
# Exploring Lifelong Learning in Neural Machine Translation

June 29, 2021

## Master's Thesis

**Master's degree in Advanced Telecommunication Technologies**

**Autor:** Lluís Guàrdia Fernàndez

**Directors:** Marta R. Costa-Jussà

Magdalena Biesialska


## Abstract

Lifelong learning is a technique defined as the ability to learn new or maintain older knowledge over time. This factor is very important for humans, as it is one of the keys to the ability to learn by humans, which artificial intelligence tries to replicate. Additionally, it supposes the possibility for the systems to adapt to new inputs without the necessity to train the models from scratch every time.

This technique, although being already applied on autonomous agents or machine learning for computer vision, has not been evaluated in the field of Machine Translation (MT). In the case of MT, currently, the majority of processes are still using sets of static data or traditional techniques which force to train the model only once without taking into account the possible variation of the language in the context.

This project will be carried out on the context of a system implemented on the BEAT platform and thought for the evaluation of a lifelong learning task, which uses two sets of data: one for training and the other to apply the learning without the translations, having a simulated person to whom we can request the translations for the sentences we think are necessary.

The objective is to suggest and analyze the usage of an active learning technique: Quality Estimation (QE), and the following comparison with the results obtained using random selection. In this project, we work with the language pairs of English-French and English-German.

As results with QE we achieve a score of 26.7 and 15.9 points for EN-FR and EN-DE, respectively, using a penalizing n-grams precisions BLEU score. These means an improvement of 0,5 for EN-FR and 0,7 for EN-DE over the results obtained using random selection.

In conclusion, the usage of lifelong learning in machine translation is feasible, although still is in an initial phase. As future possible actions over QE would be interesting to make a more extensive search of parameters or using an adaptive QE model.

## Resum

L'aprenentatge continu és una tècnica que es defineix com l'habilitat d'aprendre nou coneixement o mantenir el ja obtingut prèviament de forma contínua en el temps. Aquest factor és molt important per ser una de les claus en l'aprenentatge humà que la intel·ligència artificial intenta replicar. A més, suposa la possibilitat dels sistemes a adaptar-se a certs inputs nous sense la necessitat d'entrenar el model des de zero cada vegada.

Aquesta tècnica, tot i ja estar aplicada en agents autònoms o aprenentatge automàtic en visió per computador, encara falta d'estar avaluada en el camp de Traducció Automàtica. En aquest cas, actualment, en la majoria de processos se segueix fent servir conjunts de dates estàtiques o tècniques tradicionals que obliguen a entrenar una sola vegada el model sense tenir en compte la possible variació del llenguatge en el context.

Així doncs, aquest projecte es durà a terme en el context d'un sistema implementat en la plataforma BEAT i pensat per l'avaluació d'una tasca d'aprenentatge continu, en el qual es disposa de dos conjunts de dades: una per entrenament i una per aplicar l'aprenentatge sense les traduccions, disposant d'una persona simulada a la qual li podem sol·licitar les traduccions de les frases que creiem necessàries.

L'objectiu és proposar i analitzar l'ús de una tècnica d'aprenentatge actiu: Estimació de Qualitat (EQ), i la comparació posterior amb els resultats obtinguts davant l'ús d'una selecció aleatòria de les frases a traduir. En aquest projecte es treballa amb els parells de llenguatges d'Angles-Francès i Angles-Alemany.

Com a resultats amb EQ hem obtingut uns valors de 26.7 i 15.9 per EN-FR i EN-DE, respectivament, fent servir una puntuació BLEU penalitzadora amb precisió n-gram. Això suposa una millora de 0,5 punts per EN-FR i 0,7 punts per EN-DE sobre els resultats obtinguts fent servir selecció aleatòria.

En definitiva, l'ús d'aprenentatge continu en traducció automàtica és factible, tot i que encara està en una situació molt inicial. Com a possibles accions futures en EQ seria interessant fer una cerca més extensiva dels paràmetres o fer ús d'un model EQ adaptatiu.

## Resumen

Aprendizaje continuo es una técnica que se define como la habilidad de aprender nuevo conocimiento o mantener el ya obtenido previamente de forma continua en el tiempo. Este factor es muy importante por ser una de las claves en el aprendizaje humano el cual la inteligencia artificial intenta replicar. Además supone la posibilidad de los sistemas a adaptar-se a ciertos inputs nuevos sin la necesidad de entrenar los modelos desde cero cada vez.

Esta técnica, aun ya estar aplicada en agentes autónomos o aprendizaje automático en visión por computador, no ha estado evaluada en el campo de Traducción Automática. En este caso, actualmente, en la mayoría de procesos se sigue usando conjuntos de dadas estáticas o técnicas tradicionales que obligan a entrenar una sola vez un modelo sin tener en cuenta la posible variación del lenguaje en el contexto.

Así pues, este proyecto se llevará a cabo en el contexto de un sistema implementado en la plataforma BEAT y pensado para la evaluación de una tasca de aprendizaje continuo, en el cual se dispone de dos conjuntos de datos: uno para entrenamiento y uno para aplicar el aprendizaje sin las traducciones, disponiendo de una persona simulada a la cual le podemos solicitar las traducciones de las frases que creamos necesarias.

El objetivo es proponer y analizar el uso de una técnica de aprendizaje activo: Estimación de Calidad (EC), i la comparación posterior con los resultados obtenidos delante el uso de una selección aleatoria de las frases a traducir. En este proyecto se trabaja con los pares de lenguajes de Ingles-Francés e Ingles-Alemán.

Como resultados con EC hemos obtenido unos valores de 26.7 y 15.9 para EN-FR y EN-DE, respectivamente, usando una puntuación BLEU penalizadora con precisión n-gram. Esto supone una mejora de 0,5 puntos para EN-FR y 0,7 puntos para EN-DE sobre los resultados obtenidos con la selección aleatoria.

En definitiva, el uso de aprendizaje continuo en traducción automático es factible, aunque aún se encuentra en una situación muy inicial. Como posibles acciones futuras para EC sería interesante hacer una cerca más extensiva de los parámetros o hacer uso de un modelo EC adaptativo.

## Acknowledgments

First I want to thank both my tutor Marta R. Costa-Jussà for guiding me through this project, and Magdalena Biesialska, who was always trying to help me whenever I needed and without her would have been impossible to finish this project. It was a pleasure to work with you.

I want to thank my parents for all the support, dedication, affection and all the invaluable things they gave me during all my life.

This last year and a half was a little strange because, due to the pandemic, and there wasn't any presential class nor other students interaction almost. And so I want to thank even more Maria, which was an esential emotional support during this rough time having infinite patience bearing me and listening all my technical verbosity.

# Revision history and approval record

| Revison | Date | Purpose |
|---|---|---|
| 0 | 14/05/2021 | Document creation |
| 1 | 17/06/2021 | Document revision |
| 2 | 28/06/2021 | Document revision |
| 3 | 29/06/2021 | Document approbation |

DOCUMENT DISTRIBUTION LIST

| Name | |
|---|---|
| Lluís Guàrdia Fernàndez | |
| Marta R. Costa-Jussà | |
| Magdalena Biesialska | |

| Written by: | | Reviewed and approved by: | |
|---|---|---|---|
| **Date** | 14/05/2021 | **Date** | 29/06/2021 |
| **Name** | Lluís Guàrdia Fernàndez | **Name** | Marta R. Costa-Jussà<br>Magdalena Biesialska |
| **Position** | Project Author | **Position** | Project Supervisors |

# Contents

# List of Figures

# List of Tables

# 1 Introduction

From long to the past, humans have been interested in the possibility of communicating in an automatic way between different languages. From Johan Joachim Becher in 1661, with the first MT resembled approach system [Becher1962], until the more recent Alan Turing and his decipherment of the German Enigma machine during WW2 [Lee1997]; different approaches and inventions have been developed with that goal in mind.

Machine Translation has the translation of documents without help from any other person as its goal. Since a few years ago, MT systems have been improving a lot. This is especially thanks to the progress in Machine Learning, the increased number of source texts available in more and more languages from the internet and the improvement in accessibility, both for companies and consumers.

Despite this, the majority of the works have been focused on the translation process between different languages with different methods and techniques, but always within a context of stability, using data that does not evolve with time.

This is a differentiating factor for humans, their languages and vocabulary changes over time, as an example, the actual English from America is very different from one of its predecessors, the English talked in the middle ages in England. But you can see this same factor taking a smaller step in time, during their own adulthood people are able to learn new words which were not on their prior knowledge (such as a lot of us did not know about the word/concept of cryptocurrency) and use them in the adequate context in the future. In machine learning, normally if you use a word that was not on the model training distribution, it will hardly know what to do with it or it will be too overconfident with it, although there have been some works addressing this [Sennrich et al.2016b].

In order to treat this problem, continual or lifelong learning is a technique defined as the ability to continually learn new and retain older knowledge. This ability has started to gain attention and been applied in several artificial intelligence systems ([Parisi et al.2018]; [Biesialska et al.2020]), although it has yet to be considered solved, as it still presents some problems which have not been solved, such as catastrophic forgetting [French1999] among others.

Currently, the majority of the evaluations to systems with this ability are gathered on other machine learning areas such as autonomous or learning agents [Isele2018] or machine learning for computer vision ([Zhai et al.2019a]; L3ViSU project[1]) and still have not been properly evaluated on the Machine Translation (MT) field, probably due to the lack of a benchmark to deal with it [Biesialska et al.2020]. This could be a very important factor in the following years as it could allow the MT systems to adapt to new vocabularies and topics and produce accurate translations without having to retrain the model from zero every time.

[1] http://pub.ist.ac.at/ chl/erc/

## 1.1 Statement of purpose and contributions

The main goal of the project is developing and evaluating the efficacy of different lifelong learning techniques, developing systems that adapt to data which is evolving with time. In the future is expected to participate in the 2nd Lifelong Learning for Machine Translation WMT Task on both English-German and English-French translation directions on the Sixth Conference On Machine Translation (WMT21).

The main contribution of this project is the implementation of a lifelong learning system using the open-source platform BEAT [Anjos et al.2017], along with the usage and evaluation of the Quality Estimation (QE) and Active Learning by Processing Surprisal (ALPS) techniques for the adaptation of the deep learning model using active learning.

## 1.2 Requirements and specifications

As the requirements for this project, for the system implementations we used the Open-Source Web-Based Open-Science Platform BEAT [Anjos et al.2017], which requires a Pytorch version greater than or equal to 1.6.0. along with a Python version greater or equal to 3.7.

