Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Máster Universitario en Inteligencia Artificial

Trabajo Fin de Máster

Enhancing the Quality of Parallel Corpora through Classic and Neural Classification

Autor(a): Ana Rodero Paredes
Tutor(a): Mariano Rico

Madrid, Julio 2023

Este Trabajo Fin de Máster se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Máster
Máster Universitario en Inteligencia Artificial

Título: Enhancing the Quality of Parallel Corpora through Classic and Neural Classification

Julio 2023

Autor(a): Ana Rodero Paredes
Tutor(a): Mariano Rico
Departamento de Inteligencia Artificial
ETSI Informáticos
Universidad Politécnica de Madrid

Resumen

La evaluación de la calidad de traducción y el filtrado de corpus paralelos son aspectos vitales para desarrollar sistemas de traducción automática precisos y de alta calidad. Este trabajo se centra en mejorar el rendimiento de Bicleaner, una herramienta ampliamente utilizada para el filtrado de corpus paralelos. Presentamos nuestros hallazgos mejorando el modelo clásico de Bicleaner utilizando un conjunto de corpus más amplio y optimizando el modelo de IA de Bicleaner mediante aprendizaje curricular, aumento de las unidades de la capa oculta y ajustes de los parámetros para la generación de ruido durante el entrenamiento.

Para evaluar la calidad de la traducción, exploramos varias métricas, incluida la métrica más utilizada, BLEU (Bilingual Evaluation Understudy), y algunas otras métricas menos utilizadas. Mediante la comparación de estas métricas, evaluamos los puntos fuertes y las limitaciones de cada una, proporcionando información valiosa en el área de la evaluación de la calidad de la traducción.

Los corpus paralelos sirven como datos de entrenamiento para los modelos de traducción automática. Sin embargo, normalmente contienen ruido y frases desalineadas, lo que afecta a la calidad de la traducción. En nuestro estudio, estudiamos en primer lugar algunas herramientas, centrándonos principalmente en Bicleaner, una herramienta eficaz para filtrar pares de frases ruidosas de los corpus paralelos. Bicleaner hará uso de otra herramienta estudiada también para la limpieza de los datos, Bifixer, cuyo objetivo principal es identificar y corregir los pares de frases desalineadas dentro del corpus paralelo.

Además de Bicleaner, investigamos OpusFilter y OpusCleaner, herramientas especializadas para limpiar y filtrar corpus paralelos. OpusFilter emplea técnicas de preprocesamiento, filtrado, puntuación y clasificación para eliminar el ruido, mientras que OpusCleaner se centra específicamente en el proceso de limpieza, garantizando la coherencia e integridad de los pares de frases paralelas. Mediante análisis detallados, destacamos la eficacia de estas herramientas y su potencial para mejorar la calidad de la traducción.

El objetivo principal de este trabajo es la mejora de Bicleaner. En primer lugar, la mejora del modelo clásico de Bicleaner mediante su entrenamiento con un conjunto de corpus más amplio, incorporando diccionarios probabilísticos y archivos de frecuencias de palabras. Nuestros experimentos muestran que nuestro modelo mejora la precisión y la eficacia con respecto al modelo clásico. Además, profundizamos en la mejora del modelo de Bicleaner AI, empleando el aprendizaje curricular, ordenando el corpus de menor a mayor longitud de frase, aumentamos así gradualmente la dificultad de los ejemplos de entrenamiento, lo que permite al modelo aprender de pares

de frases más sencillas a otras más complejas. También aumentamos las unidades de las capas ocultas de 2046 a 4096, lo que permite al modelo captar patrones y dependencias más complejos. Además, ajustamos los parámetros utilizados para la generación de ruido durante el entrenamiento, introduciendo más ruido informativo y preservando al mismo tiempo la integridad de las frases. Nuestros resultados revelan unas mejoras significativas conseguidas con estas optimizaciones, que aumentan la precisión de la detección de ruido y preservan los pares de frases limpias, lo que nos lleva a una mayor calidad general de la traducción. La evaluación del modelo base en nuestro conjunto de pruebas produjo las siguientes métricas: una precisión de 0,841, un recall de 0,883, puntuación F1 de 0,862 y MCC (Coeficiente de Correlación de Matthews) de 0,846. Sin embargo, con la introducción de nuestro nuevo modelo, se observaron mejoras significativas en todas estas métricas. El modelo mejorado alcanzó valores de precisión, recall, F1 y MCC de 0,954, 0,970, 0,962 y 0,951, respectivamente. Estos resultados indican una mejora sustancial del rendimiento, lo que pone de manifiesto la eficacia y el éxito de nuestro modelo.

Para poder hacer uso de estas herramientas, y todos los modelos entrenados, encontramos todos los datos, resultados obtenidos e instrucciones en este repositorio: https://github.com/anaarodeero/TFM-TUs_results.

Abstract

Translation quality evaluation and parallel corpus filtering are vital aspects of developing accurate and high-quality machine translation systems. This work focuses on enhancing the performance of Bicleaner, a widely used tool for parallel corpus filtering. We present our findings by improving the Bicleaner classic model using a larger corpus and optimizing the Bicleaner AI model through curriculum learning, increased hidden layer units, and parameter adjustments for noise generation during training.

To evaluate translation quality, we explore various metrics, including the widely used BLEU (Bilingual Evaluation Understudy) and some other less used metrics. By comparing these metrics, we assess the strengths and limitations of each, providing valuable insights in the area of translation quality evaluation.

Parallel corpora serve as the training data for machine translation models. However, they often contain noise and misaligned sentence pairs, which impact translation quality. In our study, we firstly study some tools, we mainly focus on Bicleaner, an effective tool for filtering noisy sentence pairs from parallel corpora. Bicleaner will make use of another studied tool for the cleaning of the data, Bifixer, which primary objective is to identify and fix misaligned sentence pairs within the parallel corpus.

In addition to Bicleaner, we investigate OpusFilter and OpusCleaner, specialized tools for cleaning and filtering parallel corpora. OpusFilter employs preprocessing, filtering, scoring, and classification techniques to remove noise, while OpusCleaner focuses specifically on the cleaning process, ensuring the consistency and integrity of parallel sentence pairs. Through detailed analyses, we highlight the effectiveness of these tools and their potential for improving translation quality.

The main objective of this work is the enhancement of Bicleaner. Firstly, the improvement of the Bicleaner classic model by training it on a wider corpora, incorporating probabilistic dictionaries and word frequency files. Through experiments, we demonstrate the improved accuracy and efficiency of the enhanced classic model.

Furthermore, we delve into enhancing the Bicleaner AI model, employing curriculum learning, we gradually increase the difficulty of training examples, allowing the model to learn from simpler to more complex sentence pairs. We also increase the hidden layer units from 2046 to 4096, enabling the model to capture more intricate patterns and dependencies. Additionally, we adjust the parameters used for noise generation during training, introducing more informative noise while preserving sentence integrity. Our results reveal the significant improvements achieved by these optimizations, enhancing the noise detection accuracy and preserving clean sentence pairs, leading to higher overall translation quality. The evaluation of the base model

on our test set yielded the following metrics: precision of 0.841, recall of 0.883, F1 score of 0.862, and MCC (Matthews Correlation Coefficient) of 0.846. However, with the introduction of our new model, significant improvements were observed across all these metrics. The enhanced model achieved precision, recall, F1, and MCC values of 0.954, 0.970, 0.962, and 0.951, respectively. These results indicate a substantial enhancement in performance, highlighting the effectiveness and success of our proposed model.

In order to make use of these tools, and all the trained models, we find all the data, results obtained and instructions in this repository: https://github.com/anaarodeero/TFM-TUs_results.

Contents

Chapter 1

Introduction

Translation is a fundamental aspect of global communication, facilitating effective interaction between individuals who speak different languages. In our increasingly interconnected world, the demand for accurate and high-quality translations is ever-growing. Researchers and developers continuously strive to enhance automated translation systems, which serve as valuable tools for human translators, improving translation efficiency and ensuring precise and fluent translations.

The process of achieving accurate and fluent translations is intricate and multi-faceted. It involves navigating the complexities and subtleties inherent in different languages, understanding the context of the source text, and accurately conveying the intended meaning in the target language. Despite the advancements made in machine translation technologies, these automated systems often encounter challenges that can result in errors, inconsistencies, and inaccuracies, thereby compromising the overall quality of translations and hindering effective communication.

1.1. Objectives

The objectives of this work will be to first understand the need for the evaluation of translation quality, and the enhancing of parallel corpora filtering, which is the main objective, and will be achieved by improving Bicleaner models. Parallel corpora are collections of texts in two or more languages that are aligned at the sentence or segment level. They consist of corresponding sentences or segments in different languages, allowing for direct comparison and analysis. These corpora serve as valuable resources for training and improving machine translation models, enabling researchers to study language patterns and develop accurate translation systems. Each pair of sentences is called Translation Unit (TU), it represents the basic unit of translation, where the source language segment is aligned with its corresponding translation in the target language.

By carefully planning and executing these following objectives, we aim to advance the field of parallel corpus filtering, with the objective of using this Bicleaner models to help filter parallel corpora, so it can be later use for any translation system.

1.1.1. Evaluating translation quality

To address these challenges and drive improvements in translation quality, researchers have dedicated significant efforts to the field of translation studies. One crucial aspect of this research revolves around the evaluation of translation quality. Evaluating translation quality plays a vital role in assessing the performance of translation systems, identifying areas for improvement, and comparing different translation approaches.

1.1.1.1. Evaluation metrics

A central focus of our exploration lies in the widely used metric known as BLEU (Bilingual Evaluation Understudy). BLEU serves as a measurement tool for calculating the similarity between machine-generated translations and one or more reference translations. It employs a scoring system based on the matching of word sequences, enabling the evaluation of translation quality. While discussing BLEU, we will provide a detailed analysis of its strengths, limitations, and potential pitfalls. Additionally, we will explore alternative metrics that are less commonly used but offer different perspectives on the evaluation of translation quality. Our aim is to offer insights that are not only understandable but also informative to researchers, practitioners, and individuals interested in the field.

1.1.2. Parallel corpus filtering

In addition to the evaluation of translation quality, another critical aspect of improving translation systems is the filtering of parallel corpora, and what will be the main objective of this work. Parallel corpora consist of aligned texts in both the source and target languages and serve as the training data for machine translation models. However, these corpora often contain sentence pairs with incorrect alignments or noise, which can negatively impact the training process and, consequently, the quality of translations.

1.1.2.1. Exploration of tools used for the filtering of parallel corpora

Throughout our exploration of parallel corpus filtering, we will examine two widely used tools: Bifixer, but mainly, Bicleaner, and will be used together. And additionally, we will understand OpusFilter and OpusCleaner.

1.1.2.1.1. Bifixer and Bicleaner

These tools employ distinct techniques to identify and eliminate noisy or misaligned sentence pairs from parallel corpora, thereby enhancing the quality of the training data. We will provide clear explanations of how Bicleaner utilizes machine learning algorithms to classify sentence pairs as clean or noisy. Furthermore, we will delve into the advancements in the field with the introduction of Bicleaner AI, which was created to overcome limitations of the classic version and which is a more sophisticated model that leverages deep learning techniques to enhance noise detection and correction in parallel corpora.

1.1.2.1.2. OpusFilter and OpusCleaner

These two specialized tools designed for cleaning and filtering parallel corpora. OpusFilter employs a series of steps, including pre-processing, filtering, scoring, and classification, to remove noise and improve the quality of parallel corpora. We will examine the specific filters, scoring mechanisms, and classification techniques employed by OpusFilter. Furthermore, we will brively delve into OpusCleaner, which is dedicated to the cleaning process of parallel corpora.

1.1.2.2. Example of usage of the tools

Throughout our exploration, we will provide detailed examples and analyses to illustrate the usage and effectiveness of these tools. By comparing different approaches and methodologies, we aim to get a comprehensive understanding of the strengths and limitations of each tool, enabling them to make informed decisions regarding the evaluation and filtering of parallel corpora.

1.1.3. Enhancing the Bicleaner models

The primary focus of this work is to train enhanced models for both the classic and AI versions of Bicleaner. The pre-trained models of this parallel corpus filtering tool have shown promising results, but our objective is to surpass their performance and effectiveness.

To achieve this goal, we will embark on a comprehensive training process that involves refining the existing models. Our aim is to optimize their accuracy, efficiency, and noise detection capabilities, enabling them to outperform the pre-trained versions.

1.1.3.1. Reproducing the pre-trained models for Bicleaner

After studying and understanding all the state of the art of the filterring of parallel corpora, we will aim to improve this. Putting our focus en Bicleaner, with the main objective of improving the pre-trained models, for both Bicleaner classic and Bicleaner AI.

Before this, we would need to level with the existing models, reproducing the training of both classic and AI full models. Firstly, we begin by preparing and cleaning the data, ensuring its suitability for training, this will involve applying probabilistic dictionaries to align words across parallel sentence pairs and generating word frequency files to inform the training process. Subsequently, we delve into training the model, where we employ machine learning algorithms to classify sentence pairs as clean or noisy. The classification process is a key component of the Bicleaner classic model, and we aim to reproduced the pre-trained model training, so later, we can optimize its accuracy and efficiency.

The process for the Bicleaner AI is similar, we again prepare and clean the data, ensuring its compatibility with the Bicleaner AI full model. We incorporate word frequency files to inform the training process, enabling the model to better understand the importance and relevance of words in classifying sentence pairs. The training phase involves leveraging the power of deep learning algorithms to train the Bicleaner

AI full model, enhancing its ability to accurately distinguish between clean and noisy sentence pairs.

1.1.3.2. Improvements for the Bicleaner classic model

Building upon the base models, we explore further improvements in the Bicleaner classic model. Our objective is to optimize its performance by incorporating additional techniques and methodologies. We will investigate the use of AI model data to train the classic model, aiming to leverage the knowledge and insights gained from the Bicleaner AI model. Additionally, we will explore the different types of classifiers, to assess their impact on the classic model's classification accuracy. We will also delve into generating noisy samples to augment the training data, examining different approaches to introduce informative noise while preserving the integrity of the original sentences. Curriculum learning, a training methodology that gradually increases the difficulty of training examples, is another avenue we will explore to enhance the classic model's performance. Furthermore, we analyze the effects of enabling different training parameters and cleaning a new corpus with different hardrules, evaluating their potential to improve the classic model's classification accuracy.

1.1.3.3. Improvements for the Bicleaner AI model

After that, we will focus on enhancing the Bicleaner AI full model. Our objective is to refine its performance by implementing various approaches. We increase the number of epochs during training to observe the model's progression and ensure stability. Adjusting the noise generation parameters allows us to introduce more informative noise while preserving the integrity of the original sentences, improving the model's ability to distinguish between noise and clean data. Curriculum learning is employed to provide a structured learning experience for the model, gradually increasing the complexity of training examples. Hyperparameter tuning, including the modification of activation functions, learning rates, hidden layer units, and decay rates, is explored to optimize the model's performance.

1.2. Structure

The structure of this article will be organized the same way as the objective presented in the previous section.

- State of the art: understanding what it is already achieved.
  - Evaluation of translation quality: Explores the widely used BLEU metric for evaluating translation quality and alternative metrics.
  - Filtering parallel corpora: Introducing tools such as Bifixer, Bicleaner, OpusFilter, and OpusCleaner. It explains their techniques and processes for enhancing the quality of parallel corpora.
- Experiments: Presents the development of the work.
  - Training the base Bicleaner classic model: Explains the process of preparing and cleaning the data, training the model, and classification for the Bicleaner classic model.

  - Training the base Bicleaner AI full model: Discusses the preparation and cleaning of data, training process, and classification for the Bicleaner AI full model.
  - Improvements on Bicleaner classic model: Focuses on enhancing the performance of the Bicleaner classic model by incorporating different techniques, such as using AI model data, generating noisy samples, applying curriculum learning, and enabling different training parameters.
  - Improvements on Bicleaner AI full model: Explores the enhancements made to the Bicleaner AI full model, including increasing the number of epochs, adjusting noise generation parameters, using curriculum learning, and tuning hyperparameters.
- Results and conclusions: Presents the results of the experiments conducted. It discusses the improvements achieved for both models and comparison to the pre-trained models.

