TÉCNICO LISBOA

Energy Efficiency in Natural Language Processing

Pedro José Guerreiro Castelo Ramos

Thesis to obtain the Master of Science Degree in

Computer Science and Engineering

Supervisors: Prof. André Filipe Torres Martins
Dr. Ricardo Rei

Examination Committee

Chairperson: Prof. João António Madeiras Pereira
Supervisor: Prof. André Filipe Torres Martins
Member of the Committee: Prof. Bruno Emanuel Da Graça Martins

November 2023

Acknowledgments

I would like to take this opportunity to express my deepest gratitude and appreciation to the people who have played a significant role in the completion of my thesis. First and foremost, I extend my heartfelt thanks to my two supervisors, who have worked closely with me throughout this research. I am truly grateful for having professors André Martins and Ricardo Rei as mentors during this thesis. Their academic expertise was critical in shaping the trajectory and success of my thesis. Their dedication was felt in the weekly meetings and in their readiness to respond to any of my questions throughout the weeks and months. Finally, their insights into my area of study were crucial to the accelerated learning curve I experienced during this thesis. Next, I want to thank Unbabel and everyone involved in our weekly meetings, these moments were an opportunity to learn from others and to have a broader view of the field, as well as additional feedback and help from a more vast group of people.

This challenging journey was smoother due to everyone else in my life who was not directly involved in this thesis but played a crucial role in providing me with the support and encouragement to bring this thesis to a successful completion. I would like to start by expressing my profound gratitude to my parents and brother, whose constant support and encouragement have been vital to my academic and personal lives. They were the people who sparked my curiosity at a young age and are the driving force behind my accomplishments. I would also like to acknowledge my grandparents, cousins, aunts, and uncles for the special moments we share and their belief in my capacity to pursue my passions and embrace new challenges.

Furthermore, I would like to extend my gratitude to my friends. Just being alongside them makes me happy, and they encourage me to do more and be a better person every day. I'm grateful to have Sofia, Wemans, and other friends from high school. Sancha, Soares, and many others from my bachelor's degree. João, Maria, Serralheiro, Sara, Matilde, Bernardo, Catarina, José, and so many others from Junitec.

To everyone who has shaped or continues to shape my life, I am truly fortunate to have you. I look back on my academic and personal life as one full of great memories and fascinating learnings that have made me the person I am today.

Abstract

The rise of large language models (LLMs) has led to an ever-growing demand for computational power in natural language processing (NLP). This increase in demand is a serious environmental concern since larger models imply an increase in energy consumption. In addition, these models translate into higher training costs, which means less equitable access for researchers, and longer training and inference times, which invalidates various applications where there is a restriction in computing power or where one wants to perform a large number of inferences. In this work, we focus on the evaluation of machine translation, a task that uses LLMs to assess the quality of a given translation. Since this task is computationally expensive, it results in high energy consumption, mostly during inference. At the same time, it is a task that demands low inference times, which generally means trading off model performance for a smaller and faster model. Our main contribution is the introduction of a reference-free machine translation evaluation model, which despite being 2.3 times larger, is 2.1 times faster than our previous work on the reference-based COMETINHO. This new model shows large improvements in correlation with human evaluation using multidimensional quality metrics (MQM) scores, improving by 27 Kendall's Tau points (9%) when compared to COMETINHO and only 11 Kendall's Tau points (3%) below the state-of-the-art at the start of the development of this thesis (COMETKIWI-22).

Keywords

Model Compression; Knowledge Distillation; Energy Efficiency; Evaluation of Machine Translation

Resumo

O surgimento de large language models (LLMs) levou a uma procura cada vez maior por poder computacional em NLP. Este aumento de procura é uma séria preocupação ambiental, uma vez que modelos maiores implicam um aumento no consumo de energia. Adicionalmente, estes modelos traduzem-se em maiores custos de treino, o que significa um acesso menos equitativo para os investigadores, e maiores tempos de treino e inferência, o que invalida diversas aplicações onde existe uma restrição de poder computacional, ou se pretende realizar um grande número de inferências. Nesta tese, focamo-nos na avaliação de tradução automática, um problema que utiliza LLMs para avaliar a qualidade de uma determinada tradução. Como esta tarefa é computacionalmente intensiva, resulta em alto consumo de energia, principalmente durante a inferência. Ao mesmo tempo, é uma tarefa que exige tempos de inferência baixos, o que geralmente significa balancear a necessidade de alta precisão com a necessidade de ter um modelo menor e mais rápido. A nossa principal contribuição é a introdução de um modelo de avaliação de tradução automática que não necessita de utilizar referências, e apesar do modelo ser 2.3 vezes maior, é 2.1 vezes mais rápido que o nosso anterior trabalho no COMETINHO com referências. Este novo modelo mostra uma evolução significativa na correlação com avaliação humana utilizando MQM, melhorando a correlação em 27 pontos no coeficiente Kendall's Tau (9%) quando comparado com o COMETINHO e apenas 11 pontos (3%) abaixo do estado da arte no início do desenvolvimento desta tese (COMETKIWI-22).

Palavras Chave

Compressão de modelos; Distilação de conhecimento; Eficiência energética; Avaliação de tradução automática

Contents

List of Figures

List of Tables

Acronyms

ANN	artificial neural network
NLP	natural language processing
MT	machine translation
LLMs	large language models
DA	direct assessment
HTER	human translation edit rate
MQM	multidimensional quality metrics
PE	post-edited translation
QE	quality estimation
PEFT	parameter-efficient fine-tuning
LSTM	long short-term memory

1

Introduction

Contents

1.1 Motivation

In recent years the field of natural language processing (NLP) has seen significant improvements largely due to the advancements in deep learning [Bahdanau et al., 2014, Luong et al., 2015, Vaswani et al., 2017].

The introduction of transformer-based models [Vaswani et al., 2017] reduced substantially the training time compared to previous neural architectures due to the highly parallelizable architecture. This efficiency gain coupled with ever-growing available data means that it is possible to train a model with more data in less time, creating a space for self-supervised large pre-trained models such as BERT [Devlin et al., 2018a] and GPT [Radford et al., 2018]. Moreover, these improvements do not seem to be reaching a plateau anytime soon. As described in GPT-3 [Brown et al., 2020], an increase in the number of model parameters or the number of examples has a positive correlation with the quality of the model, with no appearance of the trend slowing down, as shown in Figure 1.1.

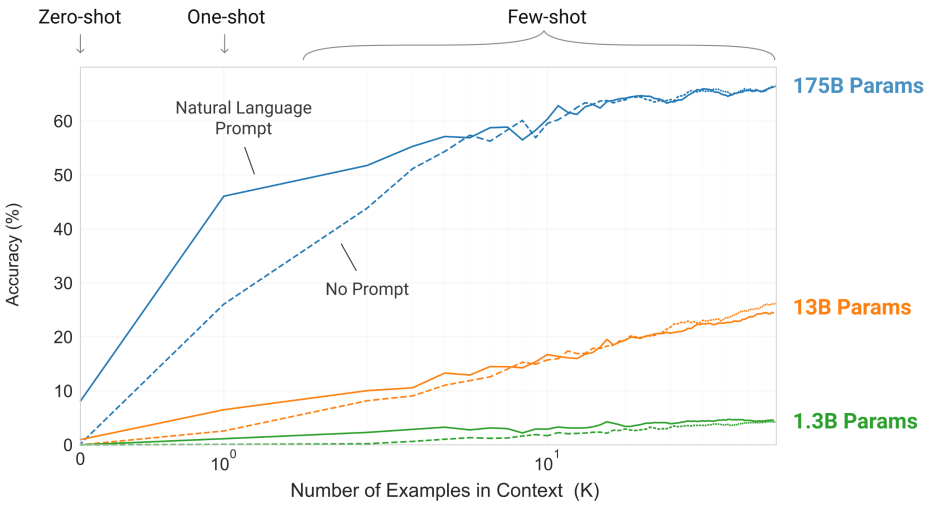


Figure 1.1: "Larger models make increasingly efficient use of in-context information". Taken from [Brown et al., 2020]

Computing power is still increasing at a fast pace, but we are no longer following Moore's Law [Moore, 1998]. The computing power required by the most recent models doubles every 5.7 months [Sevilla et al., 2022], making it more expensive to acquire the needed hardware to use the state-of-the-art. Regardless of the increase in available data, the collection, processing, and labeling of such data is getting more difficult and time-consuming as the required data to feed large models increases. Despite decreasing energy costs, especially green energy, the share of renewables worldwide remains only 28.7% [IEA, 2022]. Although it is growing at a fast pace, it is still not enough to keep the exponential growth of machine learning models environmentally sustainable. This growth will imply increased training

costs, simply training a model with 1.5B parameters can cost from 80k dollars up to 1.6M dollars [Sharir et al., 2020], making it harder for the entire research community to contribute to state-of-the-art progress dominated by large tech companies. Another downside of very large transformer models is that they become impractical for use cases where there are low memory or low inference time requirements. Therefore, it is imperative to develop more efficient models by compressing existing ones into smaller and more accessible ones that can be used without restriction while maintaining a performance close to state-of-the-art [Treviso et al., 2023]. Therefore, energy efficiency has become a hot topic in NLP and a core pillar of responsible AI, an area of increasing importance to ensure ethical, transparent, and accountable model design.

The area in which we chose to focus this thesis was the evaluation of MT. It is a significant use case since the metrics used to evaluate such MT systems are vital to ensuring the quality of the translations and minimizing critical errors. To achieve increasing levels of performance, these models have been using larger and larger pre-trained encoders. On the other side, when one wants to extensively use a certain metric, for example, to apply N-best reranking to several translation hypotheses [Fernandes et al., 2022], it becomes expensive and ineffective to use increasingly larger models. For that reason, many older and faster metrics are still popular, despite having a low correlation with human evaluation. In this thesis, we develop a metric that is distilled from a larger teacher to achieve a fast model with a much better correlation with human evaluation.

1.2 Main Contributions

Model compression is the process of reducing the size of machine-learning models while maintaining similar accuracy, enabling models to achieve faster inference times and thus reducing their energy consumption. We achieved preliminary results on MT evaluation in our work in COMETINHO [Rei et al., 2022a], where we applied several model compression techniques to COMET, an MT evaluation model. In this thesis, the work is further extended to achieve a COMETINHO-like model that is faster and has a performance similar to previous COMET models while being a reference-free version of COMETINHO, unlocking further potential applications. During the study of model compression techniques to improve COMETINHO, we answered the following research questions:

- Why should the next version of COMETINHO be reference-free?
- How many times more data is needed to train a student on teacher labels to achieve the same accuracy as a model trained using DA scores?
- To what extent can better teacher models reduce the amount of data that the student model needs to be trained on?

- Can labels from sufficiently good teachers be better than human evaluations for fine-tuning?

Our main contribution is the introduction of a reference-free COMETINHO, which despite being 2.3 times larger, is 2.1 times faster than the reference-based COMETINHO and shows large improvements in correlation with human evaluation using MQM scores, by 27 Kendall's Tau points (9%) when compared to COMETINHO and only 11 Kendall's Tau points (3%) below the state-of-the-art (COMETKIWI-22) at the start of development of this thesis.

1.3 Outline

The next chapters are organized as follows: Chapter 2 compiles the key topics that provide the background necessary to understand the work developed in this thesis and also summarizes the existing related work. Chapter 3 details the implement systems developed and explains the approach we followed to conduct this research. Chapter 4 defines the evaluation criteria we used to assess the work and presents the results achieved in the different experiments, both in terms of performance and efficiency. Finally, Chapter 5 summarizes the key findings and proposes avenues for future work.

2

Background

Contents

This chapter provides background for the topics covered in this thesis and explains the main related work. First, we present a brief introduction to neural networks, text representation, and large pre-trained models. Then, we explain the concepts of model compression and fine-tuning. Lastly, we cover the topics of machine translation and evaluation of machine translation, where our work in this thesis is focused.

2.1 Artificial Neural Network

The artificial neural network (ANN) is the bedrock of deep learning. It consists of a collection of nodes (neurons), connected by weights and biases. These nodes are organized in several layers, each with a matrix of weights and a vector of biases. To get an output given an input, a feedforward operation in equation 2.1 is performed. The input, represented as the vector $A^{[0]}$, is multiplied by the input layer $W^{[1]}$, and then the bias $b^{[1]}$ is added.

$$Z^{[1]} = W^{[1]}A^{[0]} + b^{[1]}. \tag{2.1}$$

To feed the results to the next layer, a non-linear activation function g is applied, as shown in equation 2.2. This function must be non-linear to introduce non-linearity between layers, allowing the model to predict more complex tasks that cannot be solved using simply a linear combination of the inputs.

$$A^{[1]} = g(Z^{[1]}). \tag{2.2}$$

This process is repeated in the next hidden layers until we reach the output layer. The choice of activation function depends on the type of ANN used, the most common one is the sigmoid function, defined as follows:

$$\sigma(x) = \frac{1}{1 + e^{-x}}. \tag{2.3}$$

During the design of an ANN, an important aspect to take into consideration is if we have a regression or classification task. Regression is the task of predicting a single real value in a continuous space, while classification predicts a single class out of a discrete set of possible classes. The classification task needs a different activation function in the output layer than the activation function used in its hidden layers. Since we define the probability for each class and then choose the one with the higher probability, the commonly used activation to transform the output layer results in probabilities is the $\mathrm{softmax}$ function, defined as:

$$q_i = \frac{\exp(z_i/T)}{\sum_j \exp(z_j/T)}, \tag{2.4}$$

where q_i is a probability, z_i is a logit, and T is a temperature that controls the smoothness of the output distribution. Once we get the output of our model, the initial result is no better than random guessing since the weights and biases are initialized randomly. To achieve better outputs during the training process, an optimization algorithm is used, an example is the gradient descent algorithm, shown in equation 2.5. For a given loss function f that we want to find a local minimum, for each iteration, the algorithm calculates the gradient of a point a_n, denominated $\nabla f(a_n)$, and subtracts from a_n that gradient multiplied by a small learning rate constant η to obtain the next point a_{n+1}.

$$a_{n+1} = a_n - \eta \nabla f(a_n). \tag{2.5}$$

The gradient calculates the direction in which the function locally increases faster in value. In order for the next iteration a_{n+1} to decrease in the fastest direction possible, we need to subtract the gradient multiplied by a sufficiently small learning rate. The larger the learning rate, the faster we can potentially reduce the loss, but the probability of the locality still being valid reduces significantly, so we most likely get a point a_{n+1} with a larger loss than a_n.

To know what we want to optimize for, we have to define a loss function, which compares the output model with the desired output and gives an absolute measure of how far the output is from the desired label. The example in equation 2.6 is the mean squared error loss.

$$MSE = \frac{1}{n} \sum_{i=1}^{n} (\hat{y}_i - y_i)^2, \tag{2.6}$$

where $\hat{y}_i$ is the value predicted by the model at prediction i, y_i is the correct output at instance i, and n is the number of predictions over which we are calculating the loss since this update is usually done in a batch.

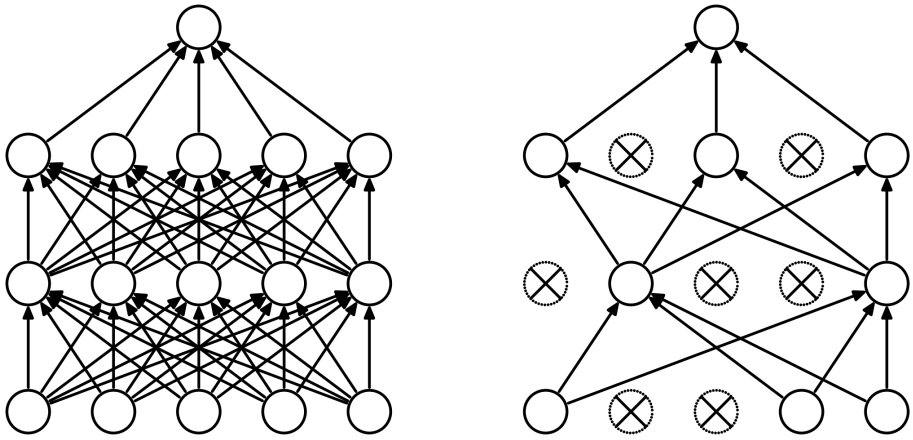
To propagate the error efficiently, we use the backpropagation algorithm, which calculates the derivative layer by layer, starting with the output layer. Propagating the error through the various weights and biases, updating them to better predict a single sample or a batch of samples.