For the active learning sampling strategy with a QE model, we used the Pytorch-based open-source OpenKiwi framework v0.1.3 [Kepler et al.2019], and for the sampling strategy with ALPS we used the ALPS architecture [Yuan et al.2020].

All the software has been launched in the CALCULA cluster, which consists of 8 servers from the TSC department of the UPC, each with 2 IntelR©XeonR©E5-2670 v3 2,3GHz 12N processors, and a total of 16 NVIDIA GTX Titan X GPUs. Each GPU has 12GB of memory and 3072 CUDA Cores. Among those servers, there was a great heterogenous variety of CPUs for this task due to the difference in their manufacturing year. The specifications for them are in the table 1:

| | processor | model name | cpu MHz | cache size | cpu cores |
|---|---|---|---|---|---|
| veuc01 | 39 | Intel(R) Xeon(R) CPU E5-2630 v4 @ 2.20GHz | 2399.864 | 25600KB | 10 |
| veuc05 | 47 | Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz | 1915.323 | 30720 KB | 12 |
| veuc06 | | | 2599.866 | | |
| veuc07 | 23 | Intel(R) Xeon(R) CPU X5660 @ 2.80GHz | 1652.740 | 12288 KB | 6 |
| veuc08 | | | 1606.261 | | |

Table 1: Specifications of the CPUs in CALCULA machines

## 1.3 Methods and procedures

The main idea of this project was originally proposed by my supervisors and it didn't come from any previous work. It is a combined effort between me and Magdalena Biesialska, who carried out the implementation of the structure of the system and helped me during the project.

In this project, the structure of the system is based on the BEAT platform [Anjos et al.2017], an open platform for research in computational sciences related to pattern recognition and machine learning, to help on the development, reproducibility and certification of results obtained in the field.

Also for the QE sampling strategy is based on the QE model implementations in the Openkiwi framework [Kepler et al.2019]. This framework implements the best QE systems from WMT 2015-18 shared tasks. Also, the ALPS sampling strategy is based on the implementation by Michelle Yuan [Yuan et al.2020], which tries to address both uncertainty sampling and diversity sampling.

The majority of the code during this project was written in Bash scripting, along with parts on Python and yaml/yml files.

## 1.4 Work Plan

The project was structured in the Work Packages listed below and illustrated in the Gantt Diagram.

- WP 1: Project proposal and work plan
- WP 2: Related Work
- WP 3: BEAT platform preparation
- WP 4: Data preparation
- WP 5: QE implementation
- WP 6: ALPS implementation
- WP 7: BEAT Integration
- WP 8: Evaluation
- WP 9: Final Report
- WP 10: TFM presentation

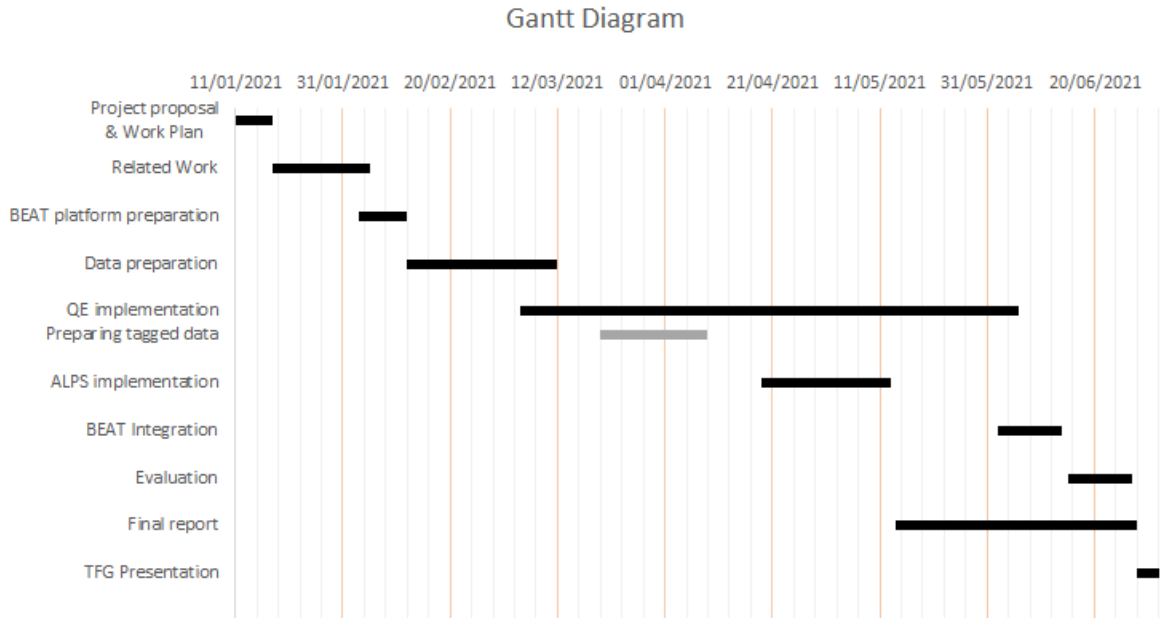



Figure 1: Gantt Diagram

## 1.5 Incidences

We had an incident as an unknown error popped up when training the QE model using the data from one of the language pairs, which did not allow us to continue in that direction. While we were working on other approaches, we tried later and it worked correctly, this delayed the work on the language pair for a considerable time. Additional time delays were encountered when the CUDA Memory error ocurred, and trainings had to be repeated with lower batch sizes, which delayed even more all the process.

Additionally training the QE models and the integrated training on BEAT took more than we expected and we were a little short-timed at the end, which did not allow us to implement the ALPS integration on BEAT nor obtain the final results with it too.

# 2 Background

In this section, we overview the neural-based MT architectures that we use in this study and the evaluation metrics used.

## 2.1 Artificial Neural Networks

Artificial neural networks (ANN) are a type of machine learning algorithm inspired by the functioning of biological neurons in animal brains. Such systems "learn" to perform tasks by considering examples, generally without being programmed with any task-specific rules.
They normally consist of a group of basic processing units called artificial neurons (AN) or perceptrons (Figure 2) which are wired together in a complex communication network. They can compute an output given input data by decomposing it in different representations in order to identify different characteristics.

Each AN model is a simplified model of a real neuron, which sends off a new signal if it receives strong enough input signal from the other nodes to which it is connected, allowing it to perform some basic operations such as AND, OR or NOR.
This model was first proposed by Warren McCulloch and Walter Pitts in 1943 [McCulloch and Pitts1943]. Among many other proposed models, the most simple AN architecture was the perceptron [Rosenblatt1961], which improves the usage of binary values for the McCulloch and Pitts model to be operational with any numbers. It works through an algorithm that computes the so-called activation function:

$$ouput = f(\sum_{\forall i} w_i x_i + b) \tag{1}$$

Where $w_i$ are the weights of the input values $x_i$ and $b$ is a bias, used to give some extra degree of freedom. These values are computed using gradient descent techniques [Barzilai and Borwein1988], which idea consists of taking proportional steps to the negative of the gradient of the function iteratively at the current point. So the result approaches the global minimum of the function.

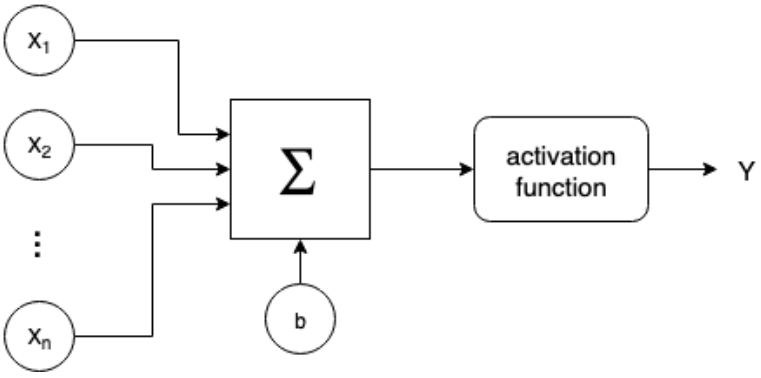



Figure 2: Structure of a perceptron

The output ($Y$ in Figure 2) has an internal threshold, so the output values of the perceptron are binary and they depend on if the output of the activation function exceeds or not this threshold. This allows the perceptron to linearly separate samples into two classes, that is why it can compute basic operations like AND or OR, but not non-linear separable functions or problems like XOR (Figure 3).

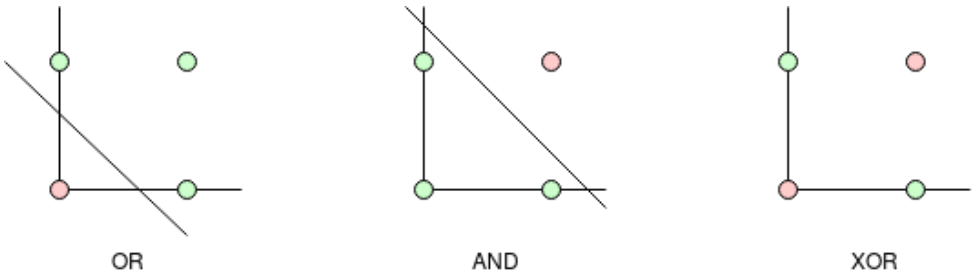



Figure 3: Operations AND, OR and XOR as linear separation problems

To solve this more complex structures are needed. One basic neural network structure, among many others, is the Multilayer Perceptron (MLP), which basically consists of multiple layers of perceptrons.
The basic structure of MLP consists of 3 layers of nodes: the input layer feeds all the data to the hidden layers for multiple representations of data and characteristic identification, and finally the output layer, which can use a different activation function depending on the nature of the task.

### 2.1.1 Recurrent Neural Network

Recurrent Neural Network (RNN) is a type of ANN formed by a sequence of concatenations of the same unit along a temporal sequence (Figure 4), this structure allows the multiple units to retain information from previous data like a temporal memory. This approach is used in the majority of NMT models today and it was based on David Rumelhart's work in 1986 [Rumelhart et al.1986].