Chapter 2

State of the art

Machine translation refers to the automated process of translating text from one natural language to another through the use of computer applications. This involves inputting text in the source language into machine translation software, which then automatically transfers the text to the desired target language.

Machine translation has rapidly evolved over the years, with advancements in natural language processing and artificial intelligence leading to significant improvements in translation quality. While machine translation can be a quick and convenient method of translating text, it is not without its limitations. The accuracy of machine translation can vary widely depending on factors such as the complexity of the language being translated, the context in which the text is used, and the quality of the translation software being used.

Despite its limitations, machine translation has become an increasingly popular tool in various industries, including e-commerce, travel and tourism, and international business. It allows businesses and individuals to communicate and exchange information with people who speak different languages, which can be especially useful in a globalized world where multilingual communication is becoming more and more necessary.

With this said, machine translation models are typically trained using parallel corpora as the primary source of information. Parallel corpora refers to a collection of texts in two or more languages that are aligned at the sentence or phrase level. These texts serve as a reference for machine translation models to learn how to accurately translate text from one language to another.

Previously, in machine translation, statistical models were able to conceal imperfections in the input data without causing significant issues. Therefore, filtering out noise from parallel corpora was not considered a crucial task. However, with the emergence of neural models, which have replaced statistical models in many machine translation applications, it has become essential to be more diligent in removing noise from the input data. This is because noise in the input can have a more significant impact on the accuracy of the output generated by neural models.

2.1. Evaluation of translation quality

When it comes to evaluating the quality of machine translation, there are two main approaches: human evaluation and automatic evaluation. Human evaluation involves having professional translators manually assess the translation quality. This method is highly effective in identifying errors at the sentence level, but it is also time-consuming and expensive.

Automatic evaluation, on the other hand, utilizes AI-based metrics to assess the quality of machine translation without human intervention. While automatic evaluation may not be as reliable as human evaluation when it comes to identifying errors at the sentence level, it is a scalable option that can quickly evaluate the overall quality of translation across multiple documents.

It's worth noting that each evaluation method has its own advantages and disadvantages. Human evaluation provides a more accurate assessment of the quality of machine translation at the sentence level but is more resource-intensive. Automatic evaluation, while less accurate at the sentence level, is faster and can evaluate translation quality across a larger volume of text. Ultimately, the choice of evaluation method depends on the specific needs and resources of the user.

Like mentioned before, the focus here will be in machine translation, and while human evaluation is widely considered the benchmark for evaluating machine translation quality, it can be a resource-intensive undertaking. The key advantage of automatic evaluation over human evaluation is its scalability. It is much quicker to run hundreds of instances of automatic evaluation than to conduct one round of human evaluation. This makes automatic evaluation an ideal solution when making tweaks or optimizing an MT system, where quick results are needed. However, it is important to note that automatic evaluation may not always capture the nuances of human evaluation, especially at the sentence level. Therefore, it should be used in conjunction with human evaluation to provide a more comprehensive assessment of MT quality.

2.1.1. The BLEU metric

The most commonly used metric is BLEU [1]. It has emerged as one of the earliest and most popular metrics for evaluating the quality of machine translation output. Over the years, it has undergone several refinements and modifications to make it more robust and accurate.

The fundamental principle of BLEU is to compare machine-generated translations against a set of high-quality human reference translations. The metric uses a sliding window approach to break down the translations and references into smaller chunks of text known as n-grams. These n-grams can be of different lengths, and their matching is determined based on their exact sequence and occurrence in the reference translations.

This metric then computes a precision score for each n-gram in the machine translation, which represents the degree of overlap with the reference translations. The scores are then aggregated using a geometric mean to compute the final BLEU score, which ranges from 0 to 1. The closer the score is to 1, the higher the quality of the machine translation output.

BLEU has been widely used in the machine translation community due to its simplicity and interpretability. However, it also has some limitations, such as:

1. Lack of correlation with human judgments: BLEU scores often do not align well with human judgments of translation quality. It is known to be biased towards translations that are more similar to the reference translations, regardless of their overall quality [2].
2. Inability to capture linguistic nuances: BLEU is a n-gram-based metric that does not account for the context or meaning of words. It is therefore unable to capture the nuances of language, such as idioms, metaphors, and wordplay [2].
3. Sensitivity to length: BLEU scores are sensitive to the length of the translations, with longer translations typically receiving lower scores. This can penalize translations that are more fluent and natural-sounding [2] [3].
4. Limited scope: BLEU only measures the similarity of n-grams between the machine-generated translation and the reference translations. This means that it may miss other aspects of translation quality, such as fluency, grammar, and the overall meaning conveyed by the translation [4].
5. Inability to handle synonyms: BLEU is unable to capture the meaning of synonyms, which can lead to lower scores for perfectly valid translations that use different words than those in the reference translations [2].
6. Lack of robustness: BLEU is known to be less robust than other metrics in the presence of noise, such as spelling errors or minor variations in word order [2].

Despite these limitations, BLEU is still a widely used metric for machine translation evaluation, but it is often used in combination with other metrics and human evaluation to provide a more comprehensive evaluation of the translation quality.

To fully understand the metric, it would be beneficial to examine an example on the way the metric is calculated.

2.1.1.1. Calculation of the BLEU metric

Starting from a model that produces a predicted sentence, and the supposed target sentence, the BLEU metric could be calculated by using n-gram precision and brevity penalty.

- Target Sentence: I arrived late because it was raining
- Predicted Sentence: I arrived late because of the rain

The first step for obtaining the score would be to tokenize each sentence, being each token a word, such as:

- Target sentence tokens: ['I', 'arrived', 'late', 'because', 'it', 'was', 'raining']
- Predicted sentence tokens: ['I', 'arrived', 'late', 'because', 'of', 'the', 'rain']

Then, the second step would be to calculate the precision scores for the n-grams (from 1 to 4)

$$Precision\ n-gram = \frac{correct\ predicted\ n-grams}{total\ of\ predicted\ n-grams}$$

For each of the n-grams, the tokens will be group in n words each. The target and predicted sentence would be divided like it is showed after this. So let us make a comparison of the target and predicted sentence of each n-gram, to calculate the precision.

- 1-grams: 6/7
  - Target: 'I', 'arrived', 'late', 'because', 'it', 'was', 'raining'
  - Predicted: 'I', 'arrived', 'late', 'because', 'of', 'the', 'rain'
- 2-grams: 3/6
  - Target: 'I arrived', 'arrived late', 'late because', 'because it', 'it was', 'was raining'
  - Predicted: 'I arrived', 'arrived late', 'late because', 'because of', 'of the', 'the rain'
- 3-grams: 2/5
  - Target: 'I arrived late', 'arrived late because', 'late because it', 'because it was', 'it was raining'
  - Predicted: 'I arrived late', 'arrived late because', 'late because of', 'because of the', 'of the rain'
- 4-grams: 1/4
  - Target: 'I arrived late because', 'arrived late because it', 'late because it was', 'because it was raining'
  - Predicted: 'I arrived late because', 'arrived late because of', 'late because of the', 'because of the rain'

Then all of this values must be combined, for this, a geometric average is used:

$$\begin{aligned}\text{Geometric Average Precision } (N) &= \exp\left(\sum_{n=1}^{N} w_n \log p_n\right) \\ &= \prod_{n=1}^{N} p_n^{w_n} \\ &= (p_1)^{\frac{1}{4}} \cdot (p_2)^{\frac{1}{4}} \cdot (p_3)^{\frac{1}{4}} \cdot (p_4)^{\frac{1}{4}} \\ &= \left(\frac{6}{7}\right)^{\frac{1}{4}} \cdot \left(\frac{3}{6}\right)^{\frac{1}{4}} \cdot \left(\frac{2}{5}\right)^{\frac{1}{4}} \cdot \left(\frac{1}{4}\right)^{\frac{1}{4}} = 0.45499...\end{aligned}$$

A range of values for N and various weight combinations can be used to calculate the metric. It is common to set N to 4 and assign uniform weights, where each weight w_n is equal to N divided by 4.

The third step involves calculating a 'Brevity Penalty' The computation of precision in the previous step allows the possibility of predicting a sentence with a single word, such as "The" or "late," resulting in a 1-gram precision score of 1, which appears perfect but is misleading. This scenario can incentivize the model to generate fewer words to attain a high score. Therefore, to avoid this issue, the Brevity Penalty is applied to penalize short sentences.

$$Brevity\ Penalty = \begin{cases} 1, & \text{if } c > r \\ e^{(1-r/c)}, & \text{if } c <= r \end{cases}$$

being c the predicted length (number of words in the predicted sentence) and r the target length (number of words in the target sentence).

By setting an upper limit of 1, this guarantees that the Brevity Penalty does not become excessively high, even if the predicted sentence is substantially longer than the reference sentence. Conversely, if the predicted sentence is quite brief, the Brevity Penalty will be correspondingly low.

In this example, both c and r are 7, so the brevity penalty will be 1.

After this three steps, the BLEU score can be calculated multiplying the brevity penalty and the geometric average precision.

$BLEU = 1 * 0.45499 = 0.45499$

2.1.2. Less used metrics

Knowing that the BLEU metric is the most commonly used one in the evaluation of machine translation, there are some other metrics, created to overcome the limitations that the BLEU metric has. The metric will be listed in the same other of the limitations each of them were created to overcome:

1. METEOR (Metric for Evaluation of Translation with Explicit ORdering) [5] is a metric designed to explicitly correlate better with human judgments of translation quality, by using a weighted combination of several similarity measures such as unigram precision, recall, and alignment-based measures.

2. BERTScore [6] uses contextualized word embeddings generated by BERT to capture the nuances of language, such as idioms, metaphors, and wordplay.

3. NIST [7] (N-gram-based Evaluation Metric for Machine Translation) is less sensitive to length, and accounts for variations in translation length in its scoring.

4. ROUGE [8] measures the overlap between the machine-generated summary and the reference summaries at the word and phrase level, which allows it to capture other aspects of summarization quality, such as content and coherence.

5. ChrF [9] (Character n-gram F-score) is designed to address the issue of synonyms by using character-level n-gram overlap instead of word-level n-gram overlap.

6. TER [10] (Translation Error Rate) is known to be more robust in the presence of noise, such as spelling errors or minor variations in word order, by measuring the number of edits required to transform the machine-generated translation into the reference translation.

It's important to note that each metric, including BLEU, has its own strengths and weaknesses and should be chosen based on the specific needs of the evaluation task.

2.2. Filtering parallel corpora

Recent studies have highlighted the significance of filtering parallel corpora to enhance the quality of machine translation, as demonstrated in the shared tasks at WMT18 [11] and WMT19 [12]. These studies shed light on the level of noise present in some of the most widely used publicly available corpora and the resulting impact on the quality of output produced by state-of-the-art neural machine translation models. In these shared tasks, participants utilized various techniques to extract high-quality data from noisy corpora, including pre-filtering rules to identify obvious noise, scoring functions, and classification to differentiate between high and low-quality sentence pairs. These techniques were applied to both high-resource and low-resource languages.

Several of these techniques have been implemented and evaluated in Bicleaner, while Bifixer offers a new approach to corpus cleaning through restorative cleaning. This method focuses on fixing content and obtaining unique parallel sentences before filtering out noise, aiming to enhance the overall quality of the output.

With this said, and before examining Bicleaner with the goal of training a model to classify parallel corpus, it is important to prepare the data beforehand. So firstly, Bifixier will be introduced and explained how it works.

After that an alternative tool will also be explain, OpusFilter, as well as a brively explation of a subproject of this, OpusCleaner.

2.2.1. Bifixer

Bifixer [13] is a crucial tool in the pre-processing stage of machine translation. Its focus on restoring and cleaning the parallel corpus ensures that the content fed to the machine translation system is of the highest quality possible. The removal of empty lines that do not contain any content in either the source or target languages reduces noise in the corpus, making it easier for the machine translation system to learn. Fixing encoding and character issues such as HTML entities, wrong alphabet characters, and punctuation errors enhances the consistency and accuracy of the corpus.

Bifixer also performs orthography fixing by rewriting words with frequent and straightforward typos. This helps to correct common errors and ensures that the corpus is of a higher quality. In addition, the tool splits sentences that are too long into several shorter sentences, but only retains the new splitting if the number of splits on both the source and target sides are equal. This ensures that the parallel corpus is properly aligned and can be effectively used by the machine translation system.

Duplicates and near-duplicates are identified and removed by Bifixer using a unique hash identifier for each sentence pair. The tool computes a score to determine the most appropriate near-duplicate to select, which is then utilized to identify them. Removing these duplicates and near-duplicates reduces noise in the corpus and helps to create a more effective training set for the machine translation system.

Overall, Bifixer plays a crucial role in preparing a high-quality parallel corpus for machine translation. Its various sub-steps ensure that the content is consistent, accurate, and free of noise, which allows the machine translation system to learn effectively and produce high-quality translations.

2.2.2. Bicleaner

Bicleaner [13] is the tool to focus here. The goal will be to understand how it works, from the filtering a parallel corpus to even being able to train our own Bicleaner model.

As previously mentioned, after the restorative cleaning process is completed, the next step is to pass the sentence pairs to Bicleaner. Bicleaner is a Python-based tool that filters and categorizes noise in parallel sentences by assessing the likelihood of two sentences being mutual translations. The tool generates a score for each pair of sentences in the parallel corpus, ranging from 0 to 1. A score close to 1 indicates a high probability of a mutual translation, whereas a score close to 0 suggests either no mutual translation or significant noise in both sentences.

Bicleaner offers the flexibility to train custom models with personalized criteria for filtering noise through `bicleaner-train`. The tool allows users to personalize the selection of noisy sentences based on their specific needs. Additionally, Bicleaner also provides pretrained models for different languages that can be used directly without the need for customization.

On the other hand, `bicleaner-classify` is used for detecting the noisy sentences mentioned before. Those pairs of sentences that are very noisy and have a score close to 0, bicleaner will change their score to 0, instead of having a very low score, this way, those that are considered very noisy can be seen quicker.

Starting off with an input file and metadata file, this tool will return an output file with the score. On one hand, the input file will contain the parallel corpus to classify and it has to have four columns, the first two correspond to the url where the sentences are from, and the next two columns are for the source and target sentence. The position of these last two columns can be customized with the flags `-scol` and `-tcol`, this way it would not be compulsory to have the first two columns with the URLs. On the other hand, there is the metadata file, which like said before, you could generate by training your own classifier or using the pretrained model.

There is more flags that could be used with the tool. Another flag commonly used is `-disable_hardrules`, where bicleaner does not change the score of those very noisy sentences with 0, it will leave the very low score so it can be seen what score is truly returning bicleaner.

This hard-rules are the first step of Bicleaner. It pre-filters based on these rules. This step involves implementing a set of rules, which are designed to filter out obvious noise, such as sentences with significantly different lengths in the source and target languages. The majority of these rules were originally designed to target noise from web-crawled content, but most of them apply to any corpus. When a pair of sentences matches a rule in this step, it is assigned a “0” score so it can be later discarded. This pre-filtering step helps to improve the quality of the corpus and reduce noise before it is used for training a machine translation system.

2.2.2.1. Classification

Bicleaner incorporates an automatic classifier that assigns a score to a pair of sentences, indicating the likelihood that they are translations of each other. In the 2018 submission of the tool [14], the algorithm used as classifier was a Random Forests.

Latter, in 2020, this algorithm was switched from Random Forests to Extremely Randomised Trees [15], as the latter performed better in preliminary experiments.

In Random Forests [16], each decision tree is trained on a bootstrap sample of the original training set, and at each split, a random subset of features is considered to find the best split. In contrast, Extremely Randomised Trees (Extra Trees) further randomize the splitting process by selecting random cut-off points for each feature, without searching for the optimal cut-off based on impurity measures like Gini index or information gain. This additional randomness makes Extremely Randomised Trees faster to build and less prone to overfitting. The key idea behind Extremely Randomised Trees is to introduce more randomness during the tree construction, which helps to decorrelate the individual trees in the ensemble. This decorrelation leads to reduced variance and increased robustness against noise and outliers in the data.

