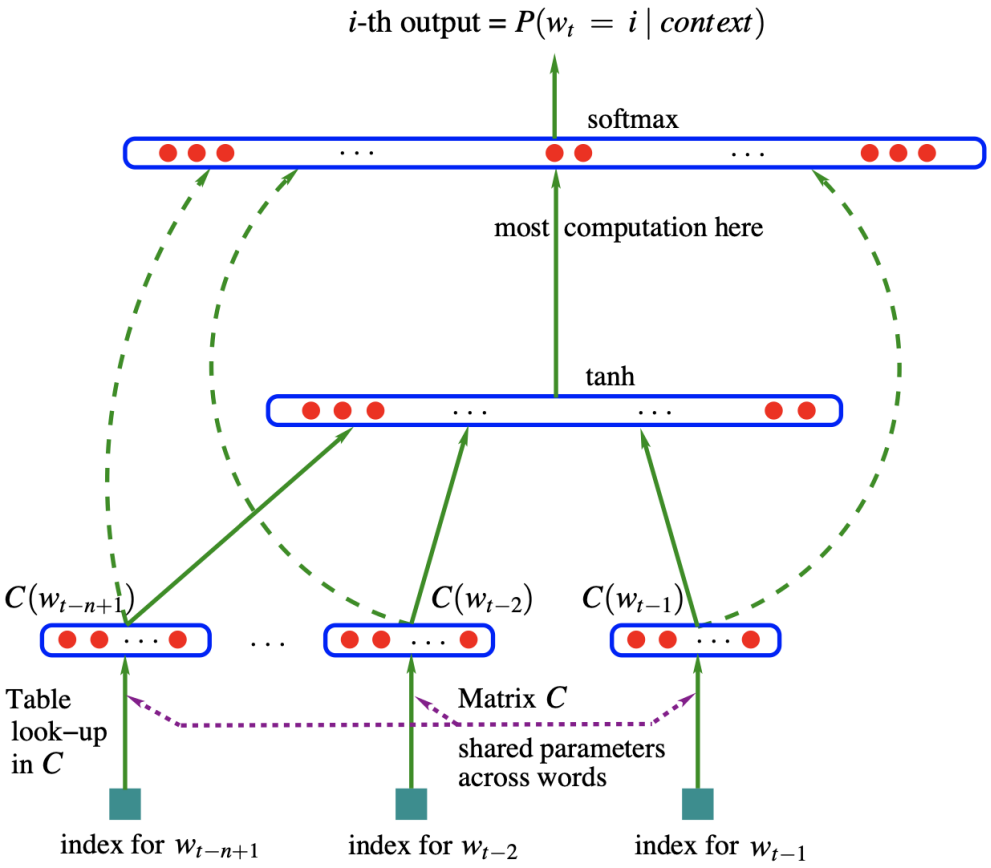

Figure 3. A neural architecture for language model [21]

real vectors $C(i)$. Embedded words $C(i)$ are concatenated into one word embedding vector and this vector can optionally go through the hidden layer where there is the computation of weighted sum and bias adding between weight matrices and the concatenated vector. After going through the hidden layer, the vector is "activated" by the activation function and then again it is computed as the weighted sum in the output layer, A Softmax function will then transform the obtained result from the output layer into values of probability (0-1). There is an optional connection between the input and the hidden layer which is claimed to speed up training [21]; however, final results show that this does not yield better performance.

2.3.1 Recurrent Neural Language Model

As mentioned earlier, Figure 3 represents a feed-forward neural network with concatenated word vectors embedding. It is still a fixed-context word window approach which easily breaks the connection between the first and the last word in a long sequence. To tackle this, RNN has the trick on conditioning previous words for next

word prediction, achievable via feeding extra previous hidden layer every time the network accepts a new word. Thus, the last word in a sequence still depends on the first word [23].

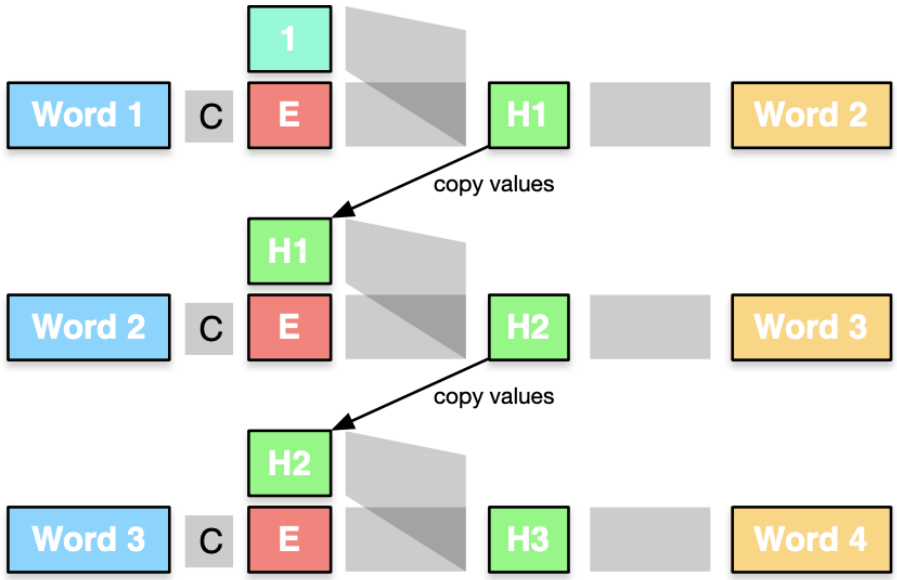

Figure 4. A simple RNN used for language model (Reprinted with the Author's permission) [23]

Figure 4 depicts a simple RNN that predicts the next words conditioning on previous words. For example, $Word3$ is mapped into continuous vector E through function C, usually, it only connects to the hidden layer $H3$, but in the RNN approach, the previous layer $H2$ would be fed together too. The more words and the previous layer feed into the network; the richer the context of the new coming words.

2.3.2 Long-short-term Neural Language Model

RNN dominated in language models for a short time until a new problem emerged, known as exploding/vanishing gradients issue in which RNN models tend to "forget" or shrink memory on some beginning words of the sentence. This is because of the nature of sequential learning where long-term information needs to sequentially flow through many cells and each time gradients are calculated, matrix multiplication can diminish the values so quickly.

According to Köln [23], there are three main problems with RNN. First, the hidden layer has two functionalities: serving as a memory block and being a continuous space representation for previous words. Second, it is sometimes necessary to pay more attention to not the recent previous words, but the words located at the beginning of the sentence, and obviously, no such attention mechanism exists

in RNN. Finally, recently learned information may affect older learned information when training long sentences.

Fortunately, thanks to the Long-short-term memory (LSTM) architecture, those problems can be tackled. The main idea behind LSTM is controlling the memory of the model through each building block/cell which is inspired by the memory mechanism in the computer.

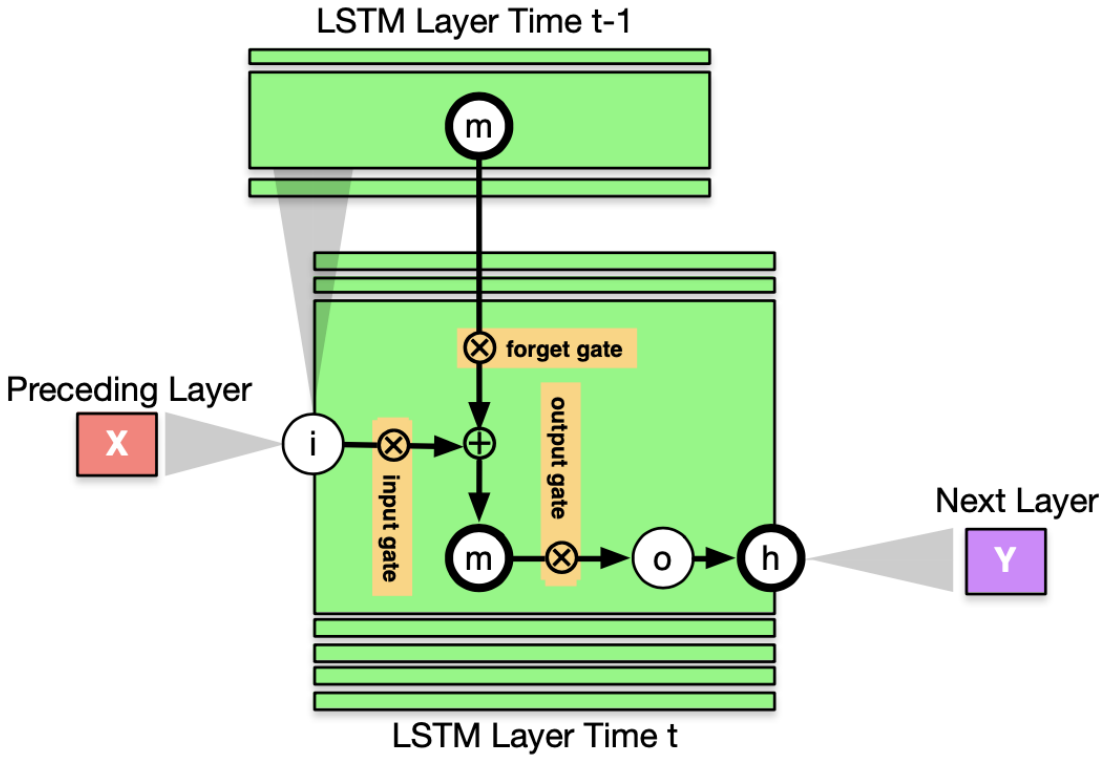

Figure 5. A core building block of LSTM (Reprinted with the Author's permission) [23]

Figure 5 describes a building block in LSTM. There are three different gates that ensure controllable information flow: input gate i, forget gate f, and output gate o. The input gate i controls how the new input information affects the memory state in the current cell. The forget gate f is responsible for regulating how much information from the previous memory state we want to retain or forget, while the output gate defines which part of the cell state to output. The output from the output gate then goes through the activation function h [23].

2.4 Encoder-Decoder Learning in Translation

As mentioned earlier in section 2.1, Kalchrener and Blunsom [12] introduced the first complete NMT system that used CNN for encoder module and RNN for decoder module in 2013. After that, Cho et al. [14] and Sutskever et al. [13] continued to extend the use of encoder-decoder or seq2seq technique to solve the sequential problem in NMT, which achieved very promising results. Indeed, achieved results show that there is a possibility to extend neural language models to solve translation problems.

Figure 6 represents the idea of using encoder-decoder in Sutkskever et al. [13]'s approach, each block in the chain is an LSTM cell.

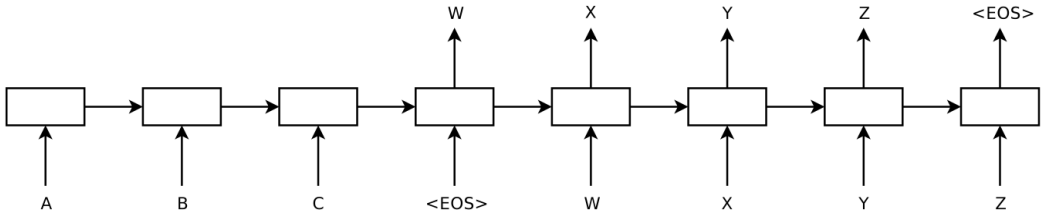

Figure 6. A simple example of using seq2seq learning in MT [13]

In Figure 6, given the source sentence ABC and the target sentence $WXYZ$, the encoder-decoder technique uses two LSTM neural networks with one to encode ABC into a vector and the other decode that vector into target sequence $WXYZ$. Technically, the encoding phase begins with transforming input sequences into embedding vectors. After this, there is a repetitive process of feeding input to each LSTM unit with previous hidden state and a new word embedding vector, this process iterates until it reaches the "EOS" token. The decoding phase, however, is a reversed process of encoding, which turns real-value vectors back to target words. Initially, an LSTM cell accepts the last hidden state vector from the encoder and the embedded vector of the initial word/token "SOS" as its input and; then generates a new predicted vector of the target word and a new hidden state. This generation process continues until reaching the predicted "EOS" token or the maximum length of the target sentence. Interestingly, Sutksever et al. [13] found that feeding input sequence in reversed order seems to boost model performance although they did not have a complete explanation for that.

The following subsections cover literature reviews of major neural network modeling techniques that have been used in NMT such as word embedding, attention mechanism, and specifically, state-of-the-art and de facto standard architecture in NMT: Transformer.

2.5 Word Embedding technique: Word2vec

Regardless of which type of neural networks is used in the seq2seq approach, one of the crucial tasks in building NMT systems is to mathematically transform a set of word inputs into real value vectors (word embedding). There are many word embedding techniques, and the choice is mainly dependent on the domain of data. One of the common data-driven word embedding methods is Word2vec, which has two model architectures: CBOW (Continuous Bag of Words) and Skip-Gram. While the CBOW model learns a set of given words and predicts the next word, Skip-Gram on the other hand has the reverse learning process which is to predict source context words given a target word. According to Mikolov et al. [24], they obtained high-quality vector representation words from a 1.6 billion- words data set in less than 12 hours of training time in the Word2vec approach.

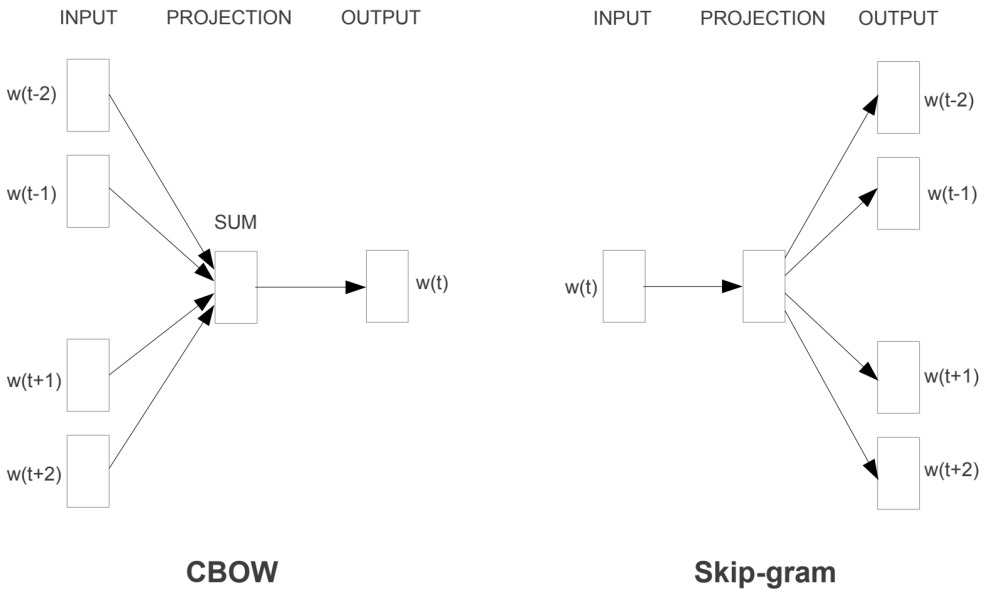

Figure 7. Two model architectures in Word2vec word embedding technique [24]

Figure 7 shows the two mentioned Word2vec architectures. The two core techniques in Word2vec consist of 1) ignoring the non-linear hidden layer to reduce training time, thus allowing more data to be trained and 2) utilizing the final weight matrices between the projection layer and the output layer to represent the actual target vectors of real numbers of input word(s).

2.6 Attention Mechanism and Transformer Architecture

Although LSTM based seq2seq learning seemed to improve NMT, the improvement was not considered much and it was not surprising that the neural network still found it challenging to cope with long sentences, Cho et al. [25] concluded that there is a rapid drop in performance in the case of unseen longer test sentences compared to training sentences in their empirical study. Later, more researchers have identified many better approaches for decoding also longer sentences. In 2016, Bahdanau et al. [26] proposed using an attention model to additionally align input words and output words beside translating target words. In other words, paying attention to some specific parts of the encoded sequence which might be related to each new predicted output word. In their proposed novel architecture, the encoder is a bi-directional RNN used for annotating sequences while the decoder module tries to perform searching through a source sentence with support from the learnable attention model.

At the time of writing, NMT has achieved current state-of-the-art with Transformer architecture. Interestingly, instead of using RNN, LSTM, or gated recurrent unit (GRU), Google's Transformer approach, introduced in [15] concentrates on utilizing attention mechanisms on feed-forward networks (FFN), relieving the heavy computing costs of recurrent networks' sequential computation. More details of the architecture are depicted in Figure 8.

As Figure 8 shows, there are two main modules: the encoder on the left half and the decoder on the right half of the Figure. The encoder module is connected to the decoder module through the Multi-Head-Attention module inside the decoder module. The decoder employs the masked self-attention mechanism to ensure auto-regressive learning for target sentences using attention masks [12]. The encoder is responsible for transforming an input sequence into a sequence of continuous vectors before the sequence is fed into the decoder, together with the output of the decoder at the previous time step. There are many notable techniques used in Transformer such as Positional Encoding, Multi-Head Attention, Masked Multi-Head Attention in decoder, skip connections, layer normalization as well as multiple stacks of both encoder and decoder modules.

The transformer's Positional Encoding (pe) is motivated by the fact that the architecture has neither RNN nor CNN implementation while preserving the order of the sequence is of great importance. Hence, two trigonometric functions sine 2 and cosine 3 are used to encode positions and ensure pe vectors have same dimension as in word embedding vectors before adding them together and applying dropout [15].

$$pe(p, 2i) = sin(p/10000^{\frac{2i}{d}}) \tag{2}$$

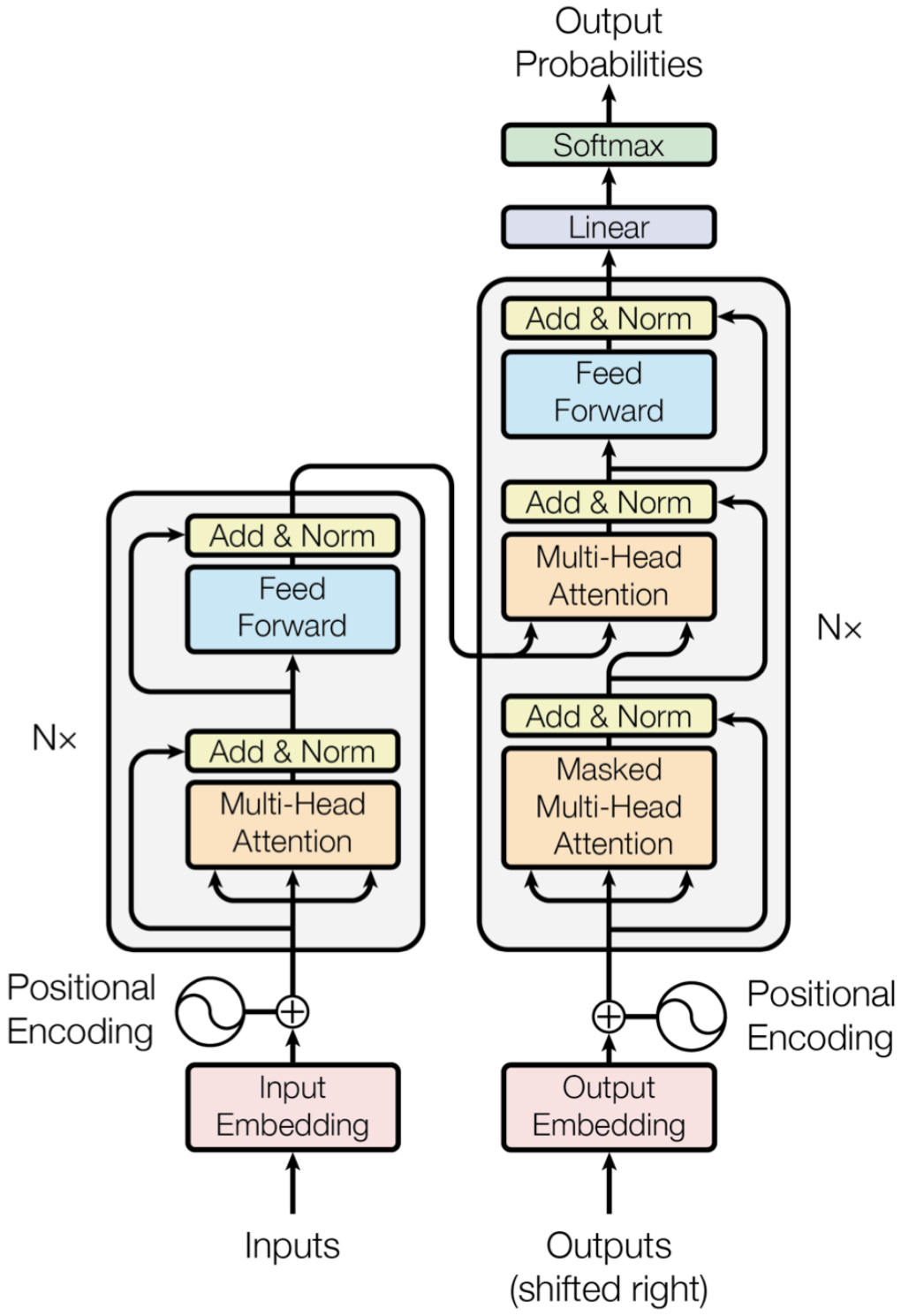

Figure 8. The transformer-model architecture [15]

$$pe(p, 2i + 1) = cos(p/10000^{\frac{2i}{d}}) \tag{3}$$

The right half of Figure 9 illustrates the computation of the Multi-Head-Attention block values by concatenating the scaled dot-product attention results from three matrices: Query Q, Key K, and Value V that are projected into lower-dimensional spaces before feeding the result through a linear layer (Equation 5). The mathematical logic of a single scaled dot-product can be formulated in the equation 4 in which attention is paid to value vj with key kj that is related to query qi.

