TÉCNICO LISBOA

Glass-Box Quality Estimation for Neural Machine Translation

João Pinto Correia de Moura

Thesis to obtain the Master of Science Degree in

Mechanical Engineering

Supervisors: Prof. André Filipe Torres Martins
Prof. João Miguel da Costa Sousa

Examination Committee

Chairperson: Prof. Carlos Cardeira
Supervisor: Prof. André Filipe Torres Martins
Members of the Committee: Prof. Bruno Emanuel da Graça Martins
Dr. Fabio Natanael Kepler

October, 2021

Para o meu avô Álvaro, colega Engenheiro.

Declaration

I declare that this document is an original work of my own authorship and that it fulfills all the requirements of the Code of Conduct and Good Practices of the Universidade de Lisboa.

Acknowledgments

First, I would like thank my supervisors Prof. João Sousa, and especially Prof. André Martins; his support and meaningful discussions provided me with the most rewarding intelectual experience I had in my time at IST, and kindled my passion for Artificial Intelligence, which I'm sure will accompany me for the foreseeable future. This was also paramount for the success of my work, and I could not have asked for better guidance.

I had the opportunity to develop this thesis in an incredibly dynamic environment at Unbabel, a company for which I have deep appreciation, and regard as a beacon of talent and innovation in Portugal. I want to especially thank Fabio Kepler, Miguel Vera, and Daan van Stigt, ever-present to share their knowledge, and who also strongly collaborated and guided me through the process of publishing a scientific paper with our findings.

Additionally, I am very grateful for the group of amazing people at IST who I worked with during all these years, and that I am honoured to call my friends. My time here would not be the same without them, and I hope to see them flourish for many years to come in their professional and personal lives.

Last, but absolutely not least, a big thank-you to my parents and brothers, who relentlessly strove to make my path possible, and without whom no good thing in life would be as rewarding; to my girlfriend, who has supported me unwaveringly through thick and thin, and everyday helps me become a better person; and to my late grandfather, my role model, who I would have given everything to share the news of my graduation with.

Resumo

A Estimação de Qualidade de Tradução tem-se tornado cada vez mais relevante nos últimos anos para o desenvolvimento de aplicações de Tradução Automática prácticas, com avanços recentes no campo de Processamento de Linguagem Natural a desbloquear novas abordagens à tarefa. Apesar das grandes melhorias que os sistemas de Estimação de Qualidade mais avançados demonstram, a maior parte negligencia uma fonte de informação promissora: o sistema de tradução sob avaliação é tratado como uma caixa negra, e apenas o seu *input* e *output* são considerados.

Nesta tese, introduzimos um método que integra informação extraída dos mecanismos internos de modelos de Tradução Automática, no processo de treino de modelos de Estimação de Qualidade - ao qual chamamos Estimação de Qualidade de Caixa de Vidro. Primeiro, com o objectivo de extrair esta informação interna, aproveitamos métodos de quantificação de incerteza existentes baseados em *Monte Carlo dropout*, os quais publicações recentes demonstraram levar à criação de representações relevantes à estimação de qualidade de traduções automáticas. Depois, propomos uma arquitectura de modelo original baseada no padrão *Predictor-Estimator*, acompanhada de um método que permite integrar as representações extraídas no processo de treino deste mesmo modelo. Finalmente, realizamos uma análise empírica, baseada em seis pares de linguagens no contexto da *WMT Quality Estimation Shared Task*, com resultados animadores. A análise do modelo proposto que levamos a cabo sugere várias direções para exploração e melhorias no futuro.

Palavras-chave: Aprendizagem Profunda, Processamento de Linguagem Natural, Estimação de Qualidade, Quantificação de Incerteza

Abstract

Quality Estimation has become increasingly relevant in the last few years for practical and confidence-aware Machine Translation applications, with recent advancements in the field of Natural Language Processing having enabled new approaches to the task. Despite the great improvements that state-of-the-art Quality Estimation systems boast, most overlook a promising source of information: the translation system under evaluation is treated as a black box, with only its input and output being regarded.

In this thesis, we introduce a method which allows for the integration of information extracted from the internal mechanisms of Machine Translation models, into the training process of Quality Estimation models, which we call Glass-Box Quality Estimation. First, in order to extract this internal information, we leverage existing model uncertainty quantification methods based on Monte Carlo dropout, which recent work has shown to yield features highly relevant to estimating the quality of machine translated text. We then propose a novel model architecture based on the Predictor-Estimator framework, and an accompanying method to integrate the extracted features into the model's training procedure. Finally, we provide an empirical evaluation based on six language pairs in the context of the *WMT Quality Estimation Shared Task*, with encouraging results. Our analysis of the proposed model suggests various directions for future improvements.

Keywords: Deep Learning, Natural Language Processing, Quality Estimation, Uncertainty Quantification

Contents

List of Tables

List of Figures

Acronyms

DL Deep Learning.

DNN Deep Neural Network.

GPU Graphics Processing Unit.

MT Machine Translation.

NLP Natural Language Processing.

NMT Neural Machine Translation.

QE Quality Estimation.

RNN Recurrent Neural Network.

SMT Statistical Machine Translation.

XLU Cross-Lingual Language Understanding.

Chapter 1

Introduction

The beginning of the 21st century has seen developments in the field of Artificial Intelligence (AI) at an unprecedented rate. It has ever since revolutionized several industries, as the adoption of its methods to solve specific problems increases, and AI-based systems become part of everyday digital life.

The use of Deep Neural Networks (DNN) in particular has seeped into virtually every field of scientific study [1]. Although these were invented in their simplest form in the late 50's [53], it was only in 2009 that advances in parallel computing hardware and software enabled the feasibility and practicality of their training in large-scale applications [8], namely with the novel use of Graphics Processing Units (GPU's) for the purpose [51]. This method was consolidated in 2012, when AlexNet [36] was trained on such accelerated hardware to classify images from the popular dataset ImageNet using a deeper, wider network than ever before [13]. This resulted in a large performance increase in comparison to former traditional Machine Learning and Computer Vision approaches, becoming an achievement widely regarded as the inflection point in academic interest in Deep Learning (DL) [2].

Recently, the field of Natural Language Processing started enjoying its "ImageNet moment". The invention of the Transformer model in 2017 (originally applied to Machine Translation) [70] has been fueling the development of very useful models, which achieve state-of-the-art performance and robustness in NLP tasks, leveraging large amounts of data through novel unsupervised learning approaches. Most importantly, the fact that most of these models are made available for research purposes, exploited by the use of transfer learning techniques, has empowered and democratized Research & Development in many different downstream tasks.

Machine Translation (MT) in specific has enjoyed a big transformation, both as a research field and industry. The replacement of Statistical MT (SMT) by Neural MT (NMT) is widespread in commercial settings, and translation engines are now almost ubiquitously powered by Deep Neural Networks. The improvement of translation quality obtained from these models, more user-friendly tools and higher demand for translation has universalized the use of MT models in the translation industry; this setting brings added importance to the development of solutions for a different set of problems, such as:

- which segments need revision by a human translator?
- how much effort will be needed to fix a poorly translated segment?
- which one of many translations created by different models should be picked as the best one?

This thesis' work will address and explore methods that can be used to answer the question stated above.

1.1 Motivation and Related Work

MT evaluation methods have typically relied on manually produced references obtained from professional translators, in order to assess translation quality; such methods compare the translations generated by an MT model to one or many provided reference translations. Various MT evaluation metrics have been proposed over the years, such as BLEU [46], NIST [15], METEOR [5], TER [58], and more recently COMET [52], PRISM [69] and BERTScore [73]. These are in general computationally inexpensive to calculate, and enable MT technology research and development, by providing the frequent evaluation of models that is needed in order to assess their improvement over previous iterations.

However, one of the shortcomings of these methods is that they cannot be used to provide quality scores when a reference translation is not available; this is above all impractical in a live translation scenario.

MT Quality Estimation (QE) is the task that addresses this situation. It consists of predicting the quality of a system's output for a given input, without any information or reference about the expected output, therefore being aimed at MT models in use [62]. This type of system can be designed for prediction at different granularity levels, from word or sentence, to paragraph or document. Table 1.1 shows an example of the type of data involved in the QE task, as well as the desired predictions - *OK* and *BAD* tags for word-level, HTER or DA score for sentence-level (these metrics will be detailed in section 2.3.1). Figure 1.1 shows a high-level schematic of a QE model's functioning.

Source	*And it's SO HORRIBLE when I do not get that hour.*
MT	*Y es HORRIBLE cuando no consigo esa hora.*
QE (word)	OK OK BAD BAD OK BAD OK OK OK
PE (reference)	Y es TERRIBLE si no tengo esa hora.
QE (sentence)	HTER =33.3 % ‖ Z-standardized DA =-0.42

Table 1.1: Data involved in both the word and sentence-level QE tasks. Poorly translated words are labelled *BAD* and correct words labelled *OK*; HTER is calculated from the number of operations needed to correct - or post-edit (PE in table) - a translation, and Z-standardized DA is derived from direct human judgements. These quality scores are described in detail in section 2.3.1.

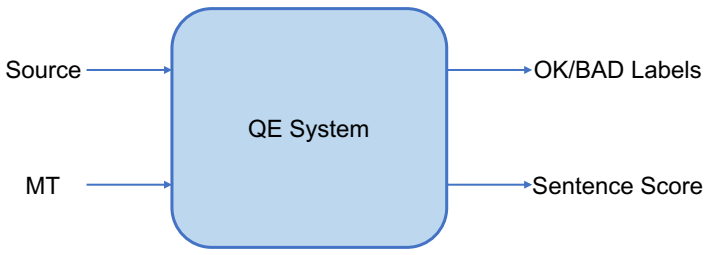

Figure 1.1: Schematic of QE system, trained to predict word labels and/or sentence scores, with no access to reference translations.

The first efforts in estimating MT quality were mainly focused on assessing how confident the MT model was in its output translation. This task was called Confidence Estimation (CE) [7, 50, 19], and differed from what is understood today as QE in that it made indispensable the availability of the MT model, and the usage of features internal to it. Being based on information obtained from structures inherent to a specific MT model implementation, these are refered to as *glass-box* features. On the other side of the spectrum, *black-box* features are those independent of any MT model, and extracted solely from source and target sentences, therefore treating the MT system opaquely.

Systems that leverage the latter have the advantage of being useful even when inspection of MT models' internals is not a possibility, which is often the case. With the increase of interest in this task, fostered and supported by the many insights gathered in annual editions of the *WMT Shared Task on Quality Estimation*[1] [9] which started in 2012 - and also its practical applicability - the usage of black-box, linguistics-based features was explored more heavily in the literature. The QuEst framework [63] proposed 17 quality indicators - some glass-box, some black-box - that became for a long period of time the standard baseline features for the task, leveraged and adapted in many pieces of further research. On a somewhat distinct direction, works such as [59] and [56] used features based on the comparison between the translation being evaluated and *pseudo-references* - translations generated by different MT models, for the same input.

Generally across this era of QE research, in order to predict a sentence score, sets of the aforementioned features - glass-box, black-box or both - were used to train a classification or regression model, depending on what type of score was being predicted. Several supervised learning algorithms were used for this purpose, perhaps the most commonly applied being those based on Support Vector Machines [59, 25, 6]. Other algorithms such as regression trees [59] and partial least squares [61] were put to use. Earlier work focused on predicting "*OK*"/"*BAD*" labels or integer scores from 1-4/1-5, but later the focus shifted to predicting a score in the range of [0,1], meant to reflect the post-editing effort needed to correct a machine translated sentence, in order for it to match the semantic meaning of a reference translation(s). This measure is the Human-targeted Translation Error Rate, or HTER [58], and it has since then become the standard prediction target and quality indicator on a sentence level, due to its stronger correlation with human judgements, when compared to contemporary counterparts like TER, BLEU and METEOR.

Regardless of the features used, all work in this line of research was bound in common by one aspect: the hand-crafting of the features. The choices made on how to engineer individual features, or what features to use for a given task, are part of a completely empirical process, mainly based on linguistic intuition at an initial stage, proceeded by trial and error and feature selection. This process is also task dependent, and therefore posits added research effort for every new or updated NLP task. In 2011, Collobert et al. [11] was one of the first works to propose the use of Neural Networks to address the problem in a radically different way. Instead of using engineered features derived from text and

[1] http://www.statmt.org/wmt12/quality-estimation-task.html

thought to be relevant for a given task, the system was fed raw text and learned internal representations - or embeddings - in an end-to-end fashion, iteratively trained by backpropagation [55] to be relevant to the task. In fact, the authors argued that the benchmark features commonly used were indirect measurements of the relevance of these representations which, discovered by the learning procedure, were more general than any of the benchmarks.

The success achieved by this method sparked a research direction more focused on improving the NN architectures that learned the embeddings, and less influenced by prior linguistic knowledge. QUETCH [35] proposed a window-based Feed-Forward NN architecture predominantly based on the one proposed in Collobert et al. [11]. In this word-level approach, the input to the NN is a bilingual context window, comprised of a target context window centered on the target word, and a source context window centered on the corresponding source word, from a source sentence that has been ”aligned” - or mapped back, on a word or multiword basis - to the target sentence; this was obtained using a word alignment tool [16]. Advancing on this work, Patel and M [47] developed an approach based on Recurrent Neural Networks (RNN)’s. Despite putting the same bilingual context window to use, the leveraging of RNN’s allowed for the modelling of sequential dependence between output QE labels. Martins et al. [41] proposed three variations of QUETCH, with slight changes to the bilingual context window: a convolutional model, a bidirectional RNN model, and a convolutional RNN model. Additionally taking advantage of linguistic knowledge, they ensembled all three models, along with a feature-enriched linear model trained on hand-crafted features.

The bilingual context windows used as inputs across these contributions involved the need for an additional Statistical MT module, given that a word alignment component was additionally required. In a seminal publication by Kim et al. [33], the Predictor-Estimator architecture was proposed as the first entirely neural approach to QE, a modification of the RNN encoder-decoder [10]) widely used for translation. Here, Predictor and Estimator address two different tasks, and are therefore separately trained with different training data. The Predictor component is pre-trained with the task of correctly predicting each word in a target sentence, conditioned on all other source and target words (except for the one being predicted). A bidirectional bilingual RNN is employed for this purpose, and parallel data is fed to the model for training. In the process, the Predictor learns internal representations relevant to the task; these are then extracted, neurally combined into QE Feature Vectors (QEFV’s), and used to train the Estimator component, with the objective of estimating translation quality at different granularity levels. It is shown that the QEFV’s approximate the role of the transferred knowledge from word prediction to quality estimation. Although modified in terms of the neural components used, the Predictor-Estimator architecture continues to be the state-of-the-art for both word and sentence level QE.

The invention of the Transformer model in 2017 by Vaswani et al. [70] had a technical paradigm shifting effect. Akin to RNN’s, the Transformer is effective at modelling sequences and dependencies within them. However, it strongly leverages a self-attention mechanism, which removes the need for recurrence; this factor allows for sequences to be processed in parallel (i.e. all words at the same

time) instead of sequentially, and enables great improvements on the long-range dependency modelling capabilities of RNN's. Originally proposed for the task of NMT, the Transformer is composed of an Encoder - which creates a high-dimensional bidirectional representation of the source sentence -, and a Decoder - which takes this representation and tries to correctly predict the target sentence word by word.

Taking notice of the sentence-level representational power of the Transformer Encoder, Devlin et al. [14] proposed the BERT model short after, a standalone modified Encoder. Like the original, its attention mechanism is allowed to attend to all words in a sequence, but is instead pre-trained using a "masked language model" (MLM) objective. Inspired by the Cloze Task [68], this consists of randomly masking some of the tokens from the input, and training the model to predict what words correspond to the masks. In addition, a second objective of "next sentence prediction" was used, which trains text-pair representations jointly with the MLM task. One of the most emphasised observations from this work was the state-of-the-art performance achieved by fine-tuning BERT on a variety of different NLP tasks, by adding a simple task-specific final classification layer. More impressively, performance reduction was shown to be almost negligible whether BERT was fine-tuned along with the new classification layer on the downstream task, or not updated at all, and just used for its word and sentence embeddings, already independently valuable due to the vast amount of data used in pre-training. Their quality solidified the transformer encoder/MLM objective pair as the state-of-the-art and tool of choice for creating language representations, launching a research direction that has since seen many proposals of BERT descendants: multilingual versions of BERT, models with optimizations/add-ons to the original training objectives, benefiting from architectural changes which improved performance [38, 39], and most interestingly for QE, models that are trained on a vast set of languages [37, 12]. In these publications, it is shown that training on many languages at the same time greatly benefits the model's power to learn common representations between them, resulting in improved performance when putting the model to use on downstream cross-lingual tasks (especially benefiting ones where low-resource languages are involved, for which little data is available), while at the same time maintaining a performance level comparable to monolingual models, tested on monolingual tasks.