With improvements in computational capacity over several years, the models increased their number of layers and parameters. The term deep learning was coined to refer to ANNs with a large number of hidden layers. With deep learning, a problem with the sigmoid activation function and other functions, such as the hyperbolic tangent, became apparent. In the backpropagation algorithm, the value of the derivative decreases layer by layer until it approaches a value close to zero. To address the vanishing gradient problem, new activation functions were proposed, such as the ReLU [Nair and Hinton, 2010] function shown in equation 2.7.

$$f(x) = \begin{cases} x & \text{if } x > 0 \\ 0 & \text{otherwise} \end{cases}. \tag{2.7}$$

This function is capable of mitigating the vanishing gradient problem since it doesn't have a maximum activation value.

With deep learning, the new limiting factor started to be the size of the labeled training sets in which the models are trained. One can train the model for several epochs on the same data, which tends to create overfitting, meaning that the model fits too closely to the training data and starts generalizing worse when predicting the labels in the test set. A solution to this problem is to use dropout [Srivastava et al., 2014], a method that, for each training step, randomly drops weights that will not be updated as shown in Figure 2.1, mitigating the effect of overfitting.



(a) Standard Neural Net (b) After applying dropout.

Figure 2.1: On the left is a standard neural network. On the right is a sparse neural network produced by applying dropout to the network on the left. Crossed units have been dropped. Taken from [Srivastava et al., 2014].

To address the lack of labeled data, the concept of self-supervised learning was introduced. Instead of needing labeled data, the data is masked, and the dataset labels are those missing data points. In the context of NLP, these data points are words in any form of text representation explained in the next Section.

2.2 Text Representation

For an ANN to understand words and sentences, it is necessary to first convert them to a machine-interpretable representation, since as explained in Section 2.1, ANNs receive as input a vector of real values $A^{[0]}$.

The first step is called tokenization, which consists of separating a sentence or document into words, characters, or sub-words. Then define the vocabulary Σ that needs to be encoded, we can use the full English dictionary or a large set of words or sub-words based on a preprocessing of the data we want to analyze, avoiding this way out-of-vocabulary words. We can then map the tokens from a vocabulary Σ to a d-dimensional vector using either discrete or distributed representations.

2.2.1 Discrete Representations

One-Hot Encoding

The one-hot encoding is a method that can be used when the size of the vocabulary ($|\Sigma|$) is not very large. It uses a vector with dimension $d = |\Sigma|$, and for each word w_i, the one-hot encoding is a vector v_i of size d:

$$v_i[j] = \begin{cases} 1 & \text{if } i = j \\ 0 & \text{otherwise} \end{cases}, \forall_{0 \leq j < d}. \tag{2.8}$$

If one encodes an entire sentence of size s using one-hot encoding, each word would need a vector of size d, and thus we would have a matrix of size $s \times d$.

Bag-of-words

The bag-of-words representation (BOW) is a different method of encoding text that uses a vector also with the dimension of the vocabulary as in one-hot encoding, but to encode an entire sentence instead of a single word, which makes it more space-efficient than one-hot encoding. As in one-hot encoding, each word is mapped to an index, the method then counts the number of times each word appears in the sentence and assigns that frequency value to the respective word index. A problem with this approach is that it ignores the positional information of the words, by losing the connection between them, it is harder for an ANN to grasp the meaning of the sentence.

TF-IDF

The TF-IDF method tackles a second problem of the bag-of-words, which is the overweight of high-frequency tokens such as stop-words and the disregard of low-frequency tokens that actually make a difference in a certain sentence.

TF-IDF has two factors, the first ($TF(w, d)$) is the frequency of the word w in the document d as explained in the bag-of-words method. The second factor is the inverse document frequency (IDF), defined as follows:

$$IDF(w) = \log(\frac{N}{df(w)}), \tag{2.9}$$

where N is the total number of documents and $df(w)$ is the frequency of documents containing the word w. TF-IDF is simply $TF(w, d) * IDF(w)$, by multiplying the bag-of-words method with the inverse document frequency, TF-IDF can assign more importance to words that are unusually frequent in a given document compared to the remaining documents and assign less value to words that are common in a given document simply because they are common in all documents.

2.2.2 Distributed Representations

Although TF-IDF is good at identifying the importance of each word in a sentence or document, the encoding still scales linearly with the size of the vocabulary and the method does not identify semantically similar words, for example, by considering synonyms as completely independent words. Distributed representations solve these two issues by having the information about a word distributed through the encoding vector.

Global Matrix Factorization Methods

Global matrix factorization is a family of factorization methods that obtain low-rank approximations of the entire document. An example is the hyperspace analog to language (HAL) [Lund and Burgess, 1996], which for a vocabulary of size d, computes a matrix of size $d \times d$ where the coordinate (i, j) indicates the level of presence of the token i in the context of the token j, offering a simple way to find associations between tokens. However, this method still uses a sparse matrix that scales quadratically with the vocabulary size, meaning that it is infeasible to use with large vocabularies. Another method is latent semantic analysis (LSA) [Deerwester et al., 1990], which for a vocabulary of size d and n documents, computes a matrix of size $d \times n$ and the value 1 or 0 is added based on whether or not the token i is present in document j.

Local Context Window Methods

Local context window methods learn word representations by focusing on specific context windows. An example is Word2Vec [Mikolov et al., 2013a], which uses two local context window model architectures. It uses an ANN with a single hidden layer, and the embedding of a certain word is the output of the hidden layer.

The method proposes two model architectures illustrated in Figure 2.2, the continuous bag-of-words (CBOW) and the Skip-gram. The CBOW is an architecture that predicts a given word based on the surrounding words. It accomplishes that by projecting all words into the same position, hence the name

bag-of-words, since the positional information is lost. Then, a log-linear classifier with the previous four words and the next four words is used to predict the middle word. The Skip-gram model predicts the surrounding words based on a given word, the larger the span of words predicted, the better the quality of the resulting word embeddings, but the training complexity increases.

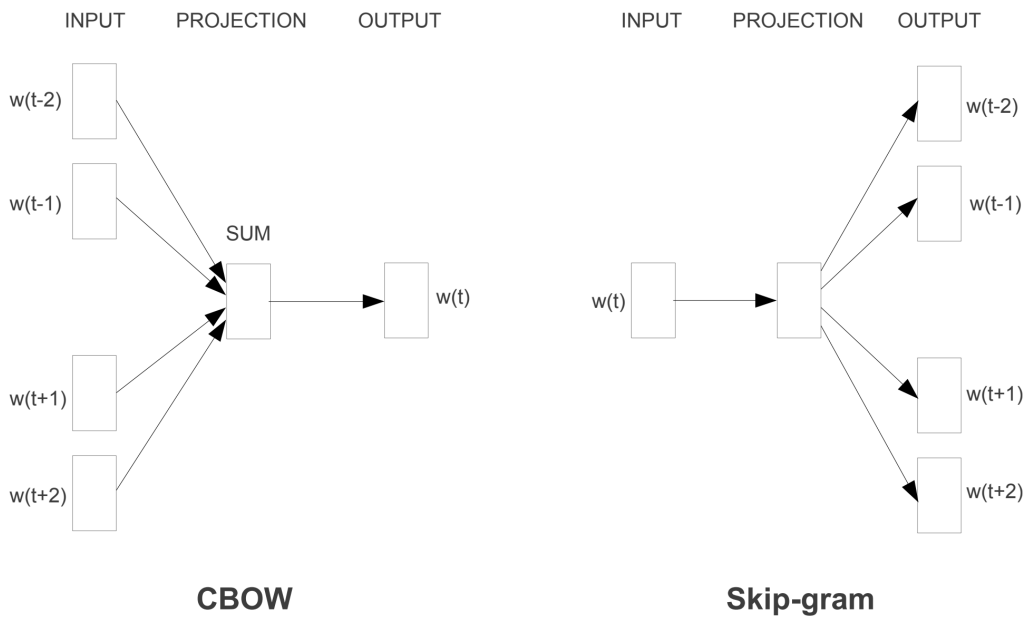


Figure 2.2: CBOW and Skip-gram architectures. Taken from [Mikolov et al., 2013a]

A remarkable advantage of this method compared to count-based techniques is the ability to perform simple vector algebraic operations. As shown in [Mikolov et al., 2013b], $vector(\text{"}King\text{"}) - vector(\text{"}Man\text{"}) + vector(\text{"}Woman\text{"})$ results in a vector that is close to $vector(\text{"}Queen\text{"})$.

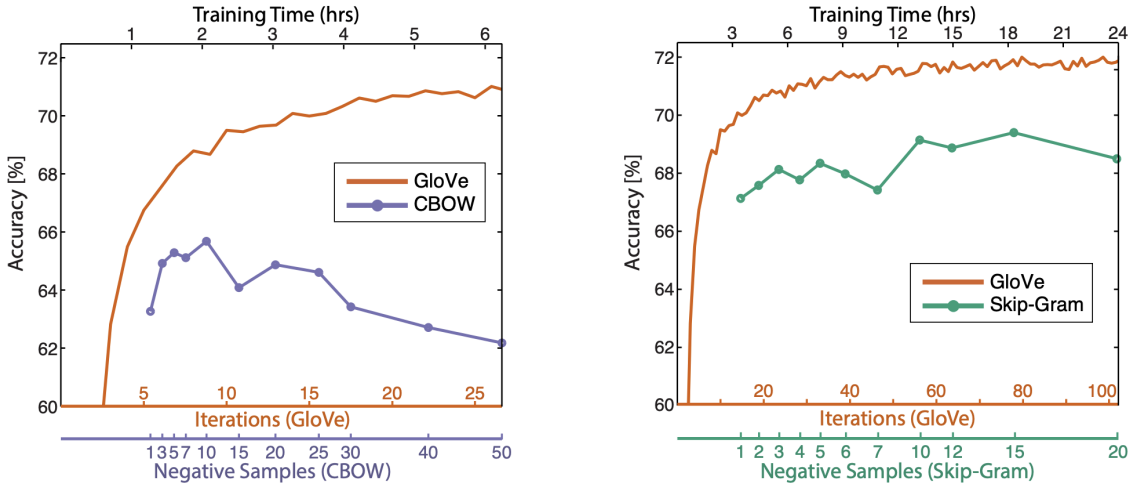
GloVe

A problem with Word2Vec is that it learns the semantics of a word based on the surrounding words, and the model ends up with a local understanding of words. GloVe [Pennington et al., 2014], an abbreviation for Global Vectors, is a method that addresses this problem by combining the advantages of global matrix factorization and local context window methods. This is accomplished by using a loss function that enforces the following relationship:

$$v_i^\top v_j = \log P(i|j), \tag{2.10}$$

where v_i and v_j are learned word vectors, and $P(i|j)$ is a globally computed statistic that represents the probability of word j appearing in the context of word i in the entire corpus. As shown in Figure 2.3, the

performance difference of GloVe compared to the Word2Vec architectures is significant.



(a) GloVe vs CBOW (b) GloVe vs Skip-Gram

Figure 2.3: GloVe compared to CBOW and Skip-gram architectures. Taken from [Pennington et al., 2014].

Contextual Word Embeddings

A problem with the previous solution remains the inability to identify the meaning of a polysemous word in its context. An example is the word ”bank” which can have different meanings depending on the sentence in which it is used. Contextual embeddings offer a solution for this by using language models to generate embeddings. We explain in more detail large language models (LLMs) in the next Section 2.3. An example of this method is the ELMo architecture [Peters et al., 2018a], which solves this issue by having a many-to-one relationship between vector representations and the word they represent. This way, according to the sentence context, a different vector can be assigned to the polysemous word.

2.3 Large Pre-Trained Language Models

LLMs are trained using self-supervised learning on a massive dataset to capture in their multiple hidden layers the relationship between the sentence tokens and thus learn the intrinsic properties of the raw text.

This enables them to perform well on a wide variety of NLP tasks through multiple techniques, including prompting [Liu et al., 2021], transfer learning, and fine-tuning. This is of particular significance for tasks where labeled data is scarce since the pretrained model can leverage the knowledge already acquired during the pre-training phase to learn the downstream task. They are also capable of few-shot learning, which means training models to perform tasks using very few examples, and even zero-shot

learning, where the model is given a description of the task and no training data and is capable of generalizing its knowledge of the task at hand.

All these advancements in LLMs emerged when it became possible to scale them several orders of magnitude, due to the parallelization capabilities of the transformer model.

2.3.1 Transformer Model

The transformer model [Vaswani et al., 2017] is a model architecture that uses self-attention to perform training in a parallel fashion, leveraging the power of GPUs. Unlike other architectures like recurrent neural networks (RNNs) [Rumelhart et al., 1985] and LSTM networks [Hochreiter and Schmidhuber, 1997], the transformer architecture is not used one step at a time. Instead, the whole sentence is fed into it at once.

The original architecture is shown in figure 2.4 and consists of an input and output embedding layer, followed by a sequence of N encoder-decoder layers made of a self-attention mechanism and a feed-forward neural network. After the last decoder, a learned linear transformation and the $\mathrm{softmax}$ function are applied to obtain the output probabilities of the next token.

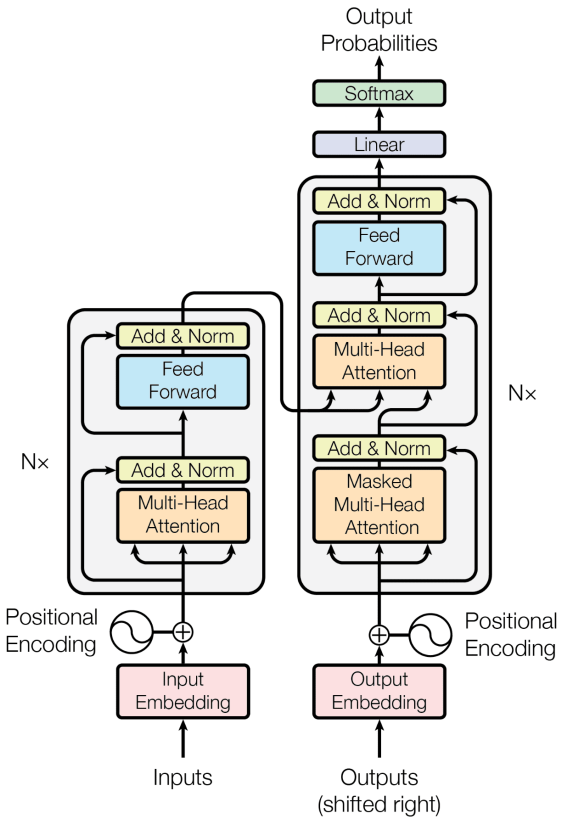


Figure 2.4: The encoder-decoder structure of the Transformer architecture. Taken from [Vaswani et al., 2017]

Input and output embeddings

The first step in the architecture consists of converting the input and output tokens to vectors using an embedding layer. The input, output, and linear transformation before $\mathrm{softmax}$ all share the same weight matrix. Before feeding the embeddings to the encoder and decoder layers, a positional encoding is added to give the model a way to use the sequence order to better understand the embeddings.

Encoder and decoder

The encoder in the original architecture is composed of 6 layers, although the number of layers is variable in many subsequent LLMs. Each layer has a multi-head self-attention sub-layer and a fully connected feedforward network.