For Bicleaner [15], the Extremely Randomised Trees algorithm operates by randomly selecting a subset of features from the entire feature set F at each internal node and then choosing the best feature among them, using a random cut-off point. The hyperparameters that govern the training of these classifiers include the method for ranking features and selecting the best one, the size of the random feature subset, and the number of trees to be utilized. To identify the optimal hyperparameters, a grid search was conducted with various parameter values. For feature ranking, both Gini importance and information gain are tested. The size of the feature subset is explored using $|F|$, $\log_2 |F|$ and $\sqrt{|F|}$, and the number of trees was tested with values of 100, 200, 300, 400, and 500. These experiments aimed to find the most effective combination of feature ranking method, feature subset size, and number of trees for the Extremely Randomised Trees algorithm in Bicleaner, and this grid search to find these values is performed during training.

The features employed in Bicleaner can be categorized into two groups: lexical and shallow features. Lexical features are based on word frequencies and bilingual dictionaries, while shallow features focus on sentence length, character distribution, and other heuristics.

By incorporating these diverse features and using the Extremely Randomised Trees algorithm, Bicleaner aims to provide a comprehensive assessment of sentence pairs in terms of their translation probability. This approach allows Bicleaner to analyze both the lexical similarities and the structural properties of the sentences, enhancing its ability to accurately classify mutual translations.

2.2.3. Bicleaner AI

As previously mentioned, the original version of the tool utilized a classifier based on Extremely Randomised Trees, which incorporated a variety of handcrafted features categorized into lexical and shallow ones. While the initial results were positive, the creators recognized the need for further improvement, leading to the development of a second version of the tool.

The revised methodology of the tool involves training a Bicleaner AI model using a clean parallel corpus. Like done for bicleaner classic, the first step would be filtering the data to result in a clean parallel corpus for the training, first through Bifixier to apply character and orthography fixing, and second to apply various filters such

as length ratio, the use of non-alphabetic characters, number of words used, among others. These filters help to ensure the quality and reliability of the corpus, ultimately improving the performance of the tool.

The second step in the methodology for training the Bicleaner AI model involves the use of a classifier to assign a score to sentence pairs based on their likelihood of being a valid translation. This score ranges between 0 and 1, with a higher score indicating a higher probability of the sentence pair being a true translation. The classifier is trained using a supervised learning approach, with positive samples being drawn from existing parallel corpora, and negative samples created by corrupting the same positive samples. Synthetic noise is introduced to simulate common errors that arise from the sentence segmentation and alignment tools used to create webcrawled corpora.

There are different types of noise used to corrupt the positive samples, including random alignment, word omission, and frequency-based noise. These same types of noise were used in the classic version of the tool. To build the training and development sets, sentence pairs from a parallel corpus provided by the user are used as positive samples. By using this approach, the Bicleaner AI model is trained to accurately distinguish between valid and invalid sentence pairs, leading to improved accuracy in identifying noisy parallel corpora.

2.2.3.1. Lite models classifiers

The lite models are based on decomposable attention [17], which basically uses attention to divide the problem into subproblems which can be solved separately, so it can also be solved parallelly.

Unlike current methodologies, this technique solely depends on alignment and is entirely computationally decomposable in relation to the input text. Decomposable attention has the following representation: When given two sentences with each word represented by an embedding vector, a neural attention mechanism creates a soft alignment matrix. This matrix is then used to break down the task into subproblems that can be solved separately, with the final classification being produced by merging the results. While performing the same amount of work as a standard LSTM encoder, this method is parallelizable across sentence length, allowing for significant speedups in low-latency scenarios.

For the creation of the Bicleaner lite model, to avoid reliance on pre-trained English embeddings, it incorporates a bilingual SentencePiece joint vocabulary with GloVe-trained embeddings [18]. This approach eliminates tokenization dependencies and results in a more concise vocabulary that requires less memory.

The lite models show good results with very little parameters, this is why it is said that it focuses on efficacy.

2.2.3.2. Full models classifiers

The full models are based on XLM-RoBERTa pre-trained language model, which is also fine-tuned. This last case is more intended for high performance inference since it focuses on efficiency.

The XLM-RoBERTa is a multilingual masked language model based on Transformer architecture, pre-trained on text from 100 languages. It achieves state-of-the-art results in cross-lingual classification, sequence labeling, and question answering. RoBERTa is a type of transformer model that has been pre-trained in a self-supervised manner on a vast corpus of text. The self-supervised approach involves pre-training the model on raw texts without any human labeling, allowing it to utilize a wide range of publicly available data.

The pre-training process involves an automatic mechanism that generates inputs and labels from the text data. To be more specific, RoBERTa was pretrained using the Masked Language Modeling (MLM) objective. In this process, the model masks 15% of the words in a sentence at random and then attempts to predict the masked words. Unlike traditional models such as RNNs, which process words sequentially, or autoregressive models like GPT, which internally mask future tokens, RoBERTa processes the entire masked sentence. This approach enables the model to learn a bidirectional representation of the sentence.

This advancement has resulted in state-of-the-art performance on cross-lingual understanding tasks. These models demonstrate successful cross-lingual transfer on various benchmarks, including named entity recognition, question answering, and cross-lingual natural language inference [19]. Nonetheless, these studies rely on Wikipedia for pretraining, which offers limited scale, particularly for languages with fewer resources. It is also evaluated the performance of monolingual fine tuning on the GLUE and XNLI benchmarks, where the XLM-RoBERTa model performs comparably to state-of-the-art monolingual models, including RoBERTa [20]. These outcomes prove that having a single large model for all languages is achievable without compromising each language performance, marking the first instance of such a feat.

The model is based on the XLM approach [21], and this XLM approach is a neural network based on the Transformer architecture that is designed for natural language processing. The model has self-attention mechanisms that enable it to focus on different parts of the input sequence when generating the output. XLM was trained on a multilingual corpus using masked language modeling and language translation tasks, where from pairs of sentences in different languages, learns to generate accurate translations. Fine-tuning the XLM model on a specific task, such as sentence classification in Bicleaner AI, allows the model to adapt to specific patterns and features in the data, resulting in better performance on that task.

Bicleaner AI is based on those approaches. A Transformer model trained on the multilingual MLM objective using only monolingual data is used, to achieve this, streams of text from each language are sampled and used to train the model to predict the masked tokens in the input. To tokenize the raw text data, Sentence Piece [22] with a unigram language model [23] is directly applied for subword tokenization. Batches from different languages are sampled using the same sampling distribution as [21], but with $\alpha = 0.3$. The value of α refers to the hyperparameter used in the fine-tuning process, specifically, it is used to balance the contribution of the monolingual and parallel training objectives during the process, namely, the training with parallel sentences so the model can learn accurate translations. A smaller value of α puts more emphasis on the parallel objective, while a larger value of it, puts more emphasis on the monolingual objective. Here, they found that 0.3 produced the best results on the XNLI task, which is a cross-lingual natural language inference task.

However, unlike XLM approach mentioned before, XLM-Roberta does not incorporate language embeddings, enabling the model to handle code-switching more effectively. This language embeddings are embeddings of entire languages, which are learned in an unsupervised manner, they capture the structural and semantic similarities and differences between languages and allow for cross-lingual transfer of information.

The model explained before is used for the full models of this version of Bicleaner, with the difference that it is fine-tuned by adding a hidden layer of 2,048 ReLU units with a 10% dropout rate in training. The fine-tuning process uses a learning rate of 0.000002, a batch size of 128 sentences, and 1000 warm-up steps. Checkpoints are saved every 2000 steps, and the process stops after three checkpoints without improvement on a development set or after a maximum of 30000 steps.

2.2.3.3. Lite vs. full models

To validate the model's performance during training and track its progress, it is utilized the development set to calculate the Matthews correlation coefficient (ϕ) between the positive and negative classes. This metric is similar to the F-score but provides more detailed information [24]. It is also a statistically robust measure that reflects a high score only when the prediction performs well across all four categories of the confusion matrix (true positives, false negatives, true negatives, and false positives), in proportion to both the number of positive and negative elements in the dataset.

Some experiments were performed to compare these two types of models and concludes that the performance of the full models is better than that of the lite models [25]. Additionally, the full models demonstrate strong performance on languages with very limited resources and on languages that were not included in the pre-training phase. Therefore, when both classic and AI versions are compared in the following sections, the full model is the one that is going to be used for the Bicleaner AI version.

2.2.4. OpusFilter

An alternative for Bicleaner could OpusFilter [26], which is also a tool for filtering parallel corpora. This toolbox incorporates several heuristic filters, language identification libraries, character-based language models, and word alignment tools, and it can be expanded with customized filters.

OpusFilter takes a YAML file as configuration for the input. This file will have a list of steps which can be for downloading corpus, selecting subsections of these and even combining them. It can also filter and score a pair of sentences, and train a classifier based on these scores.

2.2.4.1. Pre-processing

As mentioned before, the first steps involve pre-processing the raw text data. This includes tokenizing the text into words or subword units, normalizing the text by applying various text normalization techniques such as lowercasing or removing diacritics, and segmenting the text into sentences or phrases. Language identification is also performed to identify the languages of the sentences or phrases. These preprocessing steps help to prepare the data for the subsequent filtering and alignment steps.

First the corpus should be download, with the function `opus_read`, which is a component that utilizes the OpusTools [27] library to download a designated parallel corpus from the OPUS corpus collection. The corpus is then saved as two files, with one file containing the source segments and the other containing the corresponding target segments. Each segment is saved on a separate line within the files. It also provides options for users to select a particular version of the corpus to download, depending on their research or development needs. Additionally, the function allows users to choose between pre-tokenized or untokenized segments during the download process. The pre-tokenized option downloads segments that have already been processed and tokenized, which can save time and effort in the pre-processing stage. Alternatively, users can choose to download the untokenized segments and process them using their preferred tokenization method. By using this function, users can obtain a high-quality parallel corpus that has been tailored to their specific requirements, with minimal effort and maximum accuracy.

After downloading the data, the `preprocess` function performs various pre-processing tasks on the parallel corpus data to prepare it for downstream processing and analysis. The function can perform several pre-processing tasks, such as sentence segmentation, tokenization, and cleaning. Sentence segmentation involves splitting the text into individual sentences, which is a necessary step for many natural language processing (NLP) tasks. Tokenization involves dividing the sentences into individual tokens or words, which makes it easier to analyze the text further. Cleaning involves removing unwanted characters, symbols, and noise from the text, such as HTML tags, punctuation marks, and non-alphanumeric characters. By using this function, users can ensure that their parallel corpus data is properly prepared and cleaned, which can lead to better results and more accurate analyses.

2.2.4.2. Filtering and Scoring

After downloading and selecting data, it usually comes the filtering and scoring of the data. This filtering and scoring section of the OpusFilter toolbox is responsible for selecting and prioritizing the most relevant and high-quality segments from a parallel corpus. This section includes several configurable filters and scoring metrics that users can apply to the corpus data to achieve the desired filtering and scoring outcomes. The first step is to apply one or more filters to the corpus data to remove any unwanted segments, with the function.

2.2.4.2.1. Filters

Filters can be used to remove segments that contain certain words or phrases, or to remove segments that do not meet certain criteria, such as minimum or maximum length. The OpusFilter toolbox includes several pre-configured filters, which can be group in different sets. The first set of filters are length-based, which allow users to set maximum and minimum length thresholds for segments in the parallel corpus, as well as segment length ratios in words or characters. It can remove segments that are too short or too long, or that do not meet their desired length criteria. The second set are script and language identification filters, these are useful for identifying and removing segments that belong to scripts or languages that are not relevant to the user's research or analysis. The third set of filters in OpusFilter consider special characters such as numbers and punctuation marks. These filters can be used

to remove segments that contain unwanted or irrelevant characters, such as mathematical symbols or emoticons. By applying these, users can ensure that their parallel corpus data contains only relevant and meaningful text. The next set of filters use probabilities from n-gram language models, which can be used to remove segments that have low language model probabilities, which can indicate poor grammar or sentence structure. Also, it has a filter used to filtrate segments by word alignment. And lastly, the use of sentence embeddings to filter segments. These all will help select high-quality and well-formed segments for downstream NLP tasks.

2.2.4.2.2. Scores

Once the corpus data has been filtered, the next step is to score each remaining segment based on its relevance and quality. The OpusFilter toolbox includes several pre-configured scoring metrics, such as length ratio, bilingual lexicon coverage, and alignment score, which can be used to assess the quality and relevance of each segment. Users can also create their own custom scoring metrics, depending on their specific needs and objectives.

The filters used for the score function are defined in the same way as those for the filter function. However, it's important to note that the accept threshold parameters of the filters don't affect the scoring process. Instead, each filter generates one or more numerical scores. The scores produced by the filters are written in the JSON Lines format, where each line contains a single JSON object. At the top level of the object, there are class names for the filters. If there is only one filter of a specific class, and its score is a single number, the value of the score is placed below the class name. However, if the filter outputs multiple scores, they are represented in a list or dictionary. Generally, there is one score for each segment (language) if multiple scores are present.

The filters could contain the same filter class multiple times. This allows for the same filter to be used with different parameters.

2.2.4.2.3. Remove duplicates

OpusFilter provides a convenient feature to remove duplicate lines from parallel corpora, which can be useful for various tasks such as training machine learning models or improving the quality of parallel data. This feature allows the user to specify any combination of lines from the source and target segments to use as a matching criterion for identifying duplicates. For example, the user can choose to match duplicates based on an exact match of the entire source and target sentence, or on a subset of the lines such as the first n words or characters. Additionally, the user can specify whether the matching should be case-sensitive or case-insensitive.

This feature is especially useful for ensuring that each target sentence occurs only once in a parallel corpus, which is a common requirement for many natural language processing applications. By removing duplicates, the resulting corpus can be cleaner and more diverse, which can lead to improved performance for downstream tasks.

2.2.4.2.4. Ranking and sorting

Finally, the OpusFilter toolbox ranks and sorts the remaining segments based on their scores and selects the highest-scoring segments for further processing or anal-

ysis. The number of segments selected can be customized based on the user's preferences or requirements.

OpusFilter provides two functions for this, `join` and `sort`, to facilitate the filtering and scoring process of parallel corpora. The `join` function allows the user to merge separate score files into a single file, making it easier to work with the data. The `sort` function, on the other hand, sorts the input files based on the scores generated by the filters. This reordering makes it more convenient to remove noisy pairs of sentences from the end of the files.

By using `join`, multiple score files can be combined into a single file, which can be useful for downstream analysis. The function takes as input a list of score files and a file name for the output. The resulting file will contain all the scores from the input files, with each line representing a single score object in JSON format.

The `sort` function sorts the input files based on the scores generated by the filters, which allows the user to easily identify and remove noisy pairs of sentences. The function takes as input the two parallel corpus files and the corresponding score file. It sorts the files based on the scores in the score file, with the highest scores appearing first. This reordering makes it easier to spot and remove low-quality sentences from the end of the files.

2.2.4.3. Classification

OpusFilter is a tool that uses a classification approach to determine the cleanliness of sentence pairs that can be used for machine translation training. The scores calculated by different filters serve as features for a logistic regression classifier, which predicts whether a segment pair is clean or noisy. The resulting classification probability can be used to sort the data based on their expected cleanliness.

The process involves scoring a set of sentence pairs using filters and then splitting them into clean and noisy examples for training the logistic regression classifier. To select positive and negative example pairs, a percentage threshold is set for all filter scores. Only sentence pairs that score above the threshold percentile for all filters are considered clean, while the rest are labeled noisy.

OpusFilter improves upon the method used in [28], which manually placed the threshold between the two peaks of a score distribution in cases where the distribution is bimodal. Instead, OpusFilter implements an automatic selection of the optimal threshold to ensure more convenient usage of the tool. Multiple models can be trained with different training data splits using different thresholds to find the best performing model. The configuration file allows for specifying the minimum, maximum, and initial percentage thresholds for each score, which can be optimized with a search algorithm.