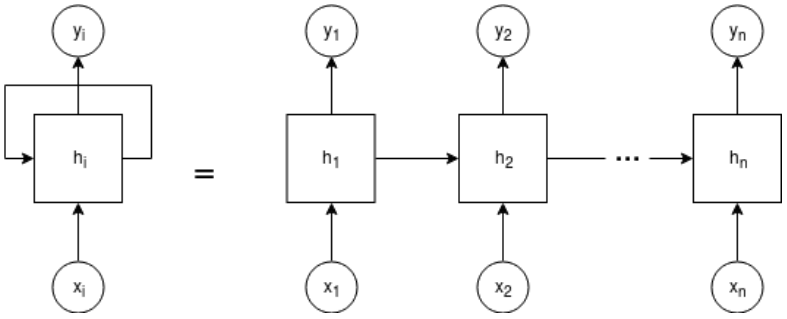



Figure 4: Diagram of an RNN

Due to their capacity to retain information, RNNs are pretty useful for sequential data, where each element of the sequence could be related to others.
One of the biggest issues with RNN in NMT is the vanishing gradient problem [Hochreiter1991], which means that these models are losing more information as "time" passes.

### 2.1.2 Long Short-Term Memory Neural Network

To try to soothe the problem that was vanishing gradient, a new model based on the RNN but with improved long term dependencies was designed: the LSTM's [Hochreiter1997]. This model is, still nowadays, of the best and most used models within NMT.

While vanilla RNN is based on a concatenation of a simple module, usually with only one operation layer, the LSTM model uses a more complex module, as shown on Figure 5, whose internal structure is based on the ideas of "memory cell" and gates, with four operation layers. This architecture lets LSTM learn longer-term dependencies.

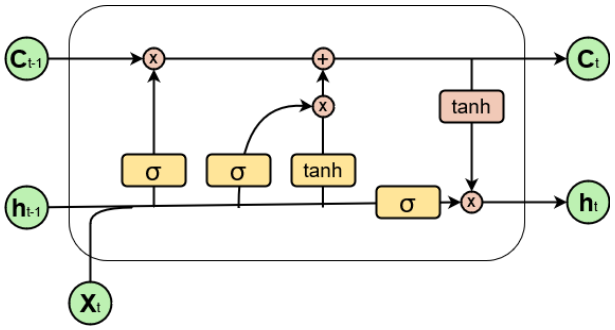



Figure 5: Internal structure of an LSTM

The cell state is responsible to carry the information through the cells, it could be considered the "memory" of the model. The fact is that the cell state is only affected by minor linear interactions is what allows it to maintain the information for long periods of time/cells.

The gates are responsible to decide which information is maintained, which is forgotten and which is updated in the cell gate, apart from the information that is also exported to the next cells as the hidden state (same as the usual RNN does). These gates are:

- forget gate: is responsible to determine how much information from the cell state ($c_t$) have to be forgotten/removed
- input gate: is responsible to determine how much of the information from the input ($x_t$) has to be saved on the cell state
- output gate: is responsible to determine the information that will be passed to the next cells as the hidden state ($h_t$)

After the data has gone through a cell, the new cell state and hidden state are passed onto the next cell, which carries out the same process.

### 2.1.3 Gated Recurrent Units

A variant of the LSTM is the Gated Recurrent Unit (GRU) [Cho et al.2014], which has a simpler structure, making the GRU computationally more efficient and faster to train. Similar to the LSTMs, GRUs use the cell state although while the LSTMs use 3 gates and 3 inputs and 2 outputs, the GRU only consists of 2 gates and the hidden state is removed, the cell state acts as such, as it can be seen on Figure 6.

About the gates, the GRUs consist of an update gate, which determines if the cell state should be updated with the state candidate (the actual activation value) or not; and the reset gate, which determines if the information of the previous cell state is important or not. The reset gate is not used on the most simple GRUs.

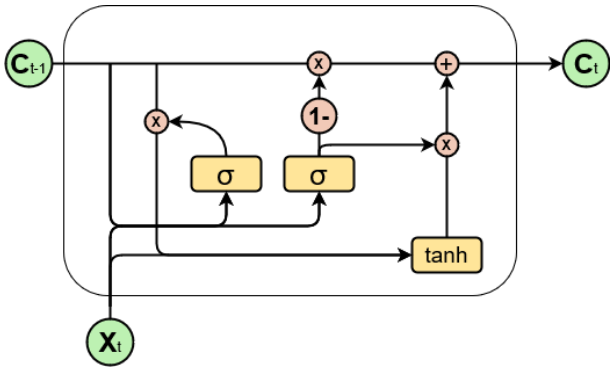



Figure 6: Internal structure of a GRU

As commented, the reduction of complexity allows to train more quickly but this also carries a decreased expressibility and flexibility.

## 2.2 Neural Machine Translation

Neural Networks for MT scientific papers started appearing around 2014 [Bahdanau et al.2014], and have been very successful since then, helped by a great number of advances in recent years. The first appearance of NMT systems in an MT public competition was in 2015 (OpentMT'15). And at the OpenMT'16 the following year, 90% of the winners were NMT systems [Bojar et al.2016] showing a similar or even better performance than the Statistical MT systems [Kalchbrenner and Blunsom2013, Cho et al.2014, Sutskever et al.2014, Bahdanau et al.2014, Sennrich et al.2016a, Zhou et al.2016, Wu et al.2016].

NMT derives from SMT phrase-based approaches [Wołk and Marasek2015] and uses large artificial neural networks, its great difference is the use of vector representations or embeddings for words and internal states. The structure is more simple than phrase-based models since there is no separation among the Language Model, Translation Model and Reordering Model, just an end-to-end sequence-to-sequence model that predicts one word at a time. However, this sequence predictor is conditioned by the entire source sequence and the translated part.

One of the most predominant NMT model used, and one we will be using in this project, is the bidirectional RNN. This RNN consists of an encoder, which is used to encode the source sentence, and a decoder, which does the word prediction in the target language [Bahdanau et al.2014]. This two consists of LSTM [Hochreiter1997] or GRU [Cho et al.2014] (Section 2.1.2) units. Additionally, they are combined with an attention mechanism [Luong et al.2015].

Other approaches are the self-attention Transformer [Vaswani et al.2017], which is almost as predominant as the LSTMs, or the less usual Convolutional Neural Networks (CNN) [Kalchbrenner et al.2016, Gehring et al.2017].

Since 2016, the majority of the best MT systems are using Neural Networks [Bojar et al.2016] such as Google, Microsoft, Yandex, among other translation services. An open source neural machine translation system, OpenNMT, has been released by the Harvard NLP group [Klein et al.2018].

## 2.3 Evaluation

### 2.3.1 Correlation Measures

A correlation coefficient measures the extent to which two variables tend to change together. The coefficient describes both the strength and the direction of the relationship [2].

With the two correlations explained in this section, the relationships measured can be linear (Pearson) and monotonic (Spearman); but other relationships can be possible on the data and so it is recommended to examine visually the associations between variables in a graph.

#### 2.3.1.1 Pearson Correlation

Pearson correlation coefficient ($r$) [Pearson1895] measures the linear relationship between two continuous variables. This coefficient indicates the strength and direction of these correlations.

The formula of Pearson correlation coefficient is defined as [3]:

$$r = \frac{cov(X, Y)}{\sigma_X \sigma_Y} \tag{2}$$

where $cov()$ in the numerator indicates the covariance between both variables, while the sigmas in the denominator indicate the standard variance of the variable.

The possible values for this coefficient can range from -1 to +1. The sign of the coefficient indicates the direction of the relationship, positive values indicate a tendency to increase or decrease the value of both variables together proportionally. Meanwhile, a negative value indicates a tendency to decrease proportionally a variable when the other increases and vice versa. On the other hand, the magnitude of the coefficient indicates the strength of the relationship, a value near 0 indicates a weak linear relationship between the variables, while an absolute value (|x|) near 1 indicates a strong relationship between variables.

#### 2.3.1.2 Spearman Correlation

Spearman correlation coefficient ($\rho$) [Spearman1904] measures the monotonic relationship between the rank variables of two continuous or ordinal variables. In a monotonic relationship variables tend to change together but unlike a linear relationship, in the monotonic case they do not have to change in a constant ratio; in a simple way, in a perfect monotonic relationship each point with a higher value on the X-axis than a particular point will also have a higher value on the Y-axis.

[2] https://support.minitab.com/en-us/minitab-express/1/help-and-how-to/modeling-statistics/regression/supporting-topics/basics/a-comparison-of-the-pearson-and-spearman-correlation-methods/

[3] https://mathworld.wolfram.com/StatisticalCorrelation.html

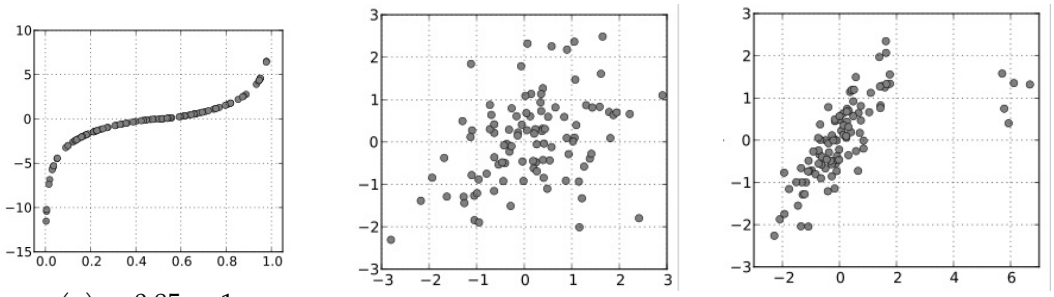
Unlike the Pearson case, Spearman does not use the raw data, it is based on the ranked values for each variable. Spearman can also be defined as the Pearson correlation coefficient between the rank variables [4].

The Spearman correlation coefficient formula is defined as [5]:

$$\rho = \frac{cov(rk_X, rk_Y)}{\sigma_{rk_X}\sigma_{rk_Y}} = 1 - 6\frac{\sum d_i^2}{n(n^2-1)} \tag{3}$$

The raw scores converted to ranks are indicated with *rk*, also the *cov*() in the numerator indicates the covariance between the rank variables; and the sigmas on the denominator the standard variance between them. The second variation of the formula can only be used if all the ranks are distinct integers, in it $d_i$ indicates the difference between the ranks in each observation while $n$ indicates the number of observations.