The self-attention mechanism works by attending to different parts of the input sequence, capturing long-range dependencies between input elements. This mechanism is defined as follows:

$$Attention(Q, K, V) = \mathrm{softmax}(\frac{QK^T}{\sqrt{d_k}})V, \tag{2.11}$$

where Q, K, and V are matrices representing the queries, keys, and values, respectively, and d_k is the dimension of both the queries and keys. The attention mechanism computes a weighted sum of the values, where the weights are computed based on the similarity between the queries and keys. Being a multi-head self-attention layer means that the self-attention mechanism is repeated h times with different learned linear projections. Then the output values are concatenated and projected again. Finally, this projection is fed to the feedforward network to obtain the output of the encoder.

The decoder starts with a multi-head attention sub-layer that receives the output embeddings, then it has another sub-layer that receives the output of the last encoder layer, and finally a feedforward sub-layer.

2.3.2 Types of pre-trained models

The transformer model was originally developed for MT, but the architecture is adaptable to many NLP tasks and soon was adopted to develop large pre-trained language models with different topologies, in common they can convert natural language into embedding vectors. These vectors represent not only the meaning of a word or token but also their meaning in the context of the sentence or document.

A masked language model, such as Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2018a], is trained to predict a word or token in a sentence. In the case of BERT, during training, 15% of the tokens in the input sentence are ”masked”, meaning they are replaced with a special token `[MASK]` with 80% probability, by a random token 10% of the time, and remain unchanged with 10% probability. Then the language model is trained to predict the original value. BERT is also trained on

a second task called next sentence prediction, where the model is given two sentences A and B and asked to predict if the sentence A is followed by the sentence B. For that, in the training set, sentence B is the actual next sentence 50% of the time and a random sentence otherwise. These two tasks require the model to receive input embeddings that are the sum of the token embeddings with segmentation embeddings, that distinguish between sentence A and B, and positional embeddings. The representation of these input embeddings is seen in Figure 2.5.

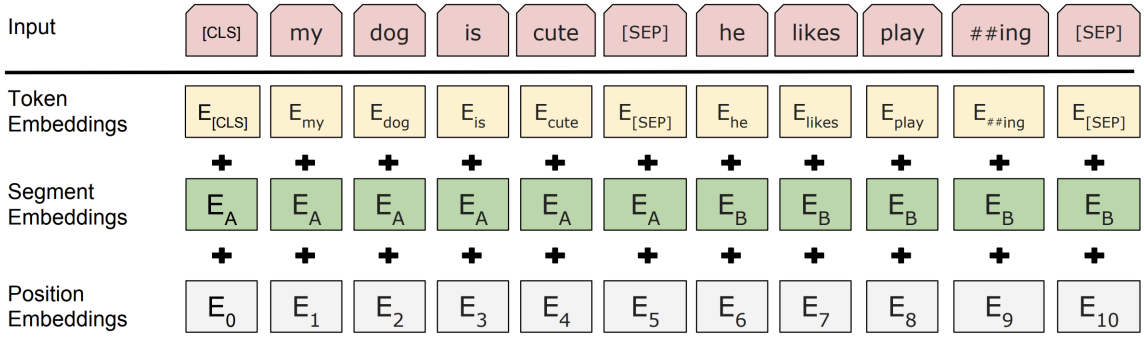


Figure 2.5: BERT input representation. Taken from [Devlin et al., 2018a].

These masked language models can then be used in downstream NLP tasks mainly related to classification and regression.

Autoregressive language models (e.g., Generative Pre-trained Transformers (GPT) [Radford et al., 2018]), on the other hand, are mainly used for generation tasks. They are trained to predict the next word in a sentence based on the previous n words [Bengio et al., 2000]. In the case of GPT, the model is trained to maximize the likelihood L_1:

$$L_1(\mathcal{U}) = \sum_i \log P\left(u_i \mid u_{i-k}, \ldots, u_{i-1}; \Theta\right), \tag{2.12}$$

where $\mathcal{U} = \{u_1, \ldots, u_n\}$, k is the size of the context window, and P is the conditional probability modeled using an ANN with parameters Θ [Radford et al., 2018].

An encoder-decoder language model (e.g., Text-to-Text Transfer Transformer (T5) [Raffel et al., 2020]) is a type of model that is the most similar to the original transformer architecture, although the two other types also leverage the transformer. T5 has two main differences compared to the transformer model, it doesn't use layer normalization bias, and it uses relative position embeddings instead of fixed embeddings, meaning that a different learned embedding is produced according to the offset between the key and query.

2.4 Model Compression

Model compression comprises several techniques to reduce the size of a model without significantly degrading accuracy. In this section, we will cover the main techniques for model compression and focus on knowledge distillation, since that is the approach our work in this thesis is centered on.

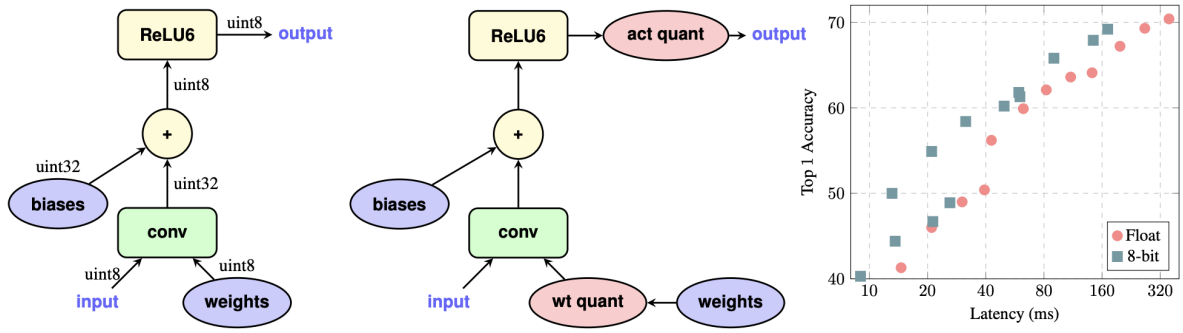
2.4.1 Quantization

Quantization is the process of reducing the size occupied by each parameter of the model, usually by converting each parameter from a floating-point into a fixed-point value [Guo, 2018]. The float-point value is most commonly represented using a 32-bit float, the most conservative quantization effort would be to use mixed-precision, meaning that the model still uses the float-point representation where it is more sensible to changes in precision and reduces the size in the rest. That reduction can range from half-precision (16-bits) [Micikevicius et al., 2017], to integers with 8-bits [Dettmers et al., 2022], and even ternary and binary representations [Bai et al., 2020]. Quantization can help reduce memory and computational costs, but it also leads to a loss in accuracy, especially if the precision of the model is reduced too much. As a result, to maintain good performance, it is important to be careful when developing and training a model that will be quantized.

The simplest quantization method is post-training quantization, one can do static or dynamic quantization. The former needs to store the model in a quantized manner and requires calibration of the parameters. The latter stores the weights in quantized form and the activations in float points and quantizes the activations during inference. The other method is quantization-aware training, achieved by training the model using floating-point weights and activations but adding extra layers that simulate the quantization of weights (”wt quant”) and activations (”act quant”). This way, the model can be trained using optimization algorithms developed for floating points while still restricting the values to be close to 8-bit integers [Jacob et al., 2018]. Then, at inference, the model is stored in the approximate form using only 8-bit integers, reducing the size four times. As seen in figure 2.6, the accuracy for a given latency is better using 8-bit integers.

2.4.2 Pruning

Pruning is a technique that removes the least necessary weights from a neural network. First introduced in [LeCun et al., 1989], the method used second-derivative information to determine the trade-off between network size and accuracy. Merely substituting a less important weight with the value 0 and freezing that weight may not be sufficient to reduce the GPU inference time since that doesn’t reduce the computation cost associated with the feedforward operations.



(a) Integer-arithmetic-only inference (b) Training with simulated quantization (c) ImageNet latency-vs-accuracy tradeoff

Figure 2.6: a) Integer-arithmetic-only inference of a convolution layer. b) Training with simulated quantization of the convolution layer. c) Improved latency-vs-accuracy trade-off using 8-bit compared to float. Taken from [Jacob et al., 2018].

Another approach used in deep neural networks is layer pruning. Since most information is encoded in the middle hidden layers, some of the last layers can be removed without a significant loss of accuracy [Dong et al., 2017]. This approach can also be done dynamically by dropping layers during inference [Fan et al., 2019].

Given the importance of transformers in NLP, studies have also been conducted to prune this specific architecture. An example is removing the least significant attention heads [Voita et al., 2019], as shown in Figure 2.7, only some heads do the heavy lifting, which means the remaining ones can be pruned without significant performance degradation.

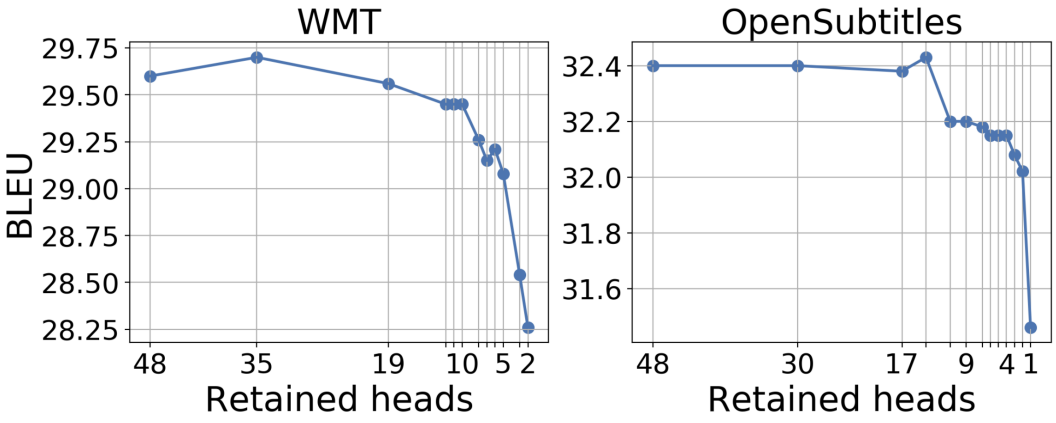


Figure 2.7: "BLEU score as a function of number of retained encoder heads (EN-RU). Regularization applied by fine-tuning trained model". Taken from [Voita et al., 2019].

2.4.3 Knowledge Distillation

Knowledge distillation consists of training a smaller student model using the knowledge of a larger teacher model or an ensemble of teachers. The simplest way to perform this knowledge transfer is for the student to mimic the outputs of the teacher, this is known as response-based distillation. One can define a distillation loss over hard or soft targets. Hard targets consist of simply the output of the model, while soft targets consist of the class probabilities generated by the teacher model, usually through the $\mathrm{softmax}$ function 2.4.

As explained in [Hinton et al., 2015], soft targets allow the student model to learn much more information per training sample since the student learns not only what is the most probable class but also the probability of the remaining classes. This results in less variance in the gradient, allowing for a higher learning rate and often needing much less data to achieve similar results. The distillation loss is then calculated using those hard or soft targets, an example is the cross-entropy loss L_{ce} defined in equation 2.13 and used in many models such as DISTILBERT [Sanh et al., 2019].

$$L_{ce} = \sum_i t_i * \log\left(s_i\right), \tag{2.13}$$

where t_i is the teacher probability and s_i is the student probability.

Feature-based distillation, on the other hand, determines a distillation loss for each layer so that the student model learns to discriminate the same specific features the teacher model is identifying in each layer.

While the previous examples focused on the distillation of task-specific models, with the popularity of the transformer architecture, there are several contributions to the distillation of pre-trained models. These models can then be fine-tuned to downstream tasks [Treviso et al., 2023], examples are DISTIL-GPT, DISTILBERT [Sanh et al., 2019], and MINILM [Wang et al., 2020]. The latter uses a method called deep self-attention distillation, where the student model is trained by deeply mimicking the self-attention behavior of the last transformer layer of the teacher. For that, the method uses several losses, as seen in Figure 2.8.

2.5 Fine-Tuning

Fine-tuning is a method used to adapt a model to a new task. Typically, a pre-trained model that was trained on a general task is then adapted to a downstream task. This process involves training the model with new data that reflects the domain of the downstream task. If the entire set of model parameters is trained, the method is called full fine-tuning. However, that is a costly process if one wants to fine-tune a model for many downstream tasks. Additionally to not being an energy and time-efficient method,

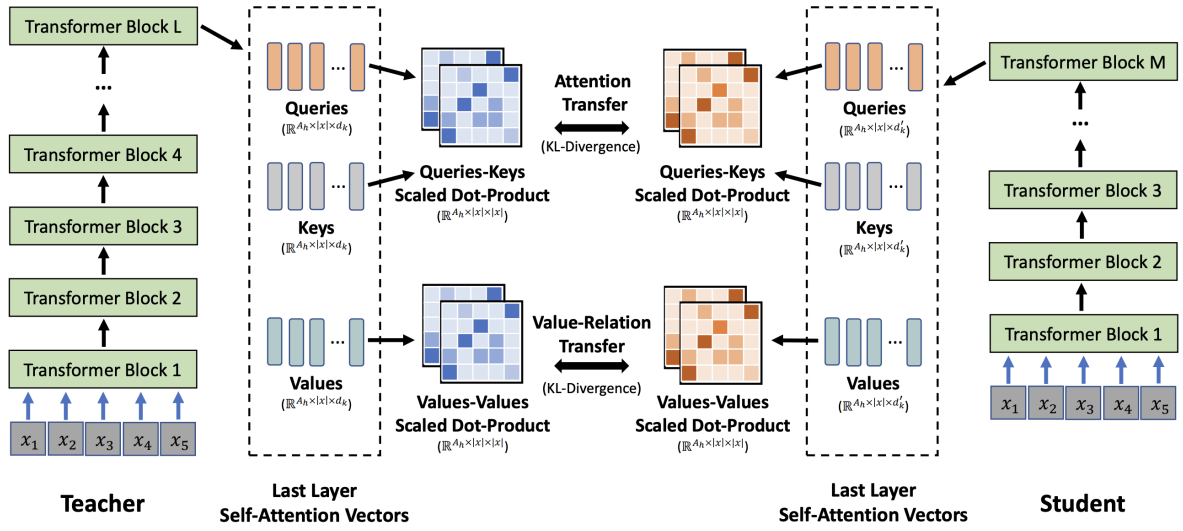


Figure 2.8: Overview of deep self-attention distillation. Through self-attention distribution transfer and self-attention value-relation transfer. Taken from [Wang et al., 2020].

one needs to store a different instance of all weights for each fine-tuned model, and since models are increasing exponentially in size, that is also an important concern. For that reason, methods for fine-tuning more efficiently are a vast area of research, and these techniques are named parameter-efficient fine-tuning (PEFT).

2.5.1 Parameter-Efficient Fine-Tuning

One method to reduce the number of parameters we fine-tune is to simply freeze part of the layers and only train the top k layers on the new data. However, this approach still requires that a considerable percentage of parameters is trained to not significantly degrade performance when compared with full fine-tuning. Another proposed method is to use Adapters [Houlsby et al., 2019] to use a significantly smaller number of parameters. The technique consists of freezing the pre-trained encoder and adding extra dense layers on top of the model that will learn to use the embeddings as input to perform the downstream task at hand. When comparing the performance against full fine-tuning, one can see in Figure 2.9 that the Adapters method needs much fewer parameters than fine-tuning on the top k layers.

One problem with this method is that by adding additional dense layers, one is effectively increasing the total number of parameters and thus the total inference time.