The characteristics of these models and the multi-lingual richness of their embeddings made them an obvious choice to serve as the Predictor in a Predictor-Estimator QE architecture, and used for QE by leveraging transfer-learning. As of this writing, said approach has proved to be the most performant method of doing Quality Estimation, being a part (in varying levels) of winning submissions in the last two editions of the *WMT Quality Estimation Shared Task* (Kepler et al. [31], Hu et al. [29]).

In parallel with supervised QE's improvements, unsupervised QE saw interesting advancements on its own front, hosted by the revisiting of *glass-box* QE features. Leveraging uncertainty quantification methods, Gal and Ghahramani [18] and Fomicheva et al. [17] showed that features engineered from the mechanisms internal to state-of-the-art Transformers, constitute a rich source of information on translation quality, competitive with supervised QE methods. These newly proposed features are much

different to the *glass-box* features used in the Confidence Estimation era, both because of the process used to extract and engineer them, but also due to the neural structures that originate them, much deeper and denser in information than their Statistical MT ancestors.

This thesis addresses the topic of combining the richness of the aforementioned features, extracted from NMT models, with the current state-of-the-art in supervised QE, described as Glass-Box QE henceforth. We hypothesize that doing so will inform the QE model's training process with valuable information on the state of the NMT model at translation time, making the former more accurate and robust.

1.2 Contributions

The main contributions of this thesis are the following:

- We introduce a new approach to Glass-Box QE, where internal NMT model features are used to enrich the training of QE models. Its goal is to provide information about the NMT model's state upon translation of the input data fed to the QE model, increasing the latter's overall accuracy and robustness. This approach is based on the features presented in Fomicheva et al. [17], shown to be highly relevant for the estimation of sentence-level translation quality scores.
- We apply the proposed method on different QE models using a novel transformer-based Predictor-Estimator architecture [33], which leverages pre-trained Language Models as the Predictor component, and applying various architectural variations. This is implemented as an extension of the OpenKiwi framework [32].
- We perform a comprehensive analysis of the proposed method's effects on the task of predicting word and sentence-level scores, namely on the accuracy and robustness of models across different languages.

Additionally, as a part of this thesis we participated in the *Quality Estimation Shared Task* and published a paper (Moura et al. [43]) on the *2020 Conference on Machine Translation (WMT)*, using the developed models and method on part of our submission, and obtaining leaderboard results.

1.3 Document outline

This dissertation starts by addressing the theoretical foundations that will serve as required background to contextualize our proposed work. Chapter 2 begins with a brief introduction of Neural Networks (Section 2.1). Then, the task of Neural Machine Translation is described in Section 2.2, along with the most relevant neural models and architectural patterns that have historically or presently been used in approaches to the problem (Sections 2.2.1, 2.2.2 and 2.2.3). The last Section (2.3) focuses on the Quality Estimation task itself, and details the roots of the current state-of-the-art approach to QE, the Predictor-Estimator architecture (Section 2.3.2).

Having established the necessary background, Chapter 3 explains the core of our developed approach. Section 3.1 explores the nature of the glass-box features extracted from NMT models, and how this extraction is implemented; Section 3.2 describes the design considerations and architecture of the developed neural model, as well as the way glass-box features are integrated into its training process.

All experiments we conducted, and their corresponding results are revealed in Chapter 4. Details on the dataset and NMT models used are included in Section 4.1, and a description of the metrics chosen to evaluate/compare our model to others is in Section 4.2. The results we obtained for all experiments are presented, and their meaning is discussed in Section 4.3.

Finally, Chapter 5 introduces future considerations for improvements and possible experiments upon the work we have developed, as well as the most important takeways from this dissertation.

Chapter 2

Concepts and Related Work

In this chapter, we present the key theoretical concepts used throughout this work. First, we provide a brief introduction to neural networks. Second, the problem of Machine Translation will be explained, as well the current paradigm and state-of-the-art of the approaches used to solve it. This will help lay the context for the third and final section, which concerns the Quality Estimation problem definition, and the approach we have used as basis for developing this work.

2.1 Neural Networks

Neural networks are defined in this section as a means of mapping a given input x $\in \mathbb{R}^N$ to an output y, which will vary depending on the objective and can take on many forms. The focus will be set on the simplest of neural networks, where information flows in only one direction, and there are no feedback connections - feedforward neural networks -, and on networks where such loops do occur - recurrent neural networks.

2.1.1 Feedforward NN's

Machine learning differs from traditional algorithms in that, instead of having rules defined upfront that will outline how to perform a task, a model that defines a set of possible rules is used, and improved by means of an optimization method by using given sample data (or "training data"). Typically, the optimization strategy is to minimize the error between the models' predictions and what is observed in that data. The simplest example of this is a linear model, where some output y is obtained for a given input $x = (x1, ..., x_N)$ by means of affine transformation, i.e. the dot product between the input vector x and a vector of weights w, added a bias term b. Each weight represents the relevance of its corresponding input element to the final output. This is represented by:

$$y(x, w) = \sum_{i-1}^{N} w_i x_i + b \tag{2.1}$$

This is the basic computation that each individual neuron in a neural network is responsible for. However, the model described above is utterly incapable of solving non-linear problems, a space where

many interesting and complex problems lay. One of the typical examples of this, is the exclusion-or boolean operator (XOR) depicted in Figure 2.1, which no linear model can solve. Intuitively we can see that it is not possible to accurately separate the data points in classes (0 or 1) by drawing a straight line.

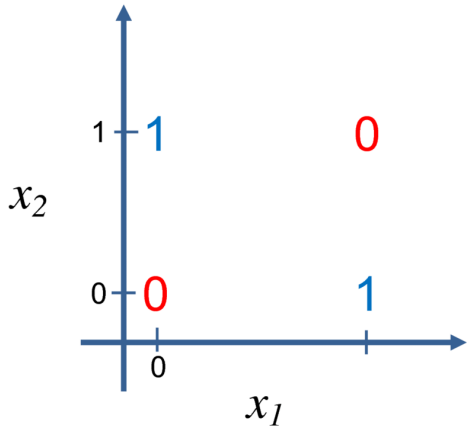

Figure 2.1: The exclusive-or (XOR), a non-linear function that returns 0 when its two binary inputs are both 0 (false) or both 1 (true). No line can be fitted to this data without incorrectly classifying some data points. Figure taken from Spears et al. [60].

The simplest form of neural network that can solve this type of problems - depicted in Figure 2.2 - is the Multi-Layer Perceptron (MLP), a class of Feed-Forward NN's.

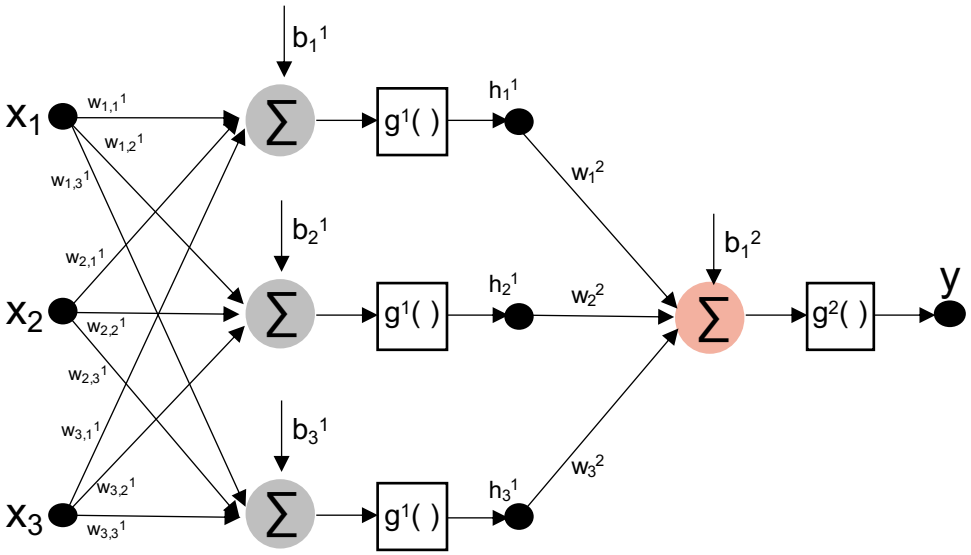

Figure 2.2: A Multi-Layer Perceptron with one single hidden layer, the simplest of Feed-Forward Neural Networks. Grey neurons are a part of the hidden-layer, while in red is the ouput neuron.

It consists of multiple layers of neurons, interconnected in a feed-forward way; this means that connections between nodes do not form a cycle, there are no feedback connections, and information moves in only one direction. The difference between these and the linear model we have just described, is that the former includes one or more intermediate hidden layers between input and output, consisting of the same affine transformation in 2.1.1, but followed by a non-linear function $g(.)$, also called an activation function. The intermediate outputs computed in hidden layers are called hidden states, and typically represented by h. An MLP with M hidden layers can be formally described as:

$$
\begin{aligned}
h_1 &= g(W^1 \cdot x + b^1) \\
h_m &= g(W^m \cdot h_{m-1} + b^m) \\
y &= g(W^y \cdot h_m + b^y)
\end{aligned}
\tag{2.2}
$$

The choice of activation function has a large impact on the capability and performance of a neural network, given that it will transform the output of every neuron within it. Different activation functions may be used in different parts of the model, but typically all hidden layers have the same activation function, and a different one is used for the output layer depending on what type of prediction is required by the model. For example, for a binary classification task - a problem where a neural network would require one output neuron to make a prediction - , the sigmoid function is commonly used, defined as:

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

Given that this function yields a real value between 0 and 1, the output can be taken as a probability, and a threshold of 0.5 can be defined to separate both classes. However, if our problem had more than one class we needed to identify (multi-class classification), we would require a probability distribution over the N classes. The Softmax activation function is used for this purpose, defined as:

$$
\text{softmax}(z)_n = \frac{\exp z_n}{\sum_{j=1}^{N} \exp z_j}, \tag{2.4}
$$

where z is the vector of raw, unnormalized outputs from the neural networks - or logits. This activation function yields a probability for each of the existing classes, and guarantees that the sum of those probabilities is 1. We can then easily take the approach of choosing the predicted class, by evaluating the highest probability attributed to any class.

The choice of activation function to be used in the hidden layers is approached quite differently, given that there is no requirement to normalize the outputs of each layer; in this context, the choice is made based on the function's observed performance on a specific task and network architecture. Many activation functions exist with different characteristics; the Rectified Linear Unit (ReLU) [44] is the most commonly used one, yielding the input value untouched, if that input is larger than 0. Figure 2.3 depicts ReLU's formal definition and graphical plot.

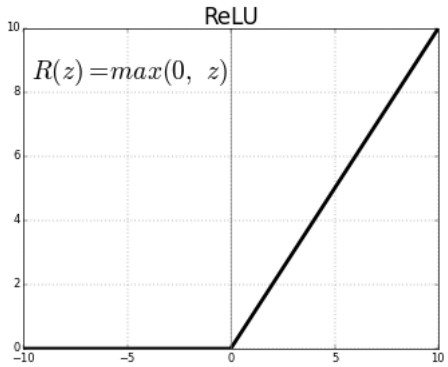

Figure 2.3: The ReLU activation function.

Training

The way to employ neural networks to a given problem is to train them on that problem. Training generally means updating the weights of neurons, described in the previous paragraphs, guided by a loss function optimization objective. The purpose of this loss function is to estimate how well a neural network is modelling the data it is presented with, and is commonly some function of the difference between a true value for an instance of data, and the value estimated by the neural network.

The choice of loss function is of high relevance to the solution itself, as it will be the single measure of error used to update the network weights in the right direction. Different functions have typically been used for different problems, and as we will see, two of them are of greater importance to this thesis, namely:

- the *Mean Squared Error Loss*, used in regression problems. It is defined by $\frac{1}{N}\sum_{n=1}^{N}(Y_n - \hat{Y_n})^2$, where N is the number of samples in the dataset or batch, $\hat{Y_n}$ is the model's prediction for that sample, and Y_n is the true value;

- the *Cross-Entropy Loss*, used in classification problems; it is defined by $\frac{1}{N}\sum_{n=1}^{N}\sum_{c=1}^{C} y_{n,c}\log(p_{n,c})$, where C is the number of existing classes, $p_{n,c}$ is the predicted probability that sample n is of class c, and $y_{n,c}$ is a binary indicator (0 or 1) of whether class c is the correct classification for sample n.

After choosing a loss function, the network weights need to be iteratively updated. The standard method to do so is by making use of the backpropagation algorithm [54]. In short, this algorithm propagates the error obtained upon estimation, from the output to the input layer. The loss function is first derived with respect to each individual weight; the magnitude of each weight's gradient is then used to update it's value, in a way defined by a chosen optimization technique - for which there are various choices, with different advantages.

2.1.2 Recurrent Neural Networks

Recurrent Neural Networks (RNN's) are another type of NN, which hold great interest and applicability for language tasks due to their recurrent nature. RNN's can process inputs of different sizes - such as sentences -, while not changing the size of the model. This type of model is capable of capturing relationships between components of the input - such as words in a sentence. Their recurrency comes from the fact that, at each time step (an RNN processes each of the input's components in sequence), the hidden state computed in the previous timestep is fed back to the network, together with the current input. In this configuration, hidden states and output are defined as:

$$\begin{aligned} \boldsymbol{h}_t &= g^h\left(\boldsymbol{W}^x \boldsymbol{x}_t + \boldsymbol{W}^h \boldsymbol{h}_{t-1} + \boldsymbol{b}^h\right) \\ \boldsymbol{y}_t &= g^y\left(\boldsymbol{W}^y \boldsymbol{h}_t + \boldsymbol{b}^y\right) \end{aligned} \tag{2.5}$$

,where g^h and g^y are non-linear activation functions applied to hidden states or the output respectively. In Natural Language Processing problems, the output space is typically discrete, therefore g^y is usually the softmax function (Equation 2.4).

For context and ease of understanding, Figure 2.4 depicts a simplified schematic of an RNN.

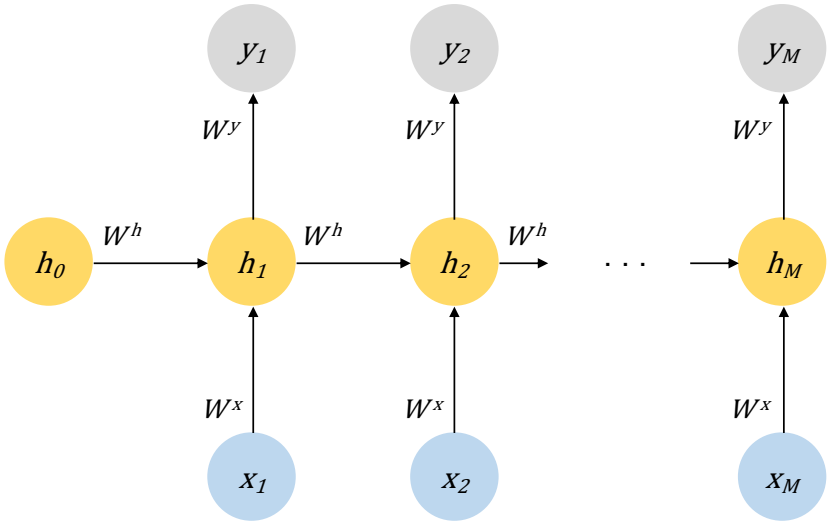

Figure 2.4: Simplified representation of a recurrent neural network.

It is the capacity to pass information from previous inputs, to be incorporated and passed to next states, which gives RNN's the ability to capture long term dependencies in data.

Training

Just like the feed-forward network we previously described, RNN's need to be trained by iteratively updating their weights, based upon the optimization of a specific loss function. The mechanism to do this is similar, although in this case - due to the RNN's recurrent nature - the loss needs to also be

propagated throughout all time steps. This is done with *backpropagation through time* [72].

A side-effect of using this adapted backpropagation is that, as we use the chain rule on the derivative of the loss, effectively multiplying gradients, the gradient gets smaller at each time step; for longer sentences, it can become so small that the network's weights will effectively not change. This phenomenon is called the *vanishing gradient problem* [27].

Various basic RNN units have been devised to overcome this problem, of which the most commonly used is the Long Short-Term Memory (LSTM) unit [28]. In short, this unit introduces a memory cell c_t, which is capable of maintaining the gradient constant across the time steps of backpropagation.

2.2 Neural Machine Translation

Machine Translation is the task of automatically converting source text in one language, to text in another language. This has historically been one of the most challenging problems in AI, due to the fluidity and ambiguity of human language; in this domain, inputs do not have a clean mapping to outputs. Classically, rule-based systems were used to create translations, which were replaced by statistical methods during the 1990's. Recently, deep neural networks have become the ubiquitous way to solve the problem for various reasons, achieving state-of-the-art performance on the task. The field is today addressed as Neural Machine Translation (NMT).