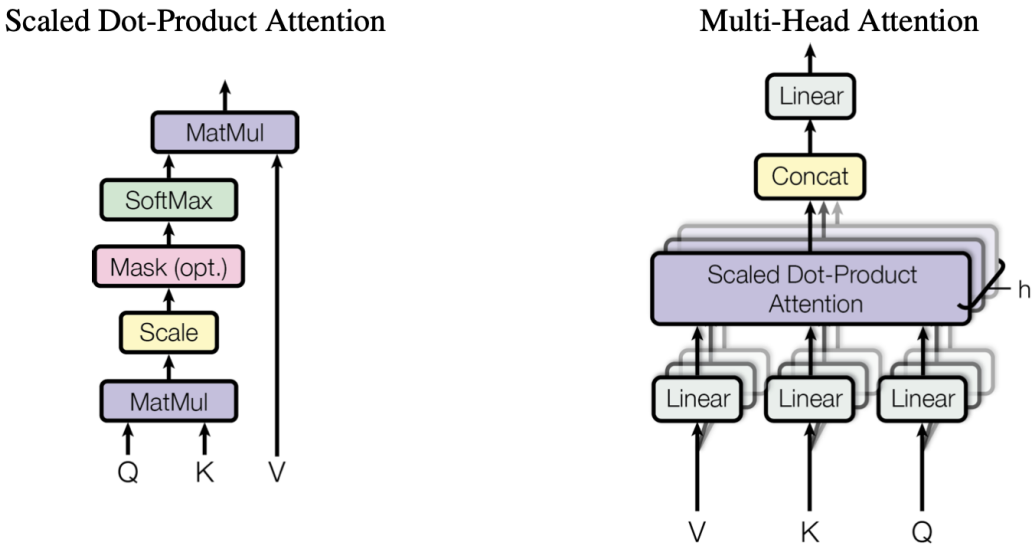

Figure 9. Scaled Dot-Product Attention and parallel running Multi-Head-Attention layers [15]

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{4}$$

where $\sqrt{d_k}$ is the dimension of the key vector k and query vector q .

$$MultiHead(Q, K, V) = Concat(head_1, ..., head_h)W^O \tag{5}$$

where $head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$

and $V \in \mathbf{R}^{mxd_v}, Q \in \mathbf{R}^{nxd_k}, , K \in \mathbf{R}^{mxd_k}$, $head_i \in \mathbf{R}^{nxd_i}, output \in \mathbf{R}^{nxd_k}$

2.7 XLM-RoBERTa Pretrained Language Model

According to Specia et al. [27], [3], XLM-RoBERTa is the large-scale pre-trained model that was used by most of the winning teams in MTQE shared task of WMT20 and WMT21. One of its important features is the ability of dealing with a large number of different languages. XLM-RoBERTa was first introduced by Conneau et al. [28] in 2020, which is based on the RoBERTa architecture appeared in the paper "RoBERTa: A Robustly Optimized BERT Pretraining Approach" by Liu et al. [29].

XLM-RoBERTa is the current state-of-the-art pre-training multilingual language model that was trained on one hundred languages. It is a Transformer encoder model which shows significant improvements in many NLP tasks such as Name Entity Recognition in low-resource SA languages [30] and MTQE [27]

2.8 Post-editing

Although MT quality has recently been improved, it is still far from the target of consistently publishable quality [31]. Post-editing (PE) remains a crucial process in translation industry-standard approach in which a draft version obtained from automatic MT and possibly translation memory (a tool to store and re-use already translated segments) is sent to translators for editing. In practice, although using the translation memory (TM) provides more confidence than a raw MT, in almost all cases, TM does not contain all new segments that need translation. Therefore, TM still cannot replace translators but rather assists them to enhance translation processes.

The main motivation behind PE is to aid or lessen the burden of traditional translation where translators translate source texts from "scratch" by providing them with pre-translations from machines. As a result, this saves labor costs while retaining good quality of translation. Some studies have demonstrated that using good quality MT can boost productivity and vice versa, bad quality MT can even slow down the work of translators, resulting in more costs [32], [33]. The thesis briefly introduces the history of PE, the current state of PE research, and industry translation pipeline in the following paragraphs.

Although there have recently been growing interests in both MT and PE effort research and application, PE is not a new term as it was one of the first applications of MT systems. According to Garcia [34], PE already appeared to be a very hot topic in the late 1950s and early 1960s until the shutdown of research on PE in the US due to the result of the APAC report which was mentioned earlier in the MT history subsection. However, in the 1970s, there were some restoration signals of using MT systems and PE processes in some organizations [34]. Later, Robert [35] argued that PE supported professional translators in improving the number of words they translated per day up to 75%, although this is usually not the case

for all languages and levels of professional translators. Following the growth of PE workflow research, there was the introduction of post-editing guidelines in 2010 by Translation Automation User Society (TAUS) which suggested to both customers and language service providers what to expect and how to utilize post-editing efficiently [31]. TAUS also defined two quality degrees of MT, 'quite sufficient' and 'publishable'. 'Quite sufficient' is described as understandable and accurate enough to carry similar meaning as the source sentences without being perfect grammatically or aesthetically. Publishable quality would be comparable to the standard of human translation, requiring grammatical and stylistic adequacy in addition to being understandable and accurate.

The PE process has indeed proven to be an important and promising topic besides MT. It also raises the challenge of improving the PE process where using an extremely good MT system is a straight-forward solution. In addition to that, assisting the PE process with a good Quality Estimation (QE) system is worth trying and there has recently been some research about the impact of using the QE system on the PE process in industry translation pipelines. One of them is a research carried out by Shenoy et al. [36] in which they explore the influence of the accuracy of a word-level QE system on the PE process and how to present word-level QE information to translators in the most effective way. Consequently, they conclude that existing state-of-the-art word-level QE models are not yet capable of supporting PE. Instead, they experimentally approximate that having at least an 80% F1 score is a minimum requirement to support the PE process. In addition, a visualization that reflects the uncertainty of the QE model is desirable for useful quality levels. Instead of studying the boundary of QE scores and QE information presentation toward PE process. Zouhar et al. [37] conducted an experimental study on the relationship between NMT and the PE process. Specifically, they chose to study MT of English to Czech and its impact on PE. The results show that having a better NMT system would yield better PE (less editing), and BLEU should not be seen as a stable predictor of time and translation output assessment.

Relatively, Automatic post-editing (APE) is an interesting sub-research area of MT, which aims at training a neural network model that can automatically "detect" the errors from MT, then correct them without having to revise the MT structure. In other words, a new separate system is employed to improve MT quality (not to confuse with the MT system that translates source segments). APE is independent of MT models and can be utilized in some domain-specific translations. The limitation of building an APE system is the cost and time needed to generate training data because such systems require access to at least three main data types: source, target, and post-edited version [38].

As mentioned earlier, the PE process is an inevitable part of the industry translation pipeline. Although the translation pipeline can vary in different organizations and translation companies, a typical translation pipeline process includes six steps of which the PE process is one of the final crucial steps as represented in the

following Figure 10.

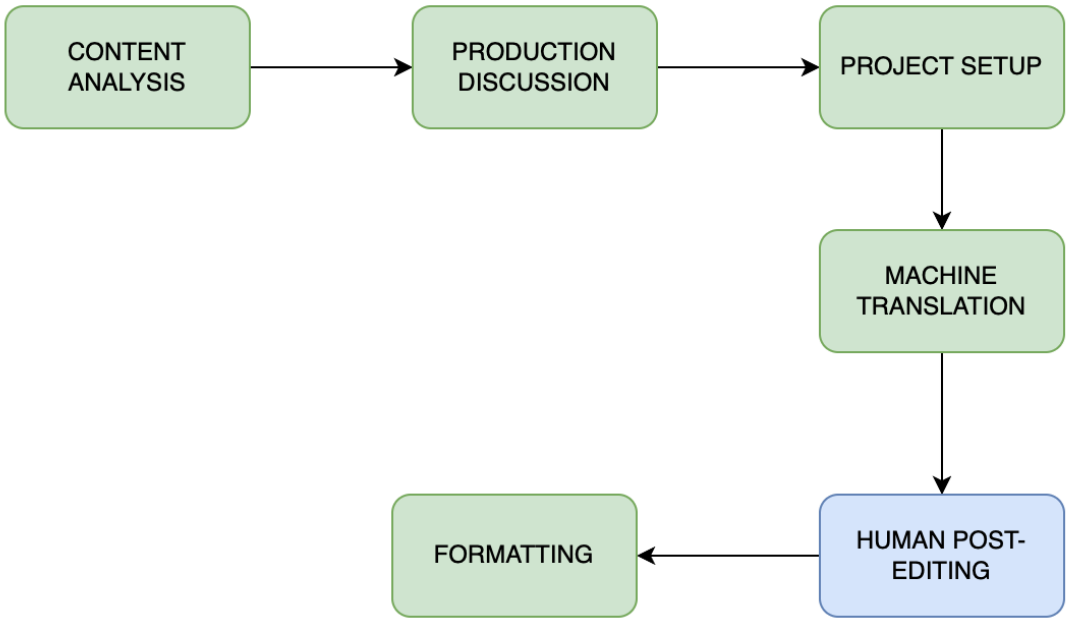

Figure 10. Industry's typical translation service pipeline[39]

In the first step, content analysis, a translation company analyzes customers' documents to obtain some specific information types such as content type, intended use/audience, and MT approach. Sufficiently acquired information in this step supports the company to discuss and set up the processes, costs, timeline, and deliverable plans with its clients in the second step, production discussion. Then, the project managers configure the production framework in the third step to meet requirements agreed by both parties. MT takes place in step 4 where MT engineers make sure the MT system flawlessly produces raw translation output before it is sent to the PE process in step 5. Professional translators who take part in this PE process are called post-editors and they are responsible for revising and correcting MT errors. The pipeline ends at step 6 where the final check on translated documents is done, and it is now in a publishable state, or it is ready to deliver to the corresponding stakeholders.

Since the PE process plays an important role in the standard translation pipeline, more research has been done in MTQE which aims at optimizing the PE process. The following Section 2.9 introduces MTQE and its potential benefits in the translation pipeline.

2.9 Machine Translation Quality Estimation (MTQE)

This section contains three subsections. Subsection 2.9.1 reviews the difference between estimating and evaluating MT output. After that, common MTQE metrics and MTQE data from WMT20, WMT21 are introduced in subsection 2.9.2. Finally, subsection 2.9.3 presents different current methods used in MTQE.

2.9.1 Evaluating MT quality and Estimating MT quality

MT evaluation is a task of evaluating the quality of MT output based on either the assessment of professional translators (manual approach) or pre-defined references (automatic approach). Because the manual approach is expensive and time-consuming, the automatic approach is often used in training and evaluating MT models. In automatic MT evaluation, reference data is used to compared with the MT output and their differences can be measured by many common metrics: METEOR [40], TER [41], NIST [42], chrf [43], and BLEU [44].

According to Scarton et al. [45], the task of estimating the quality of the output of Natural Language Processing (NLP) systems without depending on human references is known as quality estimation (QE). MTQE is one of many applications of QE that estimates MT quality on the fly without access to translation references. Useful information extracted from the MTQE system has been proven to enhance PE workflow, according to Bechara et al. [46]. In their published paper "The Role of Machine Translation Quality Estimation in the Post-Editing Workflow", the team explores the influence of MTQE on the general translation operations as well as the cognitive load on the translators by conducting experiments on the translators' productivity in terms of time and effort (keystrokes). The experiments were designed to capture three different scenarios of using or not using MTQE: translating from scratch, MTQE aided PE, and no MTQE aided PE. The results show that having good MTQE information can increase the PE efficiency and lessen the cognitive load on translators, whereas translating from scratch requires more time and cognitive loads on translators.

2.9.2 MTQE Metrics and Data

Two common metrics that have been recently used in WMT20 and WMT21 shared task on MTQE includes HTER and DA scores.

To measure human post-editing efforts in MT, HTER is one good choice since it can be automatically calculated directly from the MT and its post-edited version. The HTER score range can also be used to decently identify perfect (HTER $= 0$), near-perfect (HTER ~ 0), and bad translations (e.g. HTER > 0.5).

HTER or human-target TER stands for human translation error rate which measures the minimum post editing distance between the MT suggestion and the human post-edited version (based on MT suggestion). HTER specifically emphasizes getting the minimum TER score on one post-edited version but still mostly preserving language fluency and semantics. There is a small difference between HTER and TER: TER is used to measure the error rate between MT and reference translation(s) that were not based on the MT suggestion to be evaluated. Snover et al. [41] show that using HTER reduces the error rate by 33% and HTER is very well correlated with human judgments. Equation 10 formulates a simple computation of HTER.

$$HTER = \frac{\text{Number of edits}}{\text{Number of reference words}} \tag{6}$$

Here the numerator describes possible edits that a human needs to perform on hypothesis/MT, including insertion, deletion, the substitution of single words and one important task called shifting the word orders in hypothesis.

Recent research has shown that the better quality of data a Machine Learning algorithm has, the better results it can produce. Indeed, Valerie and Macro [7] through their experiments on the impact of data quality on Bayesian Network learning algorithm conclude that data quality has a great impact on the results and efficiency. Not very surprisingly, they also prove that inaccurate/bad data has severely affected the Bayesian algorithm. Furthermore, Boulmaiz et al [6] investigate the impact of training data size on the LSTM performances for rainfall–runoff modeling. They show that although their LSTM models achieve acceptable results when the models are trained with nine years of accumulated data, using 12 years of accumulated data clearly improves the results.

As discussed in the introduction and background chapters, having sufficient data for MTQE research can be quite challenging. One of the big contributions of MTQE data to the research community is the MLQE-PE dataset, a multilingual QE and PE dataset. According to Fomicheva et al. [47], this dataset is open source and was given in both WMT20 and WMT21 conferences. It contains up to 10,000 translations per language pair with sentence-level direct assessment, PE effort scores, and word-level good/bad labels. In addition, Fomicheva et al. [47] also contribute the MT models that were used to generate target sentences. Besides that, they provide post-edited sentences and article sources that contain the source sentences in the MLQE-PE dataset.

However, there is a potential issue related to the high correlation of QE scores (both DA and HTER) in some language pairs that has been particularly noticed and noted in two recent papers that summarize the findings of WMT20 and WMT21 shared tasks in MTQE by Specia et al. [3], [27]. There is a surprisingly high Pearson score (0.91) predicted by the top winner systems for the Romanian-English language pair, and they explain that the test data has a large number of "non-related"

translation/low-quality sentences. Hence, it is a trivial task for QE systems to detect low-quality sentences.

DA stands for Direct Assessment, of which the evaluation settings are as follow: annotators provide assessments of the perceived quality of MT. In detail, they rank the quality of MT in a scale of 0-100 and a larger score means a better quality of MT. Evaluation follows the FLORES guidelines [48]: incorrect MTs fall into the range of 0-10; MTs with few correct words but are different from the meaning of the source belong to the range of 11-29; range 30-50 contains MTs with significant mistakes; MTs are understandable and preserve overall meaning of the source will be put into the range 51-69; range 70-90 consists of MTs that have slightly loss semantic meaning compared to the source; range 91-100 includes perfect MTs. DA scores are standardized using z-score.

Evaluating MT quality using DA scores is more expensive in terms of time and costs than using HTER, not to mention that using DA is not feasible for large volumes of data.

2.9.3 MTQE methods

MTQE method used to be extracting features such as the amount of name entities, or prepositional phrases in the source and target segments [49]. Through training QE models based on these features, translation quality can be automatically estimated without human intervention. There have been recently many approaches in solving MTQE problem by using supervised learning QE system to train QE models on the given source, MT, and MT quality scores. However, they share a standard framework which is illustrated in Figure 11.

Given source, target and quality labels (eg: HTER scores), the MTQE system extracts useful features from both source (complexity) and target (fluency) and process training algorithms based on training data [45]. MTQE models once trained can be used to predict labels for unseen data. In the recent years, several MTQE frameworks have been developed, including QUEST++ [50], DeepQuest [51], OpenKiwi [52], TransQuest [53], QEMind [54], HW-TSC [55] and IST-Unbabel [56].

At the time of writing, WMT21 is the latest conference that still organizes the QE shared task at the sentence level. The baseline system is based on a multilingual transformer-based Predictor-Estimator architecture extended with stack propagation method [57] introduced by Kim et al. [58] and Kim and Lee [59]. According to Kim et al., Predictor-Estimator architecture is composed of two models: the word-prediction model and the QE model. The word-prediction model is a probabilistic model based on NMT (RNN-based) and trained on parallel corpora that conditionally takes source and target contexts to predict target words whereas the other model (QE) is a neural model trained on quality estimation data. They

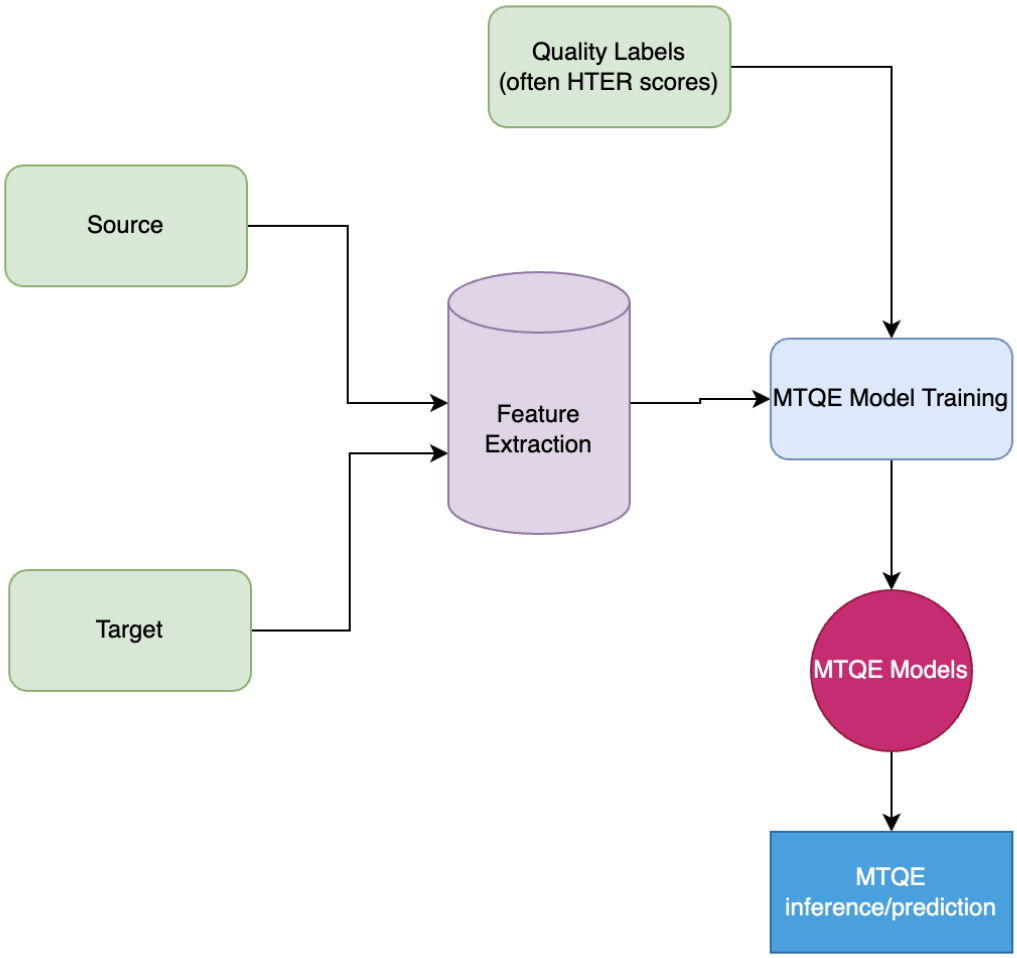

Figure 11. Standard MTQE framework

assume that the predictor is closely related to the estimator; thus, it is very useful that the predictor can transfer approximated knowledge to QE tasks. To solve the transfer learning problem, they implement quality estimation feature vectors from the predictor and then feed them into the estimator. Moreover, to achieve efficient joint learning in predictor-estimator, they utilize the stack propagation learning method which uses a continuous and differentiable link between predicting and estimating tasks hence enabling back-propagation from the estimator to the predictor [57]. The baseline system mentioned earlier has the predictor component based on the XLM-RoBERTa transformer encoder. It is first fine-tuned using ULM fine-tuning [60] on the concatenated train and development segments, and then employed in the entire predictor-estimator model for later independently fine-tuning for HTER predicting task.