Another approach is to add a separate low-rank adaptation matrix for each layer in the pre-trained model. This method is called LoRA [Hu et al., 2021], and it leverages the idea that an adaptation task has a low "intrinsic dimension" [Aghajanyan et al., 2020]. To accomplish that, the method constrains the weight matrix update by representing that update matrix with a low-rank decomposition:

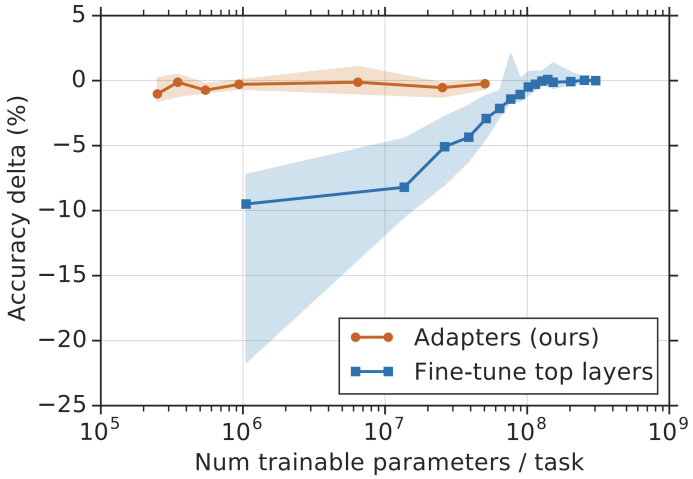


Figure 2.9: Trade-off between accuracy and the number of trained task-specific parameters for adapter tuning and fine-tuning. Taken from [Houlsby et al., 2019].

$$W_0 + \Delta W = W_0 + BA, \tag{2.14}$$

where $W_0 \in \mathbb{R}^{d \times k}$, $B \in \mathbb{R}^{d \times r}$, $A \in \mathbb{R}^{r \times k}$, and $r \ll min(d, k)$. The model reparametrization is shown in Figure 2.10, for A random Gaussian initialization is used while B is initialized as zero, so that $\Delta W = BA$ is zero in the beginning.

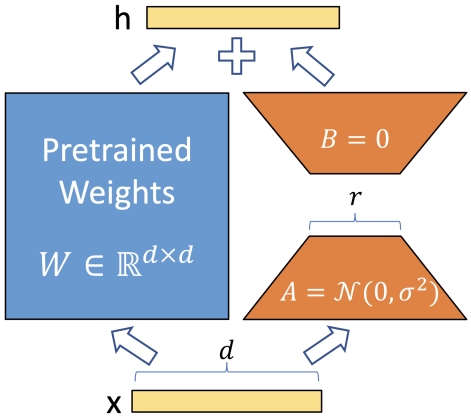


Figure 2.10: Model reparametrization. Taken from [Hu et al., 2021].

This method differs from adapters since the additional weights can be run in parallel instead of sequentially. Additionally, if one wants to have zero added inference time, one can merge the low-rank matrices with the original full-rank matrices by simply multiplying B and A to obtain ΔW and then add it to W_0.

2.6 Machine Translation

Machine translation is the process of translating text from one language to another using rule-based, probabilistic, or neural network-based approaches.

In 2014, the first MT systems using end-to-end neural network models were published [Bahdanau et al., 2014] [Sutskever et al., 2014]. By the 2016 conference on machine translation, 90% of the winning submissions were NMT models [Bojar et al., 2016].

2.6.1 Neural Machine Translation

Neural machine translation directly converts the source sentence to the target sentence. However, since neural networks receive continuous vectors as input and as output, words need to be converted into word embeddings. As explained in Section 2.2, one can map words from a vocabulary Σ to a d-dimensional vector using an embedding matrix $E \in \mathbb{R}^{d \times |\Sigma|}$, where $d \ll |\Sigma|$. A better approach is to have contextualized word embeddings, these embeddings depend not only on the word but on the entire input sentence. For that, instead of using an embedding matrix, one can use neural sequence models, such as LSTMs or Transformer architectures as pre-trained encoders to get the word embeddings used in the translation [Stahlberg, 2020].

To predict the next target language embeddings based on the source language embeddings, a simple approach is to use a connectionist sequence classification technique illustrated in Figure 2.11, which uses an LSTM to read the source language sequence word by word, and after receiving the end-of-sentence token <eos>, it starts predicting the target language sequence also word by word, while it is fed with its own predictions to do so.

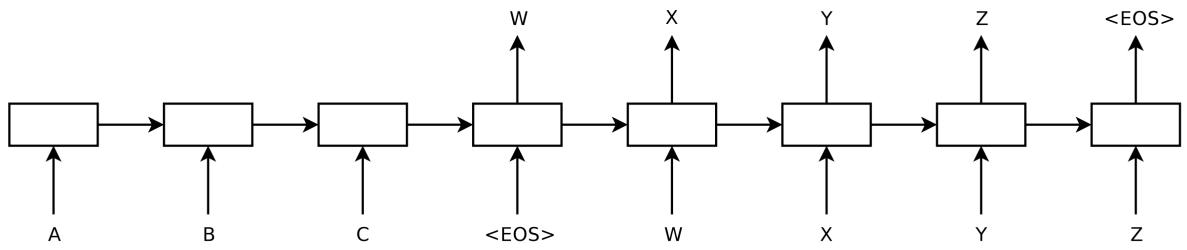


Figure 2.11: Diagram of the prediction process word by word using an LSTM. Taken from [Sutskever et al., 2014].

A problem with this solution is that it assumes an alignment between the input words and output words, which is often not the case when translating a sentence. Transformers offer a better solution since they generate the whole translated phrase in parallel, giving attention to the entire sentence.

2.7 Evaluation of Machine Translation

Evaluation of machine translation is the process of systematically assessing the quality of a certain machine translation model. One uses the source text and/or the reference text and compares them against the hypothesis produced by the MT system to attribute a certain quality score to the hypothesis. This process can be manually performed by humans, or it can be done using automatic evaluation. The main human evaluation metrics are DA [Graham et al., 2013], human translation edit rate (HTER) [Snover et al., 2006], and MQM [Lommel et al., 2014] scores.

Direct assessments consist of scoring the accuracy of the hypothesis from 0 to 100, where 0 means completely inaccurate and 100 is perfectly accurate. The main advantage of DA scores is the minimal training required, which allows the use of crowd-sourced annotators instead of professional translators. In fact, it is the method used to score the training sets of the WMT shared tasks. When there are not enough annotations per sentence [Bojar et al., 2017, Ma et al., 2018, Ma et al., 2019], one can use relative rankings (DARR), which rank several hypotheses for the same source sentence instead of giving an absolute score to each hypothesis. HTER requires a professional translator to make the necessary changes to the MT hypothesis, the resulting sentence is referred to as the post-edited translation (PE), and then the metric is defined as follows:

$$HTER = \frac{number\ of\ edits}{number\ of\ words\ in\ the\ final\ PE} \tag{2.15}$$

One problem with this metric is the difficulty of distinguishing between different types of errors in a sentence since all contribute in the same way to the final score. MQM is defined in equation 2.16 and addresses this problem by counting the number of errors and distinguishing them between minor (I_{Minor}), major (I_{Major}), and critical ($I_{Crit.}$) errors. Then the metric computes a weighted average where minor errors have a weight of 1, major errors a weight of 5, and critical errors a weight of 10.

$$MQM\ score = 100 - \frac{I_{Minor} + 5 \times I_{Major} + 10 \times I_{Crit.}}{Sentence\ Length \times 100} \tag{2.16}$$

While human evaluation usually works as the gold standard, to develop a new machine translation system, one needs to be able to automatically evaluate the quality of their models. Automatic metrics were developed for that purpose, and they are explained in detail in Sections 2.7.1 and 2.7.2.

2.7.1 MT Evaluation

There are several categories of MT evaluation metrics. The simplest methods are termed n-gram-based metrics. They work by counting the number of times n consecutive words or characters in the hypothesis align with the reference. The most common n-gram-based metric is BLEU [Papineni et al., 2002], and

although it has a low correlation with human judgment, it is a standard among the MT community. This metric works by applying a geometric mean of several n-gram sizes, since shorter n-grams measure the adequacy of the translation while longer n-grams test the fluency of such translation. The n-gram precision score for a given n is calculated as follows:

$$p_n = \frac{\sum_{\mathcal{C} \in \{Candidates\}} \sum_{\text{n-gram} \in \mathcal{C}} Count_{clip}(\text{n-gram})}{\sum_{\mathcal{C}' \in \{Candidates\}} \sum_{\text{n-gram}' \in \mathcal{C}'} Count(\text{n-gram}')}, \tag{2.17}$$

where $Count(\text{n-gram})$ is the number of matches of n-grams in a candidate sentence $\mathcal{C}$ and $Count_{clip} = min(Count, Max\_Ref\_Count)$, which means it is the same expression as $Count$ but truncated by the largest count observed in any single reference. This metric can easily score the quality of a single sentence wrongly, as explained in the original publication "...a system which produces the fluent phrase 'East Asian economy' is penalized heavily on the longer n-gram precisions if all the references happen to read 'economy of East Asia.'". However, for system evaluation, where one measures the quality of multiple MT systems over an entire test corpus, these problems tend to cancel out in comparison between systems. Several other metrics improved BLEU by taking into account the limitations of this metric. Examples of relevant metrics are NIST [Doddington, 2002], ROUGE [Lin, 2004], and METEOR [Banerjee and Lavie, 2005]. Another relevant n-gram metric is CHRF [Popović, 2015], which instead of using word-level n-grams, uses character-level 6-grams. It is much simpler than METEOR since it simply calculates an F_1 score of such 6-grams:

$$\text{CHRF}\beta = (1 + \beta^2) \frac{\text{CHRP} \cdot \text{CHRR}}{\beta^2 \cdot \text{CHRP} + \text{CHRR}}, \tag{2.18}$$

where CHRP stands for character n-gram precision, CHRR for n-gram recall, and β is a parameter to assign β times more importance to recall than to precision, the standard CHRF uses $\beta = 1$. Despite the simplicity of this metric, it has competitive results and has been shown to consistently outperform BLEU over several WMT Metrics Tasks.

While these automatic algorithmic methods are useful in many cases, they lack the ability to "weigh many aspects of translation, including adequacy, fidelity, and fluency of the translation" [Papineni et al., 2002]. A solution to achieve scores more closely correlated with human judgments is to use word embeddings instead of n-grams. An example is BERTSCORE [Zhang et al., 2019], which utilizes contextual embeddings from BERT [Devlin et al., 2018b] to compute soft alignments using cosine similarity. The cosine similarity of a reference token $\mathbf{x}_i$ and a hypothesis token $\hat{\mathbf{x}}_j$ is given by $\frac{\mathbf{x}_i^\top \hat{\mathbf{x}}_j}{\|\mathbf{x}_i^\top\| \|\hat{\mathbf{x}}_j\|}$, however since in BERTSCORE the reference and hypothesis vectors are normalized, the similarity is simply the inner product $\mathbf{x}_i^\top \hat{\mathbf{x}}_j$. Then three metrics are calculated, the recall R_{BERT}, the precision P_{BERT}, and the F_1 score F_{BERT}:

$$R_{\text{BERT}} = \frac{1}{|x|}\sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} \mathbf{x}_i^\top \hat{\mathbf{x}}_j, \quad P_{\text{BERT}} = \frac{1}{|\hat{x}|}\sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} \mathbf{x}_i^\top \hat{\mathbf{x}}_j. \tag{2.19}$$

$$F_{\text{BERT}} = 2\frac{P_{\text{BERT}} \cdot R_{\text{BERT}}}{P_{\text{BERT}} + R_{\text{BERT}}}. \tag{2.20}$$

Other embedding-based metrics are the YiSi [Lo, 2019] family of metrics and the PRISM metric [Thompson and Post, 2020].

Both n-gram and embeddings-based models directly compare the hypothesis with the reference. Another class of metrics uses neural networks to mimic the way humans evaluate translation quality. This is the class we focus more on in this thesis since the model we proposed to apply model compression (COMET [Rei et al., 2020]) uses a dense neural network to determine a score based on the given contextual embeddings. A popular neural fine-tuned metric is BLEURT [Sellam et al., 2020], it encodes the hypothesis and reference using BERT, and then the `<cls>` token is used as the input of a feedforward neural network. Other examples of important metrics of this class are ROBLEURT [Wan et al., 2022a] and UNITE [Wan et al., 2022b]. UNITE in specific is a framework that is capable of doing MT evaluation using reference-only, source-reference-combined, and also source-only or QE as explained in Section 2.7.2. The main advantage of this approach is the ability to train with triplets of `(source, hypothesis, reference)` but having an architecture that is able to output a result with only a pair of `(reference, hypothesis)` or `(source, hypothesis)` and all these results are related to each other, unifying this way all translation evaluation tasks.

2.7.2 MT Quality Estimation

Similar to MT evaluation, there are several QE metrics. The crucial difference between the two is that QE does not use a reference to determine the quality score, it works only with the source and hypothesis. Initially, these models were used as a proxy for MT evaluation metrics when there was a lack of parallel test sets, and thus no reference was available. However, as cross-lingual pre-trained encoders improved, QE started to be competitive with reference-based metrics while being more efficient at inference. Another advantage of such models is that generating reference translations is an expensive and time-consuming process since it requires professional translators, which means that QE metrics can leverage larger training sets since creating such a corpus is much faster without the need to generate the reference translation. Examples of such models are TRANSQUEST [Ranasinghe et al., 2020], OPENKIWI [Kepler et al., 2019], and COMETKIWI [Rei et al., 2022b].

COMETKIWI will be explained in the next Section 2.7.3, and it combines the COMET architecture with the predictor-estimator [Kim et al., 2017] architecture of OPENKIWI. This architecture works at the word-level and sentence-level. In word-level QE, it assigns quality labels (OK or BAD) to each word and the

gap between words, as seen in figure 2.12. For that, it uses two modules, the *predictor* which predicts a single token of the hypothesis using the remaining tokens of the hypothesis, and the entire source sentence as context. And an *estimator* that uses the features from the predictor to classify each word as OK or BAD.

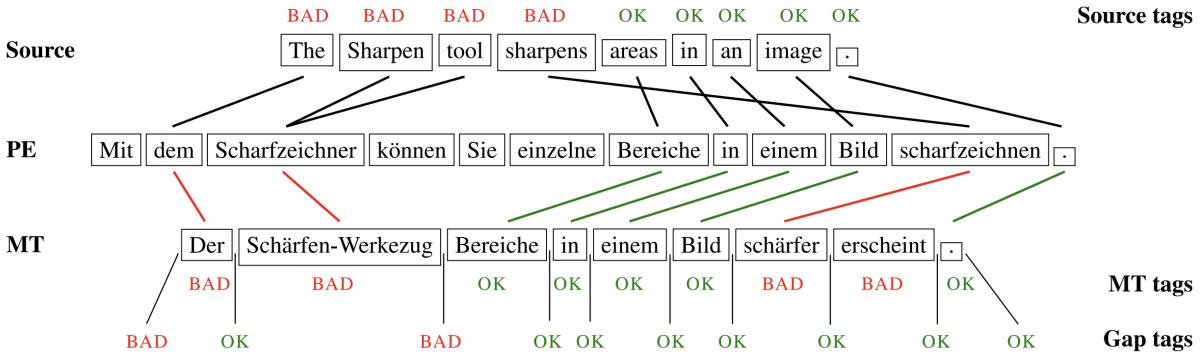


Figure 2.12: "MT (or target) tags account for words that are replaced or deleted, gap tags account for words that need to be inserted, and source tags indicate what are the source words that were omitted or mistranslated". Taken from [Kepler et al., 2019].

For sentence-level QE, the framework uses a multi-task architecture to allow the model to learn to predict the sentence-level HTER scores.

2.7.3 COMET and COMETKIWI architectures

The architecture where we apply model compression is COMETKIWI, a QE system based on the COMET MT evaluation metric. For that reason, it is important to explain both architectures and clarify where they differ.

COMET [Rei et al., 2020] is a framework for training multilingual MT evaluation models. There are two supported architectures, one estimator model that is trained to regress directly on a quality score, and one ranking model that is trained to generate DARR rankings. We will explain the regression model in more detail since it is our focus in this thesis.