Despite being very similar, Pearson and Spearman's correlations act differently as can be seen in the following examples:



(a) p: 0.85, s: 1

(b) p: 0.35, s: 0.32

(c) p: 0.63, s: 0.83

Figure 7: Cases for comparison with Pearson (p) and Spearman coefficients (s)

In the first case, we can observe a perfect monotonic relationship, although it is not linear. This makes the value of Spearman be 1 while the Pearson value is a little lower. In the second case, we can observe how the more elliptical distributed data without outliers makes both correlations yield similar low values. Finally, in the last case, we can observe why the Spearman correlation is considered more robust against outliers than Pearson. This is due to how Spearman limits the outliers to their rank value.

### 2.3.2 BLEU score

Imagine you have a Spanish sentence, and you are given a human-generated translation of it as a reference. But there could be multiple sentences considered perfectly good translations of that Spanish sentence.

[4] https://archive.org/details/researchdesignst00jero$_9$35

[5] https://statistics.laerd.com/statistical-guides/spearmans-rank-order-correlation-statistical-guide.php

The Bilingual Evaluation Understudy or BLEU [Papineni et al.2002] is a method to evaluate the quality of a machine-translated text. The basic idea behind BLEU is that, the closer a machine translation is to a professional human translator, the better it is. BLEU allows using more than one reference, which allows better robustness.

In order to illustrate better how the BLEU score works we will use an example, with the Spanish sentence ***El gato está en la alfombra*** as the sentence to translate. As a human reference, we have gotten two accepted translations, the first being ***The cat is on the mat***, and the second being ***There is a cat on the mat***.

What the BLEU method does is, given a machine-translated text, it computes a BLEU score that measures how good that MT system is. The more basic intuition behind the BLEU score is to look at the machine-generated output and see if the words it generates appear in the human-generated references. So, if we look at each word in the MT output and see if it appears in the references, we are calculating the precision of the MT output, which is a number between 0 and 1.

However, in order to resolve some deficiencies, it is common to use the modified precision measure in which we will give each word credit only up to the maximum number of times it appears in the reference sentences. Also, as we do not want to just look at isolated words, we will look at n-grams (a group of n consecutive words).

And so the algorithm goes as follows:

- First, we will count the number of distinct n-grams in the candidate.
- Then we will count the number of times each n-gram $g_i$ occurs in each reference. But we will only take the maximum of each of these values calculated.
- Finally, we will add the maximum calculated previously of each n-gram, and divide them by the total number of n-grams in the candidate (this time they do not have to be distinct).

So to start with the example, we get the slightly good translation ***The cat the cat on the mat*** as an MT output, and we will evaluate it using bi-grams.

Candidate: The cat the cat on the mat
Reference1: The cat is on the mat
Reference2: There is a cat on the mat

| bigrams | max count | sum of max counts | appearences | total of bi-grams | score |
|---|---|---|---|---|---|
| the cat | 1 | 4 | 2 | 6 | 4/6=2/3=0.66 |
| cat the | 0 | | 1 | | |
| cat on | 1 | | 1 | | |
| on the | 1 | | 1 | | |
| the mat | 1 | | 1 | | |

Table 2: Results obtained for the example candidate

So, in this example, we get that the sum of each bi-gram maximum number of appearances in the references is 4. And, although we have 5 distinct bi-grams in the candidate, the number of total bi-grams is 6 as ***the cat*** appears twice. In result, we obtain a score of 4/6 or 2/3.

So we could compact all of this in the formula 4 where the subscript $n$ indicates for what number of n-grams are we calculating and $y$ is the MT output or candidate. :

$$p_n = \frac{\sum_{n-grams \in y} \text{max count of appearances in reference}}{\sum_{n-grams \in y} \text{total n-grams in candidate}} \tag{4}$$

Finally, to obtain the final BLEU score, we calculate the Combined BLEU score (Formula 5) which is the value of all the n-grams modified precision. Where we basically exponentiate $e$ by the mean of values from unigrams to 4-grams and multiply it by BP, which stands for brevity penalty (Equation 6) and is an adjustment factor that penalizes translation systems that output translations that are too short.

$$score = BP * exp(\frac{1}{4}\sum_{n=1}^{4} p_n) \tag{5}$$

$$BP = \begin{cases} 1 & \text{if } length_y \leq length_{reference} \\ exp(1 - \frac{length_y}{length_{reference}}) & \text{otherwise .} \end{cases} \tag{6}$$

BLEU score was revolutionary for machine translation because it gave a pretty good single real number evaluation metric, although by no means perfect.

In practice, BLEU was one of the first metrics to claim a high correlation with human judgements of quality [Coughlin2003] and remains one of the most popular automated and inexpensive metrics. And so there are multiple open source implementations that you can download and use to evaluate your own system (Moses multi-bleu.perl script, NIST mteval-vXX.pl script, etc), but it is recommended to stick with only one per project since they have different implementations and their results may differ between them.

# 3 Related Work

A little overview of previous works related to the project will be done in this section.

As mentioned in the introduction, there are various works related to lifelong learning in different fields within the machine learning community such as image generation with lifelong GANs [Zhai et al.2019b] or autonomous agents in real-world data [Thrun and Mitchell1995]. Despite this, there is no work on which are evaluated the lifelong learning systems in the machine translation field. What there is, is a variety of related tasks studies that are useful in addressing the goal of lifelong learning.

For example, different types of adaptations have been studied such as domain adaptation, which is based on the premise that the system can adapt to a previously known target domain. This has been widely studied for Statistical MT [Koehn and Schroeder2007] and more recently for Neural MT [Luong and Manning2015]. Another type of adaptation studied is the instance-based adaptation, which exploits the similarity between the inference and training instances both in supervised [Li et al.2018] and unsupervised scenarios [Farajian et al.2017]. It is important to note that in these tasks there is no target domain data available. The studies in instance-based have even allowed the creation of an adaptative MT commercial toolkit [Federico2018].

Additionally, different techniques have been studied depending on the availability of the target language translated sentences serving as ground truth. The typical case is the one where we have all the target data in a parallel corpus, but this is pretty useful in the case of lifelong learning as normally we do not have the target sentences of the processed data.

On the other hand, we have cases where we do not have any translated target sentences and we are limited on the human-level translations we can obtain before training the model. Some studies tackle the case where there is not a single piece of target ground truth. In those cases, in order to train the translation model, unsupervised learning is used and, due to the lack of any parallel data, they are only relying on the usage of monolingual data ([Artetxe et al.2018], [Lample et al.2018]).

Other studies address the case where we have a limitation on the number of translated sentences we can obtain from human translators. In this case, active learning aims at selecting the most useful sentences within the available monolingual source text and query their translation. Therefore, this selection needs to minimize the number of times the human translation is requested while maximizing the improvement that this represents on the model [Liu et al.2018]. Additionally, in a similar way as active learning, there is interactive learning which through the collaboration between the machine translation system and a human tries to obtain the best translation quality while reducing the effort of the person in the process [Peris et al.2016].

# 4 Methodology

In this section we will define the systems, structures, datasets and toolkits used to design and use the lifelong learning systems during the project.

## 4.1 System Structure

To develop and evaluate the lifelong learning system we will need a structure that allows us to do so, which can be seen in Figure 8. This is the same as given by the LLMT task on the WMT21 conference [6]. In this structure, the lifelong learning process only depends on the target data, so neither domain nor instance-based adaptation will be used.

Additionally, this target data (as commented with more detail in section 4.3), consists of monolingual data which does not include the target translation ground truth, although some translations can be obtained through a human translator. This resembles a situation commented in section 3 which is usually tackled using the Active Learning approach. Following the example, we will implement and evaluate some of the active learning methods as seen in section 4.2.

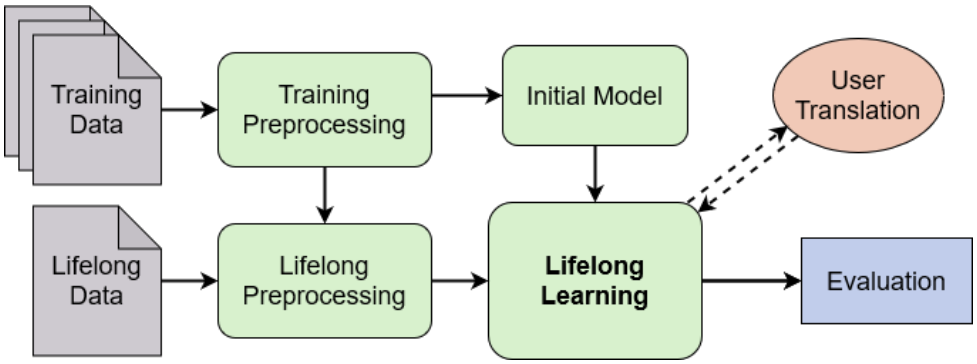



Figure 8: Structure of the complete system

### 4.1.1 Four Blocks System

The structure of the system consists of 4 differentiated blocks:

**Training Preprocessing** This block is responsible to prepare the training data and define the source and target vocabularies. The preprocessing can include tokenization, learning subword decomposition model, etc. Once the data is preprocessed, it is sent to the initial model block while the subword model and vocabularies are sent to the lifelong preprocessing block.

**Lifelong Preprocessing** The objective of this block is to preprocess the given lifelong corpus the same way as the training data using the subword models, vocabularies and other processes the training preprocessing block may have used. It has to be noted that the lifelong corpus documents are received one by one. The preprocessed documents will be sent to the lifelong learning block in order to be processed.

[6] http://statmt.org/wmt21/lifelong-learning-task.html

**Initial Model** In this block, the initial translator model will be trained from the training data preprocessed previously. This model will be the one modified by lifelong learning to adapt to the new data. The resultant model obtained in this block will be sent to the lifelong learning block.

**Lifelong Learning** This block tries to translate the sentences delivered by the lifelong preprocessing block using the model from the initial model, which can be modified. In this, we will plug the adaptation schemes implemented from section 4.2.

This block has access to all the training data, along with the possibility to store in memory the documents already processed for later use. Additionally, it also has the possibility to communicate with the "external" user translation block (section 4.1.2) and ask for the translation of sentences with the objective to obtain the best output result. The intention is that these sentences can be used to train the model previous to the document translation. Finally, each document translation obtained by this block will be passed to the evaluation block (section 4.1.3).

### 4.1.2 User Translation

This section consists of a human translating and returning the queried sentences translated. In our case though, the person will be simulated and the translations of the queried sentences will be returned automatically. As the original data used for the lifelong data is a parallel corpus, the sentences in the target language from these documents will be the translations returned.

It has to be noted that this process is arduous and involves a notorious increase in the processing time, as such it has to be considered the number of times that the human help has been requested with a penalization on the evaluation score obtained, as explained on the evaluation block on the next subsection.