In order to find the best performing model for the OpusFilter tool, an optimization criterion is used to evaluate the effectiveness of different models. This can be either cross-entropy of the classifier, which measures how well the model is able to predict clean vs. noisy segments, or the area under the receiver operating characteristics curve, which evaluates the trade-off between true positive rate and false positive rate for different threshold values. The development set used for this evaluation is a set of scores labeled as either noisy or clean by the user.

Once the logistic regression model is trained and selected, it can be applied to each segment pair in a larger set of data to produce a single cleanness score, which is the probability prediction from the model. This score can then be used to sort the data and remove noisy pairs from the end of the sentence files.

OpusFilter offers two main functions for classification:

2.2.4.3.1. Trainning the classifier

The `train_classifier` function is used to optimize a classifier that predicts the cleanliness of segment pairs. The training scores are first obtained by running the segment pairs through different filters. The function takes in these scores along with the optimization criterion, and search algorithm details. The output is an optimized classifier that is written to the specified output file.

The parameters used in this function are those mentioned before, the filter scores that were obtained in the filtering and scoring step. A criterion must be selected between:

- Cross-entropy (CE) [29]: a measure of the difference between the predicted probability distribution of the classifier and the true probability distribution of the data.
- Receiver Operating Characteristics Area Under the Curve (ROC AUC) [30]: a measure of the classifier's ability to distinguish between positive and negative examples, computed by plotting the true positive rate against the false positive rate at various threshold settings.
- Sum of Squared Errors (SSE) [31]: It calculates the squared difference between the predicted values of the model and the actual observed values. It optimizes the logistic regression classifier, where the aim is to minimize the difference between the predicted cleanness scores and the actual clean/noisy labels of the sentence pairs.
- Akaike Information Criterion (AIC) [32]: a measure of the relative quality of a statistical model, taking into account the number of parameters used.
- Bayesian Information Criterion (BIC) [33]: a similar measure to AIC, but with a penalty for models with a large number of parameters.

Another important parameter used for the trainning of the classifer is the features. This parameter `features` refers to a list of filter names that are used to extract specific features from the input data. These extracted features are then used as input to train the logistic regression classifier. The list of filter names determines which filters are used to extract the features, and hence, which aspects of the data are taken into consideration for classification. This filters can be the same as the ones mentioned before in the filtering step. For each of these filters, two subparameters should be set: clean-direction and quantiles. The first one refers to the direction that indicates higher cleanness and the available options are 'high' and 'low'. The second one is a dictionary with three items: min, max, and initial, and these items determine the minimum, maximum, and initial quantile values used for optimizing the classifier to select negative and positive training examples.

To optimize the classifier, the training data is divided into positive and negative examples based on the specified quantile boundaries in each feature, and multiple classifier models are trained. The model that attains the highest criterion score is then saved in the designated output file.

2.2.4.3.2. Classify

The function `classify` is designed to assign a cleanness score or label to every sentence in a given dataset. It takes as input the trained classifier file and the sentence pairs that need to be classified. The function then generates scores or labels for each sentence and writes them into an output file, one per line.

The parameters that this function uses are the trained model produced by the last functions explained, the scores generated by the score function mentioned before and the output file where the probabilities of cleanness of each pair of sentences is stored.

2.2.5. OpusCleaner

After seeing and understanding OpusFilter, there is another tool similar to the filtering part of OpusFilter but works visually, OpusCleaner, which is a machine translation/language model data cleaner.

The cleaning component of OpusCleaner handles the tasks of downloading various datasets, performing data cleaning operations, and preparing the datasets for translation purposes.

By simply installing the tool with `pip3 install opuscleaner` and launching it with `opuscleaner-server serve`, it will start the server, and it can accessed by visiting `http://127.0.0.1:8000/` in your web browser.

The OpusCleaner interface allows you to list and categorize datasets that you intend to use for training. You can also download additional datasets directly from the interface. The tool provides the ability to filter individual datasets and immediately see the results. Furthermore, you can compare datasets at different stages of filtering to observe the impact of each filter.

2.2.6. Example and analysis of usage

2.2.6.1. Bicleaner and Bicleaner AI

While Bicleaner, including both Classic and AI versions, is generally effective, it is not infallible and may fail like any other model. Additionally, in certain circumstances, it can be challenging to explain the disparities in the resulting scores. Nevertheless, we can still learn from the output of both versions by examining the scores they return on a set of parallel sentences. By analyzing these patterns and differences, we can gain insight into the strengths and weaknesses of each approach.

Bicleaner is a rule-based system that relies on fixed rules and thresholds to measure the quality of a translation. Therefore, it is limited by the rules it has been trained on and might not always capture the complexity of the sentence structure and nuances in meaning. For instance, it might struggle with sentences that have complex syntax or idiomatic expressions.

EN	ES	Bicleaner	Bicleaner AI
It has been said that temptations will certainly come to your door, but you are to blame if you invite them to stay for dinner.	Se ha dicho que las tentaciones ciertamente vendrán a la puerta de uno, pero que uno tiene la culpa si las invita a entrar y pasar un rato.	0.450	0.930
How may we be able to increase the joy we gain from the disciple-making work?	¿Qué podríamos hacer para disfrutar más de nuestro ministerio?	0.207	0.684
(Judges 6:11-14) David was tending sheep when God told Samuel to anoint him as Israel's next king.	El ángel de Dios se apareció a Gedeón, a quien se nombró para salvar a Israel de la opresión (Jueces 6:11-14).	0.430	0.624
Can toddlers safely be placed in day care?	¿Existe algún riesgo en llevar a la guardería a niños pequeños que apenas han empezado a andar?	0.303	0.940
One who is somewhat younger can realize what it means to do what is right and to be dedicated to his Creator.	Esto significa que aun un niño puede entender qué es correcto y dedicarle su vida a Jehová.	0.257	0.882
Adam and Eve died for their disobedience, but why do all their descendants grow old and die?	Nuestros primeros padres murieron por ser desobedientes. Ahora, ¿por qué envejecemos y morimos todos los seres humanos?	0.303	0.966
Since a woman who is pregnant is more likely to hemorrhage, heeding warning signals assumes a still greater importance.	Como las mujeres embarazadas tienen más posibilidades de sufrir hemorragias, el prestar atención a los avisos del cuerpo adquiere aún más importancia.	0.237	0.982
Instead of reading being a chore, this book makes it more like entertainment or a treat at the end of my day.	La lectura no es una tarea penosa, pues el libro me parece una forma de entretenimiento o un gusto que me doy al fin del día.	0.257	0.935
Pause periodically, and ask open-ended questions.	Conviene pausar de vez en cuando y hacer preguntas que le hagan hablar.	0.430	0.925
Saul, though, was not of that mind at all.	Saúl, no obstante, no compartía ese punto de vista.	0.437	0.956
However, Wildlife Waystation is not set up to make money.	Pero el Wildlife Waystation no se ha establecido para obtener ganancia.	0.427	0.925
But after sharing in the preaching work with the white Witnesses and enjoying hospitality in their homes, she exclaimed, "They are normal people just like us!"	Pero, después de predicar con ellos y de estar en su casa, exclamó: "¡Son como nosotros! ¡Son gente normal!".	0.327	0.927
Part of being reasonable involves administering correction in such a way that your children retain their dignity.	Ser razonable implica, entre otras cosas, corregir a los hijos sin herir su dignidad.	0.270	0.959
Remnants and particles are collected and formed into agglomerated corks and other products	Los residuos y las partículas se aglutinan para obtener corchos aglomerados y demás productos	0.373	0.817

Table 2.1: Pattern 1: Output values of the parallel corpus JW300, where Bicleaner penalizes the differences of the structures of the sentences, while Bicleaner AI looks beyond it, looking for the whole meaning of the sentences.

EN	ES	Bicleaner	Bicleaner AI
(February 22, 1998) I just wanted you to know that it is not the first time my eyes have filled with tears on reading one of your wonderful articles.	Solo quería decirles que no es la primera vez que mis ojos se llenan de lágrimas al leer uno de sus bellos artículos.	0.467	0.960
Safe driving means obeying the traffic laws. [Picture on page 7]	Conducir con seguridad significa obedecer las leyes de tráfico.	0.360	0.934
(Ecclesiastes 7:12) What is the point?	¿Cuál es la idea?	0.283	0.654
Forty per cent of all residents suffer chronic bronchitis. . . .	El 40% de los residentes padece bronquitis crónica. [...]	0.457	0.929
Of these, almost 90 percent are shooting victims.	De estos, casi el 90% muere por heridas de bala.	0.497	0.950
How Can I Recover From Drug Abuse?"	¿Cómo puedo recuperarme del abuso de las drogas?" (8 de julio de 1986).	0.477	0.970
(Luke 1:46-49) As Jehovah's Witnesses, we laud him for such great deeds as the freeing of Israel from Egyptian bondage and the miraculous conception of his dear Son.	Los testigos de Jehová lo elogiamos por sus grandes hazañas, como la liberación de Israel de la esclavitud en Egipto y la concepción milagrosa de su querido Hijo.	0.463	0.945
(Rom. 3:23, 24) What did Paul mean by saying "undeserved kindness"?	¿Qué quiso decir Pablo con "bondad inmerecida"?	0.300	0.793
(Isaiah 63:10; 1 Corinthians 6:9, 10; 2 Corinthians 7:1; Ephesians 4:30) While he does this primarily to please Jehovah God, he also receives additional benefits-better health and peace of mind.	Aunque su principal motivación es agradar a Jehova Dios, también se beneficia de otras maneras, mejor salud y tranquilidad de animo.	0.193	0.959
(Isaiah 33:24; Revelation 21:4)—As told by a paramedic in Canada.	(Relatado por un paramédico de Canadá.)	0.303	0.951
Jesus said regarding the cup: "This means my 'blood of the covenant,' which is to be poured out in behalf of many for forgiveness of sins."	Jesus dijo con respecto a la copa: "Esto significa mi sangre del pacto, que ha de ser derramada a favor de muchos para el perdon de los pecados". (Mateo 26:28).	0.217	0.976
6 Today, divinely granted peace is to be found only among true worshipers of Jehovah God.	6 Hoy la paz de origen divino se halla únicamente entre los verdaderos adoradores de Jehová Dios (Salmo 119:165; Isaías 48:18).	0.470	0.948

Table 2.2: Pattern 2: Output values of the parallel corpus JW300, where Bicleaner penalizes the use of non-alphanumeric characters not used in both sentences, while Bicleaner AI ignores it and looks for the whole meaning.

EN	ES	Before	After
I just wanted you to know that it is not the first time my eyes have filled with tears on reading one of your wonderful articles.	Solo quería decirles que no es la primera vez que mis ojos se llenan de lágrimas al leer uno de sus bellos artículos.	0.467	0.543
Safe driving means obeying the traffic laws.	Conducir con seguridad significa obedecer las leyes de tráfico.	0.360	0.787
What is the point?	¿Cuál es la idea?	0.283	0.663
Forty per cent of all residents suffer chronic bronchitis.	El cuarenta por ciento de los residentes padece bronquitis cronica.	0.457	0.647
Of these, almost 90 percent are shooting victims.	De estos, casi el 90 por ciento muere por heridas de bala.	0.497	0.623
How Can I Recover From Drug Abuse?	¿Cómo puedo recuperarme del abuso de las drogas?	0.477	0.727
As Jehovah's Witnesses, we laud him for such great deeds as the freeing of Israel from Egyptian bondage and the miraculous conception of his dear Son.	Los testigos de Jehova lo elogiamos por sus grandes hazanas, como la liberación de Israel de la esclavitud en Egipto y la concepción milagrosa de su querido Hijo.	0.463	0.757
What did Paul mean by saying "undeserved kindness"?	¿Qué quiso decir Pablo con "bondad inmerecida"?	0.300	0.587
While he does this primarily to please Jehovah God, he also receives additional benefits, better health and peace of mind.	Aunque su principal motivación es agradar a Jehova Dios, también se beneficia de otras maneras, mejor salud y tranquilidad de animo.	0.193	0.557
As told by a paramedic in Canada.	Relatado por un paramédico de Canadá.	0.303	0.677
Jesus said regarding the cup: "This means my blood of the covenant, which is to be poured out in behalf of many for forgiveness of sins"	Jesus dijo con respecto a la copa: "Esto significa mi sangre del pacto, que ha de ser derramada a favor de muchos para el perdon de los pecados"	0.217	0.577
6 Today divinely granted peace is to be found only among true worshipers of Jehovah God.	6 Hoy la paz de origen divino se halla únicamente entre los verdaderos adoradores de Jehova Dios.	0.470	0.583

Table 2.3: Demonstration of pattern 2: Differences on non-alphanumeric characters were eliminated to visualize how Bicleaner (classic) penalizes it. On the column "before" the values with the penalization and on the column "after" the output values after eliminating the differences.

EN	ES	Bicleaner	Bicleaner AI
Can any genuine comfort be derived from a pagan philosophy?	¿Puede derivarse algún consuelo genuino de una filosofía pagana?	0.827	0.979
United States Senator John L. McClellan, during an interview on reasons why crime keeps rising in the land, spoke in a similar vein:	El senador estadounidense John L. McClellan, durante una entrevista sobre razones por las cuales siguen aumentando los crímenes en ese país, habló de manera similar:	0.757	0.979
We should be at our work urgently.	Debemos ocuparnos en la obra urgentemente.	0.640	0.984
Christians are not to forget "the sharing of things with others."	Los cristianos no han de olvidar "el compartir cosas con otros".	0.873	0.979
The possibility of someday reaching the top remains a strong incentive for hard work.	La posibilidad de llegar algún día a la cima sigue siendo un incentivo para trabajar mucho.	0.863	0.968
A person must first see the need for something better than what the worldly nations have to offer.	Uno tiene que comprender primero que se necesita algo que sea mejor de lo que las naciones mundanas ofrecen.	0.713	0.981
She praised her Father for his unique holiness, his justice, and his faithfulness.	Alaba a su Padre por su singular santidad, su justicia y su fidelidad.	0.713	0.910
A Matter of Gravity	La cuestión de la gravedad	0.647	0.963
To have courage, we must pray as well as study and obey God's Word	Al ser valientes, él y su esposa salvaron a la familia de la que vendría el Mesías.	0.183	0.002
Shock Treatment	Hábito perjudicial	0.213	0.299
Remaining Organized for Survival Into the Millennium	"Llegarán a ser un solo rebaño, un solo pastor."	0.080	0.008
Money Worries, 1/8	Se empieza a aprender en la matriz, 22/1	0.343	0.056
Is there anything sinful about sexual relations between husband and wife?—Proverbs 5:15, 18, 19.	(Proverbios 5:15, 18, 19.)	0.177	0.358
The Bible likens immoral religious leaders to a bathed sow that returns to rolling in the mire	A pesar de eso, el problema se ha agravado tanto que muchos abogados y víctimas dicen que la iglesia pasa por alto esos casos y los encubre."—The Miami Herald, 3 de enero de 1988.	0.063	0.002
Especially so if it seems that your sibling—the oldest, the youngest, the best-behaved, or even the most disobedient—is in the spotlight all the time.	Tal vez llegues a sentirte como David cuando escribió: "Como alguien muerto y no en el corazón, he sido olvidado; he llegado a ser como un vaso dañado".	0.060	0.001
Some women refuse to accept the headship of their husbands.	Otros cónyuges son víctimas de maltrato.	0.110	0.053
(Genesis 33:13) Also, try to maintain a healthy view of winning and losing.	Al fin y al cabo, "los niños son delicados" (Génesis 33:13).	0.340	0.055

Table 2.4: Accurate and inaccurate translations of both tools

On the other hand, Bicleaner AI is a neural-based system that uses a deep learning algorithm to understand the context and meaning of the sentence. It is not limited by a fixed set of rules and is more flexible in capturing the nuances of the language. Neural-based systems are better equipped to handle complex syntax and idiomatic expressions.