Formally, the goal of NMT is to translate a sequence of words from the source language $\mathbf{X} = x_1, x_2, ..., x_M$, into a sequence of words in a target language, $\mathbf{Y} = y_1, y_2, ..., y_T$, by learning to model the conditional probability:

$$P(\mathbf{Y}|\mathbf{X}) = \prod_{t=1}^{T} P(y_t|y_1, y_2, ..., y_{t-1}, \mathbf{X}) \tag{2.6}$$

2.2.1 Encoder-Decoder Architecture

The most successful approach in tackling the problem of Neural Machine Translation is the encoder-decoder architecture [30, 10, 4, 66]. The concept of this architecture - which is independent of the components/networks used to accomplish it - consists of a two-step process: first, an encoder component outputs a vector, which encodes the full source language sentence; then, a decoder component uses this sentence representation, and outputs the target sentence word by word, conditioned at each step by the source sentence representation, and all the previous words it has created.

Until recently, RNN's with LSTM's as unit cells were the typical networks used for this approach; the encoder RNN would take all the source words as input, and its final hidden state would be taken as the representation of the sentence. The decoder RNN would then use it as its initial hidden state, to start generating the target sentence. Figure 2.5 depicts an RNN-based encoder-decoder architecture, taking

in a source sentence of size M, and outputting a sentence of size P.

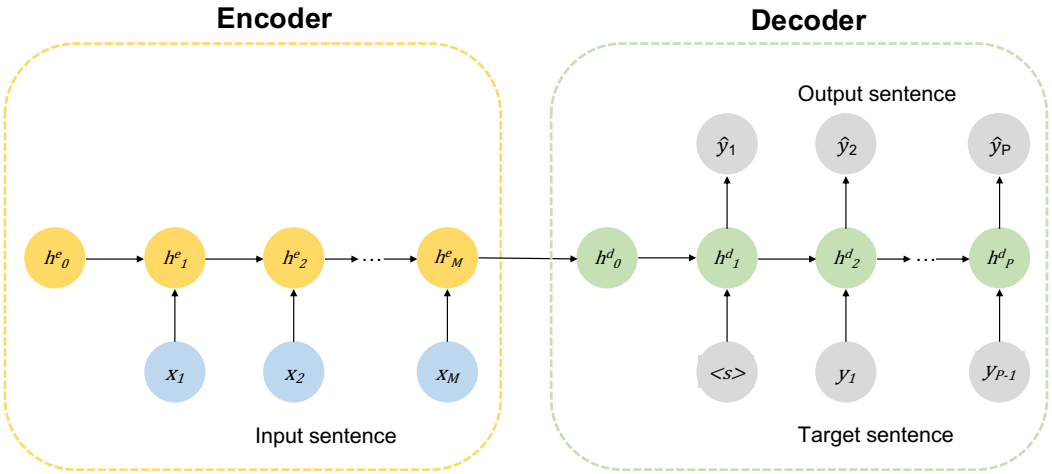

Figure 2.5: Simplified representation of the RNN-based encoder-decoder architecture.

A prominent issue with this architecture is that the final hidden state of the encoder has to hold information about the entire source sentence, no matter how long it might be. For longer sentences, this becomes exceedingly inefficient, and paired with the fact that some words might need to be translated into others that are on opposite ends of source and target sentences, the long range dependency between them might not be learned.

This problem is addressed by the attention mechanism, introduced by Bahdanau et al. [4]. Attention is a soft alignment model, which allows the decoder to partially attend to source words at each decoding step. It both replaces the need for a rich hidden state to contain all the information about the source sentence , and provides a better context for the prediction of each word in the target sentence.

Encoder

As previously described, the encoder's objective is to capture the meaning of the source sentence. Using attention with the approach devised in [4], that objective is extended to creating contextualized embeddings of all words in the source sentence, which will then be weighted by the attention mechanism to be used in decoding. Given that the encoder has access to the entire source sentence, and with the objective of getting context on both sides of each word instead of only on previous words (as explained to this point), the authors use a bidirectional RNN [57]. This entails using one RNN to process words in their normal direction - from left to right in the case of English -, and a second RNN to process words in reverse order. The resulting hidden states created for each word by each network are then concatenated to form a word embedding contextualized on both directions. Specifically, hidden states at position m of the source sentence are described as

$$\begin{aligned} \overrightarrow{\boldsymbol{h}}^{e}_{m} &= \text{RNN}\left(\overrightarrow{\boldsymbol{h}}^{e}_{m-1}, \boldsymbol{E}_{[x_m]}\right) \\ \overleftarrow{\boldsymbol{h}}^{e}_{m} &= \text{RNN}\left(\overleftarrow{\boldsymbol{h}}^{e}_{m+1}, \boldsymbol{E}_{[x_m]}\right), \end{aligned} \tag{2.7}$$

with $\boldsymbol{E}_{[x_m]}$ being the embedding of word x_m of the input sentence. The bidirectional word embedding is described by

$$\boldsymbol{h}_m^e = \left[\overleftarrow{\boldsymbol{h}}_m^e, \overrightarrow{\boldsymbol{h}}_m^e\right]. \tag{2.8}$$

The final hidden state of the encoder (h_m^e for a sentence with M words) is used as initial hidden state for the decoder h_0^d.

Decoder

The decoder's objective is to produce the target sentence, word by word, until a end-of-sentence token is output . With an exclusively RNN-based decoder, that would mean initializing it with the sentence embedding produced by the encoder, and then sequentially generating every word in the target, at each step feeding the model with both the previous hidden state $\boldsymbol{h}_{t-1}^d$, and the embedding vector for the last generated word $\boldsymbol{E}_{[y_{t-1}]}$. When leveraging the attention mechanism, at each time step the decoder will also take into consideration a context vector c_t, so that the hidden state is described by:

$$\boldsymbol{h}_t^d = \text{RNN}\left(\boldsymbol{h}_{t-1}^d, \boldsymbol{c}_t, \boldsymbol{E}_{[y_{t-1}]}\right) \tag{2.9}$$

At each time step, the output logits $\boldsymbol{y}_t$ can be used to obtain a probability distribution over the entire considered vocabulary, using the softmax (Equation 2.4). When dealing with sequential outputs such as text, choosing at each time step the word that was attributed the highest probability (in a *greedy* manner) may not be the optimal solution, and in fact can unexpectedly lead to a poor ending translation. Because of this, the output word is usually chosen according to a decoding strategy; we will not detail these strategies in the context of this thesis, as they are not directly relevant to the task of Quality Estimation.

Attention

As explained in the previous section, the attention mechanism allows the decoder to enrich its prediction at each time step with relevant information from the source sentence, namely by making use of the context vector $\boldsymbol{c}_t$. This vector is a weighted sum of the hidden states computed by the encoder for each source word , defined as:

$$\boldsymbol{c}_t = \sum_{m=1}^{M} \alpha_{tm} \boldsymbol{h}_m^e, \tag{2.10}$$

with M being the number of words in the source sentence, and α the normalized vector of attention scores, defined by:

$$\alpha_{tm} = \frac{\exp(z_{tm})}{\sum_{k=1}^{M} \exp(z_{tk})} \tag{2.11}$$

This is in fact the softmax equation (Equation 2.4), which shows us that the attention scores are nothing more than a probability distribution over all the source words, with each of its values effectively representing the importance of its corresponding source word m to the output decoding step t.

The scores z_{tm} are given by an alignment model, which models how important source words next to position m are for the output decoding at t. The approach developed in [4] uses a feedforward neural network for this purpose, which concatenates the previous decoder hidden state $\boldsymbol{h}_{t-1}^{d}$ with the hidden state for each source word $\boldsymbol{h}_{\boldsymbol{m}}^{\boldsymbol{e}}$, to obtain each value $z_{tj} = f(\boldsymbol{h}_{\boldsymbol{m}}^{\boldsymbol{e}}, \boldsymbol{h}_{t-1}^{d})$. Figure 2.6 depicts how the attention mechanism is incorporated into a standard RNN-based encoder-decoder introduced in the above Section.

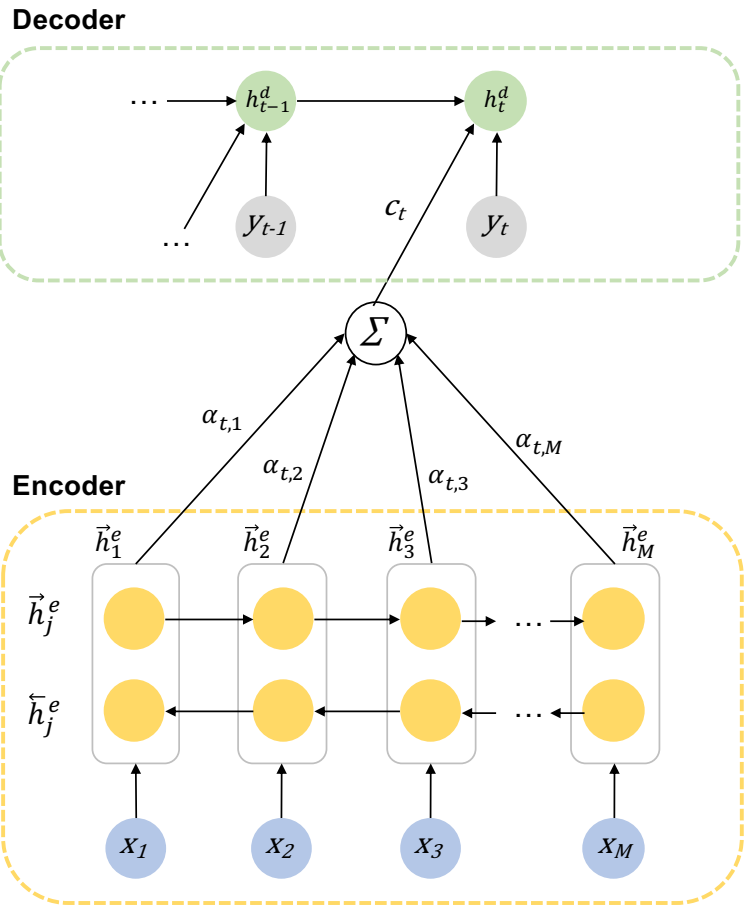

Figure 2.6: Simplified representation of the RNN-based encoder-decoder, based on the attention implementation of Bahdanau et al. [4].

2.2.2 Transformers

The fundamental process that underpins RNN's - sequential computation - is also one of its downfalls. Paralelizing the training process within each training example is not possible when using these networks, which becomes critical with longer sequence lengths, as memory constraints limit batching across examples.

The aptly named publication *Attention Is All You Need* by Vaswani et al. [70] adresses this long standing issue for on sequential data-dependent tasks, by introducing the Transfomer model. This model discards recurrence all together, and uses only attention mechanisms to learn global dependencies between input and output. Since attention weights are computed for all words in a sequence at the same time, the Transformer allows for significantly more paralelization, and since its inception has reached state-of-the-art performance in translation quality.

This model is proposed as an encoder-decoder architecture. Additionally to using the same type of attention we have described in the previous section - between output and input - it also uses self-attention; this mechanism relates different positions within a single sequence, in order to compute a representation of the sequence, and representations of each word, contextualized by the full context of the sequence. Furthermore, self-attention is much less computationally expensive than computing sentence and word representations from sequential hidden states, as is the case with RNN's.

Specifically, the Transformer follows the overall architecture described in Figure 2.7 , using stacked self-attention and point-wise, fully connected feed-forward layers for both the encoder and decoder.

Encoder

The encoder is composed of a stack of $N = 6$ identical layers. Within each a layer, the input goes through a multi-head self-attention mechanism first, followed by a fully-connected feed-forward network. A residual connection [26] is employed around each of the sub-layers (attention/feedforward + add & norm in context of the figure), followed by layer normalization [3], which together make the output of each sub-layer to be LayerNorm$(x +$ Sublayer(x)$)$.

Decoder

The decoder is also composed of $N = 6$ identical layers. Within them, two different attention mechanisms are used: one (Multi-Head attention in Figure 2.7) is similar in nature to the encoder's attention block, albeit in this case attention scores are computed between a word in the output sentence, and words in the input sentence. The other is precisely a self-attention block, slightly changed to allow for focus only on the preceding words of the output sentence: all values in the input to the attention softmax which correspond to positions subsequent to the word being predicted are masked, by being set to $-\infty$ - equivalent to zero after softmax is computed.

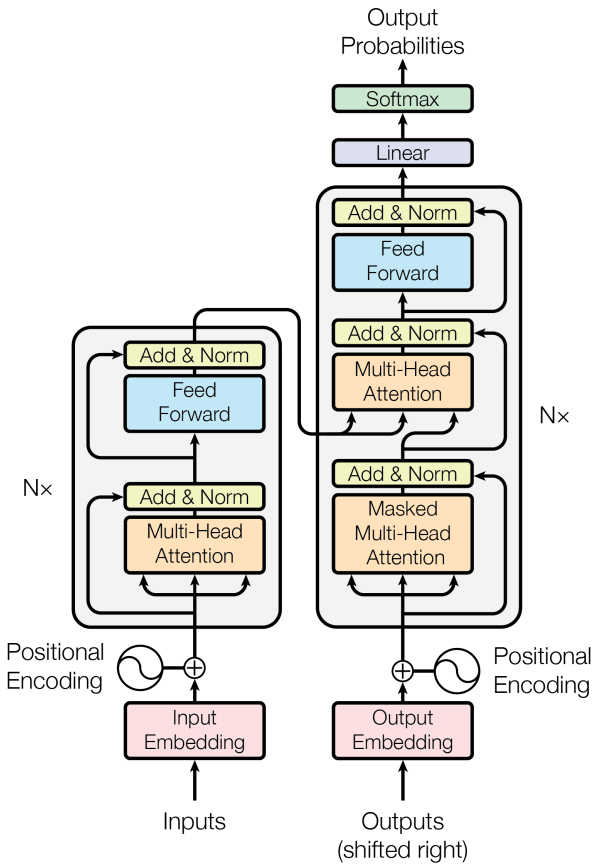

Figure 2.7: The Transformer model architecture. Figure taken from Vaswani et al. [70].

Attention

The attention mechanism used in the Transformer, which the authors call scaled dot-product attention, is presented with some changes in formulation and usage, when compared to its most basic version in the encoder-decoder context (Section 2.2.1). Namely, in [4] it is redescribed as the mapping of a query and a set of key-value pairs to an output; drawing a parallel between what we have describe thus far regarding attention, the query can be seen as the output word, whose embedding we are trying to influence by contextualizing with all source words - the keys. The attention score between each key and the query is nothing more than the dot-product of these two vectors, scaled by the square root of both vectors' dimension d_k. This dot-product calculation in fact plays the role of "compatibility function" in this case - analogous to the alignment model described in section 2.2.1 -, departing from the use of a feed forward NN, in pursuit of much faster and space-efficient computation. This is formally described as:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.12}$$

where $Q \in \mathbb{R}^{d_k}$ is the query, $K \in \mathbb{R}^{d_k}$ is the key, and $V \in \mathbb{R}^{d_v}$ is the value.

Instead of applying a single attention function, the queries, keys and values are projected h times with

different, learned linear projections - W_i^Q, W_i^K and W_i^V - , and each set then processed by a different attention head in parallel. Results from all attention heads are concatenated, and projected back to the initial dimensions that a single head would output, by use of a final learned linear projection W^0. This is described by the equation:

$$\begin{aligned} \text{MultiHead}(Q, K, V) &= \text{Concat}\left(\text{head}_1, ..., \text{head}_h\right), \\ \text{where head}_i &= \text{Attention}(QW_i^Q, KQ_i^K, VW_i^V). \end{aligned} \tag{2.13}$$

Using multiple attention heads allows the model to jointly attend to information from different representation subspaces, by focusing on different positions; averaging with a single attention head would inhibit this capability. Figure 2.8 depicts the composition of a single scaled dot-product attention head, and the configuration when h heads are used.

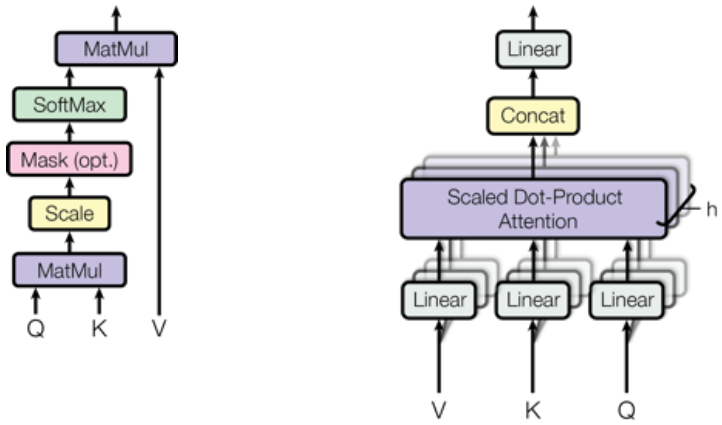

Figure 2.8: (Left) Scaled Dot-Product Attention. (Right) Multi-Head Attention, consisting of several attention layers running in parallel. Figure taken from Vaswani et al. [70].