According to Specia et al. [27], the two best frameworks in WMT21 shared

tasks of the sentence-level PE effort prediction are HW-TSC and IST-Unbabel. Although HW-TSC is the winner of almost all the given language pairs and TST-Unbabel ranks top in the majority of zero-shot language pairs, they both developed their solutions using the two-stage/ predictor-estimator framework [58] as mentioned earlier with the XLM-RoBERTa encoder for the predictor model. Motivated by the idea of adding the same language addition-translation-sentence (AMT) to the given source language and original translation as one extra criterion [55], the HW-TSC team directly concatenates high-quality translated sentences from a trained NMT system on WMT21 News with the source and original translation before training them under multi-task learning framework. On the other hand, IST-Unbabel team has an ensemble approach of using two XLM-RoBERTa multilingual encoders with stacked adapters [61]. The first is called M1-ADAPT which utilizes XLM-RoBERTa pre-trained model to further fine-tune it on QE data with stacked adapters. The other encoder M1M-ADAPT has one difference compared to M1-ADAPT, they additionally train the XLM-RoBERTa model with out-of-domain data and this approach seems to improve QE performance [56].

Another method for QE on MT is using back-translation on MT output which is translating target sentences back to source sentences and using some automatic metrics to measure the similarity between a source sentence and its back-translation version: the more similarity between them, the better the translation quality . This method is a particularly natural option in such a scenario that there is access to a good MT system with no given references. Inspired by this idea, Moon et al. [62] propose utilizing semantic-level metrics, revisiting the round-trip (RTT) based QE model that can achieve the strongest correlations with human judgments when compared to prior WMT19 quality estimation metric task's submissions.

3 Methodology

This chapter introduces the methodology that was used to answer the proposed research questions. As can be seen in Figure 12, given the four language pairs of interests with their corresponding measurable data sizes, the methodology starts with choosing open-source MTQE frameworks. Once these framework are chosen, it is important to verify whether they work as expected by replicating the published results of these frameworks. When the verification process succeeds, data are processed and packed into different batches for the experiments. Then, models are trained and evaluated in different batches. After that, prediction results are analyzed with three main objectives: evaluating the impact of data volumes on MTQE performance, investigating the impact of source segment lengths in MTQE performance, and analyzing prediction results as a classification problem. The following sections 3.1, 3.2, 3.3, and 3.4 describe in detail the methodology that covers data, chosen MTQE frameworks, MTQE evaluation metrics, source segment length impact, and near-perfect MT classification.

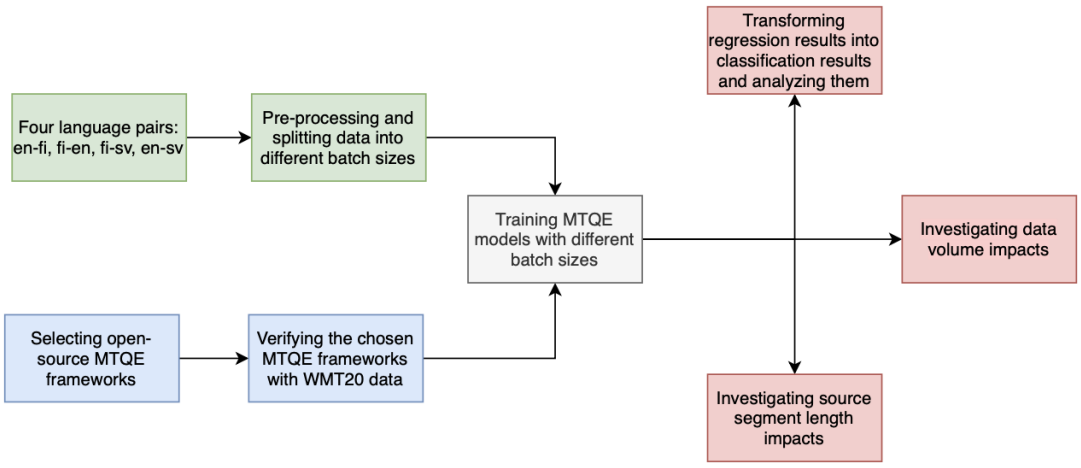

Figure 12. Methodology to measure impact of data volumes in MTQE performance.

3.1 Data

As mentioned in Chapter 2, MTQE methods are generally language dependent, and the MLQE-PE data set does not contain the four language pairs that need to be assessed. Even if the four language pairs of interest existed in the MLQE-PE dataset, perhaps it would be insufficient to achieve high metrics since the maximum size of each

language pair in the MLQE-PE dataset is limited to 10,000 segments. On the other hand, this thesis utilizes the industry data mainly from the public administration domain and is very large in volume compared to the MLQE-PE dataset. In addition, there are also some useful associated metadata that can potentially be used for further data analysis when needed.

In this thesis, a segment refers to a sequence of words in one line which can be e.g. a sentence, a number, a single word or an url. When the thesis mentions tokens, it means the word tokens extracted by splitting a sentence by white spaces.

3.2 MTQE Model Training Framework

Because this thesis does not target at optimizing MTQE models or building a new MTQE framework, the methodology solely relies on open-source MTQE frameworks to conduct experiments. Although several MTQE frameworks have been recently developed, only a few of them are open source. For example, DeepQuest [51], TransQuest [53], and OpenKiwi [52]. Since DeepQuest only supports python 2, there are only two options left. TransQuest won the WMT20's first shared task on MTQE, which is to predict Direct Assessment scores at sentence level given training data in six language pairs, with 9,000 sentences each. It achieves fairly good results also in the HTER scores prediction shared task. TransQuest is built on top of state-of-the-art transformer architecture and contains two main architectures: MonoTransQuest and SiameseTransQuest. TransQuest creators also claim that at the time they conducted their research, it was the first neural-based framework to tackle MTQE for multi-language pairs cases. The second open-source framework-OpenKiwi is a PyTorch-based tool that won 2015-2018 WMT QE shared tasks. It supports training and testing MTQE models both at the word and sentence level. Since 2019, it has been used as baseline systems in MTQE shared tasks.

OpenKiwi and TransQuest were selected because of four reasons. First, both frameworks have been partially or fully used in some submissions of the MTQE shared tasks in WMT20 and WMT21. Second, having two frameworks to investigate the impact of data volumes is better than just relying on one framework. Third, both frameworks are well known for their ease of use and have decent documentation. Last, TransQuest is a neural-based state- of-the-art QE framework that takes advantage of fine-tuned cross-lingual embedding model (XLM-Roberta) which performs better than mBERT in MTQE experiments conducted by Tharindu et al [63]. Both chosen frameworks are first verified with the WMT20 data to ensure that they are usable for the experiments.

3.3 MTQE Performance Metrics

Following the same metrics that have been used in WMT20 and WMT21 for MTQE shared tasks at the segment level, this thesis chose Pearson score and Mean Absolute Error (MAE) to measure the performance of MTQE models. MAE is a popular simple measure of accuracy used in prediction tasks that is good for comparing model performance errors according to Willmott and Matsuura [64]. It is the average absolute difference between each pair (predicted and true value) of a set of observations. In this thesis, observations contain predicted HTER and true HTER values. The equation 7 [65] shows how to calculate MAE.

$$MAE = \frac{\sum_{i=1}^{n} |x_i - y_i|}{n} \tag{7}$$

Where x_i denotes true values and y_i denotes predicted values, n is the number of observations.

The other metric is the Pearson's correlation coefficient (r) for a sample. In the experiments, it will be used to measure the statistical relationship between true and predicted HTER scores. Coefficient r has [-1,+1] range and the positive correlation approaches to +1 while a zero score would mean no correlation at all between predicted and true HTER scores.

$$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}} \tag{8}$$

In equation 8 [66]: x_i denotes true values and y_i denotes predicted values, n is the number of observations. $\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$, $\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$ denote the average of all true values, predicted values of observations, respectively.

3.4 MTQE Results Analysis

To measure impact of the source segment length on MTQE performance, the prediction results are categorized into 30 unique groups based on the ascending number of tokens of the source segments. For example, group 1 contains all one token in length segments, group 30 includes all 30 tokens in length segments. Then, Pearson scores between these groups are analyzed.

The thesis also aims at identifying whether it is possible to detect near-perfect MT with high precision. Prediction results from the best MTQE models in the four language pairs are transformed into binary classification results: one class for near-perfect translation and the other class for non-near-perfect translation.

A threshold value on the HTER score will be set so that any segments that have the HTER scores below or equal to the threshold value will be classified as positive labels while the remaining segments are classified as negative labels. In addition, the HTER threshold is changeable within a specific range to inspect the trade-off between precision and recall.

4 Experiment Implementation

As mentioned earlier in Section 3.2, the initial step of the experiment implementation starts with verifying if TransQuest and OpenKiwi can replicate their results of the HTER scores prediction task when training and evaluating with WMT20 dataset. The results show that both frameworks successfully reproduced the approximately comparable metrics. Therefore, they can be used for the main experiments.

To support answering the research questions, this thesis sets up three main experiments. The first experiment evaluates the impact of data volume on MTQE model performance which is further divided into two sub-tasks:

1. Does having more data yield better metrics?

2. How is the fluctuation of metrics when training with increasing size of data volumes?

The second experiment analyzes the relationship between the Pearson scores and the lengths of source segments (in tokens) of the four language pairs. The third experiment evaluates the MTQE performance in terms of near-perfect MT detection.

4.1 Data Preparation

The data consisted of four csv files from 2018 to 2021 with more than three million segments in four language pairs. Initially, each file was a mix of four language pairs data and contained many meta-data, e.g. customer and delivery date, but for the current implementation, this thesis only focused on source segment, MT segment and human post-edited version.

In Figure 13, the four data files were vertically stacked into one merged dataset and python open-source library pyter [67] was utilized to compute HTER labels for each segment based on MT and post-editing version. In practice, to eliminate the intensive time taken for computing HTER scores, the thesis avoids using long segments by retaining only segments between 2 and 1200 characters long. In addition, segments that have hyperlinks were ignored because the MT system added unnecessary spaces around slashes that appear in hyperlinks which will cause unexpected HTER computation behaviors. Processed data were then split into four language pairs and starting from here, data were ready to use for further steps.

Since only shorter segments were retained, it is necessary to analyze the impact of filtering data by lengths and hyperlinks. It seems that most data is retained, as illustrated in Figure 14. In detail, en-fi is the language pair that has the largest number of segments, which is over 1.1 million. The language pair fi-sv has 807,468 segments while the number of segments in the language pairs fi-en and en-sv are

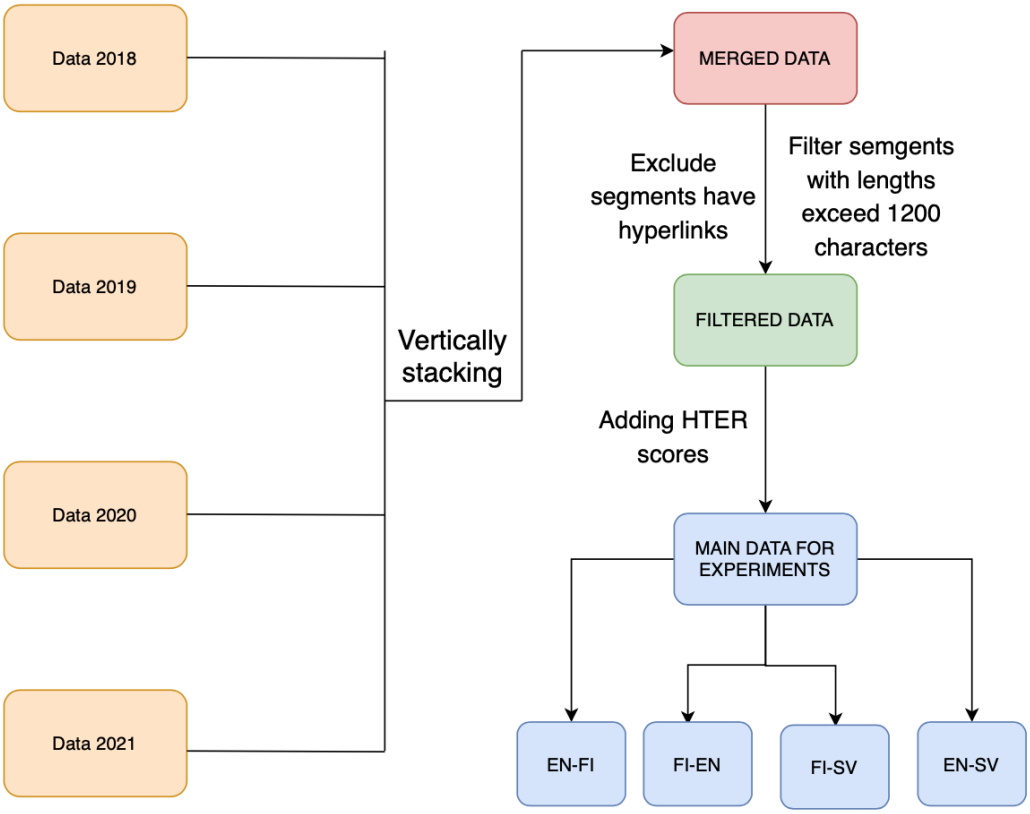

Figure 13. Data pre-processing for the four language pairs

653,771 and 253,227.

Training, validation and testing data were prepared as follows: testing dataset with the fixed size of 50,000 segments was extracted from each language pair dataset, the remaining data was used for training and validation purpose. The amount of 50,000 segments of the testing dataset was chosen because it is sufficient to investigate MTQE performance in later steps. The testing dataset was extracted by randomly shuffling the whole dataset and selecting the first 50,000 segments. The following Figure 15 shows the number of remaining data after extracting 50,000 segments for testing dataset in four language pairs.

Then, to investigate how changing data volumes would impact the performance of the trained MTQE models, the data (training + validation) were randomly sampled into different volume sizes/batches and each batch contains five sets of same-size data. The language pair en-sv includes four batches, while the other language pairs contain five batches as depicted in Table 1

As shown in Table 1, the four language pairs have the same following batch

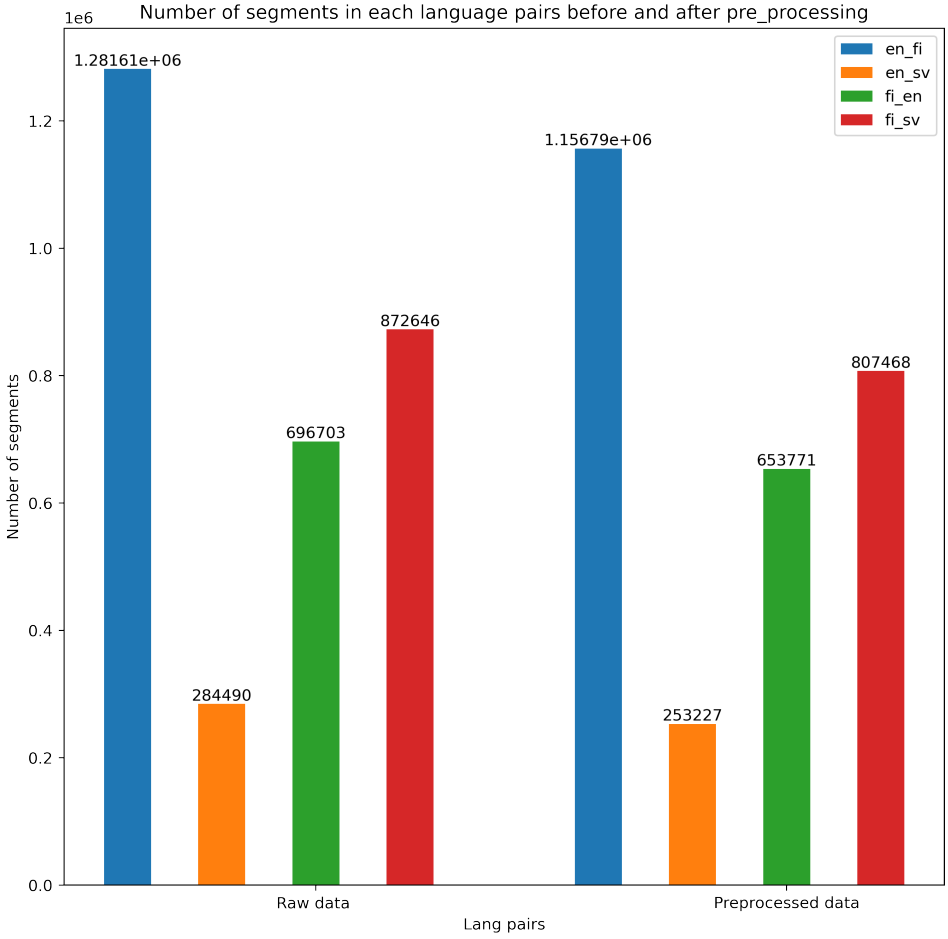

Figure 14. language pairs distribution before and after pre-processing

Language pairs	Batches				
	10,000	50,000	100,000	300,000	All
en-fi	✓	✓	✓	✓	✓(1,106,788)
fi-en	✓	✓	✓	✓	✓(603,771)
fi-sv	✓	✓	✓	✓	✓(757,468)
en-sv	✓	✓	✓		✓(203,227)

Table 1. Different batch sizes of four language pairs

sizes: 10,000; 50,000; 100,000 and their corresponding full data batch (the biggest batch). The first three language pairs: en-fi, fi-en, and fi-sv have batch size of 300,000; while the remaining language pair en-sv does not since its biggest batch size is only

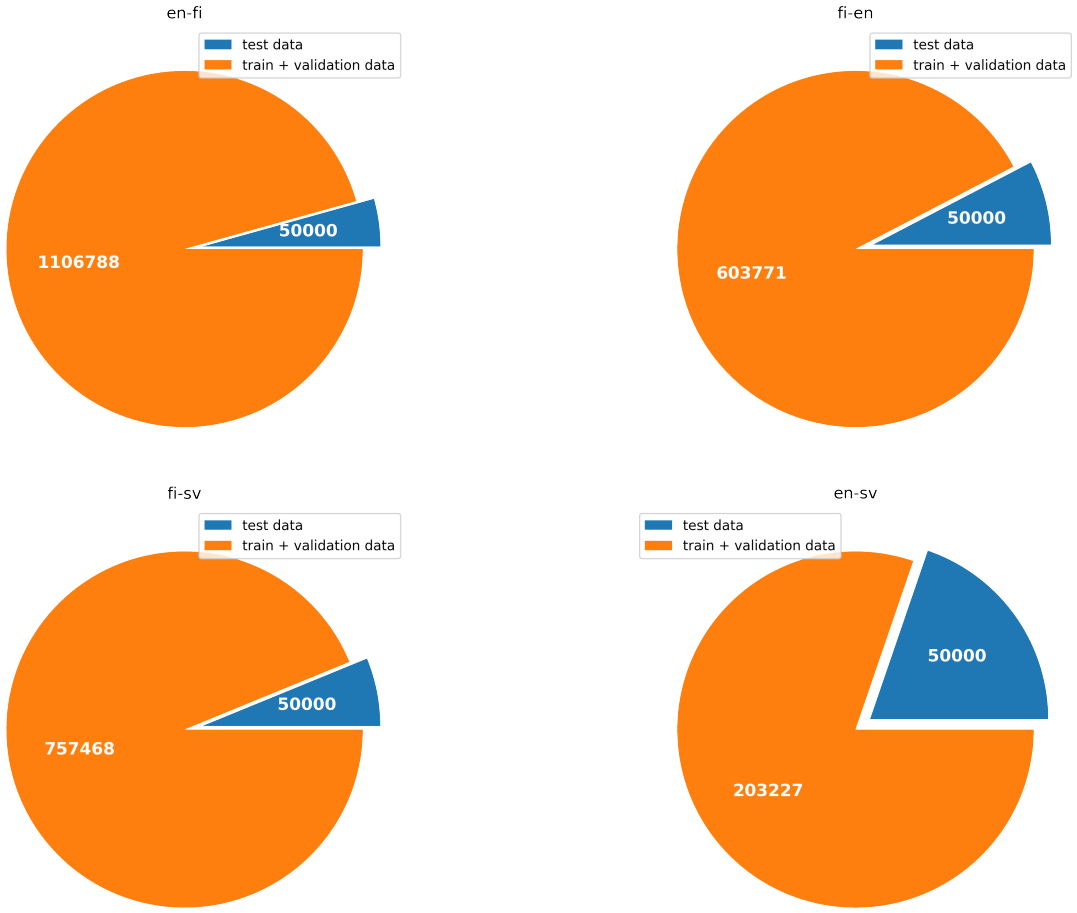

Figure 15. Training, validation and testset per language pair

203,227. The batching procedure in four language pairs followed the same steps: randomly sampling the data with the same size of the batch for five times for each batch (except the batch "All"); splitting the sampled data into training and validation sets with the corresponding ratios: 75% and 25%. For example, to generate batch "10,000" for the language pair en-fi, data from the biggest batch (1,106,788 segments) was randomly sampled with the size of 10,000 segments for five times. As a result, this created five training and validation datasets which were later used to train five corresponding models. This sampling procedure was implemented to measure the fluctuation of performance between different training sets within same volumes (e.g. 10,000; 50,000) on a fixed testset.