The architecture can use any cross-lingual pre-trained encoder, examples are BERT [Devlin et al., 2018b], XLM [Conneau and Lample, 2019], XLM-RoBERTa [Conneau et al., 2019], MINILM [Wang et al., 2020], and InfoXLM [Chi et al., 2020]. The framework then feeds the encoder with the source, MT hypothesis, and reference if we want a reference-based metric or without reference otherwise. For each sentence, it produces an embedding $e_j^{(l)}$ for each token x_j and each layer $l \in \{0, 1, ..., k\}$, to map the three input sentences in a shared feature space. Second, as shown in [Zhang et al., 2019], the levels of correlation with MT evaluation vary between layers. It is better to use all of them together than just the embeddings from the last layer. COMET uses the approach described in [Peters et al., 2018b] and

creates a single embedding for each token by combining the encoder layers using a layer-wise attention mechanism computed as:

$$\boldsymbol{e}_{x_j} = \mu E_{x_j}^{\top} \boldsymbol{\alpha}, \tag{2.21}$$

where μ is a trainable weight coefficient, $\boldsymbol{E}_{x_j} = \left[\boldsymbol{e}_{x_j}^{(0)}, \boldsymbol{e}_{x_j}^{(1)}, \ldots, \boldsymbol{e}_{x_j}^{(k)}\right]$ corresponds to the vector of layer embeddings for token x_j, and $\boldsymbol{\alpha} = \mathrm{softmax}\left(\left[\alpha^{(1)}, \alpha^{(2)}, \ldots, \alpha^{(k)}\right]\right)$ is a vector corresponding to the layer-wise trainable weights. Lastly, average pooling is applied to the token embeddings to obtain a sentence embedding. Once we have the embeddings, the COMET framework will create the following features proposed in RUSE [Shimanaka et al., 2018]:

- Element-wise source product: $h \odot s$;
- Element-wise reference product: $h \odot r$;
- Absolute element-wise source difference: $|h - s|$;
- Absolute element-wise reference difference: $|h - r|$.

These features are generated to highlight the differences between embeddings in the semantic feature space, which is then useful to the feedforward regressor that receives a single vector x (equation 2.22) of the concatenated features.

$$x = [h; r; h \odot s; h \odot r; |h - s|; |h - r|] \tag{2.22}$$

It is worth noting that the source embedding is not fed to the neural network, only the features that take advantage of the source. As explained in [Rei et al., 2020], the value added by the source embedding as an extra input feature was negligible. The dense neural network is then trained using the MSE loss (equation 2.6) between the predicted scores and the human scores. Those scores can also be normalized to z-scores between 0 and 1. The overall architecture is shown in Figure 2.13.

In COMETKIWI [Rei et al., 2022b], the source and hypothesis are concatenated and then fed to the encoder, producing the d-dimensional hidden state vectors $\boldsymbol{H}_0, \ldots, \boldsymbol{H}_L$ for each layer $0 \leq \ell \leq L$ where $\boldsymbol{H}_i \in \mathbb{R}^{(n+m)\times d}$. Then, in a similar process to COMET, the hidden states are combined in a single mix hidden state $\boldsymbol{H}_{\text{mix}}$, by feeding all layers into a scalar mix module [Peters et al., 2018b] that learns the hidden states' weighted sum of each layer of the encoder as:

$$\boldsymbol{H}_{\text{mix}} = \lambda \sum_{\ell=0}^{L} \beta_\ell \boldsymbol{H}_\ell, \tag{2.23}$$

where λ is a scalar trainable parameter, $\boldsymbol{\beta} \in \triangle^L$, is given by $\boldsymbol{\beta} = \mathrm{sparsemax}(\phi)$ using a sparse transformation [Martins and Astudillo, 2016] (instead of the $\mathrm{softmax}$ used in COMET) with $\phi \in \mathbb{R}^L$ as learnable

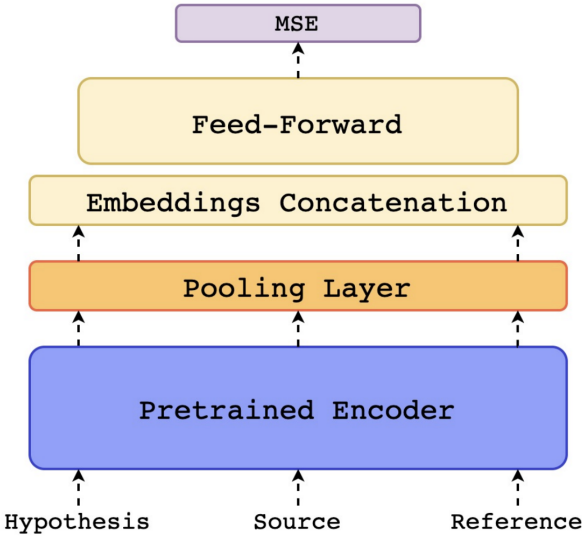


Figure 2.13: Estimator model architecture. Taken from [Rei et al., 2020].

parameters and $\triangle^L := \{\boldsymbol{\beta} \in \mathbb{R}^L : \mathbf{1}^\top \boldsymbol{\beta} = 1, \boldsymbol{\beta} \geq 0\}$, this way the model learns to ignore less important layers.

Once we have the combined hidden state, the <cls> token is used to feed a feedforward neural network the same way it is for COMET. For word-level models, the first-word hidden state vectors of each MT token are linearly projected to get word-level predictions $\hat{y} \in \{\text{OK}, \text{BAD}\}, \forall_{1 \leq i \leq n}$. The overall architecture is shown in Figure 2.14.

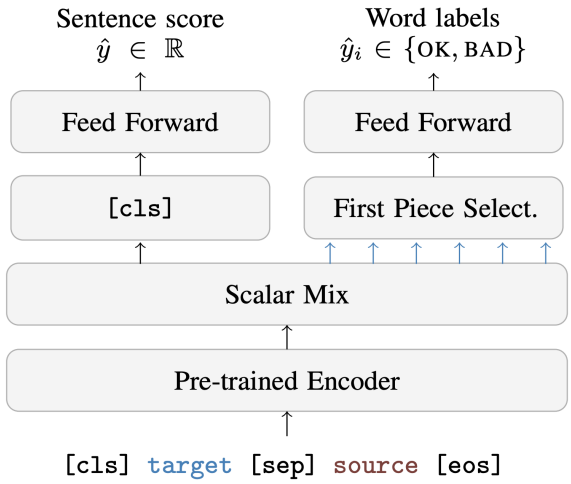


Figure 2.14: "Architecture of COMETKIWI for sentence-level (left part) and word-level QE (right part)". Taken from [Rei et al., 2022b].

According to [Rei et al., 2022b], word-level supervision was beneficial for sentence-level QE when

using models with the pre-trained encoder infoXLM. The task uses a combined loss on both DA and MQM scores:

$$\begin{aligned} \mathcal{L}_{\text{sent}}(\theta) &= \frac{1}{2}(y - \hat{y}(\theta))^2; \\ \mathcal{L}_{\text{word}}(\theta) &= -\frac{1}{n}\sum_{i=1}^{n} w_{y_i} \log p_\theta(y_i); \\ \mathcal{L}(\theta) &= \lambda_s \mathcal{L}_{\text{sent}}(\theta) + \lambda_w \mathcal{L}_{\text{word}}(\theta). \end{aligned} \tag{2.24}$$

Furthermore, the architecture that yielded the best results was an ensemble of six models, of which three used the infoXLM encoder, two used the RemBERT [Chung et al., 2020], and one used the XLM-R.

2.7.4 Previous work on COMETINHO

The previous work we developed on model compression focused on the COMET architecture, and we summarize those contributions in this section.

Length Sorting and Caching

The first approach we explored, even before model compression, was performing length sorting and caching. Length sorting consists of sorting sentences according to their length, this way, we create batches with sentences of similar length, reducing the amount of tensor padding that is necessary [Pu et al., 2021]. In COMET, the length sorting was performed according to the source length. The efficiency improvement is shown in Figure 2.15.

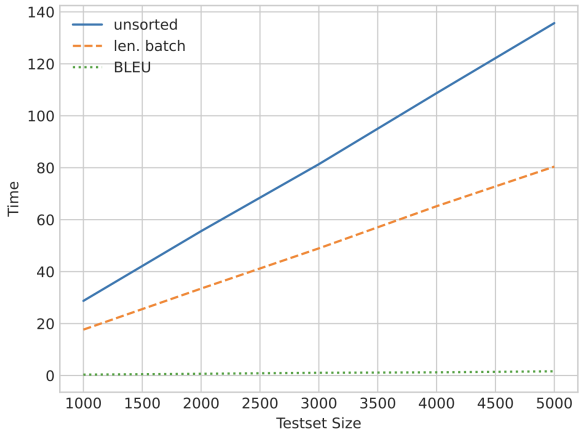


Figure 2.15: "Runtime (in seconds) with an NVIDIA GeForce GTX 1080 TI GPU, batch size of 16. Using wmt20-comet-da. BLEU is used for comparison and runs on an Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz". Taken from [Rei et al., 2022a].

Caching is another technique we use that leverages the fact that COMET uses triplet encoders,

meaning that the source, hypothesis, and reference are encoded separately instead of concatenated. Since in many applications, the same source is translated into several languages, by encoding the source, hypothesis, and reference separately, we only need to encode each unique sentence once and then cache the embeddings to use every time the sentence appears in a triplet. The efficiency improvement is shown in the right plot in Figure 2.16. After applying these two optimization methods,

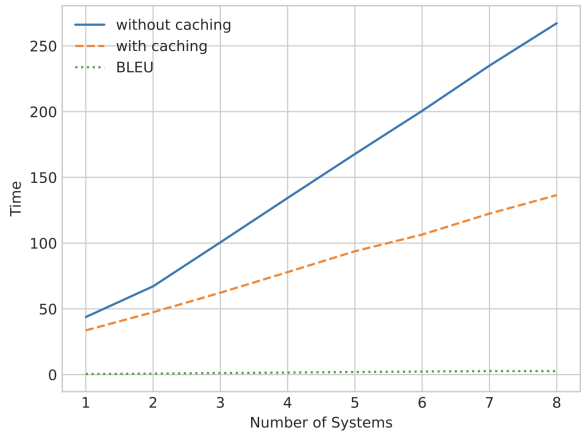


Figure 2.16: "Runtime (in seconds) with an NVIDIA GeForce GTX 1080 TI GPU, batch size of 16. Using wmt20-comet-da. BLEU is used for comparison and runs on an Intel(R) Core(TM) i7-6850K CPU @ 3.60GHz". Taken from [Rei et al., 2022a].

the inference time of COMET is reduced from 34.7 seconds to 21.1 seconds (39% faster).

Model Pruning

We then applied model pruning techniques, starting with layer pruning. Since most parameters of COMET are in the pre-trained encoder XLM-R [Conneau et al., 2019], we start by removing layers from it. Different layers impact the downstream task of MT evaluation differently. As explained in Section 2.7.3, COMET uses a layer-wise attention mechanism that pools information from all layers and combines them using a weighted average. By analyzing vector α, which corresponds to the trainable weights, we can see in Figure 2.17 that the most important layers are between 15 and 19.

However, layer 15 is dependent on the previous layers, meaning that we can only remove the topmost layers, from 20 to 25. Removing the top layers reduces the number of parameters by 10.8% while maintaining relatively constant performance.

We then experiment with reducing the size of each layer (width pruning). XLM-R is a transformer-based pre-trained encoder, each transformer block is composed of 16 attention-heads and a single hidden layer feedforward neural network with 4096 parameters. We verify that the best trade-off of size and performance is to keep only 19 layers, 3072 parameters for the hidden layer (3/4 of the original size),

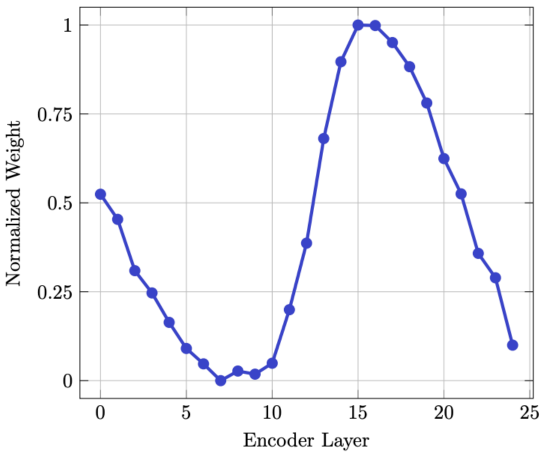


Figure 2.17: Normalized weights distribution for the COMET default model (wmt20-comet-da). Taken from [Rei et al., 2022a].

and 14 self-attention heads (2 less than the original). The result is a model named PRUNED-COMET that is 21% smaller, 37% faster, and with only 2% less performance.

Knowledge Distillation

Finally, we performed distillation using an ensemble of 5 COMET models as teachers [Glushkova et al., 2021]. For the student model, we use the same architecture and hyper-parameters and substitute the encoder XLM-R-large [Conneau et al., 2019] with a distilled version that has only 117 million parameters instead of 560 million. The corpus used contains 45 million high-quality tuples in 15 language pairs.

To generate such a corpus, we started by extracting 25 million sentence pairs from OPUS [Tiedemann, 2012], then used the Bicleaner tool [Ramírez-Sánchez et al., 2020] to filter the dataset and leave only high-quality pairs. Then we generated two translations for each source, the first using a bilingual model, which is a model that was trained to translate directly from one language to another and cannot be used for any other language pair, unlike a multilingual model that can be used for several language pairs. For the second translation, we used pivoting, meaning that to translate from language a to language b, we first translate a to a high-resource language, typically English, and then translate from that high-resource language to b. This approach is intentionally used to produce worse results than the bilingual model. Finally, we score the data to get a final corpus with 45 million tuples.

Once we have the dataset scored, we train the student model to regress over the COMET scores, and the result is a model named DISTIL-COMET that is 80% smaller, 53% faster and has only 5% less performance.

In Table 2.1, we can analyze the direct comparison between COMET, PRUNE-COMET, and DISTIL-COMET in the language pairs Chinese $\rightarrow$ English, English $\rightarrow$ German, and English $\rightarrow$ Russian.

		zh-en		en-de		en-ru		
Metric	**# Params**	**News**	**TED**	**News**	**TED**	**News**	**TED**	**avg.**
BLEU	-	0.166	0.056	0.082	0.093	0.115	0.067	0.097
CHRF	-	0.171	0.081	0.101	0.134	0.182	0.255	0.154
BERTSCORE	179M	0.230	0.131	0.154	0.184	0.185	0.275	0.193
PRISM	745M	0.265	0.139	0.182	0.264	0.219	0.292	0.229
BLEURT	579M	**0.345**	**0.166**	**0.253**	**0.332**	**0.296**	**0.347**	**0.290**
COMET	582M	0.336	0.159	0.227	0.290	0.284	0.329	0.271
PRUNE-COMET	460M	0.333	0.157	0.219	0.293	0.274	0.319	0.266
DISTIL-COMET	119M	0.321	0.161	0.202	0.274	0.263	0.326	0.258

Table 2.1: "Kendall's tau correlation on high resource language pairs using the MQM annotations for both News and TED talks domain collected for the WMT 2021 Metrics Task". Taken from [Rei et al., 2022a]

3

Proposed Approach

Contents

One of the most concerning aspects of the growth of pre-trained models is their energy consumption and environmental impact. However, in machine translation evaluation and quality estimation, accuracy is very important, and we want to keep it as close to previous models as possible. With that in mind, this thesis focuses on minimizing the quality-efficiency trade-off by reducing the size of COMET and improving the inference time as much as possible without degrading significantly the model quality.

We focus our thesis on knowledge distillation since it was the approach that offered the best trade-off in COMETINHO when we compare COMET with PRUNE-COMET and DISTIL-COMET. We then introduce model fine-tuning to further improve our results. In this chapter, we detail the considerations to determine our teacher model, the student architecture, and fine-tuning labels. And compare that process with what was done in our previous work with COMETINHO.

3.1 Generating the teacher outputs

There are several ways of applying knowledge distillation, as discussed in Chapter 2. In this thesis, we make use of a larger reference-free COMETKIWI model denominated COMETKIWI-XL [Rei et al., 2023], to leverage a more capable teacher than the ensemble of 5 COMET teachers we have used in our previous work.

COMETKIWI-XL is a model that uses the same general architecture as COMETKIWI, the main difference between them being the exchange to larger pre-trained models such as XLM-R XL and XLM-R-XXL [Goyal et al., 2021].