Even though the Transformer model was first introduced in the context of NMT, it has been used with great success on a variety of different tasks, some leveraging the whole encoder-decoder architecture that it originally modelled, and others rethinking and iterating on components of the Transformer for specific purposes. The next section explains how this has been done with relevant examples.

2.2.3 Pre-trained Contextualized Embeddings

Machine Learning models in the field of NLP have always needed to handle words in some form of numeric representation, therefore creating accurate and contextualized word embeddings has been evolving for longer than the models and architectures described in the previous section exist.

For years, several methods of approaching this problem such as *Word2Vec* [42] and GloVe [48] had proven to be capable of capturing semantic relationships between words (such as telling if words are opposites, or that the relationship between two words like ”Lisbon” and ”Portugal” is the same as

between ”Madrid” and ”Spain”), and also syntactic ones. When training models for use in NLP tasks, it has long been a common practice to take advantage of embeddings previously trained on vast amounts of text data, instead of doing so from scratch - which might be impossible due to lack of data available for the task, or even compute power.

But the above examples of word embeddings made it so that each embedding would always be the same, regardless of the context where its corresponding word was used. ELMo [49] introduced an approach that would come to change this condition. This model was composed of a bidirectional LSTM followed by a feedforward layer, and trained on a simple objective, refered to in the literature as *Language Modelling*: given all previous words in a sentence, predict the next one. Given a sequence of M words (or tokens) $t_1, t_2, ..., t_m$, the probability that the next word being predicted is correct, is formally described as:

$$p(t_1, t_2, ..., t_m) = \prod_{k=1}^{M} p(t_k | t_1, t_2, ..., t_{k-1}). \tag{2.14}$$

This objective allows for training on larger and larger amounts of data, because being self-supervised (uses next word as ground truth, with no need for human labelling), no limitations due to data labelling effort apply.

Training for this objective, and capturing context on both sides of each word in a sentence by using a bidirectional setting, gave ELMo the power to output different embeddings for the same word, depending on what context was used in a sentence.

Although we will not detail all the developments made in this field up until the current state-of-the-art, it is worth pointing out the leap that was made from this era of Language Models, to the Transformer-based ones which have become today’s standard. The increase in performance that Transformers fueled in different NLP tasks warranted great interest by the research field. In an important breakthrough, Devlin et al. [14] took the Encoder component of the Transformer model, and proposed a pre-training method which is still improved and iterated upon in various directions to this day.

Bidirectional Encoder Representations from Transformers - or **BERT** - is a model trained with the overarching purpose of providing a solid Language Understanding basis for easy use and minimal fine-tuning on downstream tasks, which equates to training powerful embeddings. Its architecture is the exact same as the original encoder component of a Transformer, albeit having more Transformer blocks (12 for the `BASE` version of the model, and 24 for the `LARGE` version). There are two steps in BERT’s training: *pre-training* and *fine-tuning*. This is depicted in Figure 2.9, where different fine-tuning objectives are used, like Textual Entailment (MNLI), Named Entity Recognition (NER) and Question Answering (SQuAD):

BERT is trained on two distinct tasks in the *pre-training* stage. First is a modification of the objective described in 2.2.3, called *Masked Language Modelling* (MLM) - originally the *Cloze* task [67]. Instead

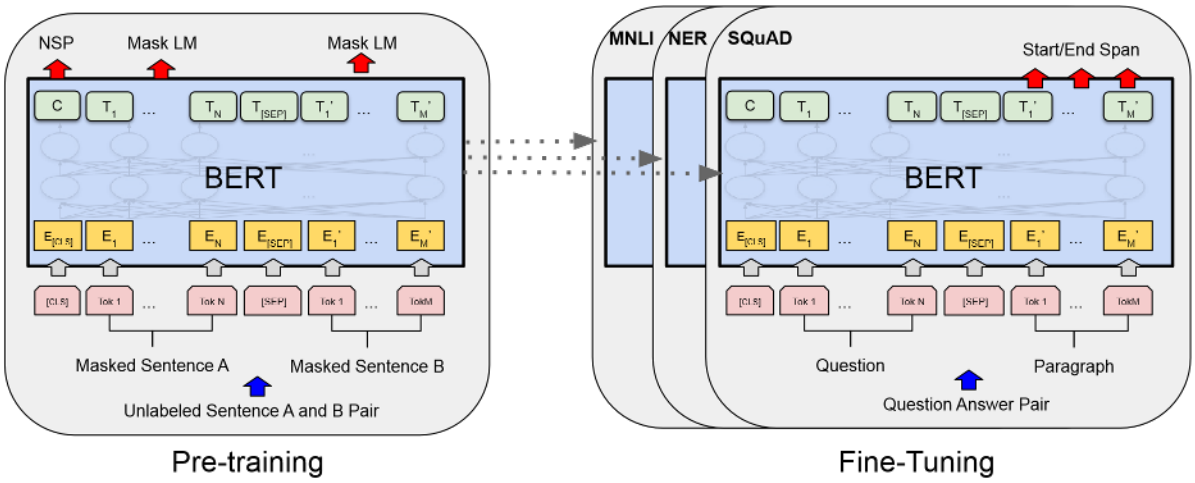

Figure 2.9: High level pre-training and finetuning procedures for BERT. The same pre-trained model parameters are used to initialize models for different down-stream tasks. During fine-tuning, all parameters are updated (classification heads + BERT). `[CLS]` is a special symbol added at the beginning of every input example, and `[SEP]` is a special separator token (e.g. separating two consecutive segments of text). Figure taken from Devlin et al. [14].

of next word prediction, it focuses on predicting masked words; 15% of all input token positions are replaced with a special token `[MASK]`, and the final hidden vectors corresponding to those tokens are fed into an output softmax over the vocabulary, effectively yielding a prediction of the missing word. This special token would not appear in any downstream task however, so as to mitigate the creation of a mismatch between both pre-training and fine-tuning, the 15% of total token positions chosen at random for prediction are replaced with `[MASK]` 80% of the time, by a random token 10% of the time, and are left unchanged the remaining 10% of the time. This task allows for capturing context on both sides of the predicted words, circumventing the problem shared by standard conditional language models: they could only train left-to-right *or* right-to-left, given that a bidirectional conditioning would allow each word to indirectly "see itself", enabling the model to trivially predict the target word.

The other task in pre-training is Next Sentence Prediction (NSP). Many important downstream tasks such as Question Answering and Natural Language Inference are based on understanding the *relationship* between two sentences, something which is not directly captured by language modeling. The authors demonstrate that NSP trains BERT to understand sentence relationships, while still being able to take advantage of self-supervision. They do this by passing in two sentences `A` and `B` as input to BERT, separated by a special [SEP] token; 50% of the time, sentence `B` is the actual next sentence that follows `A`, and is therefore labeled as `IsNext`. The other 50%, it is a random sentence from the corpus (labeled as `NotNext`). The first embedding in any input to BERT, the special classifier [CLS] token, is the one taken as the feature to train a binary classifier for NSP; because of its training objective, this embedding is not a meaningful sentence representation without fine-tuning, and that's what the authors show can be done relatively inexpensively in terms of computational resources, yielding state-of-the-art results on sentence-level tasks. Figure 2.10 depicts both tasks in BERT's pre-training.

Fine-tuning is straightforward with BERT, given that the self-attention mechanism in the Transformer encoder allows for the modeling of many downstream tasks, by simply swapping out the appropriate

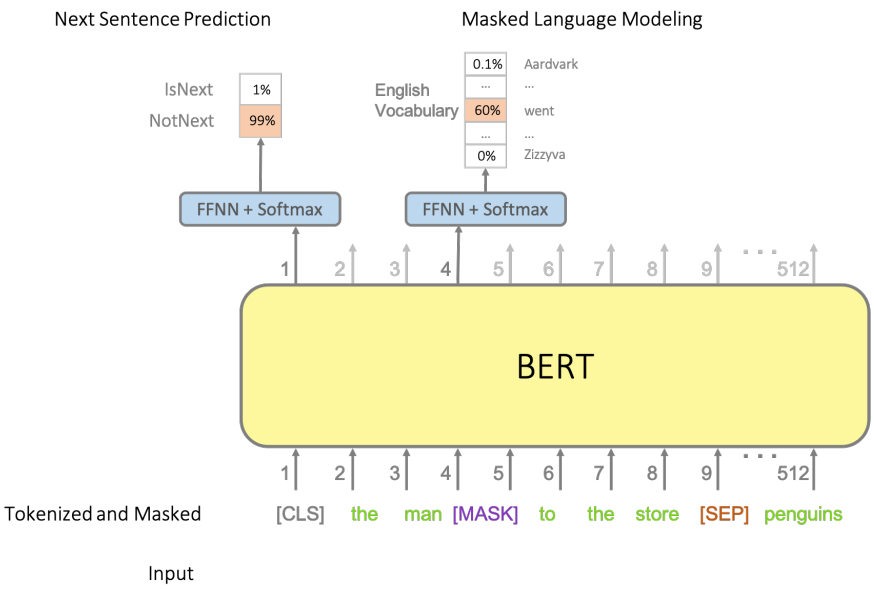

Figure 2.10: Next Sentence Prediction (NSP) and Masked Language Modeling (MLM) tasks in BERT's pre-training. Embeddings output by BERT can be used in a straightforward way for classification.

inputs and outputs. For tasks depending on text pairs the benefit is even greater, given that self-attention will effectively include *bidirectional* cross attention between the two sentences. In general, fine-tuning involves feeding the pre-trained token or [CLS] embeddings to an output layer, and then training the whole model end-to-end, including BERT's parameters.

A key in BERT's whole training process - and a dominant factor in its capabilities - is the amount of data with which it is trained. The BooksCorpus (800M words) and English Wikipedia (2,500M words) dataset are leveraged; this data volume, paired with MLM and NSP, creates a model with powerful word and sentence embeddings, which has been the bedrock of many tasks in NLP research for the past years.

2.3 Quality Estimation

As previously explained in Section 1.1, Quality Estimation is the task of predicting the quality of a system's output for a given input, without any information or reference about the expected output. In this section, we will clarify and formally describe this task regarding its different granularity levels (Section 2.3.1), and also describe the originally proposed Predictor-Estimator architecture (Section 2.3.2), which still serves as fundamental framework to modern QE approaches.

2.3.1 Sentence and Word-Level Task

The two QE sub-tasks we will be exploring are those of predicting quality labels/scores for words and/or sentences. Different techniques are applied at a document-level, which were not included in the scope of this thesis.

As detailed in Section 1.1, both tasks have evolved a great deal and been approached quite differently throughout the years; this Section formalizes them, and lays the foundational concepts for the work we have developed.

Word-Level

Word-level QE focuses on predicting quality labels - *BAD* or *OK* - for all tokens in a translated sentence. If we consider that no internal information was leveraged about the NMT system that generated the translation - a black-box approach -, the task can be described as learning to predict the right label (or class) $\boldsymbol{c}$, given a sequence of words in a source language $\boldsymbol{X} = x_1, x_2, ..., x_M$, and its machine translation in a target language $\boldsymbol{Y} = y_1, y_2, ..., y_T$, or $p(\boldsymbol{c}|\boldsymbol{X}, \boldsymbol{Y})$.

As you might recall, what we have just described is in fact a binary classification task; the loss used for this type of problem, which will guide model training, is the *Cross-Entropy Loss* (Section 2.1.1), with $C = 2$ classes. In this case, the loss for each sample becomes the log-probability predicted by the model for the correct class $\log(p(c))$.

From previous sections, we have seen that a typical and effective pattern for text classification is to create powerful embeddings of that text, with the intention of gathering all the important information for the task within it, and then simply using a classifier ”component” to predict a label. State-of-the-art word-level approaches fall into this same pattern, focusing most efforts on creating better embeddings, as we will see in Section 3.2.

Sentence-Level

Sentence-level QE, on the other hand, is perfectly described as a regression task: meaning, the objective is to predict a real value, that quantifies the quality of a sentence's machine translation. Making the same consideration of a black-box approach as we did for word-level QE, this means predicting a value $\hat{\boldsymbol{y}} = f\left(\boldsymbol{X}, \boldsymbol{Y}\right)$. The function f will be implemented by our system.

A few different measurements have been used in the literature as a true label for quality. The most common one historically is the Human-Targeted Translation Error Rate (HTER) [58]; this indicator is defined as:

$$\text{HTER} = \frac{\text{Insertions} + \text{Deletions} + \text{Substitutions} + \text{Shifts}}{\text{Number of words in reference}} \tag{2.15}$$

Each inserted/deleted/modified word or punctuation mark counts as one error, and shifting a string of any number of words, by any distance, also counts as one error. The reference translation, in this case, is the post-edit created by a human translator. A post-edit is nothing more than a ”fix”, or correction, to a machine translated piece of text.

Another commonly used quality indicator is the Direct Assessment score. This is a score directly obtained from a professional translator's assessment of a particular translated sentence, and is normally defined as a score of 0-100, 100 being a perfect translation.

A disclaimer must be raised when discussing these indicators; stemming from subjective editing and/or linguistic patterns of specific translators, both HTER and DA scores are at best strongly correlated with human judgement - increasing with the size and diversity of the sample group of translators which participated in the creation of a QE dataset. Given the inherent complexity and ambiguity of human language, where it is common that two grammatically different sentences hold the same meaning and validity as a translation, there is frequently no single true reference, or well defined theoretical quality score for a sentence.

2.3.2 Predictor-Estimator Architecture

The Predictor-Estimator, originally proposed by Kim et al. [33], is an RNN-based architecture very inspired upon the Encoder-Decoder (Section 2.5), which effectively standardized the end-to-end neural model approach still used in modern QE, and was also an early motivator of the contextualized learning of embeddings made popular with models like BERT (Section 2.2.3).

The concept of this architecture is a two-step training process, each focused on a component of the whole system: first, the Predictor is pre-trained using parallel data, i.e source sentences and reference translations. This component is nothing more than a word prediction model; very similar to the Encoder-Decoder, the big difference to it is that target context is used on both sides of the word that is being predicted, instead of just the preceding context. In the initial pre-task of training the Predictor, for each sample a random target word is replaced with X, and the model then tries to predict the original target word by conditioning on the whole source and target context. The authors assumed such a task to enable the word prediction model to transfer useful knowledge for QE (later found to be true with transformer-based "word-predictors" like BERT, and holding up for many other language tasks), which is passed forward in the form of *Quality Estimation Feature Vectors* (QEFV's). The second step is to train the Estimator, this time with QE data (source sentences, machine translations and quality annotations). This component takes the Predictor's output, and is responsible for estimating the quality of the word/sentence in question.

Figure 2.11 depicts the schematic of a Predictor-Estimator architecture.

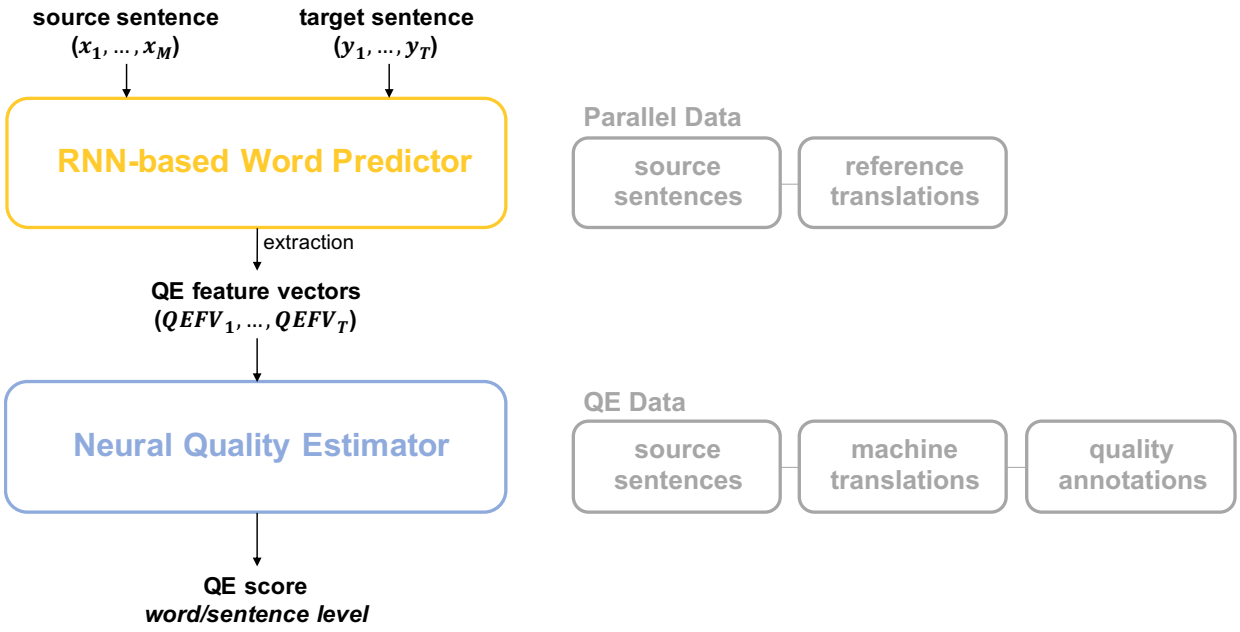

Figure 2.11: Simplified schematic of the Predictor-Estimator architecture; the Predictor is pre-trained on parallel data (generally more available), and then the whole system - Predictor + Estimator - is trained on QE data.