However, this also has one drawback of reobtaining a similar dataset when sampling large batches, which may result in similar metrics. For example, sampling the batch "100,000" in the language pair en-sv for five times may obtain quite similar segments (overlap or intersection segments) since the the batch "All" has only 203,227 segments. Fortunately, this limitation does not affect other language pairs as the population data are large, compared to required batches.

4.2 Data Volume Impact Experiment

After preparing the data, it was ready for training and testing. The experiment started with TransQuest and its default configuration [1]. In the configuration, TransQuest's Mono architecture was used, because all models trained with this architecture provide better results than models trained with Siamese architecture in all the language pairs in WMT20. Then, the same training and evaluation procedure (with same data) was repeated with OpenKiwi where BERT model setting [2] was used.

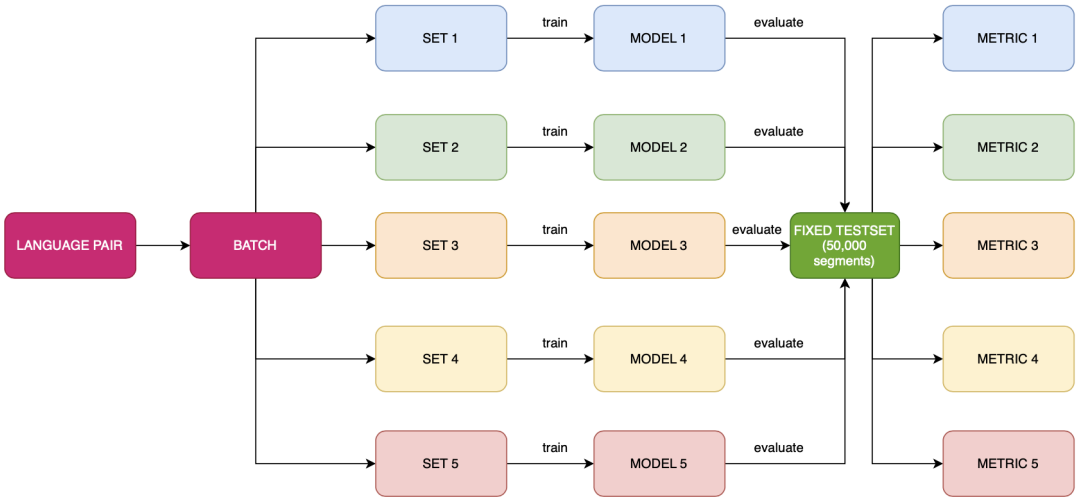

Figure 16. General pipeline of training and evaluating model from a batch data

All available batch sizes of each language pair were sequentially trained and obtained models were evaluated with the corresponding fixed testsets in four language pairs. The evaluation metrics include Pearson and MAE scores. The general pipeline for this experiment is depicted in Figure 16. This pipeline was applied in all language pairs and the corresponding available batches.

[1] https://github.com/TharinduDR/TransQuest/blob/master/examples/sentence_level/wmt_2020/ro_en/monotransquest_config.py

[2] https://github.com/Unbabel/OpenKiwi/blob/master/config/bert.yaml

4.3 Source Segment Length Impact on MTQE

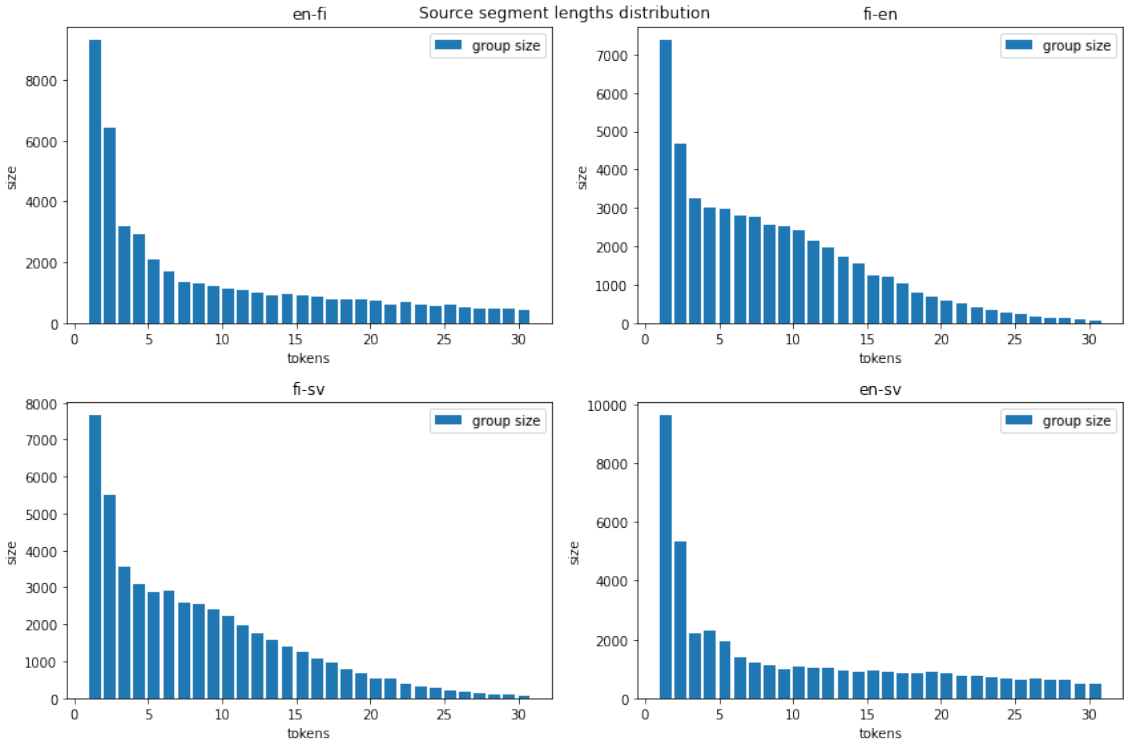

Figure 17. Distribution of source segment lengths in 4 language pairs

For each language pair, the prediction result from the best MTQE model on the corresponding fixed testset was grouped based on the length of the source segments. The distributions of source segment length groups in four language pairs are illustrated in Figure 17. The source segment lengths of all language pairs have right-skewed distributions which means most of the segments are short in length. Then, 30 groups of segments were collected and calculated Pearson scores. Because the sizes of these groups are not identical, the size of the group 30 was selected as a standard size for the remaining groups; thus, the size depends on the language pairs. From group 1 to group 29, a new standard-size subset was randomly sampled five times. The Pearson scores of these five samples were then averaged to represent the Pearson score of each group.

4.4 Near-perfect MT classification

For near-perfect MT detection, an HTER threshold value is set to represent the near-perfect translations. This means any predicted and true HTER scores that are less than or equal to the threshold (and larger than or equal to zero) will be assigned near-perfect labels, while the remaining HTER scores will be marked as non-near-perfect labels. The HTER threshold value is set from 0.0 to 0.10 to analyze

the trade-off between the precision and recall of detecting near-perfect and non-near-perfect translations. The threshold upper limit of 0.10 was chosen because the main interest is to detect near-perfect translations. In industry, publishing a non-perfect MT (false positive) without post-editing has larger costs than post-editing a near-perfect MT (false negative).

5 Experiment Results

This chapter presents in details the results of three experiments. First, it shows how different data volumes affect model performance with the chosen MTQE frameworks. Second, it presents the analytic results of the source segment length impact on model performance. Last, it introduces the near-perfect MT classification analysis.

5.1 Different Data Volumes On Model Performance

As mentioned in Chapter 3, TransQuest was used for training MTQE models and then OpenKiwi. As a result, all the experiments have been successfully conducted with TransQuest. However, OpenKiwi did not manage to train any MTQE model on the same data that was used in TransQuest, although all the supported architectures of OpenKiwi were tried. Instead, OpenKiwi returned oversized tensor error, which means that there is some issue with the tokenization. By studying the documentation and source codes of the framework, it turns out that the reason is because data contains some large segments or segments that include many rare words; thus, the tool does not truncate these segments to match the maximum tokens allowed in the pretrained model settings. Unfortunately, OpenKiwi does not support tokenization modification. To address this issue, the experiment data was filtered many times so that those segments that have the length of the source, MT, or PE version larger than 128 tokens (tokenized by the tokenizer of pretrained BERT) will be discarded.

This resulted in the tiny loss of experiment data. For example, for the batch 10,000 of the language pair en-fi, about 0.015% of the data was discarded. Nevertheless, the solution did not fully tackle the problem: OpenKiwi only successfully trained MTQE models on some batches of the prepared data, which indicates that the framework is not robust to data randomness. Although cherry-picking the data can temporarily address the issue, it is time-consuming and will create inconsistent and imbalanced datasets, not to mention that the experiment results are not comparable to the results from TransQuest. Therefore, only successful training results of the first experiment were reported. The subsection 5.1.1 shows the obtained results regarding the TransQuest framework and the results of OpenKiwi framework is shown in subsection 5.1.2. In the results, the "10,000" samples batch will be denoted as batch "10k" and similar notations will be applied for the remaining batches.

5.1.1 TransQuest Framework

5.1.1.1 Language pair en-fi

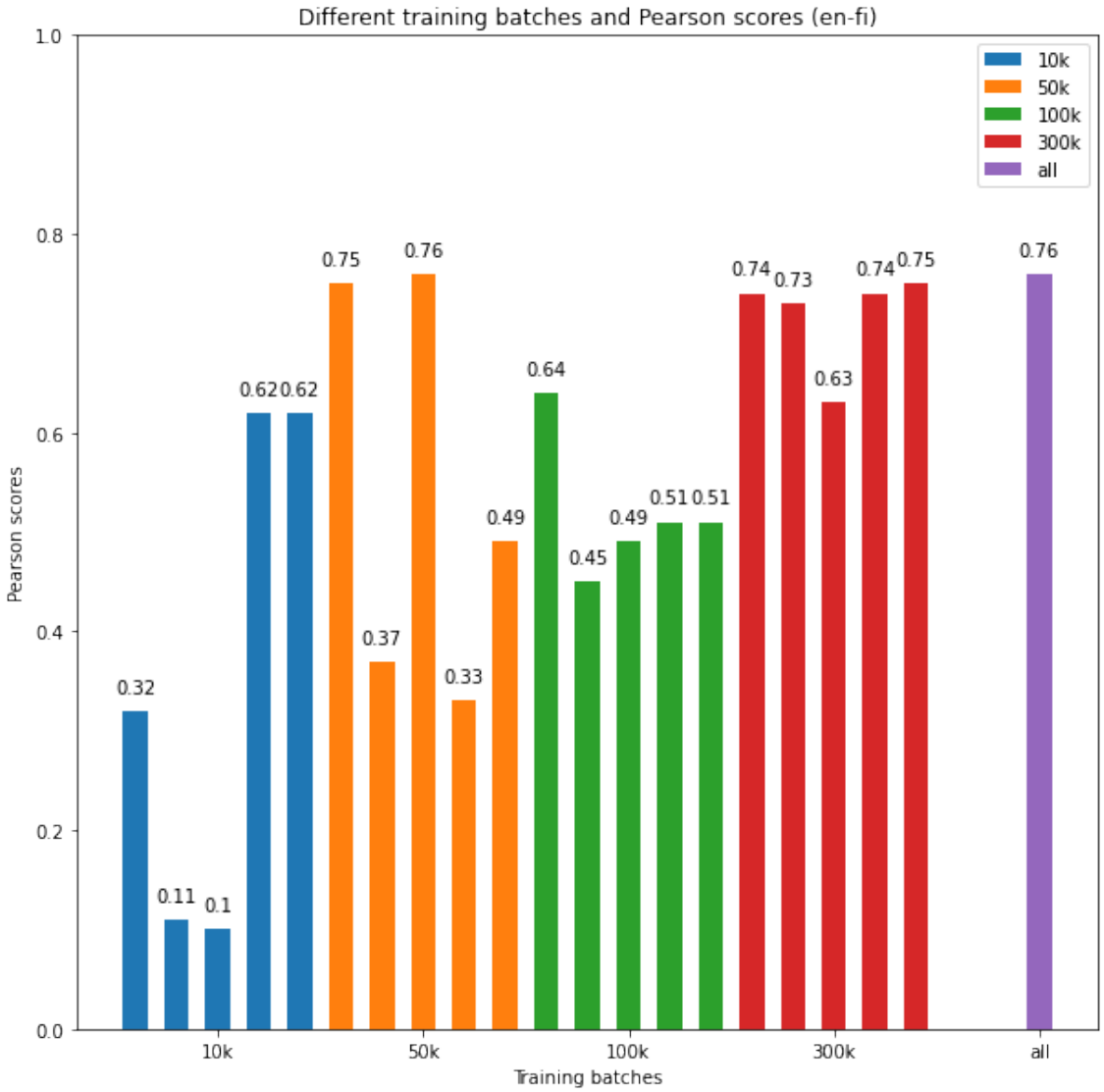

Figure 18. en-fi model evaluation metric (Pearson score) with various batches

Figure 18 illustrates the relationship between the Pearson scores of trained MTQE models when evaluating with the corresponding fixed testsets and different training batch sizes. There is a generally rising trend in model performance when the training data volumes change from small to large values, although there are exceptionally high Pearson scores in some models of small batches (10k and 50k). For example, the fourth and the fifth models of the batch 10k both obtained a 0.62 Pearson score, whereas the first and the third models of the batch 50k achieved 0.75 and 0.76 Pearson scores. However, the variation of the Pearson scores in these

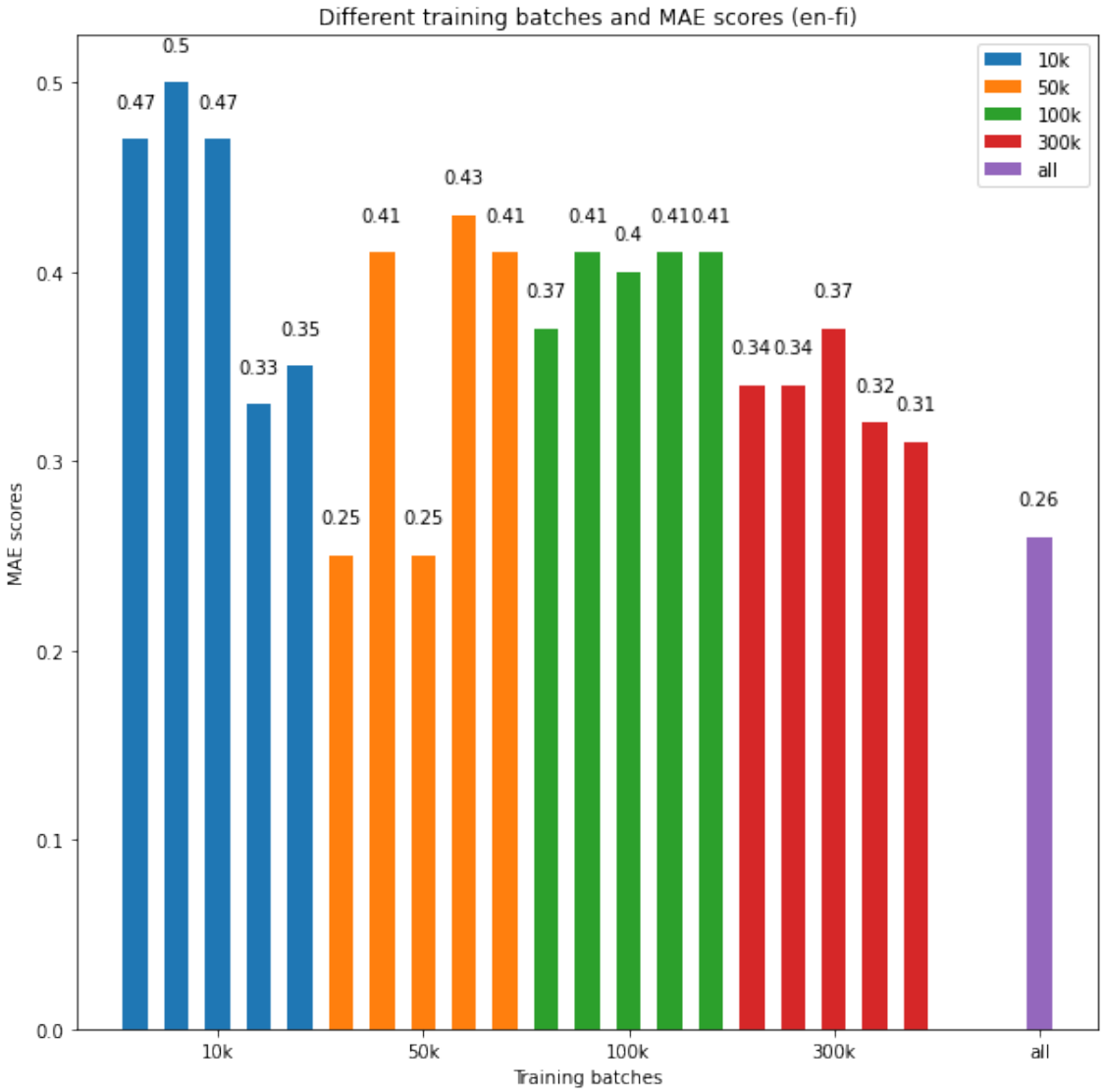

Figure 19. en-fi model evaluation metric (MAE score) with various batches

Batch size	Pearson		MAE	
	mean	standard deviation	mean	standard deviation
10k	0.354	0.258	0.424	0.078
50k	0.540	0.205	0.350	0.092
100k	0.520	0.071	0.400	0.017
300k	0.718	0.050	0.336	0.023
All (1,106,788)	0.760	0.000	0.260	0.000

Table 2. Mean and standard deviation of Pearson and MAE scores with different batches of en-fi

batches is also very high. This shows that Pearson scores seem to fluctuate with the models that were trained on small batches. In contrast, Pearson scores do not

fluctuate starting from the batch 100k. Indeed, the fluctuation behaviours in all batches can be statistically shown by analyzing the standard deviations of Pearson scores as shown in Table 2.

According to the table, the standard deviations of the Pearson scores obtained from the models trained with batch 10k and 50k are quite high: 0.258 and 0.205. On the other hand, Pearson scores from models trained with larger batches 100k and 300k are smaller: 0.071 and 0.050.

For the MAE metric depicted in Figure 19, the impacts are similar to those concluded previously for the Pearson metric. Overall, the MAE scores seem to follow a declining trend when training the models with increasing sizes of data, although there is a slight increase in the mean of the MAE scores of the models trained with batch 50k and batch 100k: 0.350 MAE to 0.400 MAE (Table 2). Models that were trained with larger batches (batch 100k, batch 300k) obtain more stable results. The standard deviation of the MAE scores from the models trained with batch 10k and 50k are 0.078 and 0.092, whereas those values are 0.017 and 0.023 in case of batches 100k and 300k (Table 2).