In table 2.1, we can see multiple examples on how Bicleaner might struggle with the different sentence structures between the English and Spanish sentences, which could lead to a lower score. However, Bicleaner AI is able to understand the underlying meaning of the sentences and thus assigns a higher score. All of those sentences have meaning in common, that is to say, they have the same meaning in the translations while having different structures. This is why, in general, neural-based systems like Bicleaner AI are more advanced and accurate than rule-based systems like Bicleaner, especially in handling complex sentence structures and nuances of the language, and for this, returning better scores.

Bicleaner and Bicleaner AI use different algorithms to calculate the score. The Classic version of Bicleaner uses a simple n-gram matching algorithm, which means that it looks for exact matches of n-grams (contiguous sequences of words) in the reference and predicted sentences. On the other hand, Bicleaner AI uses a more advanced algorithm that takes into account not only the exact matches, but also the semantic similarity between the reference and predicted sentences.

Like it is shown with all the examples of table 2.2, in cases where there are differences in capitalization, punctuation, and other minor details between the source and target sentences, the Classic version of Bicleaner may give a low score because it relies solely on exact matches of n-grams. For example, if the source sentence contains a capitalized word that is not capitalized in the target sentence, the Classic version of Bicleaner would consider this a mismatch and reduce the score accordingly. However, Bicleaner AI is designed to be more robust to such differences, as it takes into account the semantic similarity between the sentences. Therefore, even if there are minor differences in capitalization or punctuation, Bicleaner AI may still give a good score if the meaning of the sentences is preserved.

The exemplification of this second pattern is evident in table 2.3, where the disparities in non-alphanumeric characters have been rectified. As a result, the adverse scores previously generated by Bicleaner due to such penalization have now been significantly enhanced, as demonstrated in the "After" column of the table.

Despite their differences, both versions of the tool can produce similar scores when evaluating sentences that exhibit the patterns mentioned earlier. In the first section of table 2.4, we observe that both versions of Bicleaner yield good scores, which can be attributed to the similarity in the sentences' structure and meaning, as well as the absence of differences in case sensitivity and punctuation. These patterns allow both versions of the tool to effectively capture the essence of the sentences and provide accurate scores. On the other hand, on the second section of the table 2.4, we can see some example of completely wrong translation because they are directly different sentences, so that is why, both versions return bad scores. In these cases, it is clear that neither the rule-based nor the neural-based approach is able to salvage the translation.

In conclusion, while both Bicleaner Classic and Bicleaner AI have their strengths and

weaknesses, the latter is generally more advanced and accurate than the former, especially in handling complex sentence structures, idiomatic expressions, and minor differences in capitalization and punctuation. Bicleaner Classic relies on fixed rules and thresholds to evaluate translation quality, which limits its ability to capture the nuances of the language, whereas Bicleaner AI uses a deep learning algorithm to understand the context and meaning of the sentence, making it more flexible and adaptable to different sentence structures and nuances. Additionally, Bicleaner AI takes a more holistic approach and looks beyond surface-level differences to capture the overall meaning of the translated text, which leads to more accurate evaluation of translation quality. Overall, both versions of Bicleaner can be valuable tools for translation evaluation, but Bicleaner AI provides a more sophisticated and comprehensive evaluation of translation quality.

After classifying the corpus, a few test to calculate the time of execution were performed. The corpus was split three times with sizes of 10k, 50k and 100k to measure the time of the classic model and the AI model.

As shown in the table 2.5 the execution was in CPU, and not GPU, the AI version takes way more time than classic one. For the classic version, each classification was executed 50 times, to get an accurate average time. For each sizes of input, 10k, 50k and 100k pair of sentences, the following characteristic of the execution times in seconds were registered:

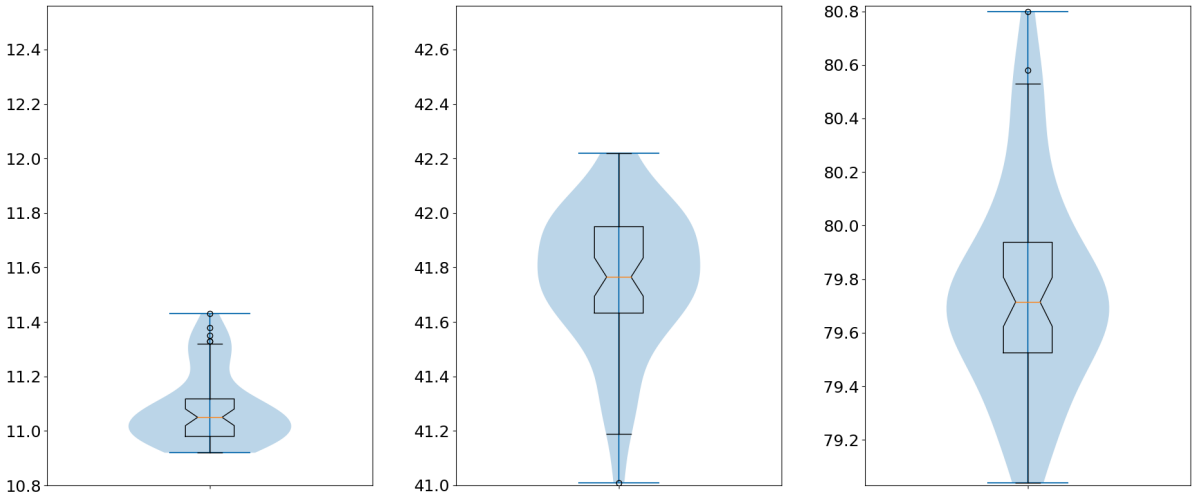
- 10k:
  - Average: 11.08
  - Variance: 0.01
  - Standard deviation: 0.13
- 50k:
  - Average: 41.76
  - Variance: 0.067
  - Standard deviation: 0.25
- 100k:
  - Average: 79.75
  - Variance: 0.148
  - Standard deviation: 0.38

In the figure 2.1 a violin + bloxplot graph can be observed with the 50 execution times of the 10k, 50k and 100k TUs. All of them are show in a range of 1.76s (the biggest of the three) to have perspective for each one.

The execution times of the table 2.5 are the average of all of this previous values. The variance and standard deviations are really low, so the values obtained are consistent.

The same was done for the execution times of Bicleaner AI, with GPU:

- 10k:



(a) 10k TUs (b) 50k TUs (c) 100k TUs

Figure 2.1: Violin graphs of the time of 50 executions each of the classification for corpus JW300 for different number of TUs (10k, 50k and 100k) for Bicleaner classic. The same interval of 1.76s is shown for each one

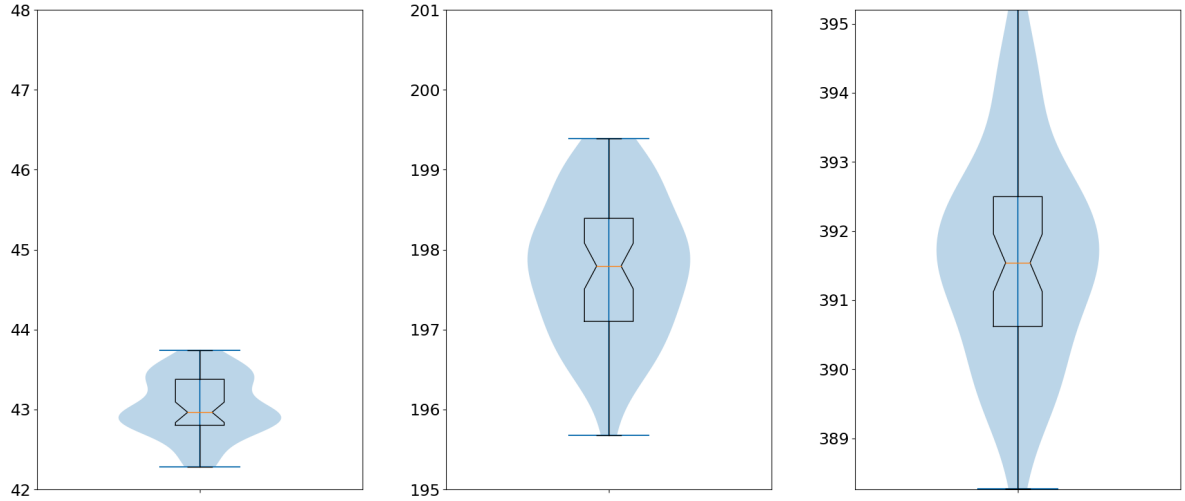
	10k	**50k**	**100k**
Classic	11.08s±0.018	41.76s±0.036	79.75s±0.054 ∼ 1min 19s
AI	2008s ∼ 33min	9866s ∼ 2h 44min	19533s ∼ 5h 25min

Table 2.5: Execution time of the classification for corpus JW300 for different number of TUs (10k, 50k and 100k) for both version of Bicleaner. The execution of the bicleaner AI version had no GPU. Approximately it is 250 times slower than the classic version.

- Average: 43.04
- Variance: 0.13
- Standard deviation: 0.36

- 50k:
  - Average: 197.77
  - Variance: 0.77
  - Standard deviation: 0.88
- 100k:
  - Average: 391.61
  - Variance: 2.27
  - Standard deviation: 1.5

In the figure 2.2 a violin + bloxplot graph can be observed with the 50 execution times of the 10k, 50k and 100k TUs. All of them are show in a range of 6s (the biggest of the three) to have perspective for each one. As it happened with the classic version in CPU, the variance in time increases with the number of TUs



(a) 10k TUs (b) 50k TUs (c) 100k TUs

Figure 2.2: Violin graphs of the time of 50 executions each of the classification for corpus JW300 for different number of TUs (10k, 50k and 100k) for Bicleaner AI. These three figures share a 6 seconds range in the vertical axis.

	10k	**50k**	**100k**
AI CPU	2008s ~ 33min	9866s ~ 2h 44min	19533s ~ 5h 25min
AI GPU	43s±0.052	197s±0.125 ~ 3min 17s	391s±0.213 ~ 6min 32s

Table 2.6: Execution time of the classification for corpus JW300 for different number of TUs (10k, 50k and 100k) for Bicleaner AI, executing with and without the GPU. Approximately it is 50x faster when using a GPU.

The table 2.6 shows differences in time when classifying with a AI model in CPU and GPU, and how beneficial is the use of the GPU when working with the AI version of Bicleaner.

2.2.6.2. OpusFilter

After analyzing the usage of bicleaner and bicleaner AI, in this section an example and analysis of usage of OpusFilter will be shown. Unlike bicleaner, OpusFilter does not have pre-trained models, so a model should be trained from the start, defining the filters and criterion that the model will have.

2.2.6.2.1. Different train and classification data

Like explained in the section 2.2.4, before training a model, the data should be pre-processed, filtered and scored, so that the model can be trained on this data, thus using these scores to know these behaviors.

All of these steps can be visualized in the repository in the file `config-train.yaml`. This file contains the steps to result a simple classifier to get the probabilities of cleanness.

In the case of OpusFilter, there is no need of having a parallel corpus beforehand, since there is a function `opus_read`, which it is used to download a corpus from

OPUS. The first step would be the `opus_read` function, where the corpus News-Commentary_v16 is downloaded from English to Spanish. The use of this particular corpus is due to the fact that for the pre-trained model for Bicleaner, the corpus used for training is also the NewsCommentary. This step returns two files with the corpus in both languages. On the other hand, the second step would be the function `subset`, to select some part of the corpus instead of the whole corpus, in this case, the subset has a size of 100k pair of sentences, this is what is normally used to train a opusFilter model. This function has as an input the English and Spanish corpus return in the previous step, and as an output also two files with the subsets for each language.

After having the corpus ready and before the training, all of this sentence pairs would need to be scored, so this scores can be used to train the model. But before this, two of the filters used need previous training, CrossEntropyFilter and WordAlignFilter, explained later. After this previous training of these filters, the sentences are scored. This step would be the function `score`, as mentioned in a previous section, would receive a list of filters, which are the following:

- LengthRatioFilter: It filters based on ratio of the segment lengths of each sentence. The parameter `unit` is used, it is the type of unit for calculating the lengths, which can be "word" for words or any whitespace-separated units, or "char" for characters. For this configuration both unit filters are used, for words and characters.

- LanguageIDFilter: It filter segments based on their confidence scores for language identification. The parameter `id_method` is used to indicate the language indentification method, which can be the `langid` library, the `cld2` library, or a `fasttext` model. In this configuration two `LanguageIDFilter` filters are used for `langid` and `cld2`. There is also another parameter `languages` where the languages in which the filter is applied, in this case, English and Spanish.

- TerminalPunctuationFilter [28]: It filters segments based on a penalty score that is determined by the occurrence of terminal punctuation marks in source and target segments. The initial score is calculated as the absolute difference between the counts of terminal punctuation marks in the source and target segments. After this, the score is increased by the number of terminal punctuation marks beyond the first occurrence in both segments. Finally, the score is updated using the formula $score = -log(score + 1)$. Having a score of 0 indicates the highest co-occurrence, while larger values indicate a greater penalty.

- NonZeroNumeralsFilter [28]: It filters segments based on the similarity measure of numbers between the segments with zeros removed. This similarity score is a result of the function `SequenceMatcher.ratio()` from Python's `difflib` library.

- CrossEntropyFilter: It filter segments by n-gram language model probabilities. The parameter `lm_params` is a list of dictionaries containing the parameters for the language models. This parameter uses two sub-parameters, `filename` to indicate the language model to use (`en.arpa.gz` and `es.arpa.gz`), and `interpolate` which is a list of the language model in ARPA format (`bg.arpa.gz`) and their weigths (0.01). The files used here are generated in the previous mentioned steps, explain here:

- `bg.arpa.gz`, `en.arpa.gz` and `es.arpa.gz` generated by steps 5, 6 and 7. These steps use `train_ngram`, which is responsible for training a character-based varigram language model using VariKN [34]. This function has 3 parameters: the data, the output generated model and two sub-parameters: `norder` for limit model order, and `dscale` for the model size scale factor (smaller values result in larger models).
- `en.arpa.gz` and `es.arpa.gz` use the filtered data generated by the step 3, which uses the function `filter`, with the whole NewsCommentary corpus as input. The function will remove the sentences that do not pass this filters, and it will return :
  - LengthFilter: It filters based on absolute segment lengths, from 1 to 100 words.
  - LengthRatioFilter: Explained before. It filters those with a ratio greater than 3.
  - LongWordFilter: It filters based on average word lengths, in this case, 40 characters as the maximum.
  - CharacterScoreFilter: It filters segments based on what proportion of their alphabetic characters are in a script, in this case, Latin. The other parameter is `threshold`, which represents the minimum proportion of characters in a script, in this case, 1.
- `bg.arpa.gz` uses the file `concatenated.gz` which is generated in step 4 with the whole corpus of NewsCommentary, joining the English and Spanish single corpora.

- WordAlignFilter: It filters segments by word alignment scores using eflomal [35]. This tool is designed to efficiently perform word alignment using the Markov Chain Monte Carlo algorithm. This filter has the following parameters:
  - `src_tokenizer` and `tgt_tokenizer` are the tokenizers for the source and target language respectively. The possible tokenizers are listed here, and in this case, `moses` is used, which uses the `fast-mosestokenizer` package.
  - `priors`: It is used to define the priors to use for the alignment. They are generated in step 8 which uses `train_alignment` to train the word alignment priors that eflomal uses [35]. It also needs the `src_tokenizer` and `tgt_tokenizer` parameters. This function returns the `align.priors` used in the filter.

When the filters are used for scoring, it does not actually filter the corpora, but returns scores for each one. This `score` function return a JSONL file with a score for each filter mentioned before for each sentence, like the following for these sentences:

- English: There is probably some slight truth – and also a certain degree of irony – to this argument.
- Spanish: Este argumento probablemente tenga algo de válido – y también un cierto grado de ironía-.