Predictor

Formally, given source and target sentences $\mathbf{X} = x_1, x_2, ..., x_M$ and $\mathbf{Y} = y_1, y_2, ..., y_T$, the RNN-based word predictor defines the probability of predicting target word y conditioned on the source context $\boldsymbol{x}$ and target context $\boldsymbol{y_{-j}}$ (target sentence without the j_{th} target word):

$$\begin{aligned} & p(y_t|y_1, ..., y_{t-1}, y_{t+1}, ..., y_T, \boldsymbol{x}) \\ & = g([\overrightarrow{h}_{t-1}; \overleftarrow{h}_{t+1}], [y_{t-1}; y_{t+1}], c_t) \\ & = \frac{\exp(y^\top W_{o_1} W_{o_2} t_t)}{\sum_{k=1}^{K_y} \exp(v_k^\top W_{o_1} W_{o_2} t_t)}, \end{aligned} \tag{2.16}$$

$$\begin{aligned} & t_t = [\max\{\tilde{t}_{t,2k-1}, \tilde{t}_{t,2k}\}]_{k=1,...,l}^\top, \\ & \tilde{t}_t = S_o[\overrightarrow{h}_{t-1}; \overleftarrow{h}_{t+1}] + V_o[E_y y_{t-1}; E_y y_{t+1}] + C_o c_t, \end{aligned} \tag{2.17}$$

where $\overrightarrow{h}_{t-1}$ and $\overleftarrow{h}_{t+1}$ are the hidden states of the forward and backward RNN's on the target sentence, c_t is the source context vector resulting from the attention mechanism when predicting the current target word as described in Section 2.2.1, K_y is the vocabulary size of the target language, v_k is the k_{th} word in that vocabulary, l is the dimension of the maxout units, and E_y is the target word embedding matrix. W_{o_1}, W_{o_2}, V_o, S_o and C_o are learned weight matrices.

After training the Predictor, Feature Vectors are extracted from the model components that affect each target word prediction, which will then be fed into the Estimator for Quality Estimation. The authors propose two types of QEFVs: pre-prediction (*Pre-QEFVs*), and post-prediction (*Post-QEFVs*).

Pre-QEFVs are the summary representation for the "pre-prediction" computational graph, which indicates the set of "pre"-computed nodes "before" making the prediction of the target word. For the t_{th} target word, this can be defined as:

$$\text{Pre-QEFV}_t = y_t W_{o_1} \odot W_{o_2} t_t, \tag{2.18}$$

where $\odot$ is the element-wise multiplication operator. This is based on word prediction using feature weights $y_t W_{o_1}$, and the t_{th} position feature vector $W_{o_2} t_t$, related to $[\overrightarrow{h}_{t-1}; \overleftarrow{h}_{t+1}]$ and $[y_{j-1}; y_{j+1}$ for surrounding target words.

Post-QEFVs are the summary representations for the "post-prediction" computation graph, defined as the set of "post"-computed nodes "after" making the prediction of the target word. For the t_{th} target word, this can be defined as:

$$\text{Post-QEFV}_t = [\overrightarrow{h}_t; \overleftarrow{h}_t], \tag{2.19}$$

which is nothing more than the direct hidden states for target word y_t.

Estimator

Finally, the Estimator takes QEFVs at all target word positions as input vectors and estimates translation qualities at word or sentence levels. Since QE data and the parallel data used to train the Predictor are not fully compatible given the qualities of their target sentences (with parallel data, the reference translation is always correct), this stage of training is considered as a form of adaptation of QEFVs to the QE task, and the Predictor can continue to be trained in conjunction with the Estimator.

For **sentence-level** QE, all QEFVs are first transformed into a single summary vector **s** and a logistic regression is then applied to it, defined as:

$$\begin{aligned} &\text{QE}_{\text{sentence}}(\boldsymbol{y}, \boldsymbol{x}) \\ &= \text{QE}_{\text{sentence}}(\text{QEFV}_1, ..., \text{QEFV}_t) \\ &= \sigma(W_s \boldsymbol{s}), \end{aligned} \tag{2.20}$$

where W_s is the learned weight matrix used for the affine transformation applied to the summary vector.

In regard to the summary vector, the authors experiment with three different approaches to create it: using a feed-forward NN (taking the average of the resulting hidden representations for each QEFV), an RNN (processing the sequence of QEFVs, and using the last hidden representation $\overrightarrow{h}_T$), and a Bi-RNN (using the concatenated last two hidden representations $[\overrightarrow{h}_T; \overleftarrow{h}_1]$.

For **word-level QE**, QEFVs are also processed in all three ways, but are used directly as summary vector of each corresponding target position, instead of composed as previously. In this model, the binary classification function is applied at the t_{th} target position, described as:

$$\begin{aligned} &\text{QE}_{\text{word}t}(\boldsymbol{y}, \boldsymbol{x}) \\ &= \text{QE}_{\text{word}_t}(\text{QEFV}_1, ..., \text{QEFV}_T) \\ &= \begin{cases} \text{OK}, & \text{if } \sigma\left(W_w h_t\right) \geq \text{ threshold} \\ \text{BAD}, & \text{if } \sigma\left(W_w h_t\right) < \text{threshold}, \end{cases} \end{aligned} \tag{2.21}$$

where W_w is the weight matrix of an affine transformation used to compute output nodes.

As we will see in the succeeding sections, the model developed in this thesis follows the architectural layout of the original Predictor-Estimator, although not RNN-based.

Chapter 3

Glass-Box Quality Estimation

This section details the approach taken in implementing the extraction of glass-box features, and their use in the QE models developed throughout the duration of this thesis. Open source tools were heavily leveraged for the purpose, and adapted in ways described in the following sections. First, in section 3.1, we describe how glass-box features are extracted according to the supporting methodology outlined in Fomicheva et al. [17], as by-products from the translation process of pre-trained NMT models provided in the context of the *WMT20 Quality Estimation Shared Task*. The `Fairseq`[1] sequence modeling toolkit is used for this task, being the same tool used to train the available MT models. Then, in section 3.2, the model architecture is explained, along with the way uncertainty measures are integrated into its training; the `OpenKiwi`[2] Quality Estimation framework, a tool which implements the best performing QE systems from previous editions of the *WMT Quality Estimation Shared Task*, is relied upon for this task. Both tools are based on the `PyTorch`[3] framework, which uses Python[4] as its main programming language.

3.1 Glass-Box QE Features

The work that serves as basis for this section (Fomicheva et al. [17]) focuses on approaching the Quality Estimation task as an unsupervised problem. State-of-the-art QE models typically treat the MT system creating the translations under evaluation as a black-box; they also require large amounts of parallel data for pretraining (namely the Predictor) and in-domain translations annotated with quality labels for training. This is a problem in general, and more so when dealing with low-resource languages, when much less data is generally available for research purposes.

The authors of [17] posit that encoder-decoder NMT models offer a rich source of information for directly estimating translation quality, with the most promising source of this richness being: the output probability distribution from the NMT system (i.e. the probabilities obtained by applying the softmax function over the entire vocabulary of the target language). The assumption supporting this hypothesis is that, the more confident the decoder is, the higher the quality of the translation.

However, while sequence-level probabilities of top MT hypothesis have been used for confidence estimation (as alluded to in section 1.1), the output probabilities from Deep Neural Networks should

[1] www.github.com/pytorch/fairseq
[2] www.github.com/Unbabel/OpenKiwi
[3] www.pytorch.org
[4] www.python.org

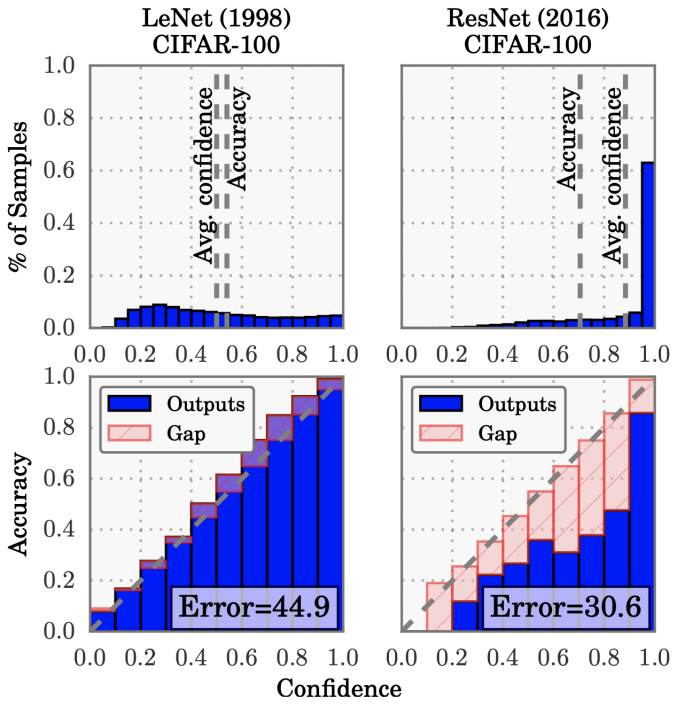

Figure 3.1: Confidence histograms (top) and reliability diagrams (bottom) for a 5-layer LeNet (left) and a 110-layer ResNet (right) on CIFAR-100. Figure taken from Guo et al. [23].

not be directly taken as reliable quality indicators, as they have been shown to generally not be well calibrated (Guo et al. [23]), i.e, the probability that they assign to a prediction does not correspond to the true likelihood of the prediction. Figure 3.1 illustrates this problem by comparing a shallow network's (LeNet) average confidence and accuracy, with that of a much deeper one (ResNet). This phenomena is not bound to computer vision tasks, as it has been observed in NLP, and MT in particular [71], where softmax output probabilities are found to often be *overconfident*, assigning large probability mass to predictions that are far away from the training data. To overcome this issue, and enable the exploit of output distributions beyond the top-1 prediction, the authors use uncertainty quantification methods and metrics derived from them, namely Monte Carlo Dropout (Gal and Ghahramani [18]). This method consists of applying dropout[5] at test time before every layer in the network, performing several forward passes for the same inputs (each affected differently by the applied dropout), and collecting posterior probabilities generated by the model. The mean and variance of the resulting distribution is then used to represent model uncertainty.

3.1.1 Feature Description

In the context of the encoder-decoder networks used for NMT, the encoder maps an input sequence of I words (or tokens) $\mathbf{x} = x_1, ..., x_I$ into a sequence of hidden states, which are summarized into a single vector using an attention mechanism. This is then taken by the decoder, which generates the output sequence $\mathbf{y} = y_1, ..., y_T$ of length T. The probability of generating $\mathbf{y}$ is therefore represented by:

[5]Dropout randomly masks neurons to zero based on a Bernoulli distribution, reducing overfitting during training [65].

$$p(\mathbf{y} \mid \mathbf{x}, \theta) = \prod_{t=1}^{T} p\left(y_t \mid \mathbf{y}_{<t}, \mathbf{x}, \theta\right) \tag{3.1}$$

The decoder produces the probability distribution $p\left(y_t \mid \mathbf{y}_{<t}, \mathbf{x}, \theta\right)$ over the system vocabulary at each time step using the *softmax function*, and the model is trained to minimize cross-entropy loss (Section 2.1.1.

Seven features are extracted and made use of for downstream QE training, which can be divided into two categories: 1) deterministic features, using the output probability distribution created upon inference not affected by dropout, and 2) uncertainty-based features, computed across different inference runs affected by MC dropout.

Softmax Distribution-Based

The first QE measure is simply the sequence-level translation probability (§3.1) in log-space, normalized by length:

$$\text{TP} = \frac{1}{T} \sum_{t=1}^{T} \log\ p\left(y_t \mid \mathbf{y}_{<t}, \mathbf{x}, \theta\right) \tag{3.2}$$

However, due to the previously mentioned over-estimation issue that top-1 predictions suffer from, TP alone would not serve as a reliable indicator. Two other metrics are used to go beyond this limitation.

The first is the entropy of the softmax output distribution over target vocabulary of size V at each decoding step, averaged across the sentence length:

$$\text{Softmax-Ent} = \frac{1}{T} \sum_{t=1}^{T} H(y_t) = -\frac{1}{T} \sum_{t=1}^{T} \sum_{v=1}^{V} p\left(y_t^v\right) \log\ p\left(y_t^v\right) \tag{3.3}$$

In this expression, the $H(y_t)$ component is called Shannon's entropy, and it is the typical measure of uncertainty used in information theory. Entropy is at its maximum value when all words in the vocabulary are assigned the same probability value, essentially representing a pick from a uniform distribution – which means the quality of the translation is likely to be low – , and at its lowest when most of the probability mass is concentrated on a few vocabulary words, which indicates that the generated target word is likely to be correct.

The second is the variance of word-level log-probabilities for each predicted sentence, expressed by:

$$\text{Sent-Var} = \sqrt{\mathbb{E}\left[P^2\right] - \left(\mathbb{E}[P]\right)^2}, \tag{3.4}$$

where $P = p(y_1), ..., p(y_T)$ represents the word log-probabilities. The rationale behind this measure is that a great disparity between probabilities assigned to individual words within the same sentence indicates very different behaviour of the NMT system and, consequently, different output quality; this information would be lost if only using the average of probabilities.

Uncertainty Quantification-Based

As explained, in this case the model's softmax probability distribution is captured for each of N stochastic forward passes through the MT model, with parameters $\hat{\theta}$ perturbed by dropout. The first two measures are the expected value and variance for the set of sentence-level probability estimates across different MC Dropout runs:

$$\text{D-TP} = \frac{1}{N} \sum_{n=1}^{N} \text{TP}_{\hat{\theta}^n} \tag{3.5}$$

$$\text{D-Var} = \mathbb{E}\left[\text{TP}_{\hat{\theta}}^2\right] - \left(\mathbb{E}[\text{TP}_{\hat{\theta}}]\right)^2 \tag{3.6}$$

where TP is sentence-level probability as defined in §3.2. A combination of the two is also used:

$$\text{D-Combo} = 1 - \frac{\text{D-TP}}{\text{D-Var}} \tag{3.7}$$

Finally, the lexical variation between the outputs generated in different runs for the same source segment is quantified. Great differences between translations might indicate model high uncertainty or a very ambiguous or complex source sentence. This is achieved by computing an average similarity score (*sim*) between the set of translation hypotheses:

$$\text{D-Lex-Sim} = \frac{1}{C} \sum_{i=1}^{\mathbb{H}} \sum_{j=1}^{\mathbb{H}} sim(h_i, h_j) \tag{3.8}$$

where $h_i, h_j \in \mathbb{H}$, $i \neq j$ and $C = 2^{-1}|\mathbb{H}|(|\mathbb{H}| - 1)$ is the number of pairwise comparisons for $|\mathbb{H}|$ hypotheses. Meteor (Banerjee and Lavie [5]) is used to compute each similarity score.

3.1.2 Implementation of Feature Extraction

The glass-box feature extraction process about to be detailed is central to this work. I decided to separate both processes - feature extraction and QE model training -, first creating separate input artifacts containing the features, and then feeding them into the QE models' training process. This decision was made for three reasons:

1. reducing development complexity, and decoupling the experimentation done within the two different processes and tools;
2. the reduced size of 7000 sentences shared by the main QE datasets used for development meant small resulting input feature artifacts, and did not justify an end-to-end ”extraction and QE training” process;
3. the computational burden of loading both MT and QE model into memory, and performing 20 to 30 forward passes with each batch of sentences before passing them to the QE model for training, was absolutely prohibitive given the available resources.

The inference step within `Fairseq`'s sequence generator component was looped over $n = 30$ times, with dropout affecting the MT model differently in each iteration. Output probability distributions are captured on a word prediction level in each run, processed on a sentence basis - as needed for deterministic features -, and again processed across different iterations - as needed for uncertainty-based features.