5.1.1.2 Language pair fi-en

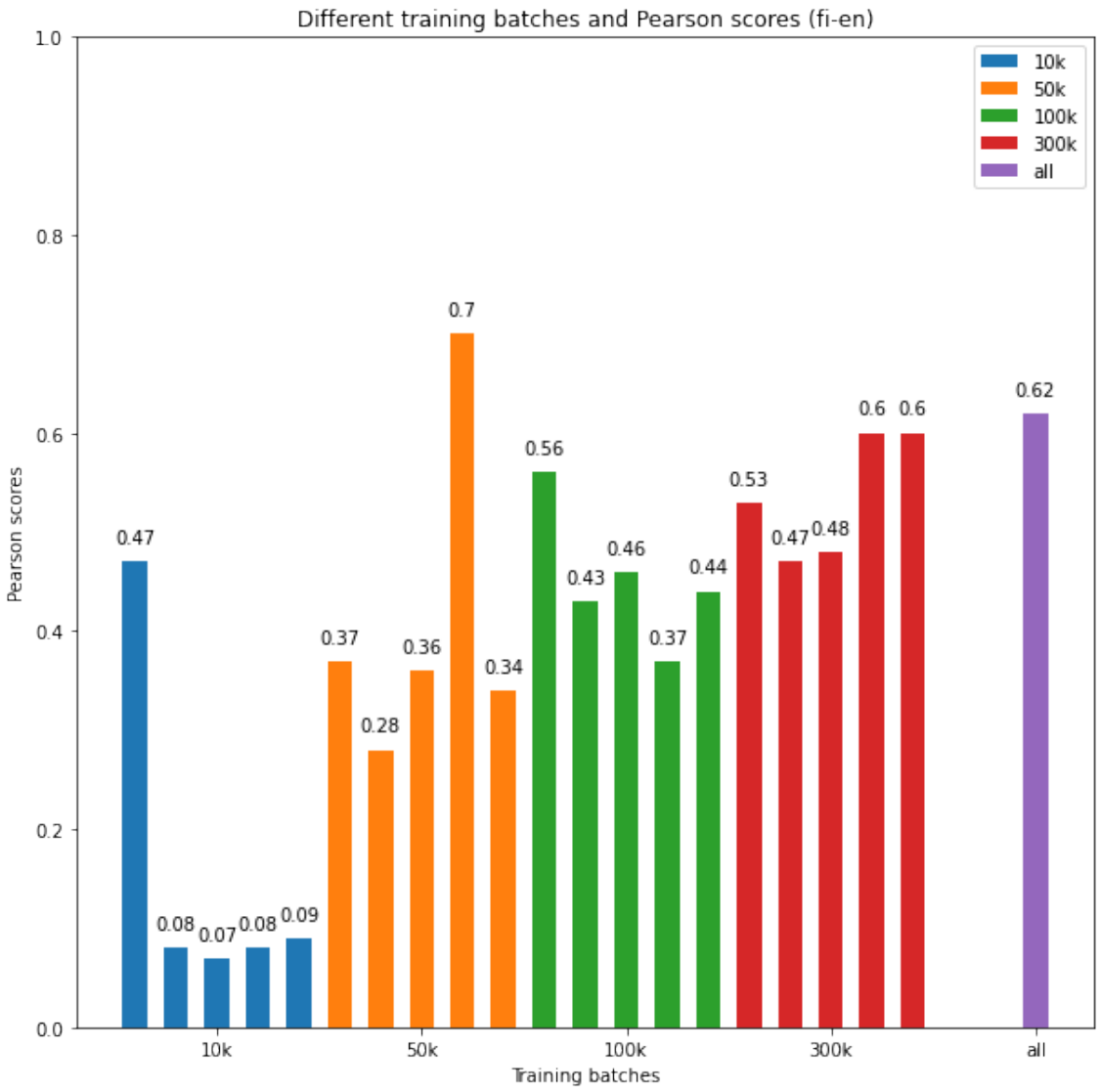

Figure 20. fi-en model evaluation metric (Pearson score) with various batches

The results of language pair fi-en are illustrated in Figures 20, 21, and Table 3. There is a similar trend of increasing Pearson scores with increasing batch sizes as previously discussed in the language pair en-fi. Although there are two exceptionally high Pearson score cases in small batches 10k and 50k where the scores are 0.47 and 0.7, the average score in these two batches 10k and 50k are still lower than other batches, with 0.158 and 0.410 compared to 0.452, 0.536, and 0.620 from batch 100k, 300k and all data. This shows similar fluctuating effect as in the language pair en-fi: models that were trained with smaller batches (10k, 50k) tended to obtain fluctuating Pearson scores. According to Table 3, the standard deviations of the Pearson scores obtained from the models trained with batch 10k and 50k are 0.175 and 0.166, whilst they are 0.069 and 0.063 in two batches 100k and 300k.

The impact of data volumes on the models' MAE scores is visualized in Figure 21 and Table 3. In general, the MAE scores follow a slow downtrend over the expansion of batch sizes and starting from the batch 100k, there is almost no fluctuation in the MAE scores which can be demonstrated by a drop of over 5 times in standard deviation from 0.05 in batches 10k and 50k to 0.009 in batches 100k and 300k.

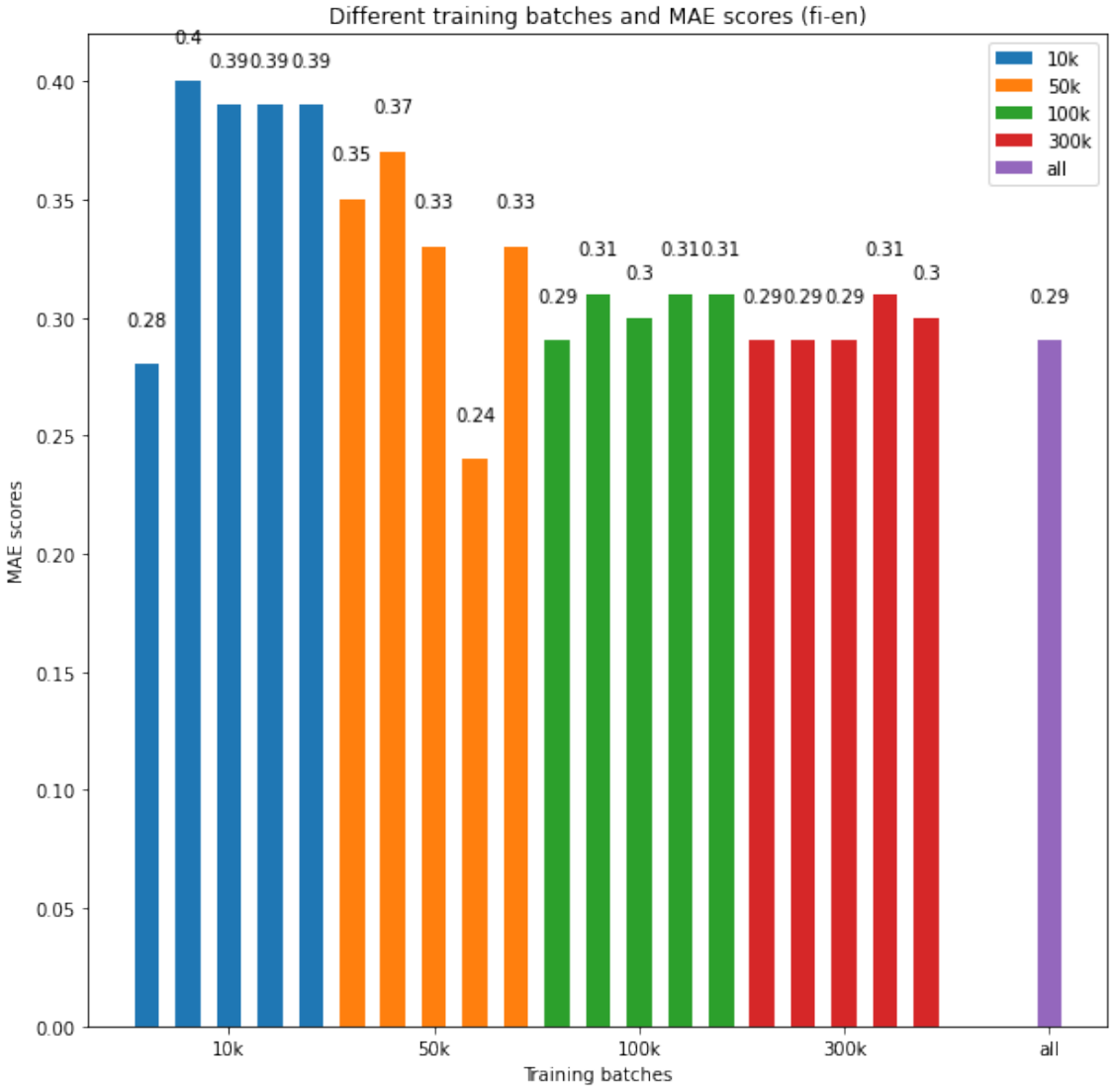

Figure 21. fi-en model evaluation metric (MAE score) with various batches

Batch size	Pearson		MAE	
	mean	standard deviation	mean	standard deviation
10,000	0.158	0.175	0.370	0.050
50,000	0.410	0.166	0.324	0.050
100,000	0.452	0.069	0.304	0.009
300,000	0.536	0.063	0.296	0.009
All (653,771)	0.620	0.000	0.290	0.000

Table 3. Mean and standard deviation of Pearson and MAE scores with different batch sizes of fi-en

5.1.1.3 Language pair fi-sv

Similar to the two language pairs en-fi and fi-en, there are some visualizations of the impacts of changing data volumes over different batch sizes with Pearson and MAE metrics in Figures 22 and 23. Table 4 supports verifying metric fluctuation.

Batch size	Pearson		MAE	
	mean	standard deviation	mean	standard deviation
10,000	0.124	0.146	0.400	0.029
50,000	0.262	0.040	0.350	0.019
100,000	0.412	0.101	0.338	0.033
300,000	0.402	0.050	0.310	0.026
All (807,468)	0.62	0.000	0.290	0.000

Table 4. Mean and standard deviation of Pearson and MAE scores with different batch sizes of fi-sv

Again, there is a slowly growing trend in model performance when the training data volume sizes change from small to large: the mean of Pearson scores nearly double between the batch 10k and the batch 50k, from 0.124 to 0.262. Increasing volume size from 50k to 100k results in higher mean of Pearson scores (from 0.262 to 0.412) and in more fluctuation, with standard deviation going up to 0.101 from 0.040. Then, in the batch 300k, the average of Pearson scores slightly decreases and standard deviation reduces by half compared to standard deviation in the batch 100k.

For the MAE metric, it seems to follow a declining trend over the increase of data volumes: the mean of MAE starts with 0.400 at the batch 10k and gradually drops along with the larger batches until it reaches the lowest value of 0.290 at the full volume batch. There seems to be no fluctuation of MAE scores in any of the batches and the batch 50k obtains the lowest standard deviation of 0.019.

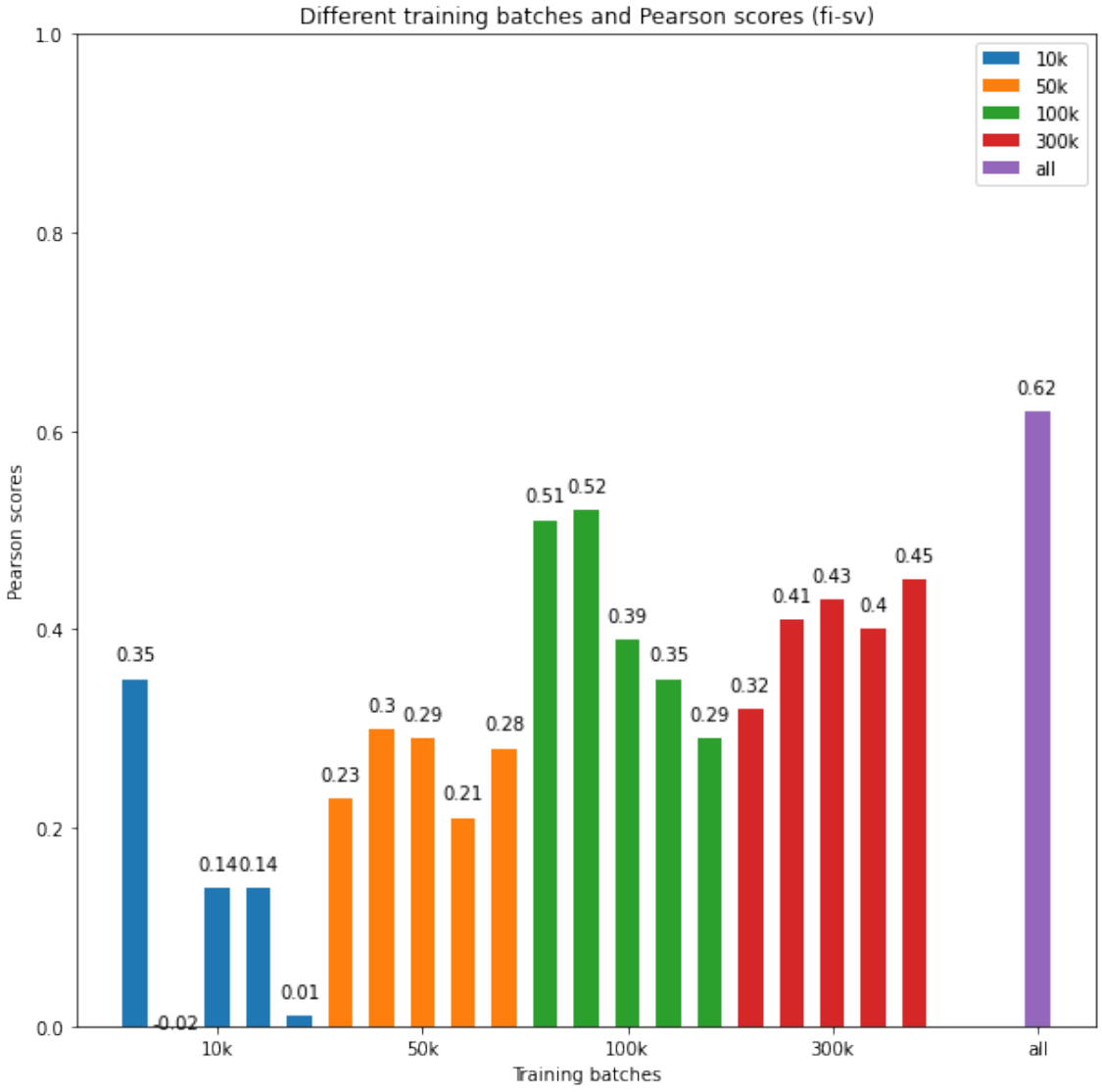

Figure 22. fi-sv model evaluation metric (Pearson score) with various batches

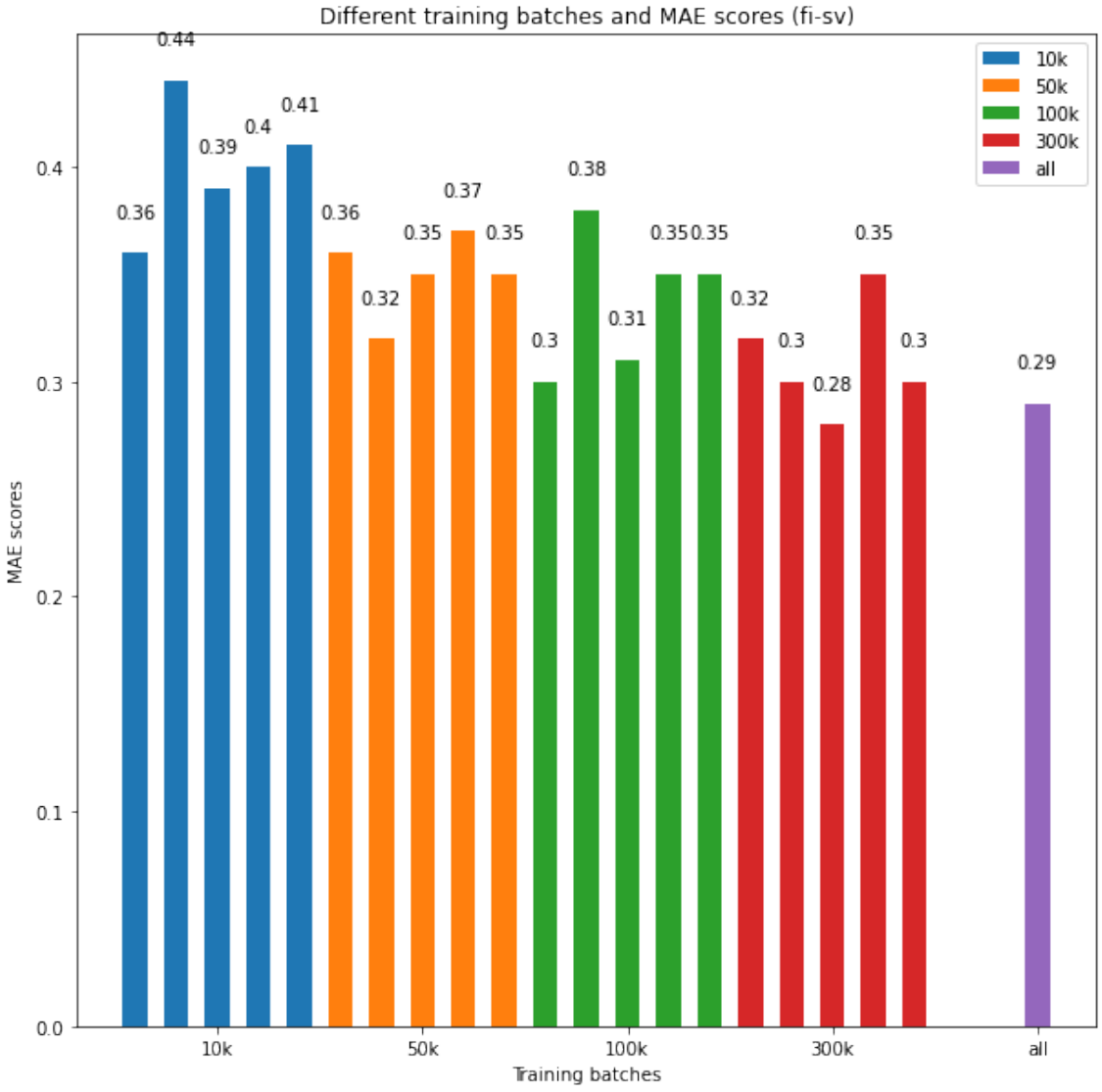

Figure 23. fi-sv model evaluation metric (MAE score) with various batches

5.1.1.4 Language pair en-sv

Out of four language pairs, en-sv has only 253,227 segments but experiments show that some models obtain especially high Pearson scores. For example, three out of five models in the batch 100k acquired a 0.95 Pearson score as shown in Figure 24.

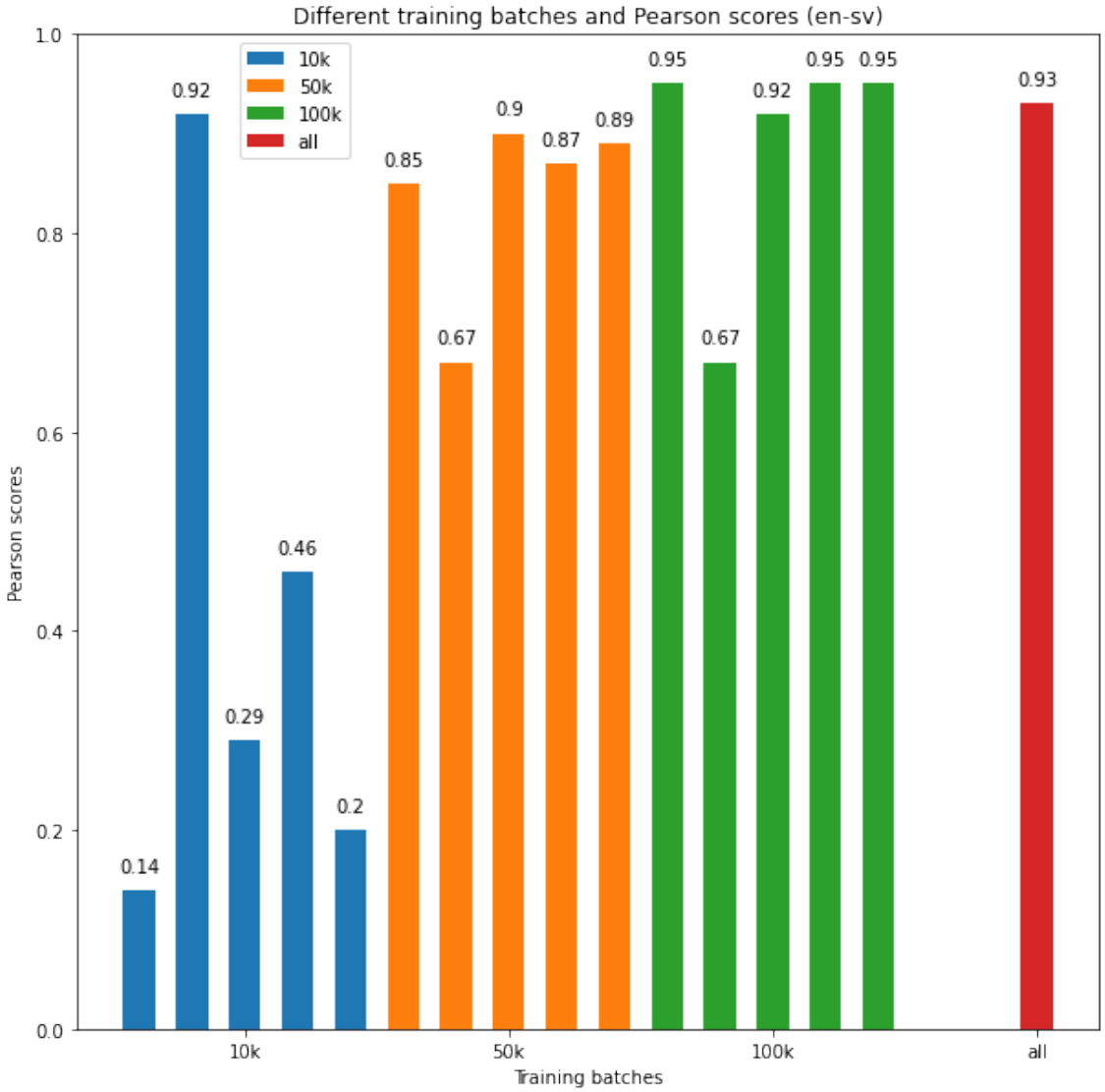

Figure 24. en-sv model evaluation metric (Pearson score) with various batches

As can be seen in Table 5, there is a significant gain of the mean of Pearson scores from the batch 10k to the batch 50k: 0.402 to 0.836. The batch 10k has one trained model that achieved a very high Pearson score of 0.92 but still, the fluctuation is relatively high with standard deviation of 0.314. Changing the batch size from 50k to 100k slightly improves the mean of Pearson scores from 0.836 to 0.888 but also raises the standard deviation from 0.095 to 0.123. The model trained with the full-size batch acquires a very good Pearson score of 0.93. As depicted in Figure 25, there is a downtrend of MAE when batch sizes change from 10k to 50k then 100k as the MAE values drop from 0.400 to 0.354 and 0.270. The batch 50k seems to have