### 4.1.3 Evaluation

This block evaluates the results of the system implemented. The final results consist of three scores $S_{sys}$, $S_w$ and $S_c$ [Prokopalo et al.2020].

$$S_{final} = S_{sys} + (S_w - S_c) \tag{7}$$

The first score ($S_{sys}$) is the modified n-gram precisions BLEU. The other two make up for the penalization score which penalizes to a certain extent the fact of having used human help (Section 4.1.2). These two are the values of the results of the system if we consider that every sentence queried to the user translation is completely wrong ($S_w$) and if we consider all the sentences queried to the user translation completely correct ($S_w$).

## 4.2 Active Learning Approaches

During the development of this project, two method implementations will be carried out with the goal to select the sentences which will be sent to the user translation during the lifelong learning block. Both methods along the baseline will be developed below:

### 4.2.1 Baseline Method

We consider the random selection of the sentences as the base method to which we will compare the results obtained by our two methods proposals.

### 4.2.2 Quality Estimation Method

The idea is to select the worst machine translated sentences as judged by a Quality Estimation model to query them into the human translator, so the model learns from where it is weakest. This method requires the machine-translated sentences from the training sentences to train and it will be done using OpenKiwi. With this framework we will train the QE model which will try to predict the HTER scores [Specia et al.2018] of the machine-translated sentences at a sentence level without any reference file, which will allow us to select the sentences with lower value to send translation requests to the user simulation block.

As commented, to train the QE model we will be using OpenKiwi, a Pytorch-based open-source framework for QE [Kepler et al.2019]. The model used will be based on the Predictor-Estimator architecture [Kim et al.2017], which consists of a predictor, trained to predict the token given the source and right and left context of the target sentence; and an estimator which uses the source sentences and the features produced by the predictor to classify each word as OK or BAD. Our goal is to obtain a prediction of the HTER score, so the model will be trained on a multi-task architecture that allows us to predict the sentence-level HTER scores.

As defined more in detail by Fabio Kepler in [Kepler et al.2019]: "Our predictor uses a bidirectional LSTM to encode the source, and two unidirectional LSTMs processing the target in left-to-right (LSTM-L2R) and right-to-left (LSTM-R2L) order. For each target token $t_i$, the representations of its left and right context are concatenated and used as a query to an attention module before a final softmax layer. /.../ The estimator takes as input a sequence of features: for each target token $t_i$, the final layer before the softmax (before processing ti), and the concatenation of the $i_{th}$ hidden state of LSTM-L2R and LSTM-R2L (after processing $t_i$)."

### 4.2.3 ALPS Strategy

A Machine Learning model usually returns a value of the confidence it has along the results obtained. The objective of uncertainty sampling is to send to the user to obtain feedback the sentences where the model is less confident about the result obtained, being it good or bad [Munro2019b].

On the other hand, when you train a model, you want it to be as good in generalizing as possible when the real data is inputted, and to do so is very important to have diverse training data. The objective of diversity sampling is to assure that the training data used is diverse [Munro2019a].

The ALPS strategy [Yuan et al.2020] tries to tackle both uncertainty sampling and diversity sampling from a simple strategy based on the masked language modelling loss. It makes use of Masked Language Modeling using BERT as the evaluation of uncertainty. MLM consists of randomly masking some of the input tokens and making the model predict those missing tokens, using BERT it makes use of the context from both left and right sides to make predictions, as it is bidirectional. The Language Modeling Loss consists of computing the Cross-Entropy Loss for the masked tokens, and it will be used as a proxy for the uncertainty evaluation. These same losses are used for the diversity sampling, as it makes use of unsupervised clustering,

such as k-means, producing as many clusters as queries to the user translation and selecting the sentences with the losses nearer the centroids.

## 4.3 Datasets

**Training and Lifelong Data** The data used to train the system will have to be delivered to two preprocessing blocks as explained in section 4.1.1, to do so the data has also been divided into two sets.

The data handed to the training preprocessing block will be called training data. This data consists of a set of documents provided on the WMT tasks between 1996 and 2013, the documents are coming from the Europarl [7] and the NewsCommentary [8] datasets on the corresponding years and also contain the complete translations of all the documents to the target language. This data is available at any stage of the system.

On the other hand, the data provided to the lifelong preprocessing block, which will be called lifelong data, consists of the sets of the same documents as in the training data but between the years 2014 and 2020. This set represents data recollected from the real world and so they are monolingual and do not contain any kind of translation to the target language. Additionally, the documents will be delivered to the system one by one.

**QE Data** As commented in Section 4.2.2, to train a QE model a minimum of 4 files are necessary: source, target, machine translated and TER.

As source and target files, the same training data files for the system will be used as explained in section 4.1.1. For the machine translated file it will be necessary to translate the source data with a model. Finally, the TER file will be obtained comparing the target and MT file with the help of the SacreBLEU library [Post2018].

[7] https://opus.nlpl.eu/Europarl-v3.php
[8] https://opus.nlpl.eu/News-Commentary.php

# 5 Implementation

In this section, we report details about the data and preprocessing along with the steps and parameters used for the implementations.

## 5.1 Data and Preprocessing

### 5.1.1 Data

The data used consists of the training parallel data delivered by WMT from 1996 until 2020, which includes the Europarl and NewsCommentary datasets for each year. The data from 1996 until 2013 is separated into the training data, and the rest is considered the lifelong data (Section 4.3).

Regarding the training data we used to train the initial model and QE model, being parallel it is separated into two files: source file with the sentences in the source language, and target file with the sentences in the target language. This target file will be considered the post-edited file (PE), the machine-translated file which has been modified by a human to achieve an acceptable translation level.

Due to the lack of an initial model at that moment and to speed up the process, to obtain the machine-translated (MT) file necessary for the QE model, the Marian NMT models provided by the University of Helsinki will be used for both pair of languages using HuggingFace [9].

| Language Pair | # Sentences | # After Cleaning |
|---|---|---|
| EN-DE | 2.455.103 | 2.312.970 |
| EN-FR | 2.516.099 | 2.386.075 |

Table 3: Number of sentences for each pair of languages before and after cleaning them.

Each of these 3 files passes through a cleaning process which consists of removing empty sentences or those with less than 4 letters/symbols (Table 3). These cleaned files are split into train, dev and test files with proportions 0,7/0,15/0,15 respectively (Table 4). Additional sentences for each file will be removed on the later preprocessing for the alignment (Section 5.3.1).

| Language Pair | Set | # Sentences | # After Removal |
|---|---|---|---|
| EN-DE | Train | 1.619.585 | 1.587.586 |
| | Dev | 346.462 | 339.495 |
| | Test | 346.923 | 346.923 |
| | TOTAL | 2.312.970 | 2.274.004 |
| EN-FR | Train | 1.671.338 | 1.554.578 |
| | Dev | 357.856 | 332.701 |
| | Test | 356.881 | 356.881 |
| | TOTAL | 2.386.075 | 2.244.160 |

Table 4: Number of sentences for each set of data before and after removing some sentences for the alignments.

[9] https://huggingface.co/transformers/model_doc/marian.html#multilingual-models

Additional input files for the training on QE models are the sentence-level TER values between the target and the MT file, obtained using the SacreBLEU library [Post2018]; and the post edited labels/tags, obtained as explained in Section 5.3.1.

### 5.1.2 Preprocessing

The preprocessing carried out on both training and lifelong data consist of normalization, tokenization and application of BPE.

With normalization and tokenization, we use the Moses tokenizer [Koehn et al.2007] and we are referring to separate the sentence, having each word and punctuation mark separated by whitespaces. As for BPE (Byte Pair Encoding) [Sennrich et al.2016b], it allows us to separate the words into smaller subunits to help the model encode strange or unseen words. To do that we use the subword-nmt repository [10].

## 5.2 System

### 5.2.1 Initial Model

The model trained as the initial model on the system consists of a 2-layer bidirectional GRU [Cho et al.2014] encoder and a 2-layer conditional GRU decoder [Sennrich et al.2017] equipped with an attention mechanism [Bahdanau et al.2014] as implemented in nmtpytorch [Caglayan et al.2017].

### 5.2.2 User Translation

In order to select the number of sentences queried to the simulated user we took the approach to query 10% of the total number of sentences in the document.

## 5.3 Adaptation Methods

### 5.3.1 QE Models

For the QE model training, various implementations are carried out with the objective to obtain the best possible results. First, it is tried to train the Predictor jointly or separately from the Estimator, and the best of those two cases will be the base for the following different implementations.

In the case of the EN-DE language pair, it is also tested to use a pre-trained OpenKiwi model. In order to be able to use the pre-trained model, it is needed the same input data as the original model which in this case includes the files with the post-edited labels, so it is necessary to generate these files first as explained in section 5.3.1. In the EN-FR case, we do not have any available pre-trained model therefore we do not have tried this approach with this pair of languages.

[10] https://github.com/rsennrich/subword-nmt

Continuing with the case for both languages pairs, once the results are obtained and the better base model is selected, different implementations are tested using different configurations of input data to train the model. On the base cases only the src, MT and TER files are used as input, in the following cases we add the PE file first, then the post-edited labels files and finally both are added simultaneously.

Once found the best input configuration, hyperparameter tuning is performed to try to improve the model obtained. The results for all the experiments can be found in the Results Section 6.

#### Post-Edited Labels

To obtain the post-edited/tags files, we use the word-level-qe-corpus-builder repository [11], an implementation of the updated version of the WMT word-level quality estimation task [Bojar et al.2017] that takes into account both fluency and adequacy issues. In it, first, a fast-align model [Dyer et al.2013] is trained, we use a random selection of 100.000 sentences from the training file. Then using this trained model, we obtain the src-PE and src-MT alignments. The PE-MT alignments are obtained using the Tercom tool [12] [Snover et al.2006] and finally the OK/BAD tags are generated.

#### Parameters

When using Openkiwi to train the base predictor and estimator in general we used the default parameters for it: a dropout of 0 for both predictor and estimator, two layers, 5 epochs, a hidden size of 125 for estimator and 100 for predictor LSTMs, an embedding size for both source and target of 50, an embedding size of 200 at the output. Additionally for the training, we used an Adam optimizer, a learning rate of 1e-3, a train and eval batch sizes of 16 and defined the seed as 42 using the wmt18 format and only the sentence level prediction as true.