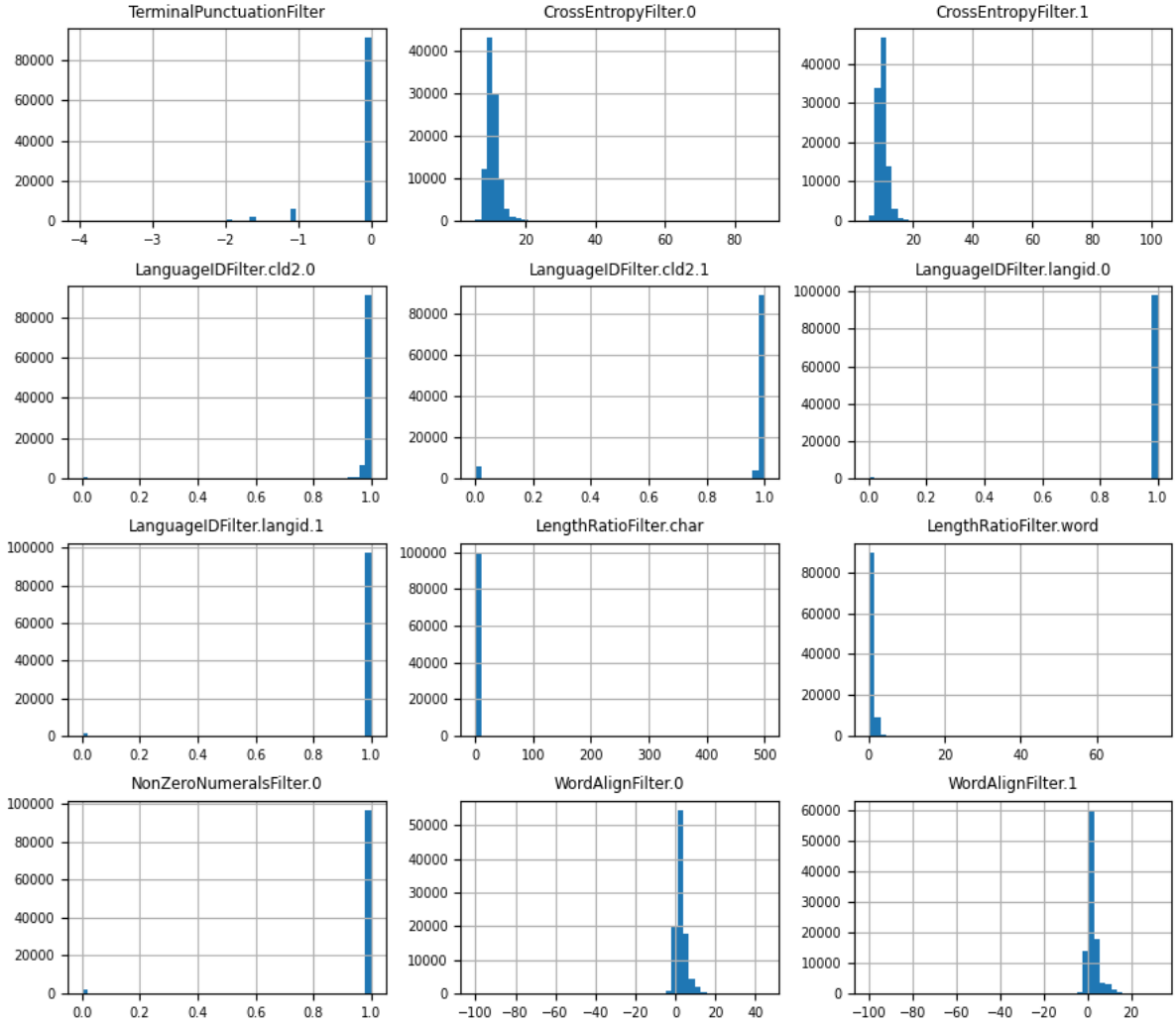

```
{
    "CrossEntropyFilter": [10.014909811809643, 11.36687484257456],
```

```
    "LanguageIDFilter": {
        "cld2": [0.98, 0.98],
        "langid": [1.0, 1.0]
    },
    "LengthRatioFilter": {
        "char": 1.0,
        "word": 1.1818181818181819
    },
    "NonZeroNumeralsFilter": [1.0],
    "TerminalPunctuationFilter": -0.0,
    "WordAlignFilter": [-1.94793, -0.0126991]
}
```

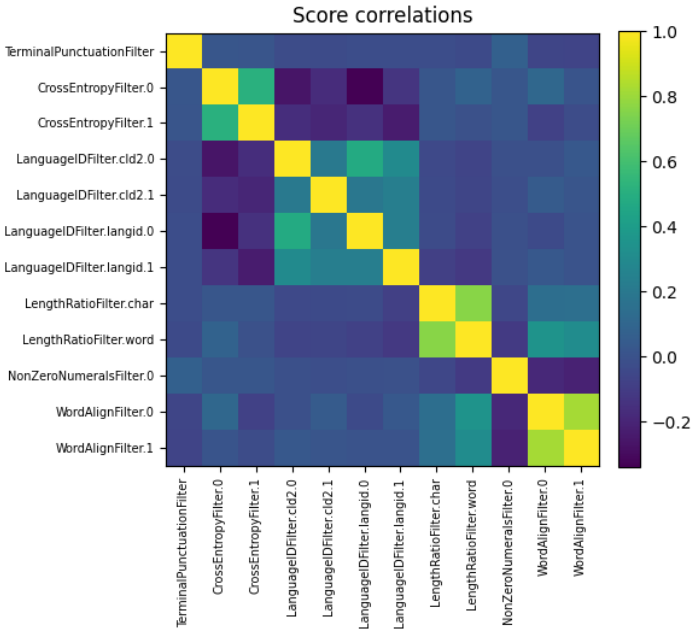

- CrossEntropyFilter similarity between the language model probabilities of the source and target sentences. A lower score indicates higher similarity. The default threshold is 50 so 10.01 and 11.36 suggest a relatively high similarity between the language models of the source and target sentences.
- LanguageIDFilter returns a 0.98 with the `cld2` library and a 1.0 for the `langid` library, this is the probability (0-1) of each sentence being from a particular language. This means that for both, the two sentences match the language, English and Spanish.
- LengthRatioFilter returns the ratio of word and character lengths between the both language sentences.
- NonZeroNumeralsFilter returns a score that indicates the degree of similarity between the numeral sequences of segments. Higher scores indicate a greater similarity between the numeral sequences, while lower scores indicate less similarity, which, in this case, indicates perfect similarity of numerals, because there are not any in both sentences.
- TerminalPunctuationFilter returns a score of -0.0, which it indicates that there is no penalty for the co-occurrence of terminal punctuation marks in both sentences. In this case, the segments have an exact match or a perfect alignment of terminal punctuation marks, resulting in the lowest possible penalty score.
- WordAlignFilter assesses the word alignment between the sentences using eflomal. Smaller scores indicate better alignment. In this case, the scores suggest relatively good alignment between the words of the English and Spanish sentences, as they are below the default threshold of 0.

OpusFilter has a separate tool called `opusfilter-scores` for the calculation and plotting statistics from scored segment pairs. This tool will give us a visual way of observing all of the scores, individually for each filter, for all of the sentences pairs, as it shows the figure 2.3.

The next step is the training of the classifier. The parameters used are these last scores, a criterion used for the classifier optimization, the Cross-Entropy was used, a logistic regression used as the model type, which is the default. The last parameter is the features, which are the same as the filters used to get the scores. As explained on a previous section, for each filter, a clean direction is set (high or low depending on each filter to indicate cleanness), and the quantiles to indicate the minimum and



(a) Histograms for the scores of each filter



(b) Correlation matrix between scores of each filter

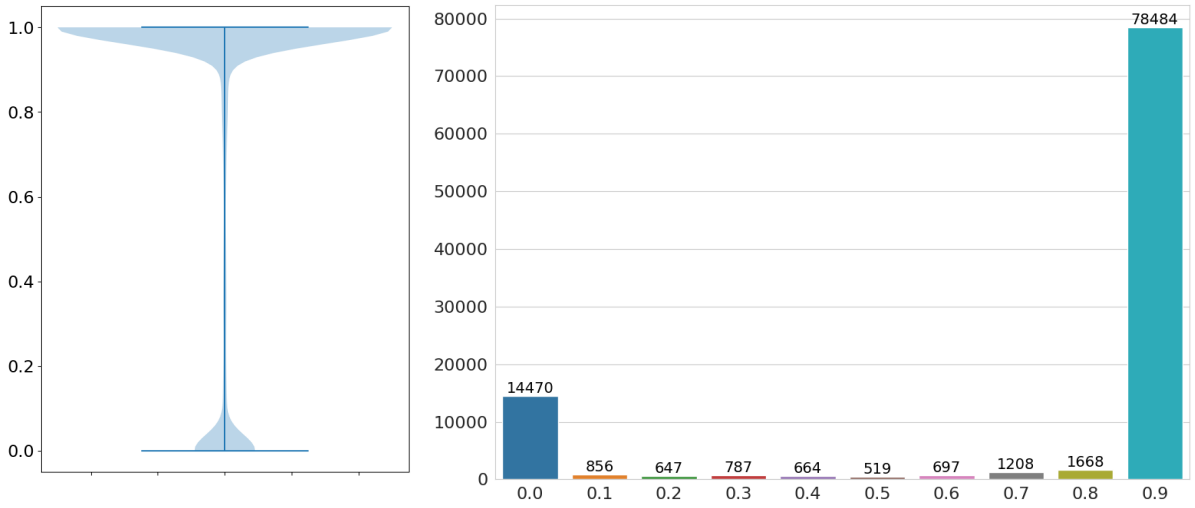
Figure 2.3: Histograms and correlations of the score values used for the training of the classifier with the NewsCommentary parallel corpus

maximum so, during the training, a selection of negative and positive examples is made.

In the end of the training process, the tool prints out this, where it shows the features cutoffs, and the weights for each filter

When all of the last steps of the training are finalized, it returns a model. Then, this model can be used to classify any corpus. As done for Bicleaner, the JW300 corpus is going to be classified through this model, shown in this file. These weights are from the trained logistic regression classifier, and the features are standardized before they are fed into the classifier (shown in this method).

As the `classify` function has as input the scores of the `score` function, the JW300 corpus is to be classified, first, it would be necessary to score it, and it has to be done the same way as the scoring done for the training, as the first step shows. Once we have the JW300 corpus scored, it can be classify with the second step, with the `classify` function, which will return, given these scores, a txt file containing the final probabilities.



(a) Violin plot

(b) Histogram

Figure 2.4: Probabilities that the classification of the Opusfilter model, trained with NewsCommentary, returns as output for the JW300 corpus

Figure 2.4 shows all the probabilities that the classification of the OpusFilter model returns with the TUs from JW300. In figure 2.4a shows a violin plot and figure 2.4b shows an histogram representing how many TUs are for each range of probabilities. As mentioned before, these probabilities indicate the probabilities of cleanness for the pair of sentences. We can see that most of the data is around 0.0 and 0.9, representing around 14% and 78% each one respectively. This indicates that the classifier is confident in its predictions. A probability of 0.9 suggests a high level of confidence that the segment is clean, while a probability of 0.0 indicates a high level of confidence that the segment is noisy.

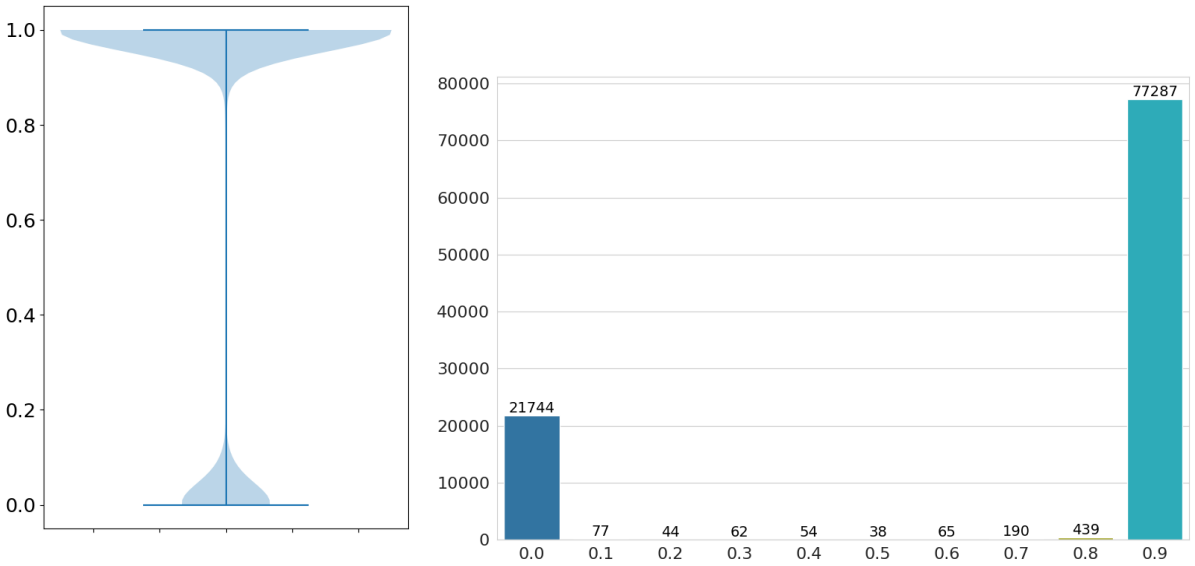
2.2.6.2.2. Same train and classification data

Although the utilization of "test" data for training purposes may seem questionable, it is important to note that the overall framework of OpusFilter is primarily unsupervised. Consequently, training the model in the same domain as the desired classification is a natural approach in this case. Because of this, the same process as the previous section will be done here, but using the same data, the JW300 corpus.

The configuration file for the training goes the same as before, but using JW300. From around the 500k TUs in total of the JW300, it is divided into two subsets, firstly `JW300-test-100k` which is the same 100k pair of sentences used in the classification before, so in this case we classify the same pair of sentences; and secondly, the rest 400k in `JW300`. This way the same corpus is used for training and classification, but not actually the same TUs.

The whole subset `JW300` (400k) is used for the training of the n_grams for the filters CrossEntropyFilter and WordAlignFilter. However, for the step of `train_classifier` an additional subset of 100k from the 400k set is randomly selected in the step 7. Figure 2.5 shows the histograms and correlation matrix for the scores of each filter.

The same configuration file for the classification as the previous section, since the same 100k pair of sentences are being classified.