Batch size	Pearson		MAE	
	mean	standard deviation	mean	standard deviation
10,000	0.402	0.314	0.400	0.086
50,000	0.836	0.095	0.354	0.035
100,000	0.888	0.123	0.270	0.067
All (253,227)	0.93	0.000	0.280	0.000

Table 5. Mean and standard deviation of Pearson and MAE scores with different batch sizes of en-sv

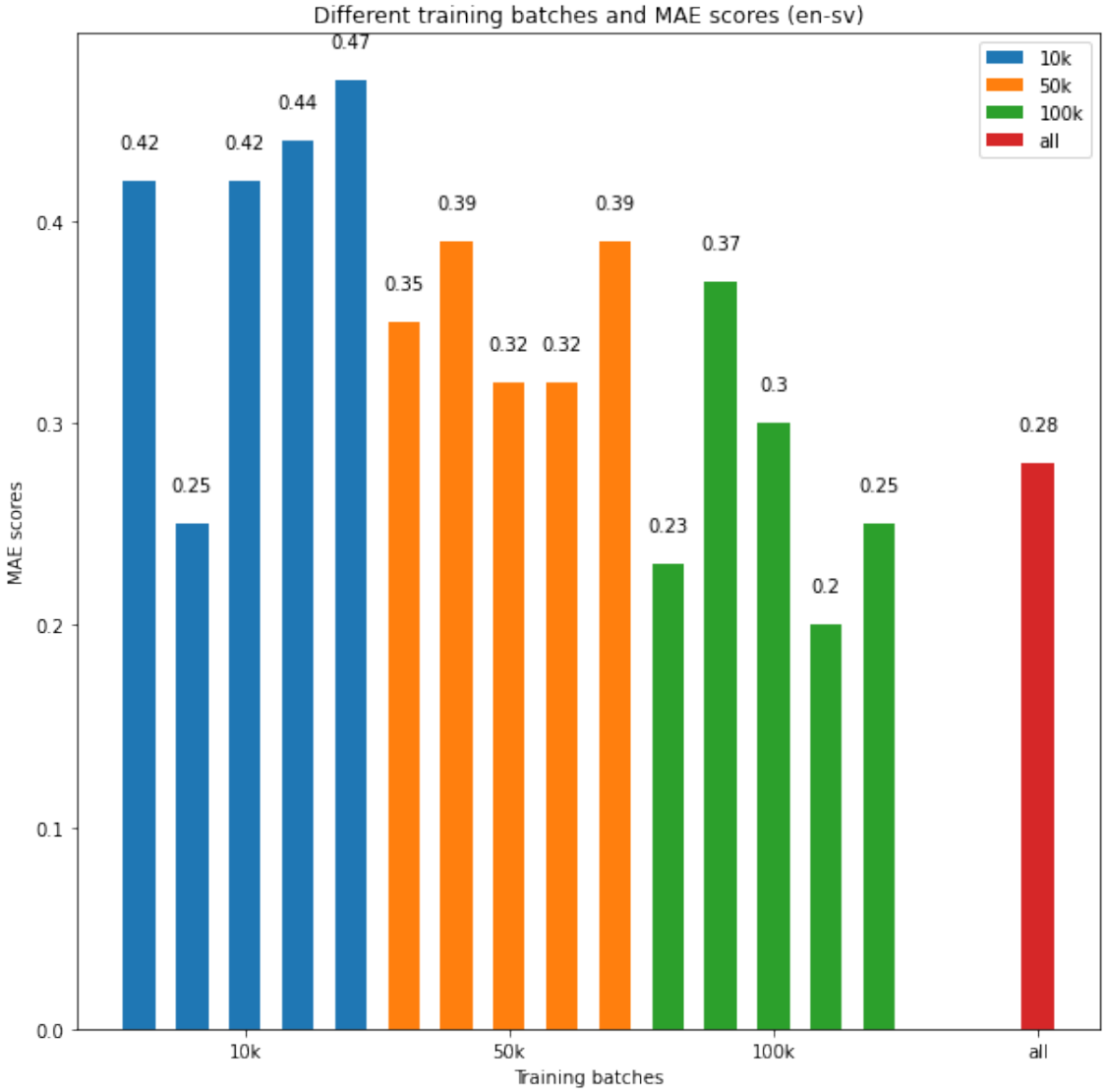

Figure 25. en-sv model evaluation metric (MAE score) with various batches

the most stable MAE with an standard deviation value of 0.035 while that number is 0.086 for the batch 10k and 0.067 for the batch 100k.

5.1.2 OpenKiwi Framework

As discussed earlier, OpenKiwi did not successfully train all the data that were earlier used with TransQuest, the common error return was "tensor 514" which violated the maximum token support (512) when segments were tokenized by the tokenizer of pre-trained BERT model. However, filtering long segments and segments with rare words seemed to partially tackle the issue. All the obtained results of the four language pairs are reported in this section. Because the framework failed at some batches of data, the population data (batch "All") is not available.

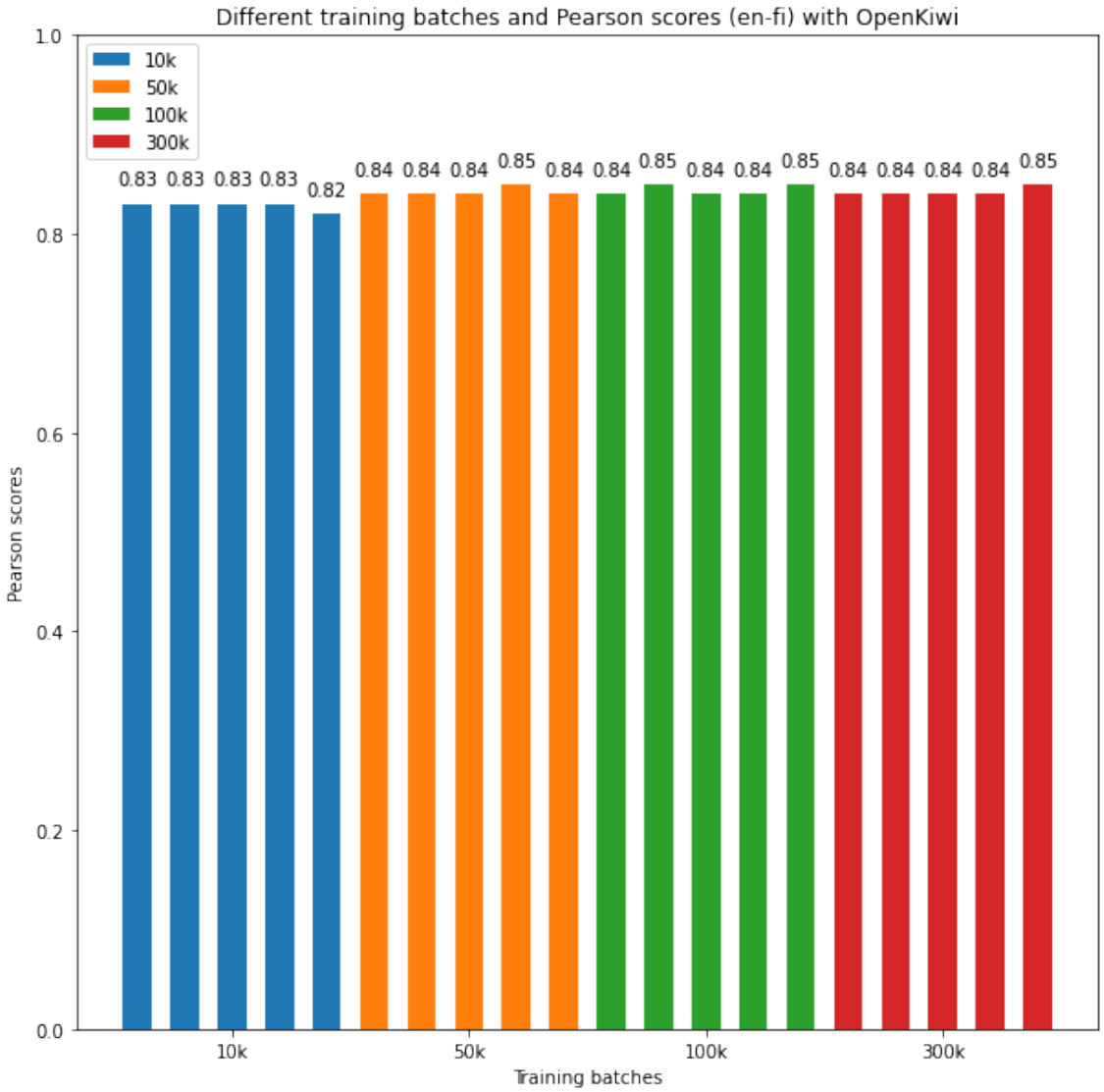

Figure 26. en-fi model evaluation metric (Pearson score) with various batches trained with OpenKiwi

Figure 26 illustrates the Pearson score results of the language pair en-fi. Overall, the results clearly show that the Pearson scores do not fluctuate within the same batch and they seem to be robust to the volume changes. The batch 50k seems

to be sufficient to obtain similar Pearson scores as in other batches.

.

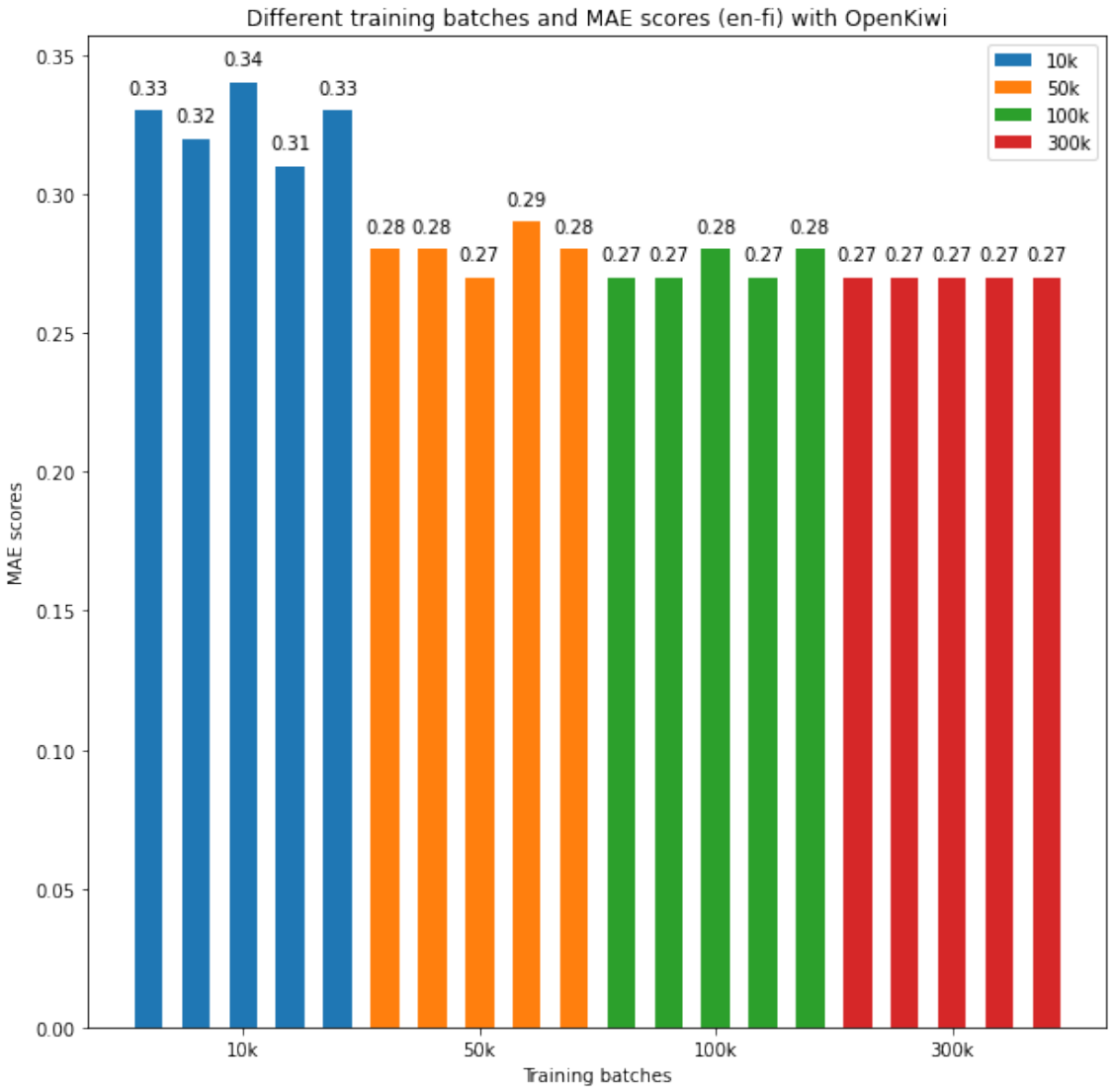

Figure 27. en-fi model evaluation metric (MAE score) with various batches trained with OpenKiwi

The MAE scores look stable within the same batch starting from the batch 50k. There is a slightly decreasing trend of MAE scores from batch 10k to other batches; again, 50,000 segments seem to be sufficient to obtain similar results as shown in other batches (Figure 27).

Figure 28 shows the evaluation results of the language pair fi-en by using OpenKiwi. Unfortunately, results are only available in some batches due to the same error mentioned earlier in the language pair en-fi. In general, there is a slightly rising trend of Pearson scores, together with the corresponding fairly declining trend of MAE scores (Figure 29). Because training only fully succeeded (in five datasets)

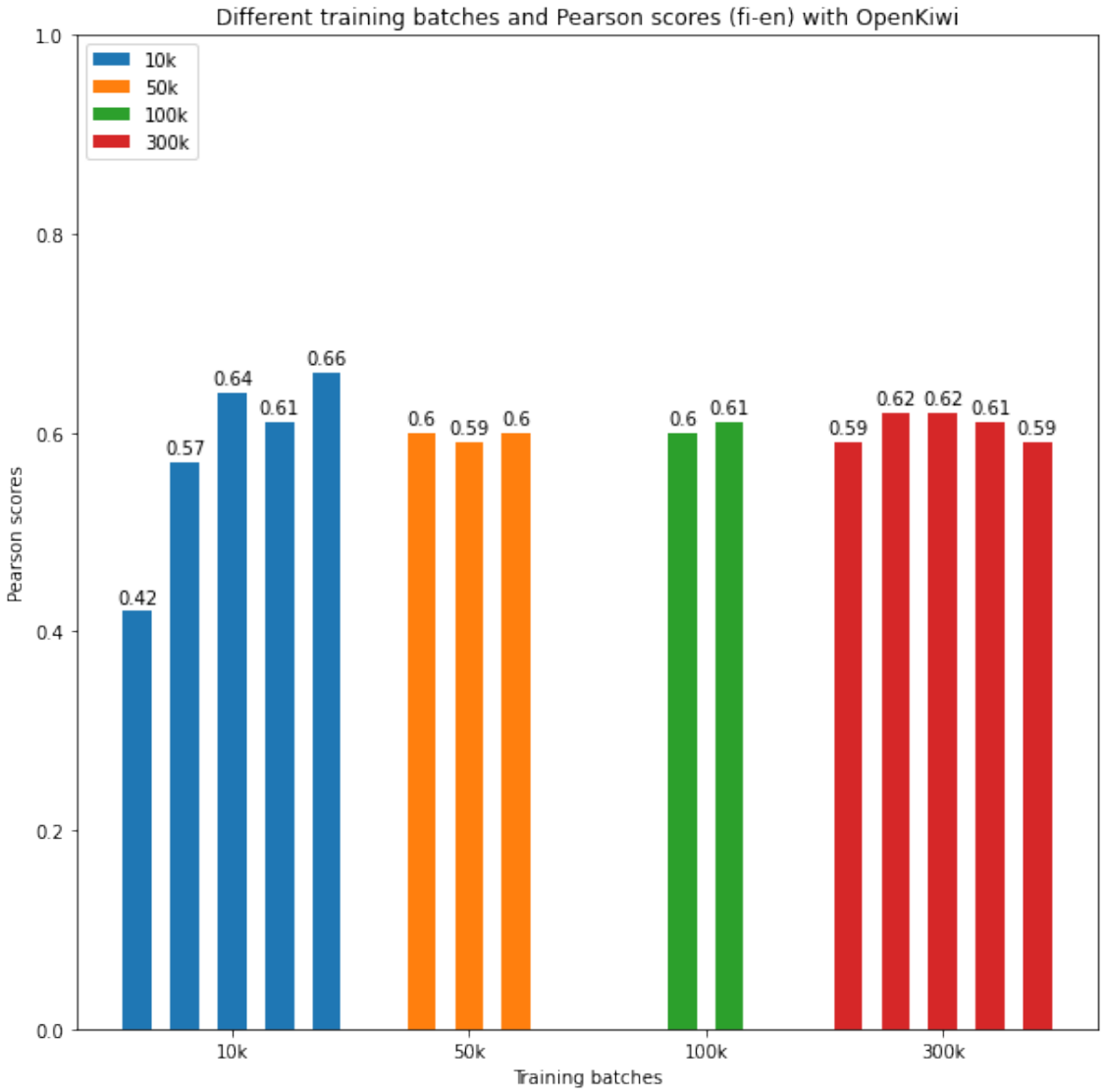

Figure 28. fi-en model evaluation metric (Pearson score) with various batches trained with OpenKiwi

in batch 10k and 300k, it is feasible to compare the fluctuation within each batch and metrics in the two batches. It seems that metrics in batch 10k fluctuate more than that in batch 300k. The results may imply that 50,000 segments are sufficient to obtain decent Pearson and MAE scores, although the missing results may prove it wrong. For example, if there were some missing low metrics in batch 50k, the average metrics would be lower and the fluctuation would be higher.

Figure 30 shows the Pearson score results of the language pair fi-sv. Although missing results from batch 50k, batch 100k and the full batch do not allow comparing metrics between each batch, available metric results seem to be robust to data volume changes.