The number of parameters and the architectures for the three models: Predictor when trained separately, joined Predictor Estimator and Finetuned Estimator can be seen on Appendix 8.1.2. Meanwhile, for the other cases, the parameters that change are only the input files or one of the parameters specified, the structure does not change. For the hyperparameter tuning the parameters which changed are the hidden size for 512, dropout for 0,2, the embedding size for 512 and the learning rate for 1e-4 and 1e-5.

### 5.3.2 ALPS Models

In the ALPS case, as commented on 4.2.3, it is used the Michelle's Yuan implementation [13] adapted to return the indexes of the selected input file sentences.

#### Parameters

In ALPS we used in general the default parameters for it which include using the BERT base uncased model with seed 42, lifelong as the task name, no cold start and ALPS as sampling. We used 16 and 8 as the train and eval batch sizes respectively. Also, 128 is defined as the maximum sequence length.

[11] https://github.com/Unbabel/word-level-qe-corpus-builder
[12] http://www.cs.umd.edu/ snover/tercom/
[13] https://github.com/forest-snow/alps

# 6 Results

This section will include findings and little analysis of the data collected.

First, there is the obtention of results related to the QE model training, which is evaluated using Pearson and Spearman Correlation. The results obtained by both approaches for the predictor-estimator model (predictor incorporated or trained externally) in both pair of languages can be found in Table 5. In it can be seen what will be recurring during all the results, the results in EN-FR will be better than in EN-DE, probably due to the more complexity on the translation and evaluation on the latter pair.

| Model | Language | Pearson | Spearman |
|---|---|---|---|
| Joint Predictor | EN-FR | 0,3325 | 0,3737 |
| Predictor->Estimator | EN-FR | 0,3113 | 0,3464 |
| Joint Predictor | EN-DE | 0,3130 | 0,3193 |
| Predictor->Estimator | EN-DE | 0,2549 | 0,2488 |
| Finetuned | EN-DE | 0,3300 | 0,3541 |

Table 5: Results obtained training the Predictor jointly/separately from the Estimator and the finetuning.

Also can be observed that in both cases the results obtained by the predictor trained jointly with the estimator obtains better results, this follows logic as this training makes the most of the connection between them, making the predictor pass better features to the estimator or making the estimator interpret better the features passed by the estimator. Additionally, as commented in Section 4.3, in Table 5 there are the results obtained by finetuning the model only for the EN-DE case, as we lack of a pretrained model for EN-FR. This finetuning results are better than the ones obtained by the predictor estimator trained from zero.

| Inputs | Language | Pearson | Spearman |
|---|---|---|---|
| Base | EN-FR | 0,3325 | 0,3737 |
| | EN-DE | 0,3130 | 0,3193 |
| Base + pe | EN-FR | 0,3277 | 0,3582 |
| | EN-DE | 0,2705 | 0,2797 |
| Base + tags | EN-FR | 0,3536 | 0,3996 |
| | EN-DE | 0,3086 | 0,3071 |
| Base+pe+tags | EN-FR | 0,3114 | 0,3426 |
| | EN-DE | 0,2769 | 0,2667 |

Table 6: Results obtained with the different input data.

Regarding the results obtained with the different kinds of inputs used can be found in Table 6 [14]. In it can be observed that, while from results obtained on the EN-DE pair, the better one is still the one where the minimum necessary source, MT and TER files are given; on the EN-FR case the addition of the tags files has supposed an increase of the results obtained. Despite this, the case where more data is given is the worse on both pair of languages, which can be surprising as usually as more information it has, better predictions can be done, such as the results with

[14] In this table we refer the usage of source, MT and TER files for input as Base

tags files with EN-FR.

Table 7 shows the results obtained when modifying the value of some of the parameters with the intention to obtain an even better model, in it can be observed that the better results are obtained when reducing the learning rate until 1e-5, but reducing it more decreases considerably the results. On other cases, no improvement has been achieved. Due to the limited time, the hyperparameter tuning has only been focused on the EN-FR case, assuming similar results on the EN-DE case.

| Hyperparameters | Value | Pearson | Spearman |
|---|---|---|---|
| learning rate | 1e-4 | 0,3682 | 0,4139 |
| | 1e-5 | 0,3765 | 0,4214 |
| | 1e-6 | 0,2858 | 0,3288 |
| hidden size | 512 | 0,2996 | 0,3309 |
| dropout | 0,2 | 0,3396 | 0,3845 |
| embeddings size | 200 | 0,2844 | 0,3040 |

Table 7: Results obtained in hyperparameter tuning for the EN-FR case.

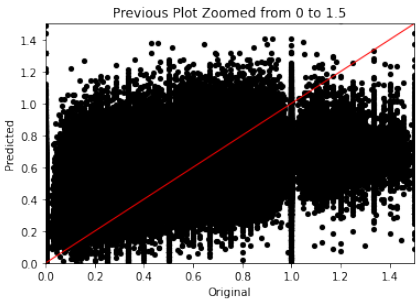
Seen all the previous results it were obtained too, on Table 8, the results of the models trained with learning rate 1e-5 with the double of epochs, to check if the training time increase helps to improve the models even more. The models were the ones which obtained the better results (finetuned in EN-DE and with tag files in EN-FR), and the ones using all the data available, as we hypothesize that the situation where that much data was given the model needed more time to learn from it compared to the other cases, giving us the reason almost matching the results obtained with tags. Seeing this improvement, another run was carried out with 20 epochs to try to improve even more although this time there was not any improvement, it seems that the peak is around the 10th epoch and from there it starts to overfit to the training data. In the case of the finetuning, the model used was pretty big which resulted in CUDA memory errors until the batch size was reduced to 2, which on the other hand increased the training time so much that we did not have time to train the finetuned model for the 10 epochs case, foreseeing this we obtained the results for the other best model on the EN-DE case (base input) for 10 epochs.

| Languages | Inputs | Epochs | Pearson | Spearman |
|---|---|---|---|---|
| EN-FR | Base+tags | 10 | **0,3765** | **0,4214** |
| | Base+pe+tags | 10 | 0,3762 | 0,4208 |
| | Base+pe+tags | 20 | 0,3471 | 0,4048 |
| EN-DE | Base | 10 | 0,3479 | 0,3680 |
| | Base+pe+tags | 10 | **0,3532** | **0,3748** |
| | Base+pe+tags | 20 | 0,3053 | 0,3322 |

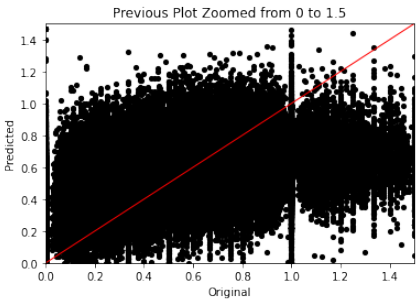
Table 8: Results obtained with learning rate 1e-5 and different inputs and epochs.

Finally, as commented before, it is no good to be guided by the strict value of the correlation values, as they depend on the data distribution [Schober et al.2018]. Therefore a little graphic analysis was carried out (Graphs 9) on the evaluation results for the models in Table 8 along with the finetuned model only for case EN-DE, as the results for the EN-FR are almost identical.

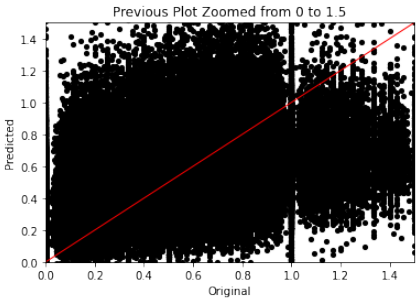
In the graphs can be observed the TER results obtained (X-axis) as explained in Section 4.3, compared to the predicted value by our model (Y-axis). Ideally, they would follow the thin red line, but as can be seen, they do not get much close despite showing some tendency in the most lower-right zone. Nevertheless, these graphs confirm that these QE models will not probably be very good to differentiate between a good or bad translation. The results obtained with the finetuned model are practically the same although the predictions seem to be more restricted in range.



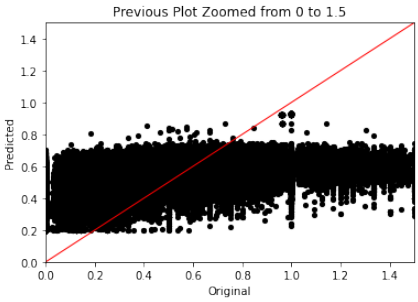

(a) Case with Base Input (10 epochs)




(b) Case with Base+pe+tags Input for 10 epochs




(c) Case with Base+pe+tags Input for 20 epochs




(d) Case with finetuned model

Figure 9: Comparison of TER results between the groundtruths and the predicted for the EN-DE case zoomed for the values between 0 and 1.5

Once treated the QE trainings, two models for each pair of languages have been selected to be tested on the lifelong learning block, being them: the tags input with 10 epochs for English - French and the finetuned model for English - German, along with the models using the base input plus the tags and the pe for both pairs of languages. The modified BLEU results (Section 4.1.3) obtained with both these models and the ALPS method on the lifelong learning block should be seen in Table 9, but, as commented in 1.5, we did not have time to implement the ALPS method within the system scope, so it will be lacking the results on that.

| Lang | Approach | Inputs | Sf |
|---|---|---|---|
| EN-FR | random | – | 26,2 |
| | QE | Base+tags | **26.7** |
| | | Base+tags+pe | 26,4 |

| Lang | Approach | Inputs | Sf |
|---|---|---|---|
| EN-DE | random | – | 15,2 |
| | QE | Base | 15,3 |
| | | Base+tags+pe | **15,9** |

Table 9: Final system results obtained for the different models and languages

These results on Table 9 come from a total set of 3.003 lifelong sentences over all the evaluation lifelong documents and evaluated using the penalized BLEU score as explained in Section 5.2.2. In them, we can observe how the results obtained with the QE models outscore the ones obtained by random sampling.

Even more, looking at Table 8 we can also observe how the QE models with higher performance obtain higher final results, indicating some correlation between the system results and the QE model performance.

# 7 Conclusions and Further Research

In this section we discuss the main findings during our research.

We have shown that the usage of Quality Estimation models improves the results obtained on lifelong learning systems for both English-French and English-German language pairs.

In the case of the QE models we trained, although the correlation scores obtained by us are worse than the scores obtained by original OpenKiwi models([Kepler et al.2019]), we hypothesize that may be due to the fact that we use different datasets with different domains. Therefore, we cannot directly compare our results with the results of state-of-the-art QE models submitted to the WMT QE shared task.