The XLM-R XL architecture is based on the XLM-R model, but it has 32 attention heads instead of 16, 36 encoder blocks instead of 24, and 3.5 billion parameters instead of 550 million.

We use COMETKIWI-XL to label the same dataset used in the previous COMETINHO model, detailed in Section 2.7.4, and an additional 13.6 million tuples from another 21 language pairs. We then used response-based distillation, which is the same method used for COMETINHO.

To choose between COMETKIWI-XL and COMETKIWI-XXL, we tested distillation with both models on 5% of the corpus. The results were similar, and since COMETKIWI-XXL takes 52.51 seconds to label a thousand sentence pairs as shown in Chapter 4, labeling the entire corpus would take approximately 35.6 days:

$$58.6M/1000 * 52.51s * 1h/3600s * 1day/24h = 35.6days. \tag{3.1}$$

The choice was to use the XL model to label the entire dataset.

Since we labeled 5% of the corpus with both models and we used response-based distillation, which is the same method used for COMETINHO, we can use this data to study to what extent better teacher models can reduce the amount of data the student model needs to be trained on.

3.2 Student model architecture

To distill the teacher model, it is necessary to choose a more compact architecture for the student model. We used Microsoft's InfoXLM-base cross-lingual pre-trained encoder [Chi et al., 2020], since the encoder used in COMETKIWI [Rei et al., 2022b] that led to superior performance in Spearman correlation was the InfoXLM-L. This finding suggests that the InfoXLM family of encoders is well-suited for reference-free MT evaluation or QE. Despite the infoXLM-base model being larger than the distilled XLM-R encoder used in COMETINHO, this was not a problem considering that even with a larger model, we can have faster inference times using a reference-free architecture.

The infoXLM architecture that follows the configuration of XLM-R is formulated to maximize the mutual information between multilingual-multi-granularity texts. The encoder used for the student model has 12 layers and 768 hidden states and is 13 times smaller than the XLM-R XL encoder used for the teacher. Additionally, the hidden layer sizes of the feed-forward neural network that follows the pre-trained encoder in the COMETKIWI architecture were also decreased. COMETKIWI has two layers with sizes 3072 and 1024, while our student model has two layers with sizes 1536 and 768.

We chose to use a reference-free architecture since the performance of reference-free models is competitive with reference-based models. As shown in the COMET submission for WMT21 [Rei et al., 2021] in Table 3.1, both the DA and MQM scores have only a decrease of 10 and 9 Kendall's Tau points, respectively.

Metric	zh-en	en-de	avg.
COMET-DA (2021)	0.454	0.309	0.382
COMET-MQM (2021)	**0.546**	**0.361**	**0.454**
COMET-QE-DA (2021)	0.436	0.308	0.372
COMET-QE-MQM (2021)	**0.531**	**0.359**	**0.445**

Table 3.1: Kendall's Tau segment-level correlation on the English→German and Chinese→English test set using DA and MQM scores from the WMT 2021 Metrics Task. Taken from [Rei et al., 2021].

These results are further confirmed in the survey "To ship or not to ship" [Kocmi et al., 2021], where COMET-src scored second on all available systems to perform pairwise comparisons, scoring only below COMET by a small margin, as seen in Table 3.2.

Both findings show that the COMET model can have a strong ability to identify translation errors given the good intuition of what is wrong with a translation just based on the source. Given the negligible impact on performance, the trade-off is worth it since reference-free architectures also allow us to use larger pre-trained encoders while still reducing the inference time. This is due to the encoding part of COMETKIWI being the most significant time-consuming part, and in a reference-based architecture, the reference is an additional sentence that needs to be encoded. Additionally, the utilization of reference-based architectures poses greater challenges in domains characterized by the limited or nonexistent

	All	0.05	0.01	0.001	Within
n	3344	1717	1420	1176	541
COMET	**83.4**	**96.5**	**98.7**	**99.2**	**90.6**
COMET-src	83.2	95.3	97.4	98.1	89.1
Prism	80.6	94.5	97.0	98.3	86.3
BLEURT	80.0	93.8	95.6	98.2	84.1
ESIM	78.7	92.9	95.6	97.5	82.8
BERTScore	78.3	92.2	95.2	97.4	81.0
ChrF	75.6	89.5	93.5	96.2	75.0
TER	75.6	89.2	93.0	96.2	73.9
CharacTER	74.9	88.6	91.9	95.2	74.1
BLEU	74.6	88.2	91.7	94.6	74.3
Prism-src	73.4	85.3	87.6	88.9	77.4
EED	68.8	79.4	82.4	84.6	68.2

Table 3.2: Column "All" shows the results for system pairs. Each following column evaluates accuracy over a subset of systems that are deemed different based on human judgment and a given alpha level in Wilcoxon's test. Column "Within" represents a subset of systems where the human judgment p-value is between 0.05 and 0.001. "n" represents the number of system pairs used to calculate accuracies in a given column. Taken from [Kocmi et al., 2021].

availability of good-quality references. This is particularly evident in low-resource language pairs, where the generation of high-quality references incurs higher costs.

3.3 Fine Tuning

To fine-tune the student model, the data was the same used to train the teacher model, which is detailed in the COMETKIWI-XL paper [Rei et al., 2023]. This corpus compiles the DA scores from the 2017 to 2020 WMT translation shared task and the MLQE-PE corpus, totaling 940k samples in 38 language pairs.

To determine the label that should be used, we performed a grid search between 0 and 1 on a hyper-parameter λ created to fine-tune the model with the following weighted score:

$$WeightedScore = \lambda * \text{COMETKIWI-XL} + (1 - \lambda) * HumanScore \tag{3.2}$$

We then concluded, as it can be seen in Chapter 4, that using exclusively COMETKIWI-XL scores was actually better than a weighted average of both. We chose to apply full fine-tuning instead of other methods explained in Chapter 2 since this dataset was much smaller than the one used to apply knowledge distillation and, for that reason, represented a much smaller percentage of the total development compute power necessary, despite full fine-tuning being a computationally expensive method. For that reason, we applied full fine-tuning with only the COMETKIWI-XL scores to our completely distilled model.

To evaluate the performance of our implemented systems and to better explain the development choices detailed in this chapter, we define our baselines and present the evaluation methodology in Chapter 4, as well as the results both of knowledge distillation and fine-tuning compared to the baselines.

4

Evaluation and Results

Contents

4.1 Evaluation Methodology

This section describes the evaluation methodology followed throughout the research developed. We define baselines to compare to our model and then describe the way we measure accuracy using Kendall's Tau and Spearman correlations.

4.1.1 Defining a baseline

For this thesis, we use as a primary baseline our previous work done compressing COMET, the COMET-INHO model [Rei et al., 2022a]. Nevertheless, we also compare our models to other widely used metrics that are representative of the techniques mentioned in Chapter 2. BLEU [Papineni et al., 2002] and CHRF [Popović, 2015] are n-gram metrics, BERTSCORE [Zhang et al., 2019] is an embedding-based model, and BLEURT [Sellam et al., 2020] is a neural fine-tuned metric. Our goal is to achieve an accuracy as close as possible to COMETKIWI-22, the state-of-the-art at the start of development.

4.1.2 Defining the model accuracy

The two accuracy metrics we use for QE are Kendall's Tau (τ) correlation [Kendall, 1938] and Spearman (ρ) correlation. Both Kendall's Tau and Spearman are used to correlate the metric scores with the MQM scores of the WMT 22 metrics shared task news test set.

Kendall's tau coefficient is a measure of rank correlation, measuring the similarity of the two corpora generated when ordering the sentence pairs by the scores given by the automatic metric and when ordering the sentence pairs by the scores given by the MQM annotations. There are several formulations of Kendall's Tau correlation coefficient that are used to handle ties in different ways when ordering the scores of the automatic metric and of the MQM annotations [Deutsch et al., 2023]. We use the formulation τ_b [Kendall, 1945] defined as follows:

$$\tau = \frac{C - D}{\sqrt{(C + D + T_h)(C + D + T_m)}}, \tag{4.1}$$

where C(Concordant) corresponds to the number of times a metric gives a better score to the best translation according to the MQM annotations, and D(Discordant) is when a metric gives a better score to the worst translation according to the MQM annotations. T_h is the number of ties in the human scores, and T_m is the number of ties in the metric scores. However, in the metrics where there are no ties, Kendall's Tau formulation can be simplified to:

$$\tau = \frac{C - D}{C + D}. \tag{4.2}$$

The measure varies between -1 and 1, where -1 means the metric always disagrees with the MQM annotations, and 1 means the metric and the annotations always agree on what is the best translation.

The second accuracy metric used is the Spearman correlation, used in the WMT23 QE shared task. The Spearman rank correlation coefficient is a measure of rank correlation, where the objective is to measure how well the relationship between two variables can be described using a monotonic function, meaning any function where the dependent variable preserves the order of the independent variable. The metric varies between -1 and 1, where 1 means that the function is monotonically increasing (for all $x \leq y$, it follows that $f(x) \leq f(y)$) and -1 means that the function is monotonically decreasing (for all $x \leq y$, it follows that $f(x) \geq f(y)$). The Spearman correlation coefficient is defined as the Pearson correlation between the rank variables:

$$\rho = \frac{\text{cov}(R(X), R(Y))}{\sigma_{R(X)}\sigma_{R(Y)}}. \tag{4.3}$$

In our use case for n scores, X_i and Y_i would be the scores given to the translation by the automatic metric and by the MQM annotations, respectively. $R(X_i)$ and $R(Y_i)$ would be the conversions of the raw scores to the ranking of that translation within all scores on X and Y, respectively. Then $\text{cov}(R(X), R(Y))$ is the covariance between the rank variables, and $\sigma_{R(X)}$ and $\sigma_{R(Y)}$ are the standard deviations of the rank variables. The standard deviations in the denominator work as a normalizer of the covariance, so that the Spearman correlation always falls between -1 and 1.

4.1.3 Defining the test set

For the test set, we used the MQM scores from the WMT22 news test set and used the high-resource language pairs English→German, English→Russian, and Chinese→English. This dataset has 4500 sentence pairs both for English→German and English→Russian and 7575 sentence pairs for Chinese →English. The WMT metrics shared task dataset has in each entry multiple MT system outputs for a given source text and also a human translation reference, along with human judgments on the quality of each MT system translation.

To explain the choice of the language pairs, it is important to understand the composition of the WMT Translation shared task corpus. As explained in Section 3.3, this training set is a compilation of the DA scores from 2017 to 2020 and consists of 940K tuples of 38 different language pairs. However, the 38 language pairs are not represented evenly, some are considered high-resource pairs due to the higher availability of quality data, and the remaining are considered low-resource pairs. In Table 4.1, we compare the 3 language pairs we used in our test set and their inverse direction with the remaining 32 language pairs. The predominance of these 3 language pairs and their inverse direction is what led us to choose them as our test set in this thesis.

Language pair	# tuples	% of total
English→German	74k	7.8%
Chinese→English	99k	10.5%
English→Russian	63k	6.7%
German→English	91k	9.7%
English→Chinese	76k	8.1%
Russian→English	70k	7.4%
Remaining 32 lps	467k	49.8%

Table 4.1: Comparitive analysis of language pairs in WMT Translation shared task DA corpus from 2017 to 2020.

Comparing the test set with the training data

The training data consists of the data used to train the student and teacher models, plus the training data used to train the respective pre-trained encoders. The data used for the teacher COMETKIWI-XL is the same data used to fine-tune our distilled model and is described in Table 4.1. The remaining training data we detail in this section are the data used to apply knowledge distillation on COMETINHO and on the student model developed in this thesis, and the corpus used to train the pre-trained encoders of the teacher and student architectures.

The COMETINHO training corpus is described in Section 2.7.4 as 45 million tuples of 15 high-quality tuples in 15 language pairs. For knowledge distillation of the student model developed in this thesis, we have an additional 13.6 million sentence pairs and 21 language pairs, totaling 58.6 million tuples of parallel data from 36 language pairs distributed as in Table 4.2.

Language pair	# tuples	% of total
English→German	3.8M	6.5%
Chinese→English	4.3M	7.3%
English→Russian	2.9M	4.9%
German→English	2.6M	9.7%
English→Chinese	1.3M	8.1%
Remaining 31 lps	43.7M	74.6%

Table 4.2: Comparitive analysis of language pairs in the training set used for knowledge distillation.

The pre-trained encoders of the student and the teacher both use the CC-100 corpora, with some differences. The student's pre-trained encoder (infoXLM) uses 94 languages in monolingual data totaling 2078GB, while the teacher's pre-trained encoder (XLM-R XL) uses all 100 languages totaling 2394GB but with a different language predominance mix that is compared in Table 4.3.

InfoXLM also uses parallel data, although on a much smaller scale, 4.2GB (10%) for English→German, 7.7GB (18%) for English→Russian, 4.0GB (9.5%) for English→Chinese, and 26.1GB (62%) for other language pairs.

	infoXLM		XLM-R XL	
Language pair	Size (GB)	% of total	Size (GB)	% of total
English	732GB	35.2%	301GB	12.6%
German	99GB	4.8%	67GB	2.8%
Chinese	97GB	4.7%	64GB	2.7%
Russian	253GB	12.2%	278GB	11.6%
Remaining languages	897GB	43.1%	1684GB	70.3%

Table 4.3: Comparitive analysis of language pairs in WMT Translation shared task DA corpus from 2017 to 2020.

4.2 Results

4.2.1 Baselines

Before getting the results of any experiment, we ran our test set on a variety of models to serve as baselines. The first baselines are popular metrics other than COMET in the top half of Table 4.4. We can see that BLEURT using the BLEURT-LARGE-512 chekpoint is largely better than the other 3 metrics since it is a neural fine-tuned metric. Compared to BERTSCORE using as an encoder ROBERTA$_{\text{LARGE}}$, it is on average 14% better in Kendall's Tau correlation and 27% better in Spearman correlation. BERTSCORE is also considerably better than the other two n-gram-based metrics since it is an embedding-based metric. It is on average 87% more accurate than CHRF in Kendall's Tau correlation and 67% in Spearman correlation. Finally, when comparing the two n-gram metrics, CHRF consistently outperforms BLEU in all language pairs, which has been shown in several other WMT metrics tasks. In this evaluation, CHRF is on average 42% and 46% better in Kendall's Tau and Spearman correlations, respectively.

Secondly, we highlight the results for the 2022 COMETKIWI version as well as the XL and XXL versions in the bottom half of Table 4.4. COMETKIWI is clearly faster due to its size, having 6.2 times fewer parameters than COMETKIWI-XL, it is able to perform inference 3.5 times faster with only a 4.5% decrease in Kendall's Tau correlation and 4.6% in Spearman correlation. Consequently, the difference compared to COMETKIWI-XXL is even greater. With only a slight increase on average in Kendall's Tau and Spearman correlations, it is not consistently better than COMETKIWI-XL. This shows that increasing the model parameters and keeping the same training dataset size only gives marginal improvements. Nevertheless, in use cases where accuracy is paramount, COMETKIWI-XXL is the most reliable model, with 0.358 on Kendall's Tau correlation and 0.477 on the Spearman correlation. The trade-off is having a model that is 19.0 times larger and 10.5 times slower at inference for a gain in performance of 5% when compared to COMETKIWI Kendall's Tau correlation.

	Metric	# Params	seconds	zh-en	en-de	en-ru	avg.
τ	BLEU	-	2.65	0.032	0.158	0.095	0.084
	CHRF	-	3.09	0.042	0.202	0.164	0.119
	BERTSCORE	355M	7.47	0.219	0.235	**0.214**	0.222
	BLEURT	335M	33.14	**0.294**	**0.241**	0.197	**0.253**
ρ	BLEU	-	2.65	0.043	0.210	0.125	0.110
	CHRF	-	3.09	0.057	0.273	0.224	0.161
	BERTSCORE	355M	7.47	0.247	0.296	**0.278**	0.269
	BLEURT	335M	33.14	**0.398**	**0.322**	0.268	**0.342**
τ	COMETKIWI-22	565M	5.02	0.326	0.308	0.401	0.341
	COMETKIWI-XL	3486M	17.36	**0.334**	0.344	0.407	0.357
	COMETKIWI-XXL	10717M	52.51	0.323	**0.346**	**0.427**	**0.358**
ρ	COMETKIWI-22	565M	5.02	0.437	0.409	0.528	0.454
	COMETKIWI-XL	3486M	17.36	**0.449**	0.457	0.540	0.476
	COMETKIWI-XXL	10717M	52.51	0.435	**0.459**	**0.565**	**0.477**

Table 4.4: Kendall's Tau (τ) and Spearman (ρ) correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set. "seconds" is the amount of time taken at inference in seconds per 1000 sentences.