On the other hand, available MAE scores are stable in batches 50k and

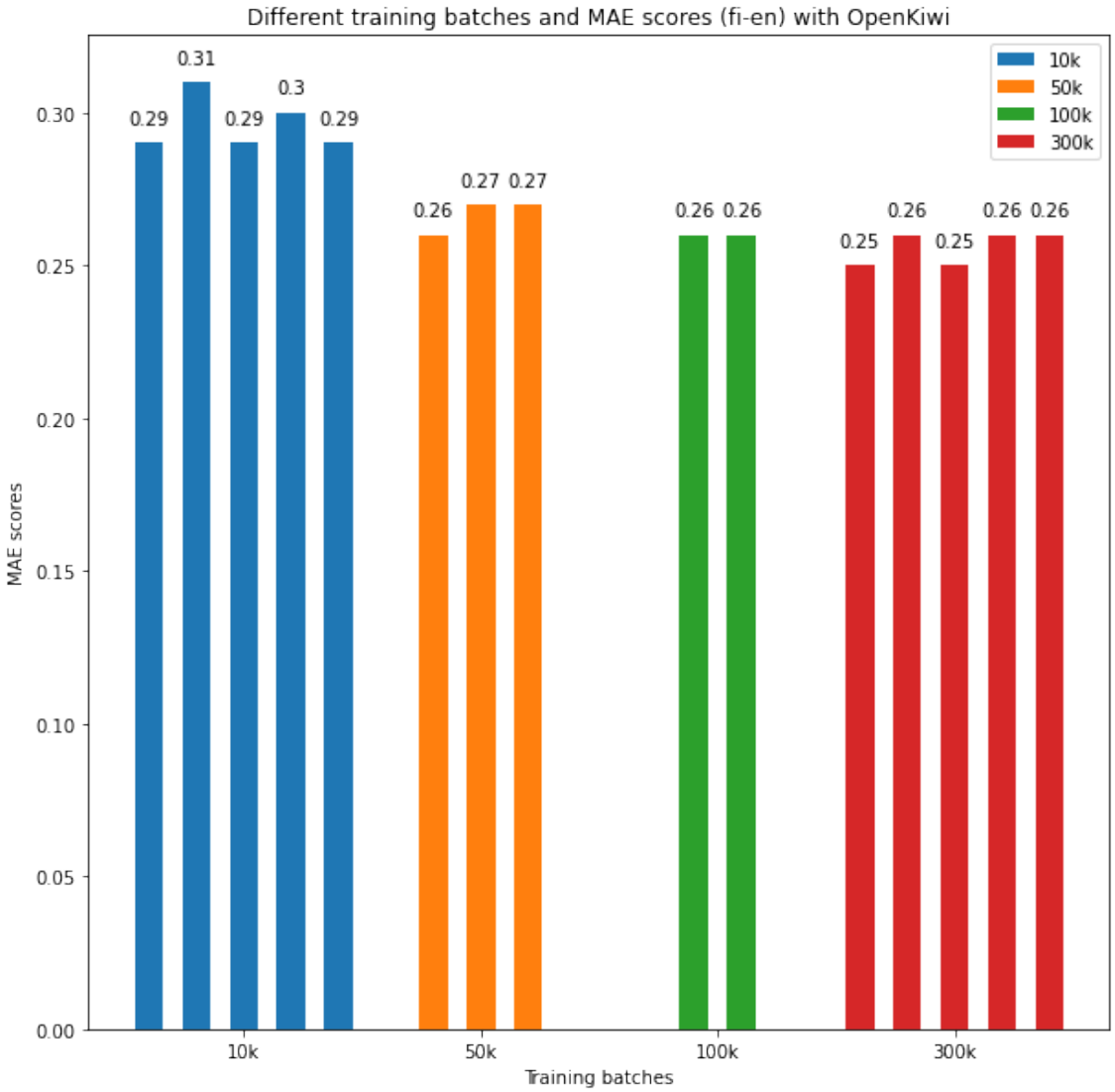

Figure 29. fi-en model evaluation metric (MAE score) with various batches trained with OpenKiwi

100k, whereas in batch 10k, the obtained metrics seem to be less stable, as shown in the Figure 31

Figure 32 shows the Pearson score results of the language pair en-sv. There are missing results from batch 50k, batch 100k, batch 300k and the full batch. Available metric results show a constant high Pearson trend (all Pearson scores are 0.94). The MAE metrics are illustrated in Figure 33. There is a slightly declining trend across the available training batches. In addition, the metrics in batch 10k are less stable than those in batches 50k and 100k.

Overall, the OpenKiwi results showed that there seemed to be no major impact of data volumes on MTQE performance when using Pearson scores, where as using MAE scores showed a slight improvement of performance. The failed training

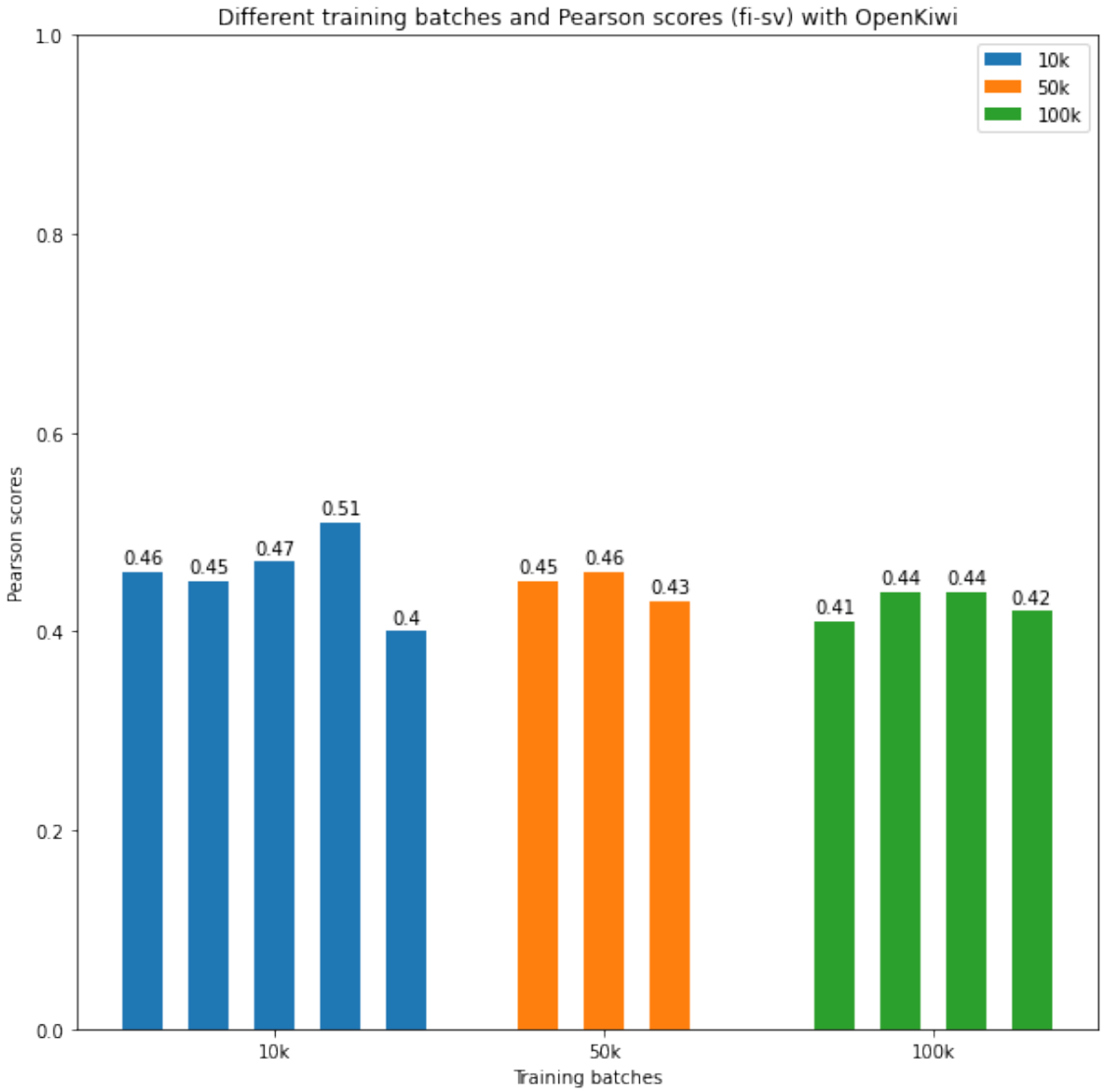

Figure 30. fi-sv model evaluation metric (Pearson score) with various batches trained with OpenKiwi

on many batches caused challenging to directly compare the results with TransQuest.

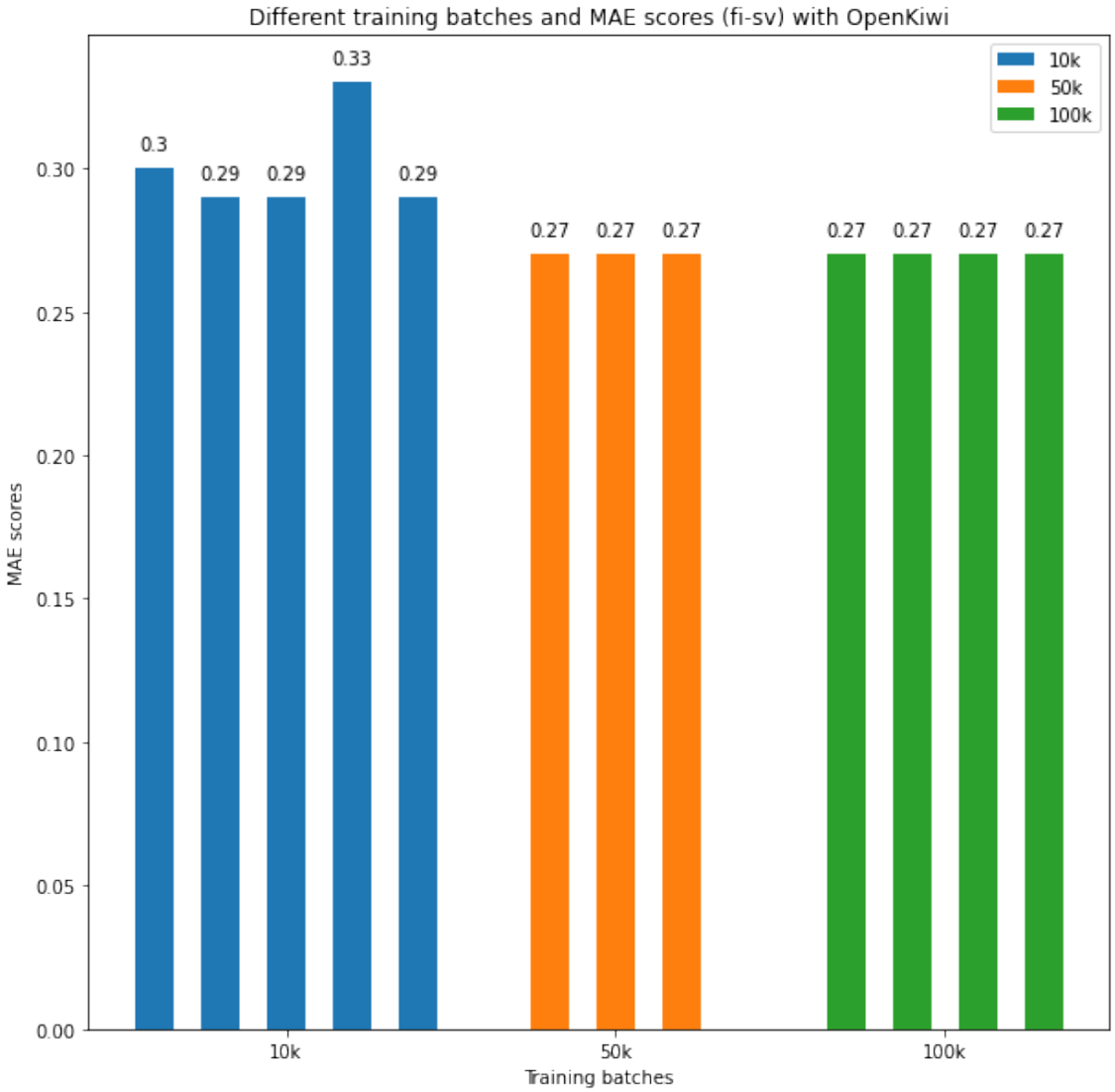

Figure 31. fi-sv model evaluation metric (MAE score) with various batches trained with OpenKiwi

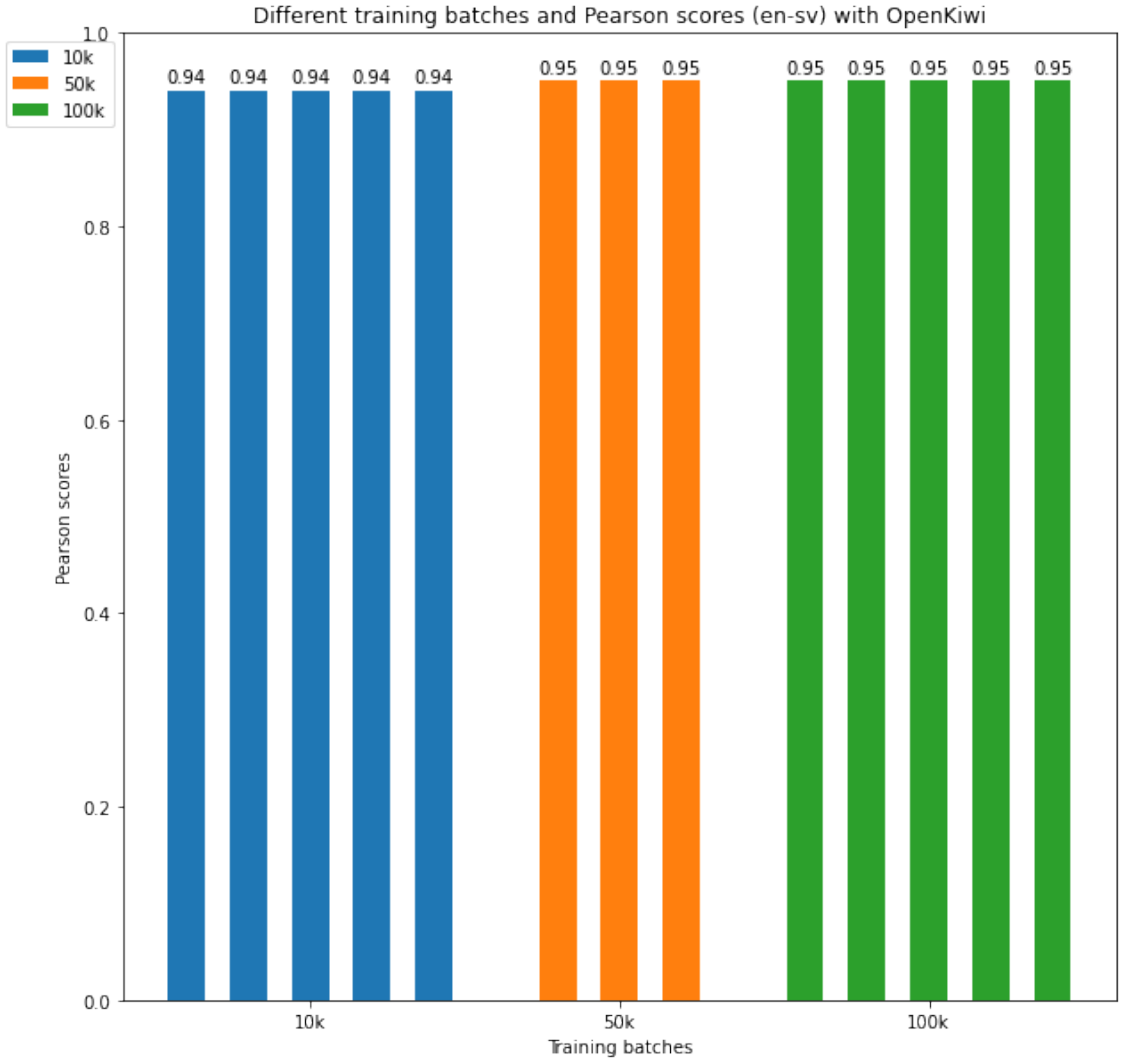

Figure 32. en-sv model evaluation metric (Pearson score) with various batches trained with OpenKiwi

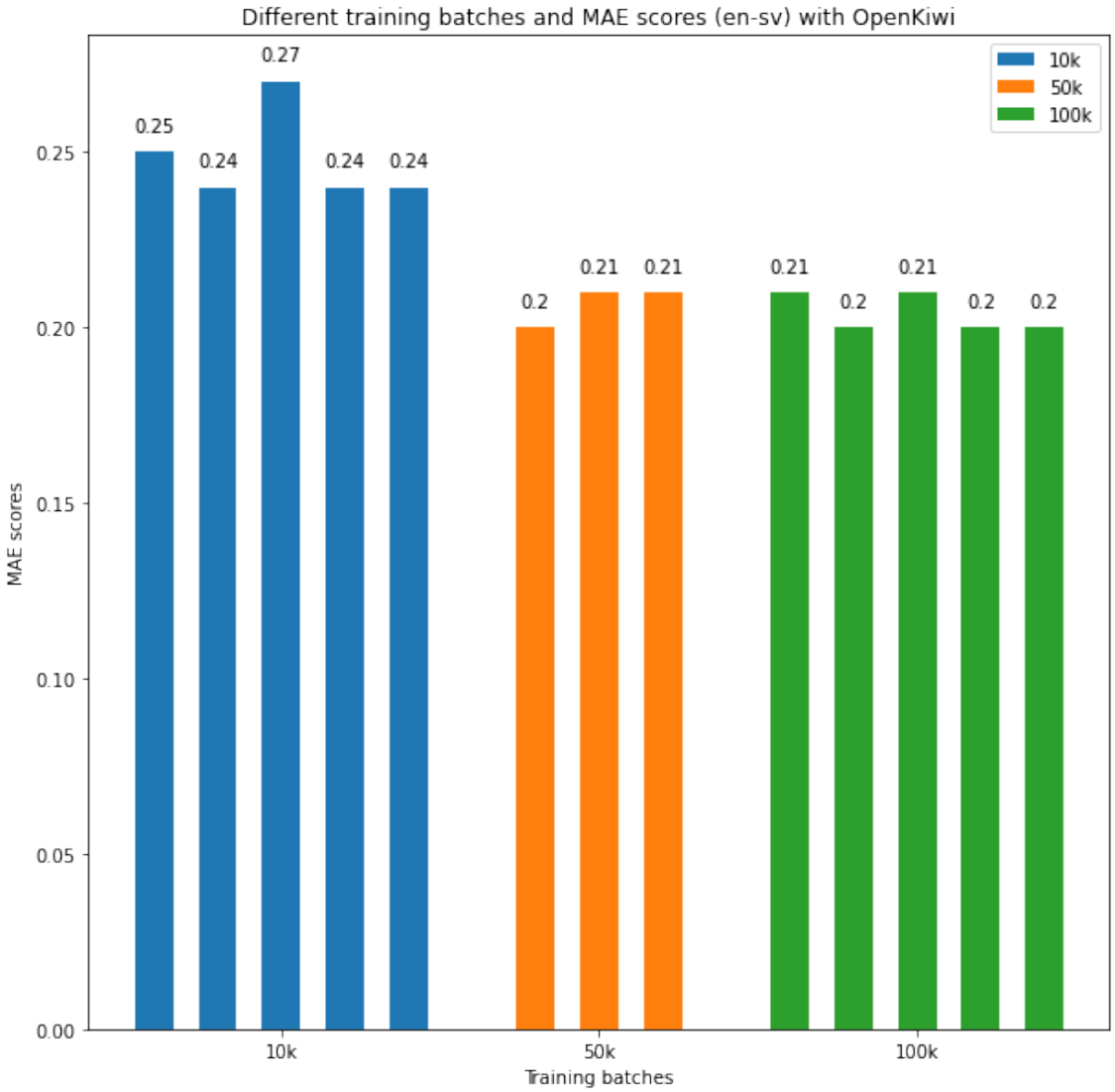

Figure 33. en-sv model evaluation metric (MAE score) with various batches trained with OpenKiwi

5.2 Model Performance for Different Segment Lengths

This section refers to the results obtained from using TransQuest. Figure 34 visualizes the impact of source segment lengths on MTQE model performance of four language pairs. The different groups of source segments length are called as follows: group 1 means it contains only samples that have source segments with the length of 1 token, and group 30 contains all source segments with the length of 30 tokens. In all the language pairs, the Pearson scores of group 1 are always the highest which can be explained by the fact that one-token source segments in the data are mostly numbers or name entities which easily make models predict the HTER scores correctly. Starting from group 4 to group 30, the Pearson scores seem to fluctuate between 0.3 and 0.4 in three language pairs: fi-en, fi-sv, and en-sv. For the language pair en-fi, there is a similar fluctuating trend of Pearson scores between 0.5 and 0.6 from group 4 to group 28.