We used documents from news domain, which are provided by the most popular machine translation shared task at the WMT conference and are widely used by the NLP researchers (and the NMT community in particular). This domain is diferent than the one usually provided on the quality estimation shared task, which uses IT-related domain datasets.

In our view, any improvements of our QE models should enable the system to select more useful sentences for active learning and the accuracy of the lifelong learning MT system should improve. We successfully implemented one method to select the number of sentences per document to query to the simulated user. This can be extended to other methods such as query the sentences which are below a certain quality threshold once passed through the QE model. Furthermore, these methods rely on a fixed parameter, either a ratio or a threshold value. This adds to a new parameter that can be finetuned for improvements on the system. As future work it would be interesting to see how other sampling methods perform, such as the ALPS method.

Finally, the general objective of the project was to train an initial MT model, which evolves over time according to the lifelong learning principles. While in our experiments we used QE models that are static, we believe that the results of the lifelong learning MT system can be improved by adapting the QE model to new data that is introduced to the lifelong learning MT system over time.

# 8 Appendix

## 8.1 Costs

In this section, we take into account the cost of the project. We considered both Magdalena Biesialska and Marta R. Costa-Jussà as the Supervisors, and me as the Teamworker.

For the salaries, we assumed that Teamworkers work 20 hours per week, and the project term is 23 weeks. Also assumed that both Supervisors work the same hours per week, also as the Leaders main task was the supervision and they were involved in other projects, they didn't work in it the same number of hours as the Teamworker, 8 hours per week for the full duration. We have to include the social security costs payment, which is a 33.4% rate.

| Description | Quantity | €/hour | h/week | Social Security | Cost |
|---|---|---|---|---|---|
| Supervisor | 2 | 40 | 8 | 4.916,48 | 19.636,48 |
| Teamworkers | 1 | 25 | 20 | 3.841 | 15.341 |
| **Total cost** | | | | | **34.977,48** |

Table 10: Total cost of the salaries

For office expenses, we needed an office. The cost for an already furnished office near our campus is 600€/month, we rented it for 6 months. So, the total cost was 3.600€.
Additionally, our team needed powerful computers to develop the project, one for each one, to work simultaneously. The cost of these computers is approximately $\frac{3*700*0.9}{6} = 315$ for a year, and as we used them for 6/12 = 1/2 of the year $\frac{375*12}{6} = 157,5$.

| Description | Quantity | €/unit | Useful life | Cost |
|---|---|---|---|---|
| Computer | 3 | 700 | 6 | 157,5 |
| **Description** | | **€/month** | **months** | **Costs** |
| Rent office | | 600 | 6 | 3.600 |
| **Total cost** | | | | **3.757,5** |

Table 11: Office expenses cost

As explained in section 1.3, our product was implemented using BEAT platform and Openkiwi framework, which are open-source. The coding of the project was written using Bash scripts. No license was necessary for any of the programs used.

As electricity consumption, we have to take into account the consumption of the office and the computers. We hired Endesa One Luz rate (0.127€/kWh).
As the electricity consumption of the office during the time we used it, is about 30 kWh approximately. The previous calculations include the electricity generated in the laboratory by the lights and other electronic devices but not the computers.
As computer electric consumption, computers use an average of 72 kWh of energy consumption per computer. This sums up to:

$$(72kWh * 3computers + 30kWh) * 0.127€/kWh = 31,24€/month \quad (8)$$

| Description | €/month | months | cost |
|---|---|---|---|
| electricity | 31,24 | 6 | 187,44 |
| **Total cost** | | | **187,44** |

Table 12: Final cost consumption of electricity

Finally, the sum of all the various costs concludes in a total of 38.022,70€, as shown in Table 13.

| Description | Cost |
|---|---|
| Salaries | 34.977,48 |
| Office | 3.757,5 |
| Supplies | 187,44 |
| **Total cost** | **38.919,42** |

Table 13: Final cost of the project

### 8.1.1 Environmental cost

Our product is not material; Consequently, the environmental impact we produce is reduced as there is no need to deal with any potentially harmful substances or exploitation of resources. However, we have to take into account the amount of impact caused by the electricity usage.

We consumed a quantity of:

$$\left(1 computers * 72kWh + 30kWh\right) * 460h + 2computer * 72kWh * 184h = 73.416kW \quad (9)$$

Taking into account a generation of $CO^2$ per electricity consumption of $0,649kgCO^2/kWh$ [15]. And with consumption of 73.416 kWh, we generate:

$$\text{Total kg } CO^2/\text{project} = 0,649kg*CO^2/kWh * 73.416kWh = 47.646,98kg\ CO^2 \quad (10)$$

[15] proposed by IDAE and described in the CALENER GT document, section 3.6

### 8.1.2 Model Architectures

#### Predictor trained separately

```
Predictor(
  (attention): Attention(
    (scorer): MLPScorer(
      (layers): ModuleList(
        (0): Sequential(
          (0): Linear(in_features=1600, out_features=800, bias=True)
          (1): Tanh()
        )
        (1): Sequential(
          (0): Linear(in_features=800, out_features=1, bias=True)
          (1): Tanh()
        )
      )
    )
  )
  (embedding_source): Embedding(45004, 200, padding_idx=1)
  (embedding_target): Embedding(45004, 200, padding_idx=1)
  (lstm_source): LSTM(200, 400, num_layers=2, batch_first=True,
                      dropout=0.1, bidirectional=True)
  (forward_target): LSTM(200, 400, num_layers=2, batch_first=True,
                      dropout=0.1)
  (backward_target): LSTM(200, 400, num_layers=2, batch_first=True,
                      dropout=0.1)
  (W1): Embedding(45004, 200, padding_idx=1)
  (_loss): CrossEntropyLoss()
)
39389601 parameters
```

#### Predictor Estimator

```
Estimator(
  (predictor_tgt): Predictor(
    (attention): Attention(
      (scorer): MLPScorer(
        (layers): ModuleList(
          (0): Sequential(
            (0): Linear(in_features=400, out_features=200, bias=True)
            (1): Tanh()
          )
          (1): Sequential(
            (0): Linear(in_features=200, out_features=1, bias=True)
            (1): Tanh()
          )
        )
```

```
      )
    )
    (embedding_source): Embedding(318370, 50, padding_idx=1)
    (embedding_target): Embedding(308876, 50, padding_idx=1)
    (lstm_source): LSTM(50, 100, num_layers=2, batch_first=True,
                    bidirectional=True)
    (forward_target): LSTM(50, 100, num_layers=2, batch_first=True)
    (backward_target): LSTM(50, 100, num_layers=2, batch_first=True)
    (W1): Embedding(308876, 200, padding_idx=1)
    (_loss): CrossEntropyLoss()
  )
  (mlp): Sequential(
    (0): Linear(in_features=400, out_features=125, bias=True)
    (1): Tanh()
  )
  (lstm): LSTM(125, 125, num_layers=2, batch_first=True, bidirectional=True)
  (embedding_out): Linear(in_features=250, out_features=2, bias=True)
  (sentence_pred): Sequential(
    (0): Linear(in_features=500, out_features=250, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=250, out_features=125, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=125, out_features=1, bias=True)
  )
  (xents): ModuleDict(
    (tags): CrossEntropyLoss()
  )
  (mse_loss): MSELoss()
)
94940679 parameters
```

### Finetuned

```
Estimator(
  (predictor_tgt): Predictor(
    (attention): Attention(
      (scorer): MLPScorer(
        (layers): ModuleList(
          (0): Sequential(
            (0): Linear(in_features=2400, out_features=1200, bias=True)
            (1): Tanh()
          )
          (1): Sequential(
            (0): Linear(in_features=1200, out_features=1, bias=True)
            (1): Tanh()
          )
        )
      )
```

```
    )
    (embedding_source): Embedding(70004, 300, padding_idx=1)
    (embedding_target): Embedding(70004, 300, padding_idx=1)
    (lstm_source): LSTM(300, 600, num_layers=2, batch_first=True,
                    dropout=0.5, bidirectional=True)
    (forward_target): LSTM(300, 600, num_layers=2, batch_first=True,
                    dropout=0.5)
    (backward_target): LSTM(300, 600, num_layers=2, batch_first=True,
                    dropout=0.5)
    (W1): Embedding(70004, 300, padding_idx=1)
    (_loss): CrossEntropyLoss()
  )
  (mlp): Sequential(
    (0): Linear(in_features=1500, out_features=125, bias=True)
    (1): Tanh()
  )
  (lstm): LSTM(125, 125, batch_first=True, bidirectional=True)
  (embedding_out): Linear(in_features=250, out_features=2, bias=True)
  (embedding_out_gaps): Linear(in_features=500, out_features=2, bias=True)
  (sentence_pred): Sequential(
    (0): Linear(in_features=250, out_features=125, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=125, out_features=62, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=62, out_features=1, bias=True)
  )
  (sentence_sigma): Sequential(
    (0): Linear(in_features=250, out_features=125, bias=True)
    (1): Sigmoid()
    (2): Linear(in_features=125, out_features=62, bias=True)
    (3): Sigmoid()
    (4): Linear(in_features=62, out_features=1, bias=True)
    (5): Sigmoid()
  )
  (xents): ModuleDict(
    (tags): CrossEntropyLoss()
    (gap_tags): CrossEntropyLoss()
  )
)
91374630 parameters
```

# 9 Bibliography

[Anjos et al.2017] A. Anjos, L. El Shafey, and S. Marcel. 2017. BEAT: An Open-Source Web-Based Open-Science Platform. *CoRR*, abs/1704.02319.

[Artetxe et al.2018] M. Artetxe, G. Labaka, E. Agirre, and K. Cho. 2018. Unsupervised neural machine translation.

[Bahdanau et al.2014] D. Bahdanau, C. Kyunghyun, and Y. Bengio. 2014. Neural machine translation by jointly learning to align and translate.

[Barzilai and Borwein1988] J. Barzilai and J. M. Borwein. 1988. Two-Point Step Size Gradient Methods. *IMA Journal of Numerical Analysis*, 8(1):141–148.

[Becher1962] Johann Joachim Becher, 1962. *Zur mechanischen Sprachübersetzung. Ein Programmierversuch aus dem Jahre 1661*.

[Biesialska et al.2020] M. Biesialska, K. Biesialska, and Marta R. Costa-jussà. 2020. Continual Lifelong Learning in Natural Language Processing: A Survey. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 6523–6541. International Committee on Computational Linguistics.