4.2.2 Knowledge Distillation

First, we choose the teacher model. Since the dataset to be labeled has 58.6 million sentence pairs, it is costly to label the entire dataset. For that reason, as explained in Chapter 3, we labeled 5% of the dataset with both COMETKIWI-XL and COMETKIWI-XXL and trained for 3 epochs. We can compare the performance difference in Table 4.5.

	Metric	zh-en	en-de	en-ru	avg.
τ	COMETKIWI-22	0.326	0.308	0.401	0.341
	COMETKIWI-XL	**0.334**	0.344	0.407	0.357
	COMETKIWI-XXL	0.323	**0.346**	**0.427**	**0.358**
	COMETKIWINHO-5%-XL	0.305	0.288	0.348	0.312
	COMETKIWINHO-5%-XXL	0.305	0.289	0.359	0.315
ρ	COMETKIWI-22	0.437	0.409	0.528	0.454
	COMETKIWI-XL	**0.449**	0.457	0.540	0.476
	COMETKIWI-XXL	0.435	**0.459**	**0.565**	**0.477**
	COMETKIWINHO-5%-XL	0.411	0.384	0.463	0.418
	COMETKIWINHO-5%-XXL	0.410	0.385	0.477	0.421

Table 4.5: Kendall's Tau (τ) and Spearman (ρ) correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set.

In Table 4.6, we compare the previous COMETINHO version with the best checkpoint on a 3-epoch knowledge distillation run on the teacher-labeled dataset. Since the dataset is reasonably large, we only freeze the pre-trained encoder for 1% of the first training epoch.

COMETKIWINHO-2.6EPOCHS represents the best checkpoint of the 3-epoch run. COMETINHO and COMETKIWINHO-2.6EPOCHS were trained mainly on the same data. However, we can achieve on av-

	Metric	# Params	seconds	zh-en	en-de	en-ru	avg.
τ	COMETINHO	119M	6.79	0.262	**0.343**	0.330	0.303
	COMETKIWINHO-2.6EPOCHS	280M	3.19	**0.322**	0.297	**0.366**	**0.327**
ρ	COMETINHO	119M	6.79	0.355	**0.453**	0.441	0.405
	COMETKIWINHO-2.6EPOCHS	280M	3.19	**0.434**	0.396	**0.485**	**0.438**

Table 4.6: Kendall's Tau (τ) and Spearman (ρ) correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set. "seconds" is the amount of time taken at inference in seconds per 1000 sentences.

erage 8% better performance on both Kendall's Tau and Spearman correlation. What introduces a big performance difference are two main changes to the architecture:

- A new pre-trained encoder (infoXLM) that has better performance than the one used in COMETINHO, which was a distilled version of XLM-R. Even though the new encoder makes the COMETKIWINHO-2.6EPOCHS model 2.35 times larger, given that the new version is reference-free, as explained in Section 3.2, there is no need to encode an extra sentence, and thus the model is able to be 2.1 times faster still.
- COMETINHO was trained on labels produced by an ensemble of 5 COMET models trained in 2021 [Glushkova et al., 2021], which had significantly lower performance than COMETKIWI-XL, which is the teacher of COMETKIWINHO-2.6EPOCHS.

We then ask the research question of how many times more data is needed to train a student on teacher labels to achieve the same accuracy as a model trained using DA scores. We perform the first experiment by training the same architecture on two different datasets. COMETKIWINHO-DAS model uses the same DA scores COMETKIWI-XL was trained on. The model COMETKIWINHO-5%-XL-1EPOCH uses 5% of the dataset used to train COMETINHO but scored by COMETKIWI-XL. In Table 4.7, we show that a smaller amount of data scored by COMETKIWI-XL, 22% fewer data than the corpus scored using DAs, can result in similar performance levels on average, 2.3% more in Kendall's Tau score and 1% less Spearman correlation score.

	Metric	tuples	zh-en	en-de	en-ru	avg.
τ	COMETKIWINHO-DAS	3.77M	0.306	0.262	0.331	0.301
	COMETKIWINHO-5%-XL-1EPOCH	2.93M	0.307	0.279	0.339	0.308
	COMETKIWINHO-2.6EPOCHS	105.48M	**0.322**	**0.297**	**0.366**	**0.327**
ρ	COMETKIWINHO-DAS	3.77M	0.411	0.352	0.444	0.404
	COMETKIWINHO-5%-XL-1EPOCH	2.93M	0.405	0.352	0.439	0.400
	COMETKIWINHO-2.6EPOCHS	105.48M	**0.434**	**0.396**	**0.485**	**0.438**

Table 4.7: Kendall's Tau (τ) and Spearman (ρ) correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set.

To better conclude this fact, we distilled the model three times, up to 24%, 59%, and 200% (2 epochs) of the full distillation dataset. All training runs had similar behaviors, indicating consistently

better Kendall's Tau correlation than COMETKIWINHO-DAS at 3.2 million tuples, as seen in Figure 4.1, which is 15% fewer data than the data used to train COMETKIWINHO-DAS. This surprising fact is related to another research question we explore further in this chapter: Can labels from sufficiently good teachers be better than humans for fine-tuning?

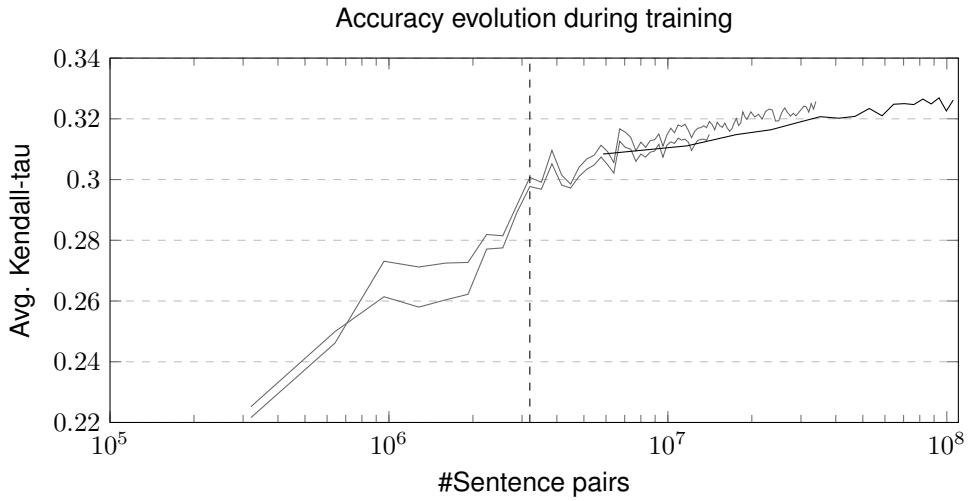


Figure 4.1: In gray are represented two training runs that were not complete, and in black a training run of 2 epochs.

4.2.3 Fine Tuning

To apply fine-tuning to our model, we first determine the label that should be used. As explained in Chapter 3, we performed a grid search on a hyper-parameter λ created to fine-tune a not completely distilled model. The results in Table 4.8 show that a larger λ, meaning more weight to COMETKIWI-XL was beneficial during the entire training period. Each column shows Kendall's Tau correlation of the model up to the number of training steps shown in the header row.

These results can be explained by the fact that automatic metrics based on pre-trained embeddings, such as COMETKIWI-XL, can outperform WMT DA scores when correlated to MQM annotations made by professional translators with access to full document context [Freitag et al., 2021]. Since WMT scores are generated using crowd-sourced annotators with minimal training and DA scores are much more flexible than MQM scores, which results in fairly noisy scores even after statistical methods are put in place to mitigate them.

It is also noted from this experiment that the accuracy of the model decreases at first while doing fine-tuning but then increases again. This performance drop is more noticeable when fine-tuning uses DA scores instead of COMETKIWI-XL labels, which can be explained by the fact that the model was distilled using the latter, so it is already used for these labels.

We then verified the same behavior on a fully distilled model and fine-tuned it for an entire epoch with

λ	2947	5894	8841	11788	14735	avg.
0	0,312	0,303	0,306	0,305	0,308	**0,307**
0.1	0,314	0,309	0,310	0,310	0,314	0,311
0.2	0,315	0,310	0,312	0,312	0,316	0,313
0.3	0,315	0,311	0,313	0,313	0,317	0,314
0.4	0,315	0,312	0,313	0,314	0,318	0,314
0.5	0,315	0,312	0,314	0,314	0,319	0,315
0.6	0,315	0,313	0,314	0,314	0,319	0,315
0.7	0,315	0,313	0,315	0,315	0,319	0,315
0.8	0,315	0,313	0,315	0,315	0,319	0,315
0.9	0,315	0,313	0,315	0,315	0,319	0,316
1	0,315	0,313	0,315	0,315	0,320	**0,316**
avg.	0,315	0,311	0,313	0,313	0,317	0,314

Table 4.8: Kendall's Tau correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set.

only DA scores and with only COMETKIWI-XL scores. The results in Table 4.9 confirmed the previous observation. The model fine-tuned on COMETKIWI-XL scores are consistently better in all language pairs and are on average 3% better in Kendall's Tau and Spearman correlation.

	Metric	zh-en	en-de	en-ru	avg.
τ	COMETKIWINHO-FINE-TUNED-DAS	0.308	0.278	0.354	0.312
	COMETKIWINHO-FINE-TUNED-COMETKIWI-XL	**0.318**	**0.292**	**0.361**	**0.323**
ρ	COMETKIWINHO-FINE-TUNED-DAS	0.416	0.371	0.472	0.419
	COMETKIWINHO-FINE-TUNED-COMETKIWI-XL	**0.429**	**0.389**	**0.480**	**0.432**

Table 4.9: Kendall's Tau (τ) and Spearman (ρ) correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set.

As in the experiment in Table 4.8, the performance declined initially in both fine-tunings and then increased, as shown in Figure 4.2. This effect is more prominent for the model fine-tuned with DA scores.

Since the Kendall Tau score was not reaching a plateau at 1 epoch, we then fine-tuned the model for 3 entire epochs, increasing the performance by 10 Kendall's Tau points, as shown in Figure 4.3.

Finally, we compare this last experiment that results in our final model, dubbed COMETKIWINHO, with the work previously done on COMETINHO [Rei et al., 2022a]. As shown in Table 4.10, COMETKIWINHO has more 27 Kendall's Tau points and 36 Spearman points than COMETINHO, which translates to approximately 9% improvement in both Kendall's Tau and Spearman correlations. When compared to the state-of-the-art at the start of the thesis, COMETKIWINHO is only 3% less performant than COMETKIWI-22 in both Kendall's Tau and Spearman correlations while being 2 times smaller and 1.6 times faster.

We should also highlight that across all experiments, the English→German language pair was the least performing language pair, and even COMETINHO is better than COMETKIWI-22 in this specific language pair. This is mostly due to the composition of the corpora used to train the pre-encoders of

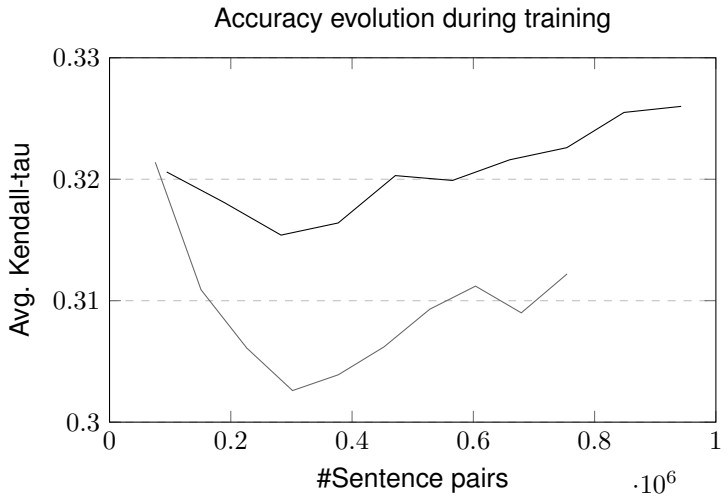


Figure 4.2: In gray is represented fine-tuning with direct assessments and in black with COMETKIWI-XL labels.

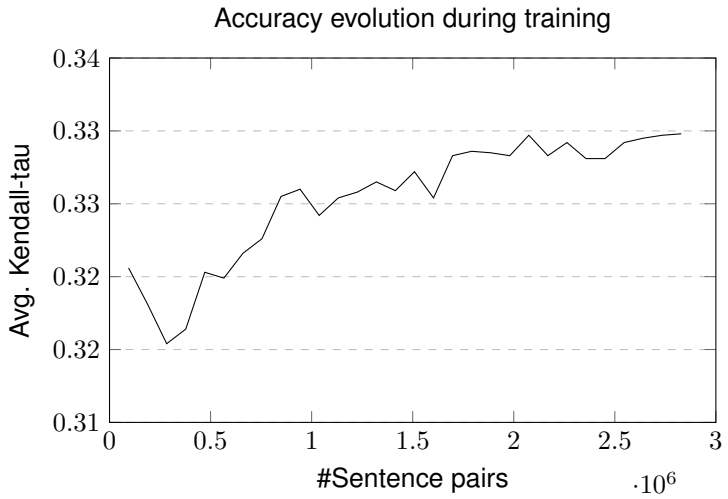


Figure 4.3: Fine tuning for with COMETKIWI-XL labels for 3 entire epochs.

COMETINHO and COMETKIWI-22. COMETINHO uses XLM-R, which has a more balanced corpus with 12.6% English, 2.6% German, and 70.3% of tuples from languages not used in our test set. While COMETKIWI-22 uses infoXLM-L, which is more skewed to English with 35.2% of tuples, followed by German with 4.8% and other languages not in the test set with only 43.1%. Although the percentage of German is higher in infoXLM-L, the fact that the pre-trained encoder is less exposed to other languages and overly exposed to English can be a reason for the difference in the specific language pair English→German.

When looking at the performance gap between COMETINHO and the state-of-the-art COMETKIWI-22, we see a difference of 38 Kendall's Tau points and 49 Spearman points. We were able to bridge 71% and 73% of those gaps, respectively. In the next chapter, we summarize the conclusions of this work and

propose future work to better close this gap with new state-of-the-art models such as COMETKIWI-XL and COMETKIWI-XXL.

	Metric	# Params	seconds	zh-en	en-de	en-ru	avg.
τ	COMETINHO	119M	6.79	0.262	**0.343**	0.330	0.303
	COMETKIWINHO	280M	3.19	0.322	0.303	0.370	0.330
	COMETKIWI-22	565M	5.02	**0.326**	0.308	**0.401**	**0.341**
ρ	COMETINHO	119M	6.79	0.355	**0.453**	0.441	0.405
	COMETKIWINHO	280M	3.19	0.434	0.403	0.491	0.441
	COMETKIWI-22	565M	5.02	**0.437**	0.409	**0.528**	**0.454**

Table 4.10: Kendall's Tau (τ) and Spearman (ρ) correlation on high resource language pairs using MQM scores from the WMT 2022 Metrics Task News test set. "seconds" is the amount of time taken at inference in seconds per 1000 sentences.

5

Conclusions and Future Work

Contents

5.1 Conclusions

In this work, our main contribution was the development of a reference-free COMETINHO-like model. The goal was to achieve a model with the best possible trade-off between efficiency and performance. We started with our previous work on COMETINHO as a baseline, focused on knowledge distillation as the primary method for model compression, and further improved our results through fine-tuning. To achieve better results, we leveraged recent developments that led to the availability of larger evaluation models to use as teachers. We use COMETKIWI-XL and a similar training set used in COMETINHO, and that alone yielded already surprising results without the need to use the reference to perform the evaluation, making the model faster, which allows the use of larger and thus more powerful pre-trained encoders and still be faster than the previous COMETINHO model.