In order to achieve scale-independence and speed up training and convergence - a common practice in ML model training - the features are then normalized, according to their mean and standard deviation in the training set. The features corresponding to the validation set are also transformed in such a way, but are not included in the calculation of mean and standard deviation, given that the QE model will only learn and have its weights updated from the distribution of features in the training set, but still needs to use validation set features in a common reference frame upon inference. The full process returns a *.tsv* file for each dataset partition, having 7 columns (each column being one type of glass-box feature), and as many rows as the size of the corresponding partition.

3.2 QE Model Implementation

Implementation details for the developed QE models are outlined in this section. As mentioned before, they come as an extension of the open-source QE framework `OpenKiwi`, specifically designed for such modular experimentation.

The previous edition of the *WMT QE Shared Task* saw great success in leveraging transfer learning techniques with pretrained Language Models to train QE engines. The best submitted systems hinted towards this promising research direction, and therefore this approach was adopted as basis for all developed models. The LM most heavily used across experiments was XLM-Roberta [12] (described in the following section, named XLM-R henceforth), an improvement on XLM [37], which was part of the *2019 Shared Task* winning individual model submission. XLM-R has recently been reported to display state-of-the-art performance on cross-lingual tasks, especially taking advantage of relationships between high and low resource languages to better perform with the latter; this was assessed and verified in preliminary experiments.

In the following two sections, both the system used as baseline, and the improved system adapted

to use glass-box features are thoroughly described.

3.2.1 XLM-Roberta

XLM-R is another example of leveraging the Transformer architecture (2.2.2) to pre-train rich and contextualized embeddings. Like BERT, the encoder module of a Transformer is used as its structure, with a few important differences in the scale and training procedure.

The focus of this model's development was to train a multilingual Language Model on a wide array of idioms, in order to improve Cross-Lingual Language Understanding (XLU); meaning, the learning of cross-lingual representations and their transfer to discriminative tasks. An important factor, which separates XLM-R from other models trained on a large number of languages, is the scale at which it is trained; 100 languages are learned, and the filtered CommonCrawl dataset composed (2.5TB) is orders of magnitude larger than the one used to train other models along the same line of research (such as mBERT and XLM), especially increasing the amount of data for low-resource languages. This follows the rationale of previous studies, which have shown both that: 1) training on larger amounts of data unlocks the potential for higher performance even when training existing model architectures (Liu et al. [39]), and 2) more low-resource language data is effective for learning high quality word embeddings in multiple languages (Grave et al. [21]). Fig 3.2 shows the comparison between CommonCrawl-100 and the dataset used to train previous models, Wikipedia-100.

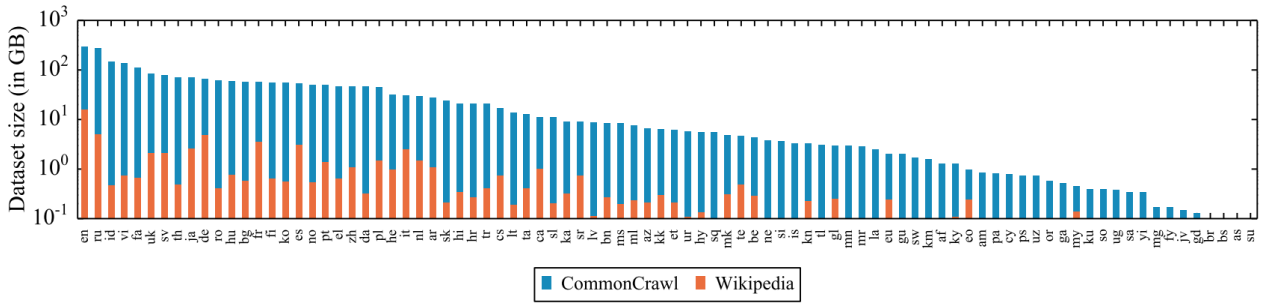

Figure 3.2: Amount of data in GiB (log-scale) for the 88 languages that appear in both the Wiki-100 corpus used for mBERT and XLM-100, and the CC-100 used for XLM-R. CC-100 increases the amount of data by several orders of magnitude, in particular for low-resource languages. Figure taken from Conneau et al. [12].

Furthermore, drawing inspiration from the RoBERTa model, the authors found that such earlier multilingual models are undertuned, and that much better performance can be obtained by simple improvements in the learning procedure of unsupervised MLM:

- the Next Sentence Prediction (NSP) objective used in the original BERT implementation is dropped, as it is found to hinder performance on down-stream tasks, and the sampling of input sequences is also changed: instead of passing in a pair of segments, which may each contain multiple natural sentences (the pair is used for the NSP objective), full sentences are sampled contiguously from one or more documents of the same language, such that the total length is at most 512 tokens (the same maximum length as BERT);

- unlike its predecessor XLM, no language embedding is passed to the model (i.e. an extra input that specifies what language is the input in), which allows XLM-R to better deal with code-switching;
- originally, the masking of tokens to be predicted was done once during data processing, and kept for the whole training procedure, being therefore a static mask. This is replaced by dynamic masking, which means that a new mask is generated every time a sequence is fed to the model. This is found to perform slightly better than static masking, with the added benefit of being more computationally efficient;
- the batch size is increased dramatically and learning rate is increased accordingly, which has been shown to improve optimization speed and end-task performance in LM pre-training;
- sub-word tokenization methods used in previous multilingual models assume that input text uses spaces to separate words - which is not the case in many languages -, relying on different language-specific tokenization tools, and making them more difficult to use on raw text; instead, a Sentence Piece model is trained and used with XML-R. This unsupervised text tokenizer is trained on raw sentences, where whitespaces are treated as any other unicode symbol, and uses either BPE or unigram algorithm to construct the appropriate vocabulary. This allows for language agnostic tokenization, in a easy end-to-end process, and also for very easy detokenization.

The authors show how the degradation in performance caused by the increase in number of learned languages - the curse of multilinguality, a previously studied phenomenon - can be counteracted by increasing model and vocabulary size. Implementing these changes, along with the previously described ones, yielded a model that performs better on low-resource languages than any other - proving that positive language transfer from high to low-resource is achieved -, while maintaining equal or even greater performance on a single language basis, than monolingual models trained solely on that specific language.

For these reasons, and in the interest of developing one simplified architecture able to deal with all 7 languages present in the main dataset used in this thesis, XLM-R was chosen as the basis for all experiments.

3.2.2 Base Kiwi System

The architecture we will explore and build upon follows the overall pattern introduced originally in the Predictor-Estimator model, described in Section 2.3.2. This is comprised of a ”Feature Extractor” module, with a ”Quality Estimator” stacked on it - or rather, multiple similar estimators, each one used for its objective: sentence score, target tag or source tag prediction. The system can be trained using either one of the estimators independently, or all at the same time, in a multi-task learning setup. Figure 3.3 depicts the general architecture, which will be refered to as *Kiwi-Base* henceforth.

The Feature Extractor module consists of a pretrained XLM-R model, with some further process-

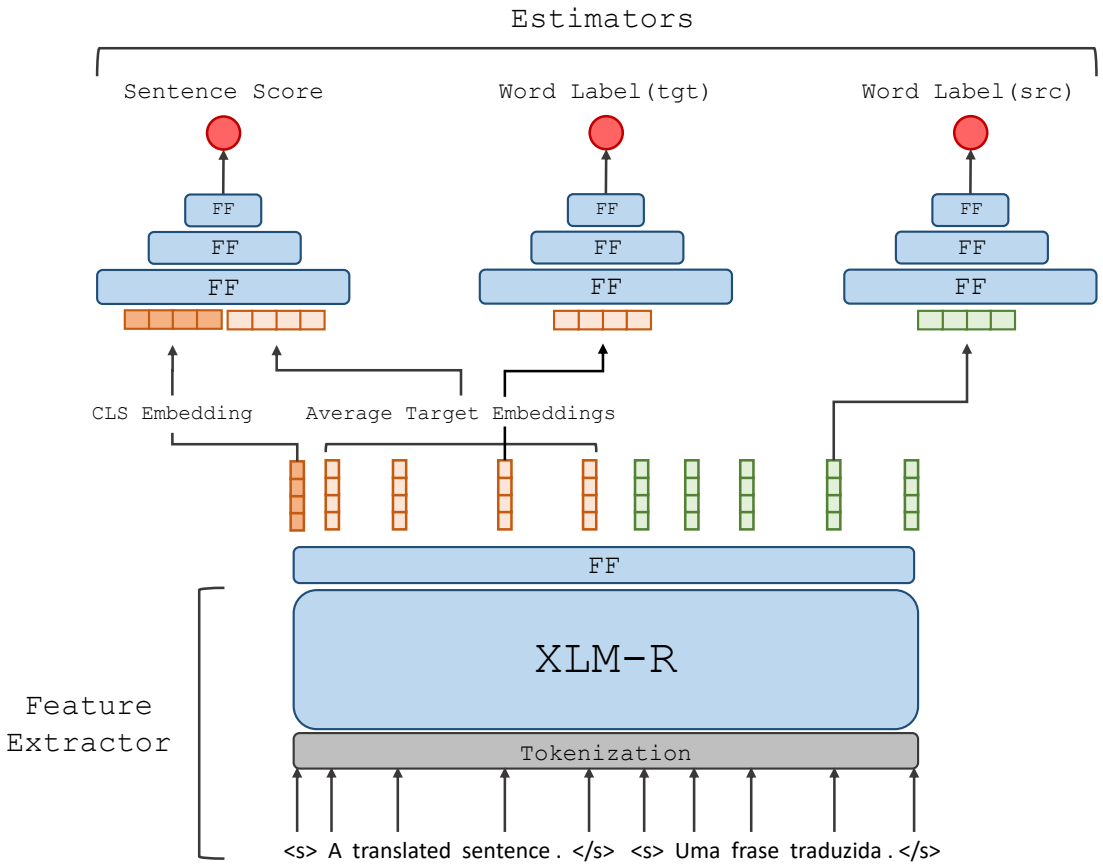

Figure 3.3: Architecture of the *Kiwi-Base* baseline system as implemented using the `OpenKiwi` framework.

ing applied on its outputs so that individual token features are combined into sentence representations. Both the model and its corresponding language-independent tokenizer (an implementation of `SentencePiece`[6]) are part of the `Transformers`[7] python package.

Source and target sentences are passed to XLMR as inputs in the format `<s> target </s> <s> source </s>`, with `<s>` representing beginning of sentence (BOS) and `</s>` representing end of sentence (EOS). In order to structure the input for the sentence-level estimator, a compact sentence representation is returned by pooling, namely averaging XLM-R's output embeddings for all tokens in the target sentence, and then concatenating the result with the classifier token embedding (first `<s>` in the input). Although XLM-R's pretraining does not include a Next Sentence Prediction objective - which means the classifier embedding has not been used to learn sentence representations -, it shows quick adaption to the new sentence-level objective, and using said token with the pooling strategy described above reliably outperforms others, like only using the pooled target embeddings, only the the classifier token itself, classifier token concatenated with pooled source embeddings, and others.

The Estimator modules are composed of neural feedforward layers, instead of a bi-LSTM (as used in

[6]https://github.com/google/sentencepiece
[7]https://huggingface.co/transformers/

the original Predictor-Estimator architecture); experiments conducted by WMT participants for the 2019 edition showed that replacing this component resulted in a similar performance, when using pre-trained LM's as Predictors. Three layers are used, progressively halving in dimension until reaching the output layer; here the model outputs either/both of the following: 1) a sentence score (Direct Assessment or HTER), optimized with a regression objective, using a Mean Squared Error loss, and 2) target and source tags, optimized with a classification objective, using a Binary Cross-Entropy objective.

If the model is trained in a multi-task learning setting , the losses calculated for each active estimator block's predictions are then summed, and used as the global system loss. In this case, the feed-forward layer placed after the Feature Extractor serves as a shared weight basis, used so that different tasks inform and benefit each other's individual performance, a practice used in various successful approaches to the QE task, such as [31].

The activation function used between all layers (excluding XLM-R) is ReLU ([45]), and the optimizer is AdamW ([40]) - an improvement on the Adam ([34]) optimizer, which decouples the gradient update from the weight decaying, instead of including an L2 regularization term in the loss function. This has been shown to yield better training loss and much better generalization capabilities than models trained with Adam.

3.2.3 Integrating Glass-Box Features

The extracted features described in section 3.1 are now introduced into *Kiwi-Base*. Different configurations were attempted in order to accomplish this, on the basis of integrating a 7-element vector (containing all glass-box features) somewhere in the architecture, where it influenced final performance.

First the feature vector was simply concatenated to the final hidden state before the output, then to other ones processed by preceding linear layers. Preliminary experiments proved this method to have no impact on accuracy, which indicated that the QE model was not able to properly incorporate the features with this setup.

A simple method worked best, showing promising results. First, the pooled sentence representation obtained from XLM-R was put through a feed-forward layer, and its dimensions reduced by about five-fold, creating a bottleneck; the feature vector is appended to the resulting hidden-state. Then, that hidden state is again put through a feed-forward layer and expanded back to a higher dimensional state. This idea draws inspiration from the autoencoder architecture (portrayed in figure 3.4) , where a bottleneck is created between the encoder and decoder parts of the model, and it is trained to reconstruct the input it was given. This requires creating a compressed representation of the input in the bottleneck, which condenses the most relevant features. While a reconstruction loss is not used to train the QE model (as is with the autoencoder), works in the area of Speech Technology (such as Grézl et al. [22]) have shown that training a feed-forward NN with a bottleneck in the middle, on a simple classification objective, results in rich features being created in the bottleneck (in the case of the cited work, the "decoder"

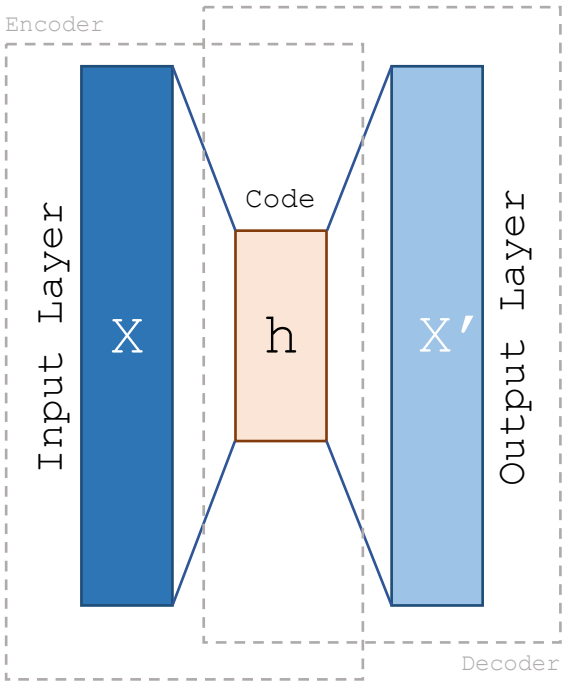

Figure 3.4: Autoencoder architecture; the model is trained to reconstruct input X by creating X'. This requires the learning of an efficient input representation - or code - at the bottleneck.

portion is removed after pretraining, and the bottleneck features are then used for downstream tasks). I posit that the same effect happens in the QE Feature Extractor, and by appending glass-box features to the bottleneck hidden state before it is processed by the feed-forward layer, each individual feature gets the chance to have more ”relevance” in the network's training dynamic. A schematic of the described integration can be seen in figure 3.5.

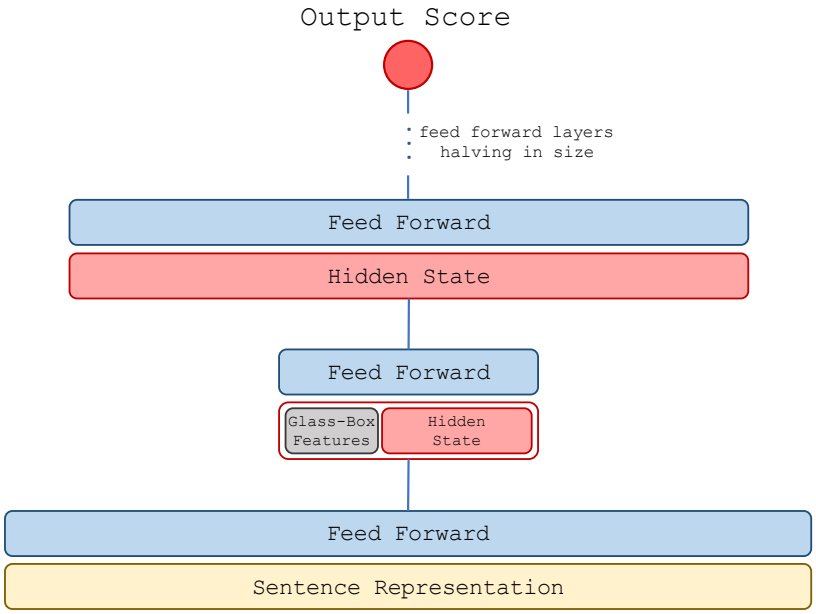

Figure 3.5: Architecture of the ”Quality Estimator” module modified to include glass-box features.