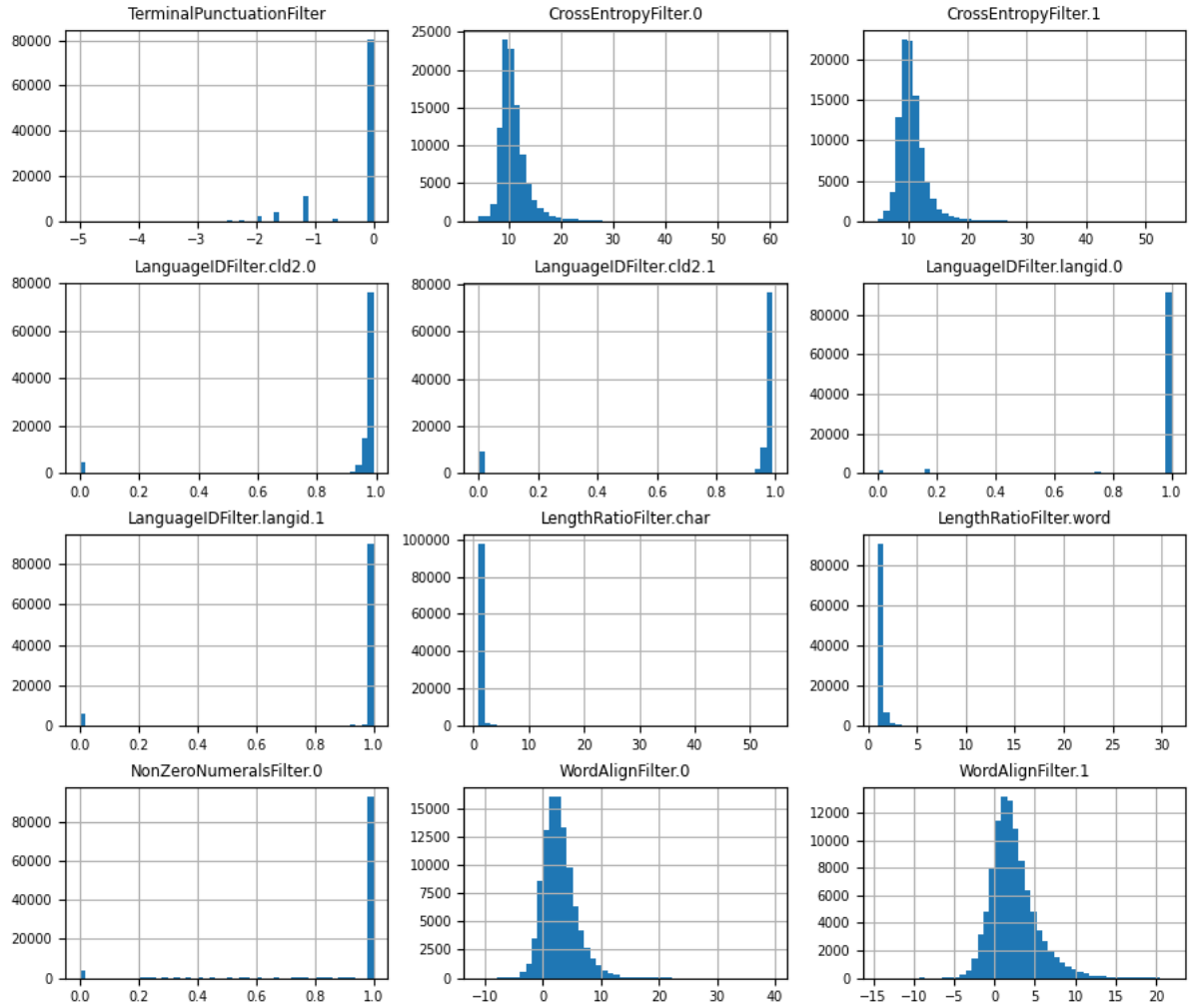


(a) Violin plot

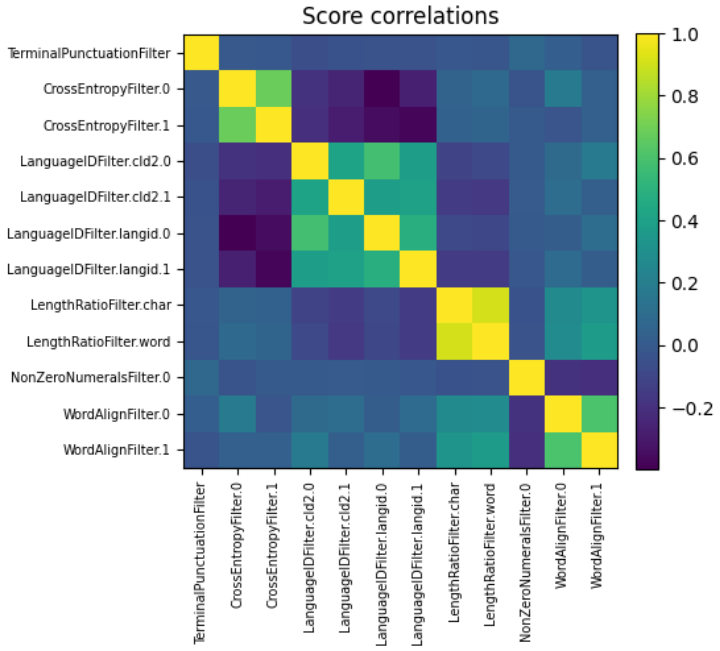
(b) Histogram

Figure 2.6: Probabilities that the classification of the Opusfilter model, trained with JW300, returns as output for the JW300 corpus

Figure 2.6 shows all the probabilities that the classification of the OpusFilter model returns with the TUs from JW300. In figure 2.4a shows the violin plot and figure 2.4b shows an histogram representing how many TUs are for each range of probabilities. As mentioned before, these probabilities indicate the probabilities of cleanness for the pair of sentences. Again, most of the data is around 0.0 and 0.9, representing in this case around 21% and 77% each one respectively, indicating that the classifier is confident in its predictions. A probability of 0.9 suggests a high level of confidence that the segment is clean, while a probability of 0.0 indicates a high level of confidence that the segment is noisy.



(a) Histograms for the scores of each filter



(b) Correlation matrix between scores of each filter

Figure 2.5: Histograms and correlations of the score values used for the training of the classifier with the JW300 parallel corpus

2.2.6.2.3. Comparison

Even tough there is not much visual difference, this second model is more confident in its probabilities, since there is less "middle" values. With the pre-trained model of Bicleaner, around 76% of TUs pass the threshold (0.5), for the first model, trained with NewsCommentary, around 82%; on the other hand, with the model trained with JW300, around 78%.

Since the second model is more accurate in the probabilities and the first one behaves differently as how it should be for some cases, we can conclude that the second model, trained in the same domain as the desired classification, works better.

As table 2.7 shows, the model trained out of domain, with the NewsCommentary corpus, misbehaves in some TUs when the JW300 corpus model does not. In the first part of the table, there are some examples where the translations are accurate and the NewsCommentary model gives them vary bad scores, and the second part of the table vice-versa, where the translations are not accurate and the model gives them good score. Meanwhile the JW300 model gives these TUs accurate scores.

2.2.6.3. Bicleaner vs OpusFilter

In this section, we present a comparative analysis of the pre-trained models: Bicleaner classic, Bicleaner AI, and OpusFilter. Despite OpusFilter exhibiting more extreme probabilities, we evaluate all models using a common threshold of 0.5, allowing us to compare the number of Translation Units (TUs) that surpass this threshold.

Utilizing the JW300 corpus, we find that the Bicleaner classic model has approximately 75% of TUs exceeding the threshold, the Bicleaner AI model demonstrates around 90%, and the OpusFilter model indicates around 80%. Additionally, the OpusFilter model aligns with the Bicleaner classic model (both exceeding or falling below the threshold) in approximately 68% of TUs. Furthermore, the OpusFilter model aligns with the Bicleaner AI model in around 75% of TUs.

As Figure 2.7 shows, it provides a visual representation of the output values for the classification of the JW300 corpus, depicting the results for the Bicleaner classic, Bicleaner AI, and OpusFilter models.

Upon analyzing Figure 2.7, we observe that the OpusFilter values exhibit greater similarity to the Bicleaner AI model, despite the latter not being as extreme. As previously mentioned, the values for the classic model are more widely distributed across all possibilities. This behavior can be attributed to the Bicleaner classic model's stricter nature, characterized by reduced flexibility in terms of sentence structure and variations in non-alphanumeric characters.

In conclusion, based on the information presented in the previous sections, the Bicleaner AI version outperforms the classic model in the case of Bicleaner. Its advantages lie in its ability to impose fewer insignificant restrictions and to consider factors beyond the character structure of sentences. Conversely, OpusFilter exhibits similarities to the AI version rather than the classic one, although with more extreme probabilities. Furthermore, OpusFilter's inflexibility stems from its lack of design for handling out-of-domain data (as it was shown in table 2.7). Considering all these factors, Bicleaner AI emerges as the preferred choice due to its capability to transcend

EN	ES	NC	JW
A Matter of Gravity	La cuestión de la gravedad	0.000	0.999
“True Knowledge Will Become Abundant”	“El verdadero conocimiento se hará abundante”	0.000	0.999
[Pictures on page 10]	[Fotografías en la página 10]	0.000	0.998
Questions in Review	Preguntas de repaso	0.000	0.715
A missionary started a study with him.	Un misionero comenzó un estudio con él.	0.000	0.944
If so, can you move mountains?	Si así es, ¿puede usted trasladar montañas?	0.000	0.999
I WAS a real enthusiast for gymnastics.	YO ERA una verdadera entusiasta de la gimnasia.	0.000	0.980
What Scriptural reasons do we have to keep on preaching?	¿Qué razones bíblicas tenemos para seguir predicando?	0.088	0.998
(Revelation 2:15) Sectarianism was taking hold.	(Revelación 2:15.) El sectarismo echaba raíces.	0.401	0.896
Releasing the Oxygen	Liberación del oxígeno	0.000	0.999
(Revelation 3:18) Such genuine wealth includes being ‘rich in fine works, liberal, ready to share.’	Es de capital importancia que obedezcan a Cristo invirtiendo en riquezas espirituales, que ‘compren de Cristo oro acrisolado por fuego’ (Revelación 3:18).	0.999	0.000
But by then the U.S. figure had risen to almost 1 car for every 2 persons.	Actualmente Alemania y Luxemburgo cuentan con 1 vehículo por cada 2 habitantes.	0.720	0.052
Effects of Marijuana	Puntos de vista de granjeros	0.999	0.000
(Ecclesiastes 7:12) What is the point?	¿Cuál es la idea?	0.999	0.000
Hence, God’s Word says: “Do not go beyond the things that are written.” (1 Corinthians 4:6) Those who truly love God will observe that counsel.—See also Proverbs 30:5, 6.	Por eso, la Palabra de Dios dice: “No vayas más allá de las cosas que están escritas”. (1 Corintios 4:6.)	0.640	0.026
Meditating on Jehovah and Jesus helps you remain zealous for the truth	Meditar en Jehová y Jesús nos ayuda a seguir sirviendo a Jehová con entusiasmo	0.999	0.000
Until such a bond is established, some may consider it better to allow the biological parent to carry out discipline.	Antes de empezar a disciplinar a su hijastro, procure ser su amigo.	0.999	0.000
We might well ask ourselves, ‘Do I feel the way David felt?’	La palabra agradabilidad denota la cualidad de ser agradable, grato o placentero.	0.999	0.000

Table 2.7: Output values of the classification of the JW300 corpus. The first column with the OpusFilter model trained with the NewsCommentary, the second column with the model trained with the JW300 corpus. It shows some example where the model trained out of domain (Newscommentary) misbehaves, while the JW model does not.

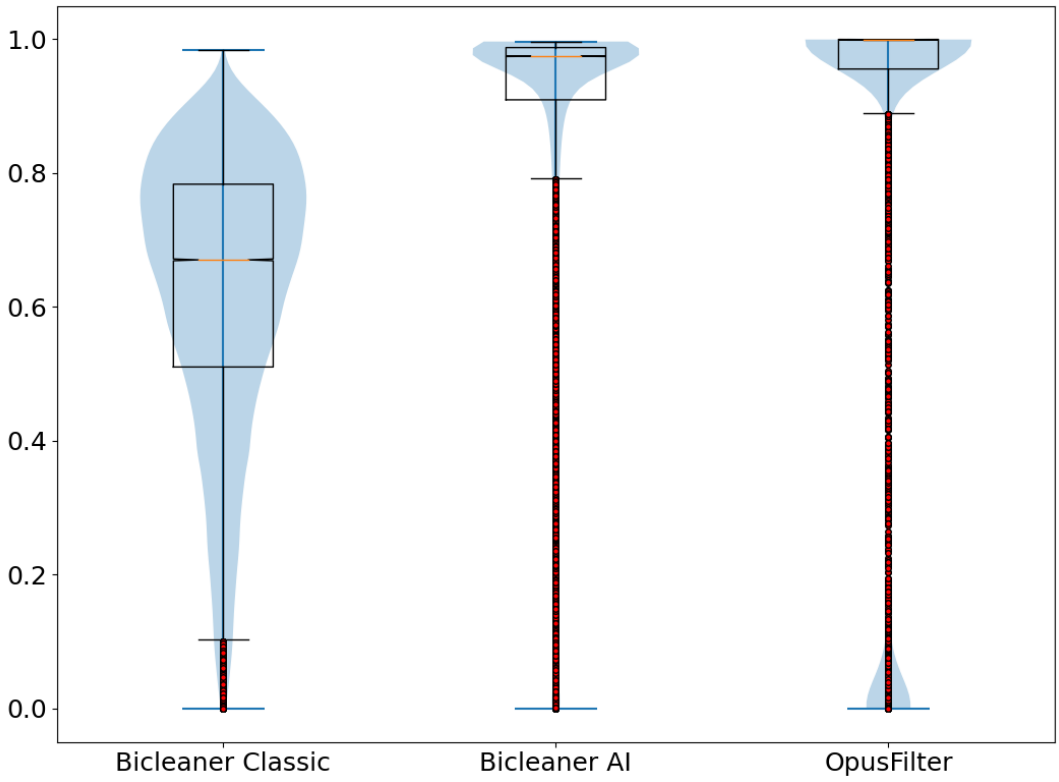


Figure 2.7: Output values of the classification for the JW300 corpus, for the Bicleaner classic, Bicleaner AI and OpusFilter models.

the restrictions that the classic model might impose and its ability to handle diverse data types effectively.

Chapter 3

Experiments

In this chapter, first of all, the training of a Bicleaner model would be explained. The first objective will be to train a model like the pre-trained one that Bicleaner offers. This way, after achieving and getting to the same start point, the search for a better one could be done, for both Bicleaner classic and AI.

Like mentioned in the state of the art, even tough Bicleaner offers pre-trained models to use (classic and full AI), it offers the training of your own model to use as classifier. Hence, in this chapter, the focus will be on training a customized model, commencing with the replication of the pre-trained model as a foundation for following progress. The languages used will be English and Spanish, therefore all the parallel corpus used will be from English to Spanish.

All of the development of this chapter for the classic version of Bicleaner will be performed in a Linux system, specifically, in Ubuntu 22.04, in a virtual environment with python 3.8. On the other hand, the Bicleaner AI version will be performed in a NVIDIA GeForce RTX 3080 GPU (10GB).

3.1. Training the base Bicleaner classic model

In this section, the steps to train a model from scratch will be explained, resulting in a model almost identical to the pre-trained one that Bicleaner has. The section will be split into five subsections, the first one, the preparation and cleaning of the corpus, the second would be the creating of the probabilistic dictionaries, thirdly the word frequency file, fourthly the training of the model, and lastly, the classification of the model comparing it with the pre-trained one.

All the steps to train a Bicleaner model from scratch are explained in Bicleaner's documentacion "How to train your Bicleaner".

3.1.1. Prepare and clean the data

The first step in the training of your own model will be the pre-processing and cleaning of the data, which is a fundamental step of the process, due to the fact that the model will be trained with this data, therefore, the cleaner the data, the better the model will be.

Firstly, we will need to obtain enough bilingual data already aligned. It would be better to used data with not to much noise and it is advisable to disregard any corpus derived from web crawling, documentation, or entity recognition. There are two options to get this data, the use of the command `mtdata`, or it can manually downloaded from OPUS. Here, the data was obtained through the second option.

Bicleaner uses for the pre-trained model the following corpus:

- Dictionaries: Opensubtitles: 4M + Eubookshop: 2M + Tilde: 2M + DGT: 2M
- Training corpus: Newscommentary: 100K

So the first step will be to download from OPUS those five parallel corpus from English to Spanish. After having all the data, the next step will be the cleanning, shown in this file:

- Step 1: Detokenize the data: the command used is `sacremoses`. This will be done for both English and Spanish corpus.
- Step 2: The two files will be combined with the command `paste` into a single file with two columns, the first one English and the second one, Spanish.
- Step 3: Use Bifixer explained in the section 2.2.1, to restore and clean the parallel corpus, orthography fixing and the identification of duplicates. The flag `-aggressive_dedup` is activated to treat similar sentences as duplicates.