We then introduced ways to further improve the quality of the model by fine-tuning it on a smaller dataset that was labeled with DAs and found that these labels were less helpful to learning MQM annotations than the labels of the teacher model on the same dataset. The final distilled and fine-tuned result was a model that bridges the performance gap between lighter metrics (COMETINHO) and state-of-the-art metrics (COMETKIWI-22), unlocking potential new applications that were less feasible without this new model.

5.2 Future Work

A primary avenue for future work is to further study techniques to perform model compression better:

- Applying knowledge distillation with computationally more expensive methods such as self-attention distillation [Wang et al., 2020];
- Using even smaller models as pre-trained encoders, such as the MINILM [Wang et al., 2020];
- Apply other model compression techniques pre and post-knowledge distillation, such as quantization-aware training and pruning attention heads of the transformer-based pre-trained encoder or dynamically pruning layers;
- Curate a better training set for knowledge distillation using data pruning [Sorscher et al., 2022].

Finally, we can further investigate the performance and robustness of our new model in more domains and in different use cases, such as N-best reranking or inference in CPU and other hardware-restricted use cases.

Bibliography

[Aghajanyan et al., 2020] Aghajanyan, A., Zettlemoyer, L., and Gupta, S. (2020). Intrinsic dimensionality explains the effectiveness of language model fine-tuning. *arXiv preprint arXiv:2012.13255*.

[Bahdanau et al., 2014] Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

[Bai et al., 2020] Bai, H., Zhang, W., Hou, L., Shang, L., Jin, J., Jiang, X., Liu, Q., Lyu, M., and King, I. (2020). Binarybert: Pushing the limit of bert quantization. *arXiv preprint arXiv:2012.15701*.

[Banerjee and Lavie, 2005] Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

[Bengio et al., 2000] Bengio, Y., Ducharme, R., and Vincent, P. (2000). A neural probabilistic language model. *Advances in neural information processing systems*, 13.

[Bojar et al., 2016] Bojar, O., Chatterjee, R., Federmann, C., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Logacheva, V., Monz, C., et al. (2016). Findings of the 2016 conference on machine translation. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 131–198.

[Bojar et al., 2017] Bojar, O., Helcl, J., Kocmi, T., Libovickỳ, J., and Musil, T. (2017). Results of the wmt17 neural mt training task. In *Proceedings of the second conference on machine translation*, pages 525–533.

[Brown et al., 2020] Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. (2020). Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

[Chi et al., 2020] Chi, Z., Dong, L., Wei, F., Yang, N., Singhal, S., Wang, W., Song, X., Mao, X.-L., Huang, H., and Zhou, M. (2020). Infoxlm: An information-theoretic framework for cross-lingual language model pre-training. *arXiv preprint arXiv:2007.07834*.

[Chung et al., 2020] Chung, H. W., Fevry, T., Tsai, H., Johnson, M., and Ruder, S. (2020). Rethinking embedding coupling in pre-trained language models. *arXiv preprint arXiv:2010.12821*.

[Conneau et al., 2019] Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. (2019). Unsupervised cross-lingual representation learning at scale. *arXiv preprint arXiv:1911.02116*.

[Conneau and Lample, 2019] Conneau, A. and Lample, G. (2019). Cross-lingual language model pretraining. *Advances in neural information processing systems*, 32.

[Deerwester et al., 1990] Deerwester, S., Dumais, S. T., Furnas, G. W., Landauer, T. K., and Harshman, R. (1990). Indexing by latent semantic analysis. *Journal of the American society for information science*, 41(6):391–407.

[Dettmers et al., 2022] Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). Llm. int8 (): 8-bit matrix multiplication for transformers at scale. *arXiv preprint arXiv:2208.07339*.

[Deutsch et al., 2023] Deutsch, D., Foster, G., and Freitag, M. (2023). Ties matter: Modifying kendall's tau for modern metric meta-evaluation. *arXiv preprint arXiv:2305.14324*.

[Devlin et al., 2018a] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018a). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Devlin et al., 2018b] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2018b). Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

[Doddington, 2002] Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145.

[Dong et al., 2017] Dong, X., Chen, S., and Pan, S. (2017). Learning to prune deep neural networks via layer-wise optimal brain surgeon. *Advances in neural information processing systems*, 30.

[Fan et al., 2019] Fan, A., Grave, E., and Joulin, A. (2019). Reducing transformer depth on demand with structured dropout. *arXiv preprint arXiv:1909.11556*.

[Fernandes et al., 2022] Fernandes, P., Farinhas, A., Rei, R., de Souza, J. G., Ogayo, P., Neubig, G., and Martins, A. F. (2022). Quality-aware decoding for neural machine translation. *arXiv preprint arXiv:2205.00978*.

[Freitag et al., 2021] Freitag, M., Foster, G., Grangier, D., Ratnakar, V., Tan, Q., and Macherey, W. (2021). Experts, errors, and context: A large-scale study of human evaluation for machine translation. *Transactions of the Association for Computational Linguistics*, 9:1460–1474.

[Glushkova et al., 2021] Glushkova, T., Zerva, C., Rei, R., and Martins, A. F. (2021). Uncertainty-aware machine translation evaluation. *arXiv preprint arXiv:2109.06352*.

[Goyal et al., 2021] Goyal, N., Du, J., Ott, M., Anantharaman, G., and Conneau, A. (2021). Larger-scale transformers for multilingual masked language modeling. *arXiv preprint arXiv:2105.00572*.

[Graham et al., 2013] Graham, Y., Baldwin, T., Moffat, A., and Zobel, J. (2013). Continuous measurement scales in human evaluation of machine translation. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*, pages 33–41.

[Guo, 2018] Guo, Y. (2018). A survey on methods and theories of quantized neural networks. *arXiv preprint arXiv:1808.04752*.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

[Houlsby et al., 2019] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., De Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-efficient transfer learning for nlp. In *International Conference on Machine Learning*, pages 2790–2799. PMLR.

[Hu et al., 2021] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

[IEA, 2022] IEA (2022). Renewables share of power generation in the net zero scenario, 2010-2030, iea, paris. https://www.iea.org/data-and-statistics/charts/renewables-share-of-power-generation-in-the-net-zero-scenario-2010-2030.

[Jacob et al., 2018] Jacob, B., Kligys, S., Chen, B., Zhu, M., Tang, M., Howard, A., Adam, H., and Kalenichenko, D. (2018). Quantization and training of neural networks for efficient integer-arithmetic-only inference. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2704–2713.

[Kendall, 1938] Kendall, M. G. (1938). A new measure of rank correlation. *Biometrika*, 30(1/2):81–93.

[Kendall, 1945] Kendall, M. G. (1945). The treatment of ties in ranking problems. *Biometrika*, 33(3):239–251.

[Kepler et al., 2019] Kepler, F., Trénous, J., Treviso, M., Vera, M., and Martins, A. F. (2019). Openkiwi: An open source framework for quality estimation. *arXiv preprint arXiv:1902.08646*.

[Kim et al., 2017] Kim, H., Lee, J.-H., and Na, S.-H. (2017). Predictor-estimator using multilevel task learning with stack propagation for neural quality estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568.

[Kocmi et al., 2021] Kocmi, T., Federmann, C., Grundkiewicz, R., Junczys-Dowmunt, M., Matsushita, H., and Menezes, A. (2021). To ship or not to ship: An extensive evaluation of automatic metrics for machine translation. *arXiv preprint arXiv:2107.10821*.

[LeCun et al., 1989] LeCun, Y., Denker, J., and Solla, S. (1989). Optimal brain damage. *Advances in neural information processing systems*, 2.

[Lin, 2004] Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

[Liu et al., 2021] Liu, P., Yuan, W., Fu, J., Jiang, Z., Hayashi, H., and Neubig, G. (2021). Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *arXiv preprint arXiv:2107.13586*.

[Lo, 2019] Lo, C.-k. (2019). Yisi-a unified semantic mt quality evaluation and estimation metric for languages with different levels of available resources. In *Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1)*, pages 507–513.

[Lommel et al., 2014] Lommel, A., Uszkoreit, H., and Burchardt, A. (2014). Multidimensional quality metrics (mqm): A framework for declaring and describing translation quality metrics. *Revista Tradumàtica: tecnologies de la traducció*, (12):455–463.

[Lund and Burgess, 1996] Lund, K. and Burgess, C. (1996). Producing high-dimensional semantic spaces from lexical co-occurrence. *Behavior research methods, instruments, & computers*, 28(2):203–208.

[Luong et al., 2015] Luong, M.-T., Pham, H., and Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

[Ma et al., 2018] Ma, Q., Bojar, O., and Graham, Y. (2018). Results of the wmt18 metrics shared task: Both characters and embeddings achieve good performance. In *Proceedings of the third conference on machine translation: shared task papers*, pages 671–688.

[Ma et al., 2019] Ma, Q., Wei, J. T.-Z., Bojar, O., and Graham, Y. (2019). Results of the wmt19 metrics shared task: Segment-level and strong mt systems pose big challenges. Association for Computational Linguistics.

[Martins and Astudillo, 2016] Martins, A. and Astudillo, R. (2016). From softmax to sparsemax: A sparse model of attention and multi-label classification. In *International conference on machine learning*, pages 1614–1623. PMLR.

[Micikevicius et al., 2017] Micikevicius, P., Narang, S., Alben, J., Diamos, G., Elsen, E., Garcia, D., Ginsburg, B., Houston, M., Kuchaiev, O., Venkatesh, G., et al. (2017). Mixed precision training. *arXiv preprint arXiv:1710.03740*.

[Mikolov et al., 2013a] Mikolov, T., Chen, K., Corrado, G., and Dean, J. (2013a). Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

[Mikolov et al., 2013b] Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In *Proceedings of the 2013 conference of the north american chapter of the association for computational linguistics: Human language technologies*, pages 746–751.

[Moore, 1998] Moore, G. (1998). Cramming more components onto integrated circuits. *Proceedings of the IEEE*, 86(1):82–85.

[Nair and Hinton, 2010] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)*, pages 807–814.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

[Pennington et al., 2014] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543.

[Peters et al., 2018a] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018a). Deep contextualized word representations.

[Peters et al., 2018b] Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018b). Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

[Popović, 2015] Popović, M. (2015). chrf: character n-gram f-score for automatic mt evaluation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 392–395.

[Pu et al., 2021] Pu, A., Chung, H. W., Parikh, A. P., Gehrmann, S., and Sellam, T. (2021). Learning compact metrics for mt. *arXiv preprint arXiv:2110.06341*.

[Radford et al., 2018] Radford, A., Narasimhan, K., Salimans, T., and Sutskever, I. (2018). Improving language understanding by generative pre-training. *Homology, Homotopy and Applications*, -.

[Raffel et al., 2020] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., Liu, P. J., et al. (2020). Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67.

[Ramírez-Sánchez et al., 2020] Ramírez-Sánchez, G., Zaragoza-Bernabeu, J., Bañón, M., and Ortiz-Rojas, S. (2020). Bifixer and bicleaner: two open-source tools to clean your parallel data. In *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, pages 291–298.

[Ranasinghe et al., 2020] Ranasinghe, T., Orasan, C., and Mitkov, R. (2020). Transquest: Translation quality estimation with cross-lingual transformers. *arXiv preprint arXiv:2011.01536*.

[Rei et al., 2022a] Rei, R., Farinha, A. C., de Souza, J. G., Ramos, P. G., Martins, A. F., Coheur, L., and Lavie, A. (2022a). Searching for cometinho: The little metric that could. In *Proceedings of the 23rd Annual Conference of the European Association for Machine Translation*, pages 61–70.

[Rei et al., 2021] Rei, R., Farinha, A. C., Zerva, C., van Stigt, D., Stewart, C., Ramos, P., Glushkova, T., Martins, A. F., and Lavie, A. (2021). Are references really needed? unbabel-ist 2021 submission for the metrics shared task. In *Proceedings of the Sixth Conference on Machine Translation*, pages 1030–1040.

[Rei et al., 2023] Rei, R., Guerreiro, N. M., Pombal, J., van Stigt, D., Treviso, M., Coheur, L., de Souza, J. G., and Martins, A. F. (2023). Scaling up cometkiwi: Unbabel-ist 2023 submission for the quality estimation shared task. *arXiv preprint arXiv:2309.11925*.

[Rei et al., 2020] Rei, R., Stewart, C., Farinha, A. C., and Lavie, A. (2020). Comet: A neural framework for mt evaluation. *arXiv preprint arXiv:2009.09025*.

[Rei et al., 2022b] Rei, R., Treviso, M., Guerreiro, N. M., Zerva, C., Farinha, A. C., Maroti, C., De Souza, J. G., Glushkova, T., Alves, D. M., Lavie, A., et al. (2022b). Cometkiwi: Ist-unbabel 2022 submission for the quality estimation shared task. *arXiv preprint arXiv:2209.06243*.

[Rumelhart et al., 1985] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1985). Learning internal representations by error propagation.

[Sanh et al., 2019] Sanh, V., Debut, L., Chaumond, J., and Wolf, T. (2019). Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

[Sellam et al., 2020] Sellam, T., Das, D., and Parikh, A. P. (2020). Bleurt: Learning robust metrics for text generation. *arXiv preprint arXiv:2004.04696*.

[Sevilla et al., 2022] Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. (2022). Compute trends across three eras of machine learning. *arXiv preprint arXiv:2202.05924*.

[Sharir et al., 2020] Sharir, O., Peleg, B., and Shoham, Y. (2020). The cost of training nlp models: A concise overview. *arXiv preprint arXiv:2004.08900*.

[Shimanaka et al., 2018] Shimanaka, H., Kajiwara, T., and Komachi, M. (2018). Ruse: Regressor using sentence embeddings for automatic machine translation evaluation. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 751–758.

[Snover et al., 2006] Snover, M., Dorr, B., Schwartz, R., Micciulla, L., and Makhoul, J. (2006). A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.

[Sorscher et al., 2022] Sorscher, B., Geirhos, R., Shekhar, S., Ganguli, S., and Morcos, A. S. (2022). Beyond neural scaling laws: beating power law scaling via data pruning. *arXiv preprint arXiv:2206.14486*.

[Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1):1929–1958.

[Stahlberg, 2020] Stahlberg, F. (2020). Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418.

[Sutskever et al., 2014] Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27.

[Thompson and Post, 2020] Thompson, B. and Post, M. (2020). Automatic machine translation evaluation in many languages via zero-shot paraphrasing. *arXiv preprint arXiv:2004.14564*.

[Tiedemann, 2012] Tiedemann, J. (2012). Parallel data, tools and interfaces in opus. In *Lrec*, volume 2012, pages 2214–2218. Citeseer.

[Treviso et al., 2023] Treviso, M., Lee, J.-U., Ji, T., Aken, B. v., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Raffel, C., et al. (2023). Efficient methods for natural language processing: A survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. (2017). Attention is all you need. *Advances in neural information processing systems*, 30.

[Voita et al., 2019] Voita, E., Talbot, D., Moiseev, F., Sennrich, R., and Titov, I. (2019). Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. *arXiv preprint arXiv:1905.09418*.

[Wan et al., 2022a] Wan, Y., Liu, D., Yang, B., Bi, T., Zhang, H., Chen, B., Luo, W., Wong, D. F., and Chao, L. S. (2022a). Robleurt submission for the wmt2021 metrics task. *arXiv preprint arXiv:2204.13352*.

[Wan et al., 2022b] Wan, Y., Liu, D., Yang, B., Zhang, H., Chen, B., Wong, D. F., and Chao, L. S. (2022b). Unite: Unified translation evaluation. *arXiv preprint arXiv:2204.13346*.

[Wang et al., 2020] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers. *Advances in Neural Information Processing Systems*, 33:5776–5788.

[Zhang et al., 2019] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2019). Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.