The rest of the architecture resembles *Kiwi-Base* in every other way, although introducing foreign features into an int ermediate hidden state requires a different range of hyperparameters in order to make training work.

In the next section, this approach will be compared to its base architecture with respect to performance, and quantitatively evaluated.

Chapter 4

Experiments and Results

This chapter presents the evaluation of the designed model, and the results of that process. Section 4.1 introduces the dataset we used for the addressed task, and other relevant model resources. Section 4.2 formalizes the metrics that measure performance for word and sentence-level estimation, which allow for a meaningful comparison with previous work, and incremental iterations on our model architecture. Finally, in section 4.3 we report the obtained results - comparing them with baselines on the task - and discuss their meaning in the context of our work.

4.1 Dataset and Model Resources

The dataset we used for developing the models presented in this thesis had a strong influence on the direction of the work itself. A part of the 2020 edition of the *WMT Quality Estimation Shared Task* [64], there were two tasks relevant to our work: task 1, for predicting Direct Assessment scores (sentence-level), and task 2, for predicting Post-Editing effort (word and sentence-level).

Task 1 was newly proposed for the 2020 workshop; previous works such as Graham et al. [20] had advised for the employment of DA scores instead of HTER for the evaluation of quality estimation systems, given that 1) although the correlation found between HTER and human assessment is fairly strong, it is not a sufficient stand-in for the latter, and 2) DA's provide a valid human assessment without an automatic component, and have been shown to produce replicable sentence-level human assessment scores for translations.

Task 2 has been standard in the workshop since its very creation in 2012; it evaluates the application of QE for post-editing purposes, and consists of 1) predicting OK and BAD labels for every token in both the target and source sentence, and additionally for each gap between two words where something is/is not missing, and 2) predicting the sentence-level HTER score, which is defined as the ratio between the number of edits (insertions/deletions/replacements) and the reference translation length.

Both tasks share the same dataset, newly sourced mainly from Wikipedia articles. It includes six language-pairs - 2 high, 2 medium, and 2 low-resource - namely English-German and English-Chinese (high), Romanian-English and Estonian-English (medium), Nepalese-English and Sinhala-English (low). An extra high-resource language-pair was added, Russian-English, however separated from the rest when it comes to content, being comprised of Russian Reddit forums (75%), and Russian WikiQuotes

(25%). Datasets for all language-pairs were divided into 7K sentences for training, and 1K sentences for development. Only a subset of the full dataset was annotated and made available for task 2, specifically for the English-German and English-Chinese language pairs.

For task 1, each sentence was annotated following the FLORES setup [24], which presents a form of DA. Here, at least 3 professional translators rate sentences from 0-100 according to their perceived translation quality. DA scores are then standardized using the z-score by rater; this score is the number of standard deviations by which the value of a raw score is above or below the mean value of all ratings that a given translator has provided, defined by:

$$z = \frac{x - \mu}{\sigma} \tag{4.1}$$

,where x is a raw DA score, μ is the mean of all ratings, and σ is the standard deviation of ratings for a given evaluator. Standardizing scores in this manner ensures that the raw score is not used as an absolute and independent value, but that the evaluator's rating distribution is taken into account. Therefore, the data available for each sentence is the original and translated sentences, the raw translations scores from the 3 evaluators and their mean, the z-scores and their mean (named z-mean henceforth), as well as the NMT model score for the sentence. The measure taken to describe quality, and used as prediction target to train the QE model, is the z-mean value.

Furthermore, the NMT models used to create the dataset were made available, so that system-internal information could be exploited for the task. For all language-pairs, these are standard Transformer models, with 6 encoder blocks and 6 decoder blocks. This resource helped motivate the idea for this thesis, as it is very in line with the efforts required for the task itself.

4.2 Evaluation Metrics

The word and sentence-level tasks are of two distinct natures; the former is a binary classification problem, while the latter is a regression problem. We will break up the two objectives in this section, which by their very nature are evaluated differently.

Sentence-Level (DA and HTER)

A regression task entails learning the mapping from input X to a continuous output variable Y - in this case, a real value.

Performance in regression tasks is usually assessed using some form of error measurement focused on the magnitude of the predicted value. When training a model to predict the appropriate market value of real-estate based on several of its features, for example, the desired outcome is that the predicted absolute value is as close as possible to the real absolute value. The most common metric to evaluate

precisely this - which is also often used as the loss function to train the model itself - is the Mean Squared Error (MSE); it measures the average squared difference between predictions and real values, and is defined by:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}\left(Y_i - \hat{Y}_i\right)^2 \tag{4.2}$$

where Y_i is the prediction, and $\hat{Y}_i$ is the real value. However, the broad research objective regarding the sentence-level QE task has veered away from an "absolute value" approach. Starting some years ago - in particular, influenced by the 2015 Quality Estimation Shared Task -, systems have typically been evaluated on the *direction* of their predictions. Plainly, the desired outcome is to achieve a greater correlation with sentence scores, be it HTER or DA.

The most common and generally used correlation metric is the Pearson correlation coefficient (or Pearson's r). This is a measure of linear correlation between two sets of data (in this case, z-mean scores and model predictions for each language pair), defined by the covariance of the two sets, normalized by the product of their standard deviations:

$$\rho_{X,Y} = \frac{\text{cov}(X,Y)}{\sigma_X \sigma_Y} \tag{4.3}$$

We use Pearson's r as the metric to assess all the models trained in the context of this thesis.

Word-Level (OK/BAD labels)

The quality of binary (two-class) predictions – such as predicting an OK or BAD label for each word – is generally measured by the Matthews correlation coefficient (MCC). In essence, this is a correlation coefficient between the observed and predicted binary classifications. It returns a value between 1 and +1, being that +1 represents a perfect prediction, 0 equivalent to random prediction, and -1 total disagreement between prediction and observation. Is is defined by:

$$\text{MCC} = \frac{TP \times TN - FP \times FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \tag{4.4}$$

, where TP, TN, FP and FN are true positives, true negatives, false positives and false negatives, in order. The MCC is a balanced measure which can be used even if the classes are of very different sizes (i.e. imbalanced). While there is no perfect way of describing a confusion matrix of true and false positives and negatives by a single number, MCC is generally regarded as being one of the best such measures.

4.3 Experimental Results

This section presents the experimental results obtained with the approach proposed in Chapter 3. First, in section 4.3.1, glass-box features will be independently assessed, and compared to past performance on this task. Section 4.3.2 will focus on the performance of the Base OpenKiwi System (3.2.2), and how that performance is affected when taking the approach described in 3.2.3.

The shared baseline for the comparison of results consists of a neural predictor-estimator system, as described in 2.3.2, and as implemented in Kepler et al. [31]. The predictor is pre-trained on the same parallel data used to train the available NMT models 4.1. This model will be refered to as `OpenKiwi` henceforth.

4.3.1 Glass-Box Features

The original premise and results of the publication which suggested the use of glass-box features (as they are presented in this thesis) as a quality indicator [17], was that these were on par with supervised quality estimation methods, in terms of correlation with human judgement. In order to test this - both for validation of the feature extraction process implementation, and for future comparison with the developed method - glass-box features were extracted from the provided models, using the validation sets for each language pair. Then, the Pearson r correlation was calculated for each feature-language pair. The results can be seen in Table 4.1.

	Feature	**Language Pair**						
		En-De	En-Zh	Ro-En	Et-En	Ne-En	Si-En	Ru-En
(i)	TP	0.0993	0.2808	0.5951	0.3992	0.3653	0.3658	0.3658
	Softmax-Ent	0.0858	0.2919	0.5595	0.3546	0.4133	0.4077	0.3790
	Sent-Std	0.0691	**0.3252**	0.5049	0.3985	0.3669	0.3912	0.3510
(ii)	D-TP	**0.1078**	0.3158	**0.6404**	**0.4936**	0.3905	0.3797	**0.4441**
	D-Var	0.0782	0.1943	0.3550	0.2780	0.2336	0.2338	0.2329
	D-Combo	0.0487	0.1259	0.2620	0.1335	0.2938	0.2244	0.2013
	D-Lex-Sim	0.0994	0.2903	0.6210	0.3940	**0.4751**	**0.4318**	0.4092

Table 4.1: Pearson correlation (r) between the employed *glass-box features* and human DA's for every language pair (validation set) - best results are in bold.

Features in group (i) are softmax distribution-based, in a standard deterministic inference scenario , while the ones in group (ii) are uncertainty quantification-based, as detailed in section 3.1.1.

As expected, group (ii) features consistently display higher correlation to human DA's across language pairs, `D-TP` being the most effective for high and medium resource languages, and `D-Lex-Sim` for low resource languages. This is in accordance with intuition, given that MT models trained on low-resource languages have had less data points to train and converge on, and might therefore create more variable outputs for the same source, when affected by dropout.

In the same line of analysis, we see that correlations on high resource languages are lower than

for the rest of the language-pairs. Since we are in fact measuring uncertainty with these features, it seems that, the more trained - and therefore confident - the model is, the less signal we get from the uncertainty quantification method used, which results in features that are less fit to independently estimate translation quality. Although this is an observable pattern for the obtained results, it cannot be generalized to all models, language pairs and datasets, as results are highly dependent on a multitude of factors, ranging from training hyperparameters, to dataset domain, content and size. A deeper analysis on this subject could is deffered to future work.

4.3.2 Glass-Box QE

In this section, comparisons are drawn between results obtained with:

- baseline Predictor-Estimator implemented in `OpenKiwi`;
- glass-box features independently (namely, the one which is the most correlated with DA's for each language pair);
- two variants of the Base Kiwi System described in Section 3.2.2: Kiwi-Base, and Kiwi-Large. These model's predictor module is respectively comprised of an XLM-Roberta base model ($\sim$270M parameters with 12-layers, 768-hidden-state, 3072 feed-forward hidden-state, 8-heads) and a large model ($\sim$550M parameters with 24-layers, 1024-hidden-state, 4096 feed-forward hidden-state, 16-heads). Our objective is to take advantage of the experiments that were done, to understand how much performance impact does the size of the predictor have on a downstream task;
- the model described in Section 3.2.3, named Kiwi-Glass-Box; this is Kiwi-Large, modified with the architectural changes developed in this thesis, which accommodate the introduction of glass-box features into the training process;
- finally, Kiwi-Glass-Box-Ensemble. As was mentioned previously, we submitted the developed models to the *2020 WMT Quality Estimation Shared Task*. The Shared Task presents a gamified approach to research, where participants use their developed models to submit predictions on a blind test set; there is then a leaderboard of participant scores, and therefore an elected winner. In the effort of obtaining a final performance boost, we ensemble the 5 best glass-box models obtained in the process of hyperparameter tuning, for each language pair. This is accomplished by using Lasso regression to tune their combined predictions on the validation set. Since we did not have access to the test set labels, these ensembles were trained using k-cross validation (k=10) on the validation set).

  Tables 4.2 and 4.3 show the results obtained on both tasks.

Pair	System	Pearson	
		VAL	TEST
En-De	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.5715	0.5230
	KIWI-GLASS-BOX	0.5263	-
	KIWI-LARGE	0.4794	-
	OPENKIWI-BASE	0.3499	0.2670
	BEST GB FEATURE	0.1078	-
	Openkiwi 1.0	-	0.1455
En-Zh	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.5711	0.4940
	KIWI-GLASS-BOX	0.5461	-
	KIWI-LARGE	0.5258	-
	OPENKIWI-BASE	0.4199	0.3460
	BEST GB FEATURE	0.3252	-
	OpenKiwi 1.0	-	0.1902
Ro-En	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.8968	0.8910
	KIWI-GLASS-BOX	0.8841	-
	KIWI-LARGE	0.8790	-
	OPENKIWI-BASE	0.6672	0.7080
	BEST GB FEATURE	0.6404	-
	OpenKiwi 1.0	-	0.6845
Et-En	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.7697	0.7700
	KIWI-GLASS-BOX	0.7611	-
	KIWI-LARGE	0.7496	-
	OPENKIWI-BASE	0.6728	0.6900
	BEST GB FEATURE	0.4936	-
	OpenKiwi 1.0	-	0.4770
Ne-En	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.7994	0.7920
	KIWI-GLASS-BOX	0.7804	-
	KIWI-LARGE	0.7711	-
	OPENKIWI-BASE	0.6987	0.6040
	BEST GB FEATURE	0.4751	-
	OpenKiwi 1.0	-	0.3860
Si-En	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.6896	0.6390
	KIWI-GLASS-BOX	0.6604	-
	KIWI-LARGE	0.6521	-
	OPENKIWI-BASE	0.5727	0.5650
	BEST GB FEATURE	0.4318	-
	OpenKiwi 1.0	-	0.3737
Ru-En	(*)**KIWI-GLASS-BOX-ENSEMBLE**	0.7391	0.7670
	KIWI-GLASS-BOX	0.7137	-
	KIWI-LARGE	0.6938	-
	OPENKIWI-BASE	-	-
	BEST GB FEATURE	0.4441	-
	OpenKiwi 1.0	-	0.5479

Table 4.2: Task 1 results on the validation and test sets for all language pairs in terms of Pearson's r correlation. Systems in **bold** were officially submitted to the *2020 Quality Estimation Shared Task*. (*) Lines with an asterisk use LASSO regression to tune ensemble weights on the validation set, therefore their numbers cannot be directly compared to the other models.

Pair	System	Target MCC		Source MCC		Pearson	
		Val	Test	Val	Test	Val	Test
En-De	KIWI-GLASS-BOX	**0.460**	**0.465**	**0.357**	**0.349**	**0.618**	**0.633**
	OPENKIWI-BASE	0.445	0.432	0.330	0.324	0.561	0.531
	(*)OpenKiwi 1.0	-	0.358	-	0.266	-	0.392
En-Zh	KIWI-GLASS-BOX	0.567	0.567	**0.348**	0.287	**0.691**	**0.651**
	OPENKIWI-BASE	**0.576**	**0.575**	0.298	0.287	0.615	0.593
	(*)OpenKiwi 1.0	-	0.509	-	0.270	-	0.506

Table 4.3: Task 2 word and sentence-level results on the validation and test sets. Results for OPENKIWI-BASE and KIWI-GLASS-BOX were obtained from a single model trained by multi-tasking on the 3 different subtasks. (*) Baseline results on the validation set were not made available by the organizers.

Most comparisons we draw from the obtained results in the following paragraphs are expressed for task 1; this task was initially worked on more deeply, and was the one used as experimentation and feedback mechanism to understand the impact of the developed method. Once validated, the method was applied in a straightforward way to task 2. Analyzing the results, we can answer a series of questions that help assess the different components and experiments we led:

Are glass-box features good unsupervised quality estimators?

As we alluded to in section 4.3.1, the extracted features by themselves achieve a very comparable - and for some language pairs, even better - performance than the best approach from previous years, OpenKiwi 1.0. This is all the more of an impressive result, considering that these features are extracted in a completely unsupervised manner, and points to potential use cases where labelled data is not available - either at all, or in a big enough quantity to train an accurate QE model.

Are pre-trained contextualized embeddings a better choice than training a Predictor from scratch?

Using pre-trained contextualized embeddings with XLMRoberta proves to greatly outperform the baseline system OpenKiwi 1.0 in both tasks, even when not taking XLMR-large into account. The fact that RNN's are replaced by a Transformer architecture, the optimization for multilingual pre-training implemented in XLMR (3.2.1), and the sheer amount of data that XLMR is pre-trained with, all contribute to this difference. It also highlights the virtues of using transfer-learning in NLP tasks; pre-training a neural model with the size and amount of data that has been proven to be required for powerful language representation is impossible in most cases, and taking advantage of the computational expense incurred by large research entities enables explorations such as the one developed for this thesis.

What effect does increasing the Predictor's capacity have?

Switching from using XLMR Base to Large as the Predictor component has a very strong impact in performance, with the increase in correlation averaging 11,3% across language-pairs (tested on task 1 only). This is in line with findings from the original XLMR paper [12], which indicated that adding capacity to a multilingual model alleviates the *curse of multilinguality* (degrading performance caused by training

on many languages), and results in higher performance for the same number of languages involved in the training process.

Do glass-box features positively inform the training of QE models?

The developed Kiwi-Glass-Box approach consistently increases performance across language pairs and in both tasks. Sentence-level improvements are more visible in predicting HTER (task 2), with the increase averaging 6,6% for both language-pairs, but DA prediction performance benefits nonetheless (task 1), averaging 2% across language-pairs, and as high as 4.7% in the case of En-De. The glass-box features are leveraged by the model during training, resulting in a stronger correlation with human judgement than either one separately. It is curious to note that, even though features extracted for high-resource language pairs show the lowest correlations independently, the QE models trained on those language pairs make the best use of them, judging by performance increase. This could be a factor of the language itself, or alternatively, the translation patterns of a more extensively trained NMT model, learned by the QE model, which might find in uncertainty measurements more signal for determining translation quality.