[Bojar et al.2016] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, M. Huck, A. J. Yepes, P. Koehn, V. Logacheva, C. Monz, M. Negri, A. Névéol, M. Neves, M. Popel, M. Post, R. Rubino, C. Scarton, L. Specia, M. Turchi, K. Verspoor, and M. Zampieri. 2016. Findings of the 2016 Conference on Machine Translation. In *ACL 2016 First Conference on Machine Translation (WMT16)*, pages 131–198. The Association for Computational Linguistics.

[Bojar et al.2017] O. Bojar, R. Chatterjee, C. Federmann, Y. Graham, B. Haddow, S. Huang, M. Huck, P. Koehn, Q. Liu, V. Logacheva, C. Monz, M. Negri, M. Post, R. Rubino, L. Specia, and M. Turchi. 2017. Findings of the 2017 Conference on Machine Translation (WMT17). In *Proceedings of the Second Conference on Machine Translation*, pages 169–214, Copenhagen, Denmark. Association for Computational Linguistics.

[Caglayan et al.2017] O. Caglayan, M. García-Martínez, A. Bardet, W. Aransa, F. Bougares, and L. Barrault. 2017. NMTPY: A Flexible Toolkit for Advanced Neural Machine Translation Systems. *The Prague Bulletin of Mathematical Linguistics*, 109(1):15–28.

[Cho et al.2014] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

[Coughlin2003] D. Coughlin. 2003. Correlating automated and human assessments of machine translation quality. In *In Proceedings of MT Summit IX*, pages 63–70.

[Dyer et al.2013] C. Dyer, v. Chahuneau, and N. A. Smith. 2013. A Simple, Fast, and Effective Reparameterization of IBM Model 2. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 644–648, Atlanta, Georgia. Association for Computational Linguistics.

[Farajian et al.2017] M. A. Farajian, M. Turchi, M. Negri, and M. Federico. 2017. Multi-domain neural machine translation through unsupervised adaptation. In *Proceedings of the Second Conference on Machine Translation, WMT 2017*, pages 127–137.

[Federico2018] M. Federico. 2018. Challenges in Adaptive Neural Machine Translation. In *Proceedings of the AMTA 2018 Workshop on Translation Quality Estimation and Automatic Post-Editing*, pages 207–242. Association for Machine Translation in the Americas.

[French1999] Robert French. 1999. Catastrophic forgetting in connectionist networks. *Trends in cognitive sciences*, 3:128–135.

[Gehring et al.2017] J. Gehring, M. Auli, D. Grangier, D. Yarats, and Y.N. Dauphin. 2017. Convolutional Sequence to Sequence Learning. *CoRR*, abs/1705.03122.

[Hochreiter1991] Sepp Hochreiter. 1991. Untersuchungen zu dynamischen neuronalen netzen.

[Hochreiter1997] J. Hochreiter, S. Schmidhuber. 1997. Long Short-Term Memory. *Neural Comput.*, 9(8):1735–1780, November.

[Isele2018] D. Isele. 2018. Lifelong reinforcement learning on mobile robots.

[Kalchbrenner and Blunsom2013] N. Kalchbrenner and P. Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

[Kalchbrenner et al.2016] N. Kalchbrenner, L. Espeholt, K. Simonyan, A. van den Oord, A. Graves, and K. Kavukcuoglu. 2016. Neural Machine Translation in Linear Time. *CoRR*, abs/1610.10099.

[Kepler et al.2019] F. Kepler, J. Trénous, M. Treviso, M. Vera, and A. F. T. Martins. 2019. 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. Association for Computational Linguistics.

[Kim et al.2017] H. Kim, J. Lee, and S. Na. 2017. Predictor-Estimator using Multilevel Task Learning with Stack Propagation for Neural Quality Estimation. In *Proceedings of the Second Conference on Machine Translation*, pages 562–568, Copenhagen, Denmark. Association for Computational Linguistics.

[Klein et al.2018] G. Klein, Y. Kim, Y. Deng, V. Nguyen, J. Senellart, and A.M. Rush. 2018. OpenNMT: Neural Machine Translation Toolkit. In *Proceedings of AMTA 2018*, volume 1, pages 177–184, March.

[Koehn and Schroeder2007] P. Koehn and J. Schroeder. 2007. Experiments in Domain Adaptation for Statistical Machine Translation. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 224–227. Association for Computational Linguistics.

[Koehn et al.2007] Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, et al. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th annual meeting of the association for computational linguistics companion volume proceedings*

*of the demo and poster sessions*, pages 177–180.

[Lample et al.2018] G Lample, A. Conneau, L. Denoyer, and M. Ranzato. 2018. Unsupervised machine translation using monolingual corpora only.

[Lee1997] A. Gladwin Lee. 1997. Alan turing, enigma, and the breaking of german machine ciphers in world war ii.

[Li et al.2018] X. Li, J. Zhang, and C. Zong. 2018. One Sentence One Model for Neural Machine Translation. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. European Language Resources Association (ELRA).

[Liu et al.2018] M. Liu, W. Buntine, and G. Haffari. 2018. Learning to Actively Learn Neural Machine Translation. In *Proceedings of the 22nd Conference on Computational Natural Language Learning*, pages 334–344. Association for Computational Linguistics.

[Luong and Manning2015] M. Luong and C. D. Manning. 2015. Stanford neural machine translation systems for spoken language domain. In *International Workshop on Spoken Language Translation*.

[Luong et al.2015] T. Luong, H.Q. Pham, and C.D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *EMNLP*.

[McCulloch and Pitts1943] W.S. McCulloch and W. Pitts. 1943. A logical calculus of ideas immanent in nervous activity. In *Bulletin of Mathematical Biophysics*, volume 5, pages 115–133.

[Munro2019a] R. Munro. 2019a. Diversity Sampling Cheatsheet. *Towards Data Science*.

[Munro2019b] R. Munro. 2019b. Uncertainty Sampling Cheatsheet. *Towards Data Science*.

[Papineni et al.2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

[Parisi et al.2018] German I. Parisi, R. Kemker, Jose L. Part, C. Kanan, and S Wermter. 2018. Continual Lifelong Learning with Neural Networks: A Review. *CoRR*.

[Pearson1895] K. Pearson. 1895. Note on Regression and Inheritance in the Case of Two Parents. *Proceedings of the Royal Society of London Series I*, 58:240–242.

[Peris et al.2016] A. Peris, M. Domingo, and F. Casacuberta. 2016. Interactive neural machine translation. *Computer Speech Language*, 45.

[Post2018] Matt Post. 2018. A Call for Clarity in Reporting BLEU Scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers*, pages 186–191, Belgium, Brussels. Association for Computational Linguistics.

[Prokopalo et al.2020] Y. Prokopalo, S. Meignier, O. Galibert, L. Barrault, and A. Larcher. 2020. Evaluation of Lifelong Learning Systems. In *Proceedings of the 12th Language Resources and*

*Evaluation Conference*, pages 1833–1841, Marseille, France. European Language Resources Association.

[Rosenblatt1961] F. Rosenblatt. 1961. *Principles of Neurodynamics: Perceptrons and the Theory of Brain Mechanisms*.

[Rumelhart et al.1986] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533–536, October.

[Schober et al.2018] P. Schober, C. Boer, and L. Schwarte. 2018. Correlation Coefficients: Appropriate Use and Interpretation. *Anesthesia Analgesia*, 126:1763–1768.

[Sennrich et al.2016a] R. Sennrich, B. Haddow, and A. Birch. 2016a. Edinburgh Neural Machine Translation Systems for WMT 16. In *Proceedings of the First Conference on Machine Translation*, pages 371–376, Berlin, Germany, August. Association for Computational Linguistics.

[Sennrich et al.2016b] R. Sennrich, B. Haddow, and A. Birch. 2016b. Neural machine translation of rare words with subword units.

[Sennrich et al.2017] R. Sennrich, O. Firat, K. Cho, A. Birch, B. Haddow, J. Hitschler, M. Junczys-Dowmunt, S. Läubli, A. V. Miceli Barone, J. Mokry, and M. Nădejde. 2017. Nematus: a Toolkit for Neural Machine Translation. In *Proceedings of the Software Demonstrations of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, pages 65–68. Association for Computational Linguistics.

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

[Spearman1904] C. Spearman. 1904. The Proof and Measurement of Association between Two Things. *The American Journal of Psychology*, 15(1):72–101.

[Specia et al.2018] L. Specia, C. Scarton, and G. H. Paetzold. 2018. Quality Estimation for Machine Translation. *Synthesis Lectures on Human Language Technologies*, 11(1):1–162.

[Sutskever et al.2014] Y. Sutskever, O. Vinyals, and Q.V. Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*, pages 3104–3112.

[Thrun and Mitchell1995] S. Thrun and T. M. Mitchell. 1995. Lifelong robot learning. In *The Biology and Technology of Intelligent Autonomous Agents*, pages 165–196.

[Vaswani et al.2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

[Wołk and Marasek2015] K. Wołk and K. Marasek. 2015. Neural-based Machine Translation for Medical Text Domain. Based on European Medicines Agency Leaflet Texts. *Procedia Computer Science*, 64:2 – 9.

[Wu et al.2016] Y. Wu, M. Schuster, Z. Chen, Q.V. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, J. Klingner, A. Shah, M. Johnson, X. Liu, L. Kaiser, S. Gouws,

Y. Kato, T. Kudo, H. Kazawa, K. Stevens, G. Kurian, N. Patil, W. Wang, C. Young, J. Smith, J. Riesa, A. Rudnick, O. Vinyals, G. Corrado, M. Hughes, and J. Dean. 2016. Google's Neural Machine Translation System: Bridging the Gap between Human and Machine Translation. *CoRR*, abs/1609.08144.

[Yuan et al.2020] M. Yuan, H. Lin, and J. Boyd-Graber. 2020. Cold-start Active Learning through Self-supervised Language Modeling. In *Empirical Methods in Natural Language Processing*.

[Zhai et al.2019a] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori. 2019a. Lifelong gan: Continual learning for conditional image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[Zhai et al.2019b] M. Zhai, L. Chen, F. Tung, J. He, M. Nawhal, and G. Mori. 2019b. Lifelong gan: Continual learning for conditional image generation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*.

[Zhou et al.2016] J. Zhou, Y. Cao, X. Wang, P. Li, and W. Xu. 2016. Deep Recurrent Models with Fast-Forward Connections for Neural Machine Translation. *TACL*, 4:371–383.