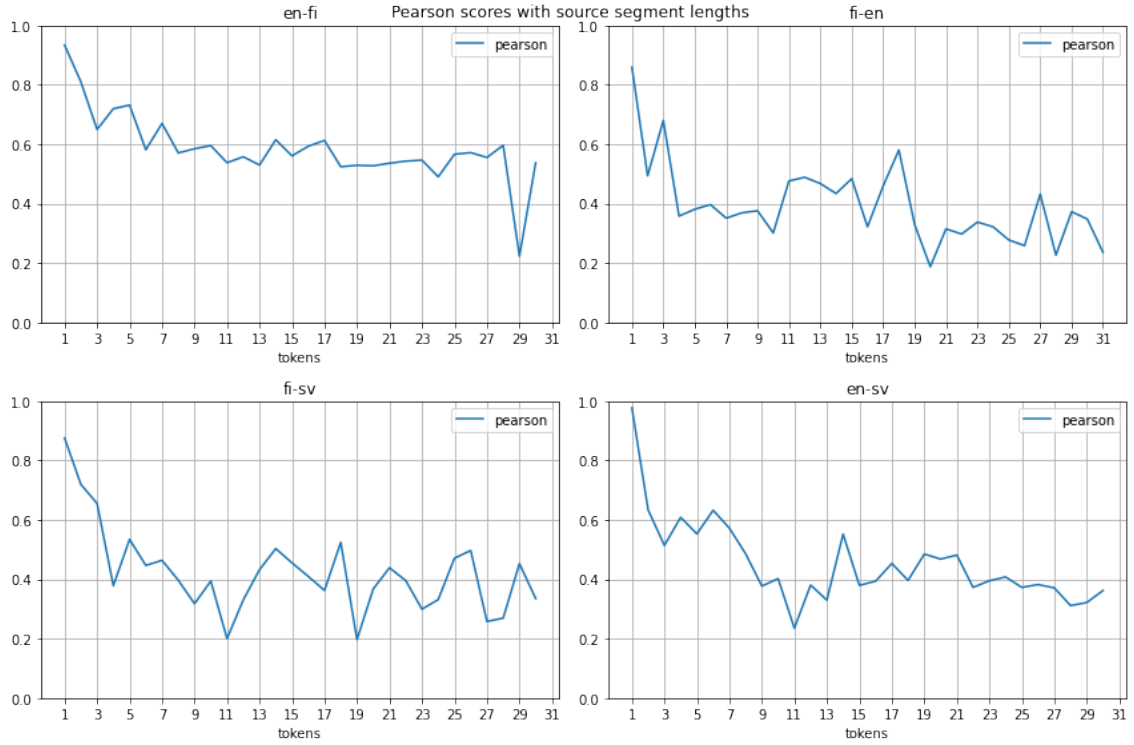

Figure 34. Pearson scores and different groups of source segment length

5.3 Near-Perfect MT detection

This section refers to the results obtained from using TransQuest. Table 6 and Table 7 show the results of detecting near-perfect MT and non-near-perfect MT in the four language pairs within a certain range of HTER threshold values. In general, setting larger HTER thresholds increases the number of near-perfect MT and reduces the

HTER threshold	en-fi						fi-en					
	near-perfect MT			non-near-perfect MT			near-perfect MT			non-near-perfect MT		
	precision	count	recall	precision	count	recall	precision	count	recall	precision	count	recall
0.00	1.00	12,744	0.17	0.78	37,226	1.00	0.00	8,980	0.00	0.82	41,020	1.00
0.01	1.00	12,744	0.27	0.80	37,226	1.00	0.00	8,980	0.00	0.82	41,020	1.00
0.02	0.99	12,775	0.37	0.82	37,225	1.00	0.00	8,983	0.00	0.82	41,017	1.00
0.03	0.99	12,785	0.43	0.84	37,215	1.00	0.00	9,023	0.00	0.82	40,977	1.00
0.04	0.98	12,822	0.47	0.84	37,178	1.00	0.00	9,156	0.00	0.82	40,844	1.00
0.05	0.98	12,905	0.50	0.85	37,095	1.00	0.00	9,345	0.00	0.81	40,655	1.00
0.06	0.98	12,985	0.52	0.86	37,015	1.00	0.00	9,590	0.00	0.81	40,410	1.00
0.07	0.97	13,067	0.53	0.86	36,933	1.00	0.00	9,847	0.00	0.80	40,153	1.00
0.08	0.97	13,218	0.54	0.86	36,782	0.99	1.00	10,209	0.00	0.80	39,791	1.00
0.09	0.97	13,307	0.55	0.86	36,693	0.99	0.97	10,432	0.01	0.79	39,568	1.00
0.10	0.96	13,514	0.56	0.86	36,486	0.99	0.94	10,914	0.04	0.79	39,086	1.00

Table 6. Classification results of 2 language pairs en-fi, fi-en with different HTER threshold values

number of non-near-perfect MT. Identifying non-near-perfect MT seems to be an easier task than detecting near-perfect MT in all four language pairs. In terms of near-perfect MT detection, en-fi is the language pair that obtains the best MTQE results (both at precision and recall), followed by the language pairs: en-fi, fi-sv, and fi-en. When predicting non-near-perfect MT, en-fi is also the language pair which has better metrics, followed by fi-en, en-sv, and fi-sv. Both language pairs en-fi and en-sv share quite similar number of near-perfect/ non-near-perfect MT when different HTER thresholds are set and en-fi clearly obtains better results. There are some cases in which recall values are zero while precision values are 1.0; this is because the number of true positives is microscopic compared to the number of false negatives which makes the rounding calculation result in zero.

As shown in Table 6, the classification results look promising in both near-perfect and non-near-perfect MT for the language pair en-fi. In near-perfect MT, setting a threshold value of 0.0 or 0.01 yields 1.0 or 100% precision but the models correctly predict only 17% and 27% of the near-perfect MT, respectively. Allowing slightly larger HTER threshold values gradually increases the recall value up to 0.56 at the HTER threshold value of 0.1 while slowly reducing the precision value down to 0.96. On the other hand, both precision and recall values are high in the non-near-perfect MT detection case which implies also high corresponding f1 scores.

For the language pair fi-en, the model predicts near-perfect MT unpromisingly but non-near-pefect MT satisfactorily in the chosen HTER threshold range. For example, it only starts to obtain 0.97 precision and 0.01 recall at HTER 0.09 in near-perfect MT detection. Following the reduction of the number of non-near-perfect MT, the precision values slightly decline while the corresponding recall values constantly stay at 1.00.

In the near-perfect MT detection case, the model predicts poor results for the language pair fi-sv as both precision and recall values are zero when thresholds are less than 0.03, as depicted in Table 7. Although the model obtains 1.00 precision at HTER 0.04, the recall is zero because of a large number of false negatives compared

HTER threshold	fi-sv						en-sv					
	near-perfect MT			non-near-perfect MT			near-perfect MT			non-near-perfect MT		
	precision	count	recall	precision	count	recall	precision	count	recall	precision	count	recall
0.00	0.00	15,942	0.00	0.68	34,058	1.00	0.71	12,519	0.00	0.75	37,481	1.00
0.01	0.00	15,942	0.00	0.68	34,058	1.00	0.71	12,519	0.00	0.75	37,481	1.00
0.02	0.00	15,945	0.00	0.68	34,055	1.00	0.71	12,533	0.00	0.75	37,467	1.00
0.03	1.00	16,002	0.00	0.68	33,998	1.00	0.71	12,594	0.00	0.75	37,406	1.00
0.04	0.99	16,238	0.02	0.68	33,762	1.00	0.71	12,776	0.00	0.74	37,224	1.00
0.05	0.97	16,593	0.04	0.68	33,407	1.00	0.71	13,018	0.00	0.74	36,982	1.00
0.06	0.96	16,957	0.07	0.68	33,043	1.00	0.68	13,269	0.00	0.73	36,731	1.00
0.07	0.95	17,361	0.11	0.68	32,639	1.00	0.68	13,541	0.00	0.73	36,459	1.00
0.08	0.95	17,861	0.13	0.67	32,139	1.00	0.93	13,885	0.07	0.74	36,115	1.00
0.09	0.94	18,232	0.15	0.67	31,768	0.99	0.92	14,160	0.37	0.80	35,840	0.99
0.10	0.93	18,991	0.16	0.66	31,009	0.99	0.91	14,685	0.40	0.80	35,315	0.98

Table 7. Classification results of 2 language pairs fi-sv, en-sv with different HTER threshold values

to a few true positives. There is a slight improvement in the recall values when the threshold values change from 0.04 to 0.10 while precision value by contrast goes down steadily from 0.99 to 0.93. Not very surprisingly, the model shows positive and stable results in non-near-perfect MT where precision and recall values are around 0.68 and 1.00 in the given HTER range.

With the language pair en-sv and near-perfect MT case, the model only obtains promising results starting from HTER 0.08 in which the precision stays at 0.93 with the corresponding recall of 0.07. The precision values then continue a slightly declining trend, while recall values follow a quickly increasing trend from HTER 0.09 as the recall value rises by five times compared to 0.07 at HTER 0.08. For detecting non-near-perfect MT, the model again follows the similar trend, which is discussed in the other language pairs, the precision values fluctuate between 0.743 and 0.80, whereas recall values mostly end up at 1.00.

6 Conclusion

This thesis defined three research problems; 1) data volume impact on MTQE performance, 2) segment length impact on MTQE, and 3) whether it is possible to detect near-perfect translations of the four language pairs: en-fi, fi-en, fi-sv, and en-sv. Given these problems, the thesis comprehensively reviewed the related works on MT and MTQE fields, including DNN, the Transformer architecture, MT, MT evaluation, and MT quality estimation. After that, the thesis showed the methodology and the experiment implementation to tackle the research problems.

According to the experiment results with the TransQuest framework, increasing the training data volumes generally had a positive impact on MTQE performance in the four languages of interest. Moreover, a very short source segment (1-3 tokens) could be considered a reliable predictor for high MTQE performance (Pearson scores). Furthermore, setting different threshold values of HTER scores in each language pair allowed controlling the MT quality to meet different requirements of the translation projects. The best model in each language pair was good at predicting non-near-perfect translations, which may support the PE process by not sending bad quality MT to translators. Exceptionally, the model of the language pair en-fi obtained good results also in near-perfect translations. Predicting good quality of MT with very high precision and even low recall can benefit the PE process. The experiment results obtained by using the OpenKiwi framework were not fully available compared to TransQuest. The results showed that OpenKiwi was less sensitive to volume and more sensitive to data quality.

Overall, although this thesis work provided reliable answers for the three research questions, more research in MTQE is needed. In addition, there is room for improvement. For example, Bayesian classifier can be used in the last experiment of detecting near-perfect MT to obtain better results.

References

[1] C Kasparek. The translator's endless toil. The Polish Review, 28, 1983.

[2] Y. Gambier and L. van Doorslaer. Handbook of Translation Studies. Number nid. 1 in Handbook of Translation Studies. John Benjamins Publishing Company, 2010.

[3] Lucia Specia, Frédéric Blain, Marina Fomicheva, Erick Fonseca, Vishrav Chaudhary, Francisco Guzmán, and André F.T. Martins. Findings of the wmt 2020 shared task on quality estimation. 5th Conference on Machine Translation, WMT 2020 - Proceedings, 2020.

[4] Lucia Specia and Atefeh Farzindar. Estimating machine translation post-editing effort with hter. Proceedings of the AMTA-2010 Workshop Bringing MT to the User: Research on Integrating MT in the Translation Industry, 2010.

[5] F Alva-Manchego, L Specia, S Szoc, T Vanallemeersch, and H. Depraetere. Validating Quality Estimation in a Computer-Aided Translation Workflow: Speed, Cost and Quality Trade-off". Users and Providers Track, Summit XVIII, 2021.

[6] T. Boulmaiz, Mawloud Guermoui, and Hamouda Boutaghane. Impact of training data size on the lstm performances for rainfall–runoff modeling. Modeling Earth Systems and Environment, pages 1–12, 2020.

[7] Valerie Sessions and Marco Valtorta. The effects of data quality on machine learning algorithms. Proceedings of the 2006 International Conference on Information Quality, ICIQ 2006, 2006.

[8] Luisa Bentivogli, Arianna Bisazza, Mauro Cettolo, and Marcello Federico. Neural versus phrase-based machine translation quality: A case study. EMNLP 2016 - Conference on Empirical Methods in Natural Language Processing, Proceedings, 2016.

[9] Quinn Dupont. The cryptological origins of machine translation, from al-kindi to weaver. amodern, 01 2017.

[10] W John Hutchins. Machine translation : a concise history. Mechanical Translation, 13, 2007.

[11] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. Moses: Open Source Toolkit for Statistical Machine Translation. Association for Computational Linguistics, Prague, Czech Republic, June 2007.

[12] Nal Kalchbrenner and Phil Blunsom. Recurrent Continuous Translation Models. Association for Computational Linguistics, Seattle, Washington, USA, October 2013.

[13] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. Advances in Neural Information Processing Systems, 4, 2014.

[14] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 2014.

[15] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in Neural Information Processing Systems, 2017-December, 2017.

[16] Marcin Junczys-Dowmunt, Roman Grundkiewicz, Tomasz Dwojak, Hieu Hoang, Kenneth Heafield, Tom Neckermann, Frank Seide, Ulrich Germann, Alham Fikri Aji, Nikolay Bogoychev, André F. T. Martins, and Alexandra Birch. Marian: Fast Neural Machine Translation in C++. Association for Computational Linguistics, Melbourne, Australia, July 2018.

[17] Patrick Henry Winston. Artificial Intelligence (3rd Ed.). Addison-Wesley Longman Publishing Co., Inc., USA, 1992.

[18] DAVID McLEAN. Kevin gurney, an introduction to neural networks , university college london (ucl) press, 1997. isbn 1-85728-673-1 hb. £14.95. xi+234 pages. Natural Language Engineering, 6, 2000.

[19] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, Cambridge, MA, USA, 2016. Accessed: 22-11-2021.

[20] Seppo Linnainmaa. Taylor expansion of the accumulated rounding error. BIT Numerical Mathematics, 16:146–160, 1976.

[21] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin. A neural probabilistic language model. Journal of Machine Learning Research, 3, 2003.

[22] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernocký, and Sanjeev Khudanpur. Recurrent neural network based language model. Proceedings of the 11th Annual Conference of the International Speech Communication Association, INTERSPEECH 2010, 2:1045–1048, 01 2010.

[23] Philipp Koehn. Statistical Machine Translation. Cambridge University Press, 2009.

[24] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. 1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings, 2013.

[25] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder–decoder approaches. Proceedings of SSST 2014 - 8th Workshop on Syntax, Semantics and Structure in Statistical Translation, 2014.

[26] Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 - Conference Track Proceedings, 2015.

[27] Lucia Specia, Frédéric Blain, Marina Fomicheva, Chrysoula Zerva, Zhenhao Li, Vishrav Chaudhary, and André F. T. Martins. Findings of the WMT 2021 Shared Task on Quality Estimation. Association for Computational Linguistics, Online, November 2021.

[28] 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, 2019.

[29] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach. ArXiv, abs/1907.11692, 2019.

[30] Ridewaan Hanslo. Deep learning transformer architecture for named entity recognition on low resourced languages: State of the art results. arXiv, 2021.

[31] Maarit Koponen. Is machine translation post-editing worth the effort? a survey of research into post-editing and effort. The Journal of Specialised Translation, pages 131–148, 01 2016.

[32] Ana Guerberof. Productivity and quality in mt post-editing. Proceedings of MT Summit XII, 2009.

[33] Mirko Plitt and François Masselot. A productivity test of statistical machine translation post-editing in a typical localisation context. The Prague Bulletin of Mathematical Linguistics, 93:7–16, 01 2010.

[34] Ignacio Garcia. Is machine translation ready yet? Target. International Journal of Translation Studies, 22, 2010.

[35] Anne-Marie Robert. Vous avez dit post-éditrice ? quelques éléments d'un parcours personnel. The Journal of Specialised Translation 19, 22:29–40, 2013.

[36] Raksha Shenoy, Nico Herbig, Antonio Krüger, and Josef van Genabith. Investigating the Helpfulness of Word-Level Quality Estimation for Post-Editing Machine Translation Output. Association for Computational Linguistics, Online and Punta Cana, Dominican Republic, November 2021.

[37] Vilém Zouhar, Aleš Tamchyna, Martin Popel, and Ondřej Bojar. Neural machine translation quality and post-editing performance. arXiv, 2021.

[38] Hyeonseok Moon, Chanjun Park, Sugyeong Eo, Jaehyung Seo, and Heuiseok Lim. An empirical study on automatic post editing for neural machine translation. IEEE Access, 9, 2021.

[39] spansource.com. Neural machine translation and post-editing. https://spansource.com/services/neural-machine-translation-and-post-editing, 2022. Accessed: 30-01-2022.

[40] Satanjeev Banerjee and Alon Lavie. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. Association for Computational Linguistics, pages 65–72, June 2005.

[41] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. AMTA 2006 - Proceedings of the 7th Conference of the Association for Machine Translation of the Americas: Visions for the Future of Machine Translation, 2006.

[42] George Doddington. Automatic Evaluation of Machine Translation Quality Using N-Gram Co-Occurrence Statistics. HLT '02. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2002.

[43] Maja Popović. chrF++: words helping character n-grams. Association for Computational Linguistics, pages 612–618, September 2017.

[44] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. Association for Computational Linguistics, pages 311–318, July 2002.

[45] Carolina Scarton, Gustavo Paetzold, and Lucia Specia. Quality Estimation for Language Output Applications. The COLING 2016 Organizing Committee, Osaka, Japan, December 2016.

[46] Hannah Béchara, Constantin Orăsan, Carla Parra Escartín, Marcos Zampieri, and William Lowe. The role of machine translation quality estimation in the post-editing workflow. Informatics, 8(3), 2021.

[47] Marina Fomicheva, Shuo Sun, Erick Fonseca, Frédéric Blain, Vishrav Chaudhary, Francisco Guzmán, Nina Lopatina, Lucia Specia, and André F. T. Martins.

MLQE-PE: A multilingual quality estimation and post-editing dataset. arXiv preprint arXiv:2010.04480, 2020.

[48] Francisco Guzmán, Peng-Jen Chen, Myle Ott, Juan Pino, Guillaume Lample, Philipp Koehn, Vishrav Chaudhary, and Marc'Aurelio Ranzato. The flores evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english. arXiv, 2019.

[49] Rowda Juan. Machine translation: The basics of quality estimation. https://tech.ebayinc.com/engineering/machine-translation-the-basics-of-quality-estimation/, 03 2016. Accessed: 28-03-2022.

[50] Lucia Specia, Gustavo Paetzold, and Carolina Scarton. Multi-level translation quality prediction with quest++. Association for Computational Linguistics and The Asian Federation of Natural Language Processing, pages 115–120, July 2015.

[51] Julia Ive, Frédéric Blain, and Lucia Specia. Deepquest: a framework for neural-based quality estimation. In the Proceedings of COLING 2018, the 27th International Conference on Computational Linguistics, Sante Fe, New Mexico, USA, 2018.

[52] Fábio Kepler, Jonay Trénous, Marcos Treviso, Miguel Vera, and André F. T. Martins. OpenKiwi: An open source framework for quality estimation. Association for Computational Linguistics, pages 117–122, July 2019.

[53] Tharindu Ranasinghe, Constantin Orasan, and Ruslan Mitkov. Transquest: Translation quality estimation with cross-lingual transformers. Association for Computational Linguistics, 2021.

[54] Jiayi Wang, Ke Wang, Boxing Chen, Yu Zhao, Weihua Luo, and Yuqi Zhang. QEMind: Alibaba's Submission to the WMT21 Quality Estimation Shared Task. Association for Computational Linguistics, Online, November 2021.

[55] Yimeng Chen, Chang Su, Yingtao Zhang, Yuxia Wang, Xiang Geng, Hao Yang, Shimin Tao, Guo Jiaxin, Wang Minghan, Min Zhang, Yujia Liu, and Shujian Huang. HW-TSC's Participation at WMT 2021 Quality Estimation Shared Task. Association for Computational Linguistics, Online, November 2021.

[56] Chrysoula Zerva, Daan van Stigt, Ricardo Rei, Ana C Farinha, Pedro Ramos, José G. C. de Souza, Taisiya Glushkova, Miguel Vera, Fabio Kepler, and André F. T. Martins. IST-unbabel 2021 submission for the quality estimation shared task. Association for Computational Linguistics, pages 961–972, November 2021.

[57] Yuan Zhang and David Weiss. Stack-propagation: Improved Representation Learning for Syntax. Association for Computational Linguistics, Berlin, Germany, August 2016.

[58] 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 Transactions on Asian and Low-Resource Language Information Processing, 17:1–22, 09 2017.

[59] Hyun Kim and Jong-Hyeok Lee. A Recurrent Neural Networks Approach for Estimating the Quality of Machine Translation Output. Association for Computational Linguistics, San Diego, California, June 2016.

[60] Jeremy Howard and Sebastian Ruder. Universal language model fine-tuning for text classification. ACL 2018 - 56th Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference (Long Papers), 1, 2018.

[61] Jonas Pfeiffer, Andreas Rücklé, Clifton Poth, Aishwarya Kamath, Ivan Vulić, Sebastian Ruder, Kyunghyun Cho, and Iryna Gurevych. Adapterhub: A framework for adapting transformers. Association for Computational Linguistics, pages 46–54, 2020.

[62] Jihyung Moon, Hyunchang Cho, and Eunjeong L. Park. Revisiting round-trip translation for quality estimation. Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, EAMT 2020, 2020.

[63] Jörg Tiedemann and Santhosh Thottingal. Opus-mt - building open translation services for the world. Proceedings of the 22nd Annual Conference of the European Association for Machine Translation, EAMT 2020, 2020.

[64] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate Research, 30:79–82, 2005.

[65] Cort J. Willmott and Kenji Matsuura. Advantages of the mean absolute error (mae) over the root mean square error (rmse) in assessing average model performance. Climate Research, 30(1):79–82, 2005.

[66] David Freedman, Robert Pisani, and Roger Purves. Statistics (international student edition). Pisani, R. Purves, 4th edn. WW Norton & Company, New York, 2007.

[67] Tanaka Hiroyuki. pyter. https://github.com/roy-ht/pyter, 2012. Accessed: 22-11-2021.