In the multi-task setting of task 2, results show that is possible for word-level performance to be positively influenced by added information at the sentence-level. Apart from the case of target label prediction for En-Zh, the word-level task and its corresponding component in the model architecture (the final binary classification block) benefits from the extra information on which the shared layers are trained (refer to Section 3.2.2).

All in all, the proposed method improves performance across the board on both word and sentence-level prediction.

Chapter 5

Conclusions

In this chapter we summarize the main contributions and findings of this dissertation. Then, we suggest some of the possible directions for future work regarding the proposed method.

5.1 Summary of Contributions

In Section 1.1, we described the task of Translation Quality Estimation and discussed its relevance for the applicability of Machine Translation systems. We also described the aim of our work: enhancing state-of-the-art QE models, by allowing them to leverage internal features from NMT models - or glass-box features.

Our starting point was the set of glass-box features introduced in Fomicheva et al. [17], originally used directly as unsupervised quality estimators, and proven to be effective at representing NMT model uncertainty. We began by implementing the extraction of these features for the NMT models used, as described in Section 3.1.

Then, in Section 3.2.2 we proposed a QE model architecture on which to apply these features, leveraging the Natural Language Understanding capabilities of a multi-lingual, pre-trained Language Model (XLM-Roberta). Independently, this architecture yielded results that surpassed the performance of RNN-based QE systems - the state-of-the-art up until this point -, as we show in Section 4.3.2.

Finally, in Section 3.2.3 we developed a method to introduce glass-box features into the proposed model's training process. We validated our design choices, confirmed the independent relevance of the features as quality indicators, and at last showed that the QE model's performance in the sentence-level task consistently increases across language pairs, when using them as extra input information. We also showed that, although features are extracted on a sentence-level granularity only, multi-task learning paired with weight sharing between sentence and word-level Estimator components has a positive influence on the model's performance in predicting word-level labels. These conclusions are part of the evaluation drawn from results across two tasks and six language pairs, obtained as the result of our participation in the *WMT Quality Estimation Shared Task* (Moura et al. [43]), and according to the proposed metrics.

5.2 Future Work

5.2.1 Explore NMT Corpus Size Influence

One of the conclusions that were taken from the obtained results, was that even though the glass-box features extracted for high-resource language pairs showed the lowest correlations with human judgements as unsupervised quality indicators, the QE models trained on those language pairs seem to make the best use of them. To explore the reason for this phenomena, an experiment could be conducted, consisting of training a pair of (or more) NMT models on each language pair (with emphasis on high resource), using increasing portions of the full training data set. Corresponding QE models would be trained on each NMT model's translation outputs, and the increase in performance by using glass-box features could then be compared for each model, when tested on a validation/test data set. This would allow to more deeply isolate the influences of specific language pairs, and size of NMT Corpus on the final results for QE model performance. However, this experiment would require the human quality labelling of all translation outputs, for each model and the respective dataset portion it was trained on. Time, and most importantly resource constraints prevented us from conducting the described experiment in the context of this thesis.

5.2.2 Feature Granularity and Integration

In our work, we extracted and used sentence-level glass-box features only. It would be interesting to extract these features - and even other suitable ones - on the basis of each word prediction made by the NMT model, and then test how these will impact prediction of word labels, using the same method as the one we have developed. Since the sentence-level features are always pooled/aggregated in some way in order to represent the prediction uncertainty of the whole sentence, important information may come from more granular glass-box features.

In another direction, it would be interesting to integrate the internal information coming from the NMT model with the QE training process in a different way; the trend we have empirically observed to work best for most Deep Learning approaches is not to hand-craft features at all, but contrarily to start with a model's raw hidden states, and let the Neural Network optimize on how to better make use of them. In this context, it would be interesting to see the effects of applying uncertainty quantification methods to internal states of NMT models, and integrating them in their raw form with the proposed QE model.

5.2.3 End-to-end Training

Carrying on with the line of thought begun in the previous section, both for the case of using crafted features that we explored in this work, and for the possible use of raw NMT model hidden states, training the NMT and QE model jointly in an end-to-end fashion would be a relevant setting to experiment. However, such an experiment would be expensive computation-wise (two large models would have to

be loaded into GPU memory and used at the same time), which may require some form of parameter sharing for computational viability.

Bibliography

[1] O. I. Abiodun, A. Jantan, A. E. Omolara, K. V. Dada, N. A. Mohamed, and H. Arshad. State-of-the-art in artificial neural network applications: A survey. *Heliyon*, 4(11):e00938, 2018. ISSN 2405-8440. doi: https://doi.org/10.1016/j.heliyon.2018.e00938. URL http://www.sciencedirect.com/science/article/pii/S2405844018332067.

[2] M. Z. Alom, T. M. Taha, C. Yakopcic, S. Westberg, P. Sidike, M. S. Nasrin, B. C. V. Esesn, A. A. S. Awwal, and V. K. Asari. The history began from alexnet: A comprehensive survey on deep learning approaches, 2018.

[3] J. L. Ba, J. R. Kiros, and G. E. Hinton. Layer normalization, 2016.

[4] D. Bahdanau, K. Cho, and Y. Bengio. Neural machine translation by jointly learning to align and translate, 2016.

[5] S. Banerjee and A. Lavie. 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, Ann Arbor, Michigan, June 2005. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W05-0909.

[6] E. Biçici, Q. Liu, and A. Way. Referential translation machines for predicting translation quality and related statistics. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 304–308, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3035. URL https://www.aclweb.org/anthology/W15-3035.

[7] J. Blatz, E. Fitzgerald, G. Foster, S. Gandrabur, C. Goutte, A. Kulesza, A. Sanchis, and N. Ueffing. Confidence estimation for machine translation. In *COLING 2004: Proceedings of the 20th International Conference on Computational Linguistics*, pages 315–321, Geneva, Switzerland, aug 23–aug 27 2004. COLING. URL https://www.aclweb.org/anthology/C04-1046.

[8] S. Boggan and D. M. Pressel. Gpus: An emerging platform for general-purpose computation. Technical report, ARMY RESEARCH LAB ABERDEEN PROVING GROUND MD COMPUTATIONAL AND INFORMATION . . . , 2007.

[9] C. Callison-Burch, P. Koehn, C. Monz, M. Post, R. Soricut, and L. Specia. Findings of the 2012 workshop on statistical machine translation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 10–51, Montréal, Canada, June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/W12-3102.

[10] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation, 2014.

[11] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuoglu, and P. Kuksa. Natural language processing (almost) from scratch, 2011.

[12] A. Conneau, K. Khandelwal, N. Goyal, V. Chaudhary, G. Wenzek, F. Guzmán, E. Grave, M. Ott, L. Zettlemoyer, and V. Stoyanov. Unsupervised cross-lingual representation learning at scale, 2020.

[13] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*, 2009.

[14] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[15] G. Doddington. Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the Second International Conference on Human Language Technology Research*, HLT '02, page 138–145, San Francisco, CA, USA, 2002. Morgan Kaufmann Publishers Inc.

[16] C. Dyer, A. Lopez, J. Ganitkevitch, J. Weese, F. Ture, P. Blunsom, H. Setiawan, V. Eidelman, and P. Resnik. cdec: A decoder, alignment, and learning framework for finite-state and context-free translation models. In *Proceedings of the ACL 2010 System Demonstrations*, pages 7–12, Uppsala, Sweden, July 2010. Association for Computational Linguistics. URL https://aclanthology.org/P10-4002.

[17] M. Fomicheva, S. Sun, L. Yankovskaya, F. Blain, F. Guzmán, M. Fishel, N. Aletras, V. Chaudhary, and L. Specia. Unsupervised quality estimation for neural machine translation, 2020.

[18] Y. Gal and Z. Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning, 2016.

[19] M. Gamon, A. Aue, and M. Smets. Sentence-level MT evaluation without reference translations: beyond language modeling. In *Proceedings of the 10th EAMT Conference: Practical applications of machine translation*, Budapest, Hungary, May 30–31 2005. European Association for Machine Translation. URL https://www.aclweb.org/anthology/2005.eamt-1.15.

[20] Y. Graham, T. Baldwin, M. Dowling, M. Eskevich, T. Lynn, and L. Tounsi. Is all that glitters in machine translation quality estimation really gold? In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 3124–3134, Osaka, Japan, Dec. 2016. The COLING 2016 Organizing Committee. URL https://www.aclweb.org/anthology/C16-1294.

[21] E. Grave, P. Bojanowski, P. Gupta, A. Joulin, and T. Mikolov. Learning word vectors for 157 languages, 2018.

[22] F. Grézl, M. Karafiát, S. Kontar, and J. ernocký. Probabilistic and bottle-neck features for lvcsr of meetings. *2007 IEEE International Conference on Acoustics, Speech and Signal Processing - ICASSP '07*, 4:IV–757–IV–760, 2007.

[23] C. Guo, G. Pleiss, Y. Sun, and K. Q. Weinberger. On calibration of modern neural networks, 2017.

[24] F. Guzman, P.-J. Chen, M. Ott, J. Pino, G. Lample, P. Koehn, V. Chaudhary, and M. Ranzato. The flores evaluation datasets for low-resource machine translation: Nepali–english and sinhala–english. pages 6100–6113, 01 2019. doi: 10.18653/v1/D19-1632.

[25] C. Hardmeier, J. Nivre, and J. Tiedemann. Tree kernels for machine translation quality estimation. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 109–113, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-3112.

[26] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition, 2015.

[27] S. Hochreiter and Y. Bengio. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. 2001.

[28] S. Hochreiter and J. Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, Nov. 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL https://doi.org/10.1162/neco.1997.9.8.1735.

[29] C. Hu, H. Liu, K. Feng, C. Xu, N. Xu, Z. Zhou, S. Yan, Y. Luo, C. Wang, X. Meng, T. Xiao, and J. Zhu. The NiuTrans system for the WMT20 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1018–1023, Online, Nov. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.wmt-1.117.

[30] N. Kalchbrenner and P. Blunsom. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709, Seattle, Washington, USA, Oct. 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D13-1176.

[31] F. Kepler, J. Trénous, M. Treviso, M. Vera, A. Góis, M. A. Farajian, A. V. Lopes, and A. F. T. Martins. Unbabel's participation in the wmt19 translation quality estimation shared task, 2019.

[32] F. Kepler, J. Trénous, M. Treviso, M. Vera, and A. F. T. Martins. OpenKiwi: An open source framework for quality estimation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics–System Demonstrations*, pages 117–122, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P19-3020.

[33] H. Kim, H. Jung, H.-S. Kwon, J.-H. Lee, and S.-H. Na. Predictor-estimator. *ACM Transactions on Asian and Low-Resource Language Information Processing (TALLIP)*, 17:1 – 22, 2017.

[34] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization, 2017.

[35] J. Kreutzer, S. Schamoni, and S. Riezler. QUality estimation from ScraTCH (QUETCH): Deep learning for word-level translation quality estimation. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 316–322, Lisbon, Portugal, Sept. 2015. Association for Computational Linguistics. doi: 10.18653/v1/W15-3037. URL https://www.aclweb.org/anthology/W15-3037.

[36] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[37] G. Lample and A. Conneau. Cross-lingual language model pretraining, 2019.

[38] Z. Lan, M. Chen, S. Goodman, K. Gimpel, P. Sharma, and R. Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[39] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[40] I. Loshchilov and F. Hutter. Decoupled weight decay regularization, 2019.

[41] A. F. T. Martins, R. Astudillo, C. Hokamp, and F. Kepler. Unbabel's participation in the WMT16 word-level translation quality estimation shared task. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 806–811, Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2387. URL https://www.aclweb.org/anthology/W16-2387.

[42] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space, 2013.

[43] J. Moura, M. Vera, D. van Stigt, F. Kepler, and A. F. T. Martins. IST-unbabel participation in the WMT20 quality estimation shared task. In *Proceedings of the Fifth Conference on Machine Translation*, pages 1029–1036, Online, Nov. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.wmt-1.119.

[44] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, page 807–814, Madison, WI, USA, 2010. Omnipress. ISBN 9781605589077.

[45] V. Nair and G. E. Hinton. Rectified linear units improve restricted boltzmann machines. In *ICML*, 2010.

[46] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu. 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, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://www.aclweb.org/anthology/P02-1040.

[47] R. N. Patel and S. M. Translation quality estimation using recurrent neural network. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 819–824,

Berlin, Germany, Aug. 2016. Association for Computational Linguistics. doi: 10.18653/v1/W16-2389. URL https://www.aclweb.org/anthology/W16-2389.

[48] J. Pennington, R. Socher, and C. Manning. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar, Oct. 2014. Association for Computational Linguistics. doi: 10.3115/v1/D14-1162. URL https://www.aclweb.org/anthology/D14-1162.

[49] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations, 2018.

[50] C. B. Quirk. Training a sentence-level machine translation confidence measure. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation (LREC'04)*, Lisbon, Portugal, May 2004. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2004/pdf/426.pdf.

[51] R. Raina, A. Madhavan, and A. Y. Ng. Large-scale deep unsupervised learning using graphics processors. In *Proceedings of the 26th annual international conference on machine learning*, pages 873–880, 2009.

[52] R. Rei, C. Stewart, A. C. Farinha, and A. Lavie. COMET: A neural framework for MT evaluation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 2685–2702, Online, Nov. 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.emnlp-main.213.

[53] F. Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

[54] D. Rumelhart, G. E. Hinton, and R. J. Williams. Learning internal representations by error propagation. 1986.

[55] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. *Learning Internal Representations by Error Propagation*, page 318–362. MIT Press, Cambridge, MA, USA, 1986. ISBN 026268053X.

[56] C. Scarton and L. Specia. Exploring consensus in machine translation for quality estimation. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*, pages 342–347, Baltimore, Maryland, USA, June 2014. Association for Computational Linguistics. doi: 10.3115/v1/W14-3343. URL https://www.aclweb.org/anthology/W14-3343.

[57] M. Schuster and K. Paliwal. Bidirectional recurrent neural networks. *IEEE Trans. Signal Process.*, 45:2673–2681, 1997.

[58] M. Snover, B. J. Dorr, R. Schwartz, and L. Micciulla. A study of translation edit rate with targeted human annotation. 2006.

[59] R. Soricut, N. Bach, and Z. Wang. The SDL language weaver systems in the WMT12 quality estimation shared task. In *Proceedings of the Seventh Workshop on Statistical Machine Translation*, pages 145–151, Montréal, Canada, June 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W12-3118.

[60] B. K. Spears, J. Brase, P.-T. Bremer, B. Chen, J. Field, J. Gaffney, M. Kruse, S. Langer, K. Lewis, R. Nora, and et al. Deep learning: A guide for practitioners in the physical sciences. *Physics of Plasmas*, 25(8):080901, Aug 2018. ISSN 1089-7674. doi: 10.1063/1.5020791. URL http://dx.doi.org/10.1063/1.5020791.

[61] L. Specia, N. Cancedda, M. Dymetman, M. Turchi, and N. Cristianini. Estimating the sentence-level quality of machine translation systems. In *EAMT*, 2009.

[62] L. Specia, D. Raj, and M. Turchi. Machine translation evaluation versus quality estimation. *Machine translation*, 24(1):39–50, 2010.

[63] L. Specia, K. Shah, J. G. de Souza, and T. Cohn. QuEst - a translation quality estimation framework. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 79–84, Sofia, Bulgaria, Aug. 2013. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/P13-4014.

[64] L. Specia, F. Blain, M. Fomicheva, E. Fonseca, V. Chaudhary, F. GuzmÃ¡n, and A. F. T. Martins. Findings of the wmt 2020 shared task on quality estimation. In *Proceedings of the Fifth Conference on Machine Translation*, pages 743–764, Online, November 2020. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/2020.wmt-1.79.

[65] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56): 1929–1958, 2014. URL http://jmlr.org/papers/v15/srivastava14a.html.

[66] I. Sutskever, O. Vinyals, and Q. V. Le. Sequence to sequence learning with neural networks, 2014.

[67] W. L. Taylor. “cloze procedure”: A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953. doi: 10.1177/107769905303000401. URL https://doi.org/10.1177/107769905303000401.

[68] W. L. Taylor. “cloze procedure”: A new tool for measuring readability. *Journalism quarterly*, 30(4): 415–433, 1953.

[69] B. Thompson and M. Post. Automatic machine translation evaluation in many languages via zero-shot paraphrasing, 2020.

[70] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

[71] S. Wang, Z. Tu, S. Shi, and Y. Liu. On the inference calibration of neural machine translation, 2020.

[72] P. Werbos. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560, 1990. doi: 10.1109/5.58337.

[73] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. Bertscore: Evaluating text generation with bert, 2020.