  Bifixer will enhance text by replacing characters from incorrect alphabets with the correct ones, standardizing punctuation and spacing, eliminating HTML tags, rectifying common spelling errors in specific languages (in this case, both English and Spanish), filtering out sentences with empty source or target, generating hashes for parallel sentences to facilitate the removal of duplicates, and offering segmentation for lengthy sentences. The output of this will the two columns with the sentence from English into Spanish, with two additional columns with the hash and ranking:

  - Hash: It is generated using the XXHash algorithm, which is applied after combining the fixed source and target sentences. Sentences that are identical during this process will produce the same hash.
  - Ranking: This number is obtained during the hash generation process. It helps determine the best sentence among those that have the same hash, when multiple sentences share the same hash and have an identical ranking number, only one random sentence is retained. However, if the ranking numbers differ within a group of sentences with the same hash, the sentence with the highest ranking number is the one to keep.
- Step 4: The commands `sort` and `awk` will be used to delete the duplicates identified before. Since Bixifer does not eliminate any sentences, it just adds the hash and ranking, this additional command will be necessary to delete those duplicates or near duplicates. For this, the `sort` command is used to sort all the sentences given those two added columns, and after, `awk` is used to write in the output file only one of the sentences with the same hash.
- Step 5: Hardrules: explained in the previous chapter. It serves as an filtering stage to identify and eliminate obvious noise, poor language quality and vulgar

language. The output file of this command will be the same parallel corpus with an additonal column indicating 0 (to discard) or 1 (to keep). Here we have two sub steps, the one that it is used now and old version which was used for the pre-trained model, so for this first model, the sub-step 5.2 will be used:

  - 5.1: It shows how the command `bicleaner-hardrules` would be used in the current version of the tool. You would have to indicate the source and target language and a configuration (yaml file), in this file, there would be a list of all the hardrules indicating "True" or "False" to activate all, any or none.

  - 5.2: The pre-trained model was created with bicleaner v0.14 which differs in some of the hardrules. Therefore, to apply the same hardrules, the code from that version will be applied, in which, all of the default hard-rules are applied except the language model filter and the porn removal, indicated with the flags (`-disable_lm_filter` and `-disable_porn_removal`). These hardrules will be explained later, in a later section.

- Step 6: The command `awk` will be used again to discard all sentences marked by the hardrules filter, namely, all of the sentences that have a 0 in the third column.

- Step 7: The last command is `cut`. It will be used just to eliminate the extra columns generated before, returning the first two columns with the English and Spanish sentences.

All of these files generated during the cleaning are stored in this dataset. The total of sentences filtered in each step for each corpus is:

	Initial	**Bifixer 1**	**Bifixer 2**	**Hardrules**	**Total filtered**	**% filtered**
OpenSubtitles	5,000,000	4,565,267	3,769,593	3,336,355	1,663,645	33.27 %
EUbookshop	5,215,515	5,330,194	4,688,614	3,865,569	1,349,946	25.88 %
DGT	5,127,624	5,146,877	3,249,531	2,245,421	2,882,203	56.21 %
TildeMODEL	3,775,496	3,783,461	3,388,253	3,220,129	555,367	14.71 %
NewsCommentary	238,511	239,856	239,746	234,326	4,185	1.75 %
Total	19,357,146	19,065,655	15,335,737	12,901,800		

Table 3.1: Sizes of the files generated when applying Bifixer and the hardrules, total of TUs in each step, and the total of TUs filtered with each percentage respectively

The table 3.1 shows the process in the filtration of the TUs for each corpus. The step for Bifixer is separated in two. The first one, the output of the bifixer tool after fixing the text, which is most cases will result in more TUs than initially, due to the fact that during this fixing, the tool will segment long sentences, splitting a single TU into two o more. The third column, Bifixer 2, is the result of applying the next step, where the duplicates identified by Bifixer, are removed. The fourth column is the result of filtering the TUs throught the Bicleaner hardrules. Lastly, the remaining columns are showing the total of TUs that were filtered and their respective percentages. This indicates that the cleanest corpus would be NewsCommentary, while the noisiest would be DGT.

3.1.2. Probabilistic dictionaries

To create the probabilistic dictionaries, a parallel corpus consisting of several million sentences is required. As mentioned in the last section, the corpora available to create the dictionaries are: OpenSubtitles (4m TUs), Eubookshop (2m), Tilde (2m), DGT(2m). It is convenient to include smaller corpora from various domains in order to obtain a diverse vocabulary. The goal is to accumulate approximately 10 million parallel sentences. After cleaning the data in the previous section, we will split and combine all of the final files of each corpus, as it can be seen in this file. The command `shuf` is used to select a number of sentences randomly, then those four subsets are combined.

After having all the data needed to create the dictionaries, the commands in this file are used to create a CSV file. The first command `sed` is used to perform a search and replace operation. The command performs different replacements:

- `s/"/\\"/g`: This expression replaces every occurrence of a double quotation mark (") with an escaped double quotation mark (\"). So in the next step the sentences can be quoted. The g flag ensures that it replaces all occurrences within each line.
- `s/\t/","/g`: This expression replaces every tab character () with a comma enclosed in double quotation marks (","). The g flag used again for the same reason
- `s/^/"/` and `s/$/"/`: This expression adds a double quotation mark (") at the beginning of each line. The ^ symbol represents the start of a line, and the $for the end of the line.

This way we would have a comma separated file with each language sentences between double quotation marks to differentiate the commas added to separate and the ones that are already in the sentences.

For the second command `sed`, the operation being performed is a substitution (s) command with the expression `"1s/^/en,es\n/"`. This expression inserts the string `"en,es"` at the beginning of the first line in the file. Therefore, this command is modifying the file by inserting the string `"en,es"` at the beginning. And lastly, with the command `gzip`, the file it compressed in a gz file.

(repositorio privado de Tudor)

3.1.3. Word frequency files

For the trainning of a Bicleaner model are also needed two word frequency files of the source and target languages. To create theses word frequency files, it is necessary to count the occurrences of each word within a corpus. This task necessitates access to substantial monolingual corpora for both the source and target languages. The corpus used is the same combined file that was created for the dictionaries.

This file shows the steps necessary to achieve this, and it is performed through pipes:

- First selecting each column for English or Spanish, then `sacremoses` to tokenize in those cases that are previously tokenized, with the option `-x` which escapes special characters for XML.
- `awk` is used to lowercase all the lines of the corpus

- `tr` is used to replace all spaces with "\n" so the sentences will be split into as many lines as words has.
- `sort` is used to sort the input lines alphabetically. By setting the LC_ALL environment variable to "C", it ensures that the sorting is performed based on the ASCII values of characters, producing a byte-by-byte comparison.
- `uniq` filters out adjacent repeated lines in the sorted input. The -c option is used to prefix each line with the count of its occurrences, indicating the number of times each line appears in the input.
- `sort` is used again to sort the last output. The -nr options are used to perform a numeric sort in reverse order, sorting the lines by the count of occurrences in descending order.
- `grep` searches for lines in the input where there are zero or more whitespace characters followed by the number 1. The -v option negates the search, causing grep to output lines that do not match this pattern. Essentially, the command filters out lines that contain whitespace followed by 1, and only outputs lines that do not have this pattern.
- `gzip` compresses the file resulting in all the past commands.

All of this is done for both monolingual corpus, resulting in two gz files containing the word frequencies of English and Spanish given the 10 millions sentences.

3.1.4. Training the model

After having all the data cleaned for the creation of the probabilistic dictionaries and the word frequency files, a Bicleaner model can be trained. Like mentioned in a previous section, the corpus used for the training of the model is the NewsCommentary, and after preparing and cleaning the same way as the rest of the corpus, with this command 100k pairs of sentences are randomly selected.

This file shows the command used to train to model with the following parameters:

- `-normalize_by_length`: It normalize by length in qmax dict feature
- `-s` and `-t` to indicate the source and target language
- `-d` and `-D` to indicate the dictionaries of the source and target language respectively.
- `-b` to indicate the block size, namely, sentence pairs per block, in this case 1000.
- `-c`: where the classifier data is stored after training.
- `-f` and `-F` to indicate the word frequency files of the source and target language respectively.
- `-m`: where the model configuration data will be stored after training.
- `-lm_file_sl` and `-lm_file_tl`: Output files with the generated source and target language models.

After all this, a Bicleaner model would be trained to use as classifier for any parallel corpus.

3.1.5. Classification

All of the data trained and generated previously can be seen in the directory "modelo base" in this directory of my repository.

After this, we will classify the corpus JW300. After executing the classification explained in the previous section, it results in this output file.

Then the classified file from the pre-trained model is compared to the scores returned from the model we just trained. The scores are mostly almost the same. It would be impossible to have the exact scores, due to the fact that the subsets of corpus used for the training are selected randomly, so, even though the same corpus were used, not the exact same pair of sentences. Because of this, it is normal to have differ a lit of bit in the scores.

In the table 3.2 we can see some values from the classification. In the first column we have the pre-trained model scores and in the second column the score returned by the model previously trained.

Also figure 3.1 shows all the classification scores of the pre-trained classic model vs. my just trained model. It shows how similar they behave, which was the objective.

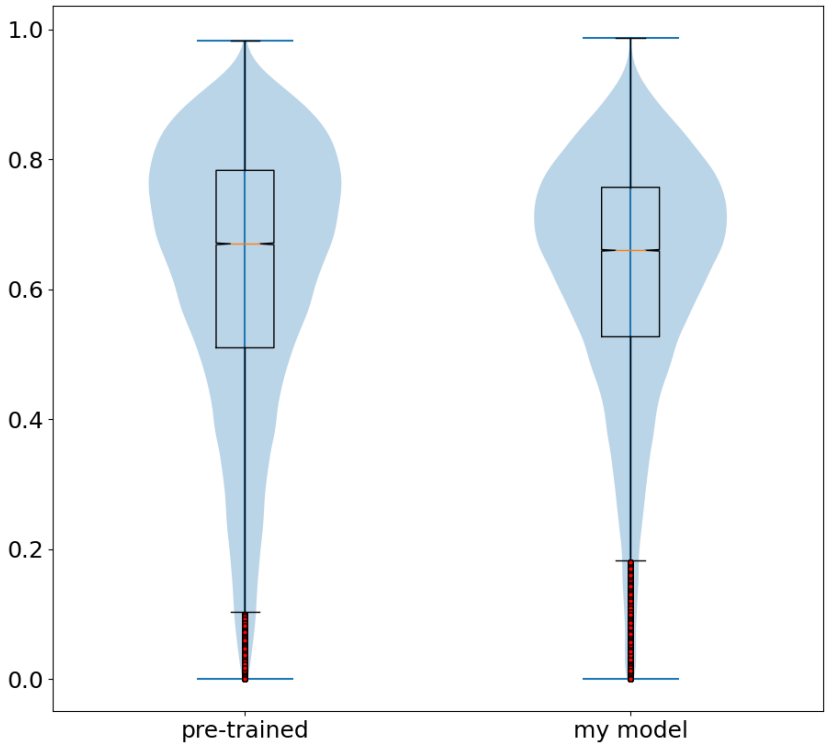


Figure 3.1: Violin+box plot of the classification scores of the pre-trained classic model and my base model.

After achieving the same point as the pre-trained model of Bicleaner, the goal will be to better this, trying to improve the existing model.

EN	ES	Pre-trained	My model
30 The Almost Indestructible Water Bear	30 El “oso de agua”: criatura casi indestructible	0.753	0.750
It has been said that temptations will certainly come to your door, but you are to blame if you invite them to stay for dinner.	Se ha dicho que las tentaciones ciertamente vendrán a la puerta de uno, pero que uno tiene la culpa si las invita a entrar y pasar un rato.	0.450	0.433
2 The man was the apostle Paul, without a doubt an effective and progressive minister.	2 Se trata del apóstol Pablo, sin duda un ministro eficiente e ingenioso (1 Timoteo 1:12).	0.630	0.627
Can any genuine comfort be derived from a pagan philosophy?	¿Puede derivarse algún consuelo genuino de una filosofía pagana?	0.827	0.813
United States Senator John L. McClellan, during an interview on reasons why crime keeps rising in the land, spoke in a similar vein:	El senador estadounidense John L. McClellan, durante una entrevista sobre razones por las cuales siguen aumentando los crímenes en ese país, habló de manera similar:	0.757	0.777
Robert Luccioni	Robert Luccioni	0.487	0.460
Now I really felt I had nowhere to go, and nobody to turn to.	Fue en aquel momento cuando verdaderamente sentí que no tenía a dónde ir ni a quién acudir.	0.593	0.583
Monuments in the famous 2,500-year-old Acropolis of Athens “have been subjected to the attacks of armies, vandals and trophy-hunters, in peace and in war,” says Greece—A Monthly Record.	Los monumentos de la famosa Acrópolis de Atenas que cuenta con 2.500 años de antigüedad “se han visto sometidos al ataque de ejércitos, vándalos y cazadores de trofeos, tanto en tiempo de paz como en tiempo de guerra,” dijo Greece—A Monthly Record.	0.877	0.887
You can see, then, why many persons today really do not want God’s kingdom to rule over them.	Se puede ver, pues, por qué muchas personas hoy día realmente no quieren que el reino de Dios las gobierne o rija.	0.760	0.773
Clearly, then, a dependable standard is needed to help the conscience to assess matters properly.	Claro está, pues, que se necesita una norma confiable para ayudar a la conciencia a justipreciar apropiadamente los asuntos.	0.813	0.783
It’s a nudibranch (pronounced nōōdē-brank).	Es un nudibranquio.	0.180	0.080
Every so often stretches of the original paving stones can be seen in this section, worn but intact after 2,000 years!	En esta sección de la carretera, de vez en cuando se pueden ver trechos con piedras del pavimento original, ¡gastadas, pero enteras, después de 2.000 años!	0.567	0.607
The declaration that “Millions now alive will never die off our earth” is not based on what, but what grounds are there for expecting this?	¿En qué no se basa la declaración “millones que ahora viven nunca morirán de sobre la Tierra”, pero qué base hay para esperar que esto se realice?	0.670	0.647
The weather could not have been better.	El clima fue ideal para la ocasión.	0.430	0.483

Table 3.2: Output values of the classification of the JW300 corpus. The first column with the pre-trained model and the second column with the model I trained

3.2. Training the base Bicleaner AI full model

In this section, the steps to train a AI full model from scratch will be explained, resulting in a model very similar to the pre-trained one that Bicleaner AI has. It is not known exactly what were the steps of the preparation of the data on the training of the pre-trained model, so two models will be trained here. On one hand, the first model will be trained with the same exact data as the pre-trained (obtained directly from the developers), however, the preparation of this is not known, because of this, a second model will be trained, with the same initial corpora but applying the default preparation of the rules, so it can be shown all the steps from the beginning.

The training of this model was performed in a NVIDIA GeForce RTX 3080 GPU (10GB).

The section will be split into four subsections, the first one, the preparation and cleaning of the corpus, the second would be the creating of the word frequency file, thirdly training of the model, and lastly, the classification of the model comparing it with the pre-trained one.

3.2.1. Prepare and clean the data

For the pretrained AI full model the following corpora was used OPUS_TildeMODEL_v2018, JW300, OPUS_Europarl_v8, neulab_tedtalksv1_train, OPUS_OpenSubtitles_v2018, OPUS_GlobalVoices_v2017q3, news_commentary_v14, OPUS_UNPC_v1_0, OPUS_SciELO_v1. This can be downloaded with `mtdata`. The initial total of pair of sentences that each of them have can be seen in the table 3.3

The preparation and cleaning of the data is the same as the one done for the classic model, explained in section 3.1.1, and shown in this file

In the first step, the data will be detokenized, to deal with those cases in which the corpus is tokenized. Then, in step 2 the single corpora in English and Spanish are combined into a single file. The third step is Bifixer to restore and clean the parallel corpus, orthography fixing and the identification of duplicates. The forth step was to delete those duplicates idetifies by Bifixer. The fifth step was the Bicleaner Hardrules, with the sixth step being the removal of all the sentences that did not pass the hardrules.

	Initial	Bifixier 1	Bifixier 2	Hardrules	Total filtered	% filtered
TildeMODEL	3,775,496	3,783,461	3,388,253	3,220,129	555,367	14.71 %
JW300	100,000	103,395	98,480	92,943	7,057	7.06 %
Europarl	2,009,073	2,032,947	1,958,596	1,937,248	71,825	3.58 %
TedTalk	210,181	223,919	222,669	208,869	1,312	0.62 %
OpenSubtitles	5,000,000	4,565,267	3,769,593	3,336,355	1,663,645	33.27 %
GlobalVoices	693,544	701,260	687,571	650,231	43,313	6.25 %
NewsCommentary	238,511	239,856	239,746	234,326	4,185	1.75 %
UNPC	4,000,000	4,095,380	3,998,365	1,307,396	2,692,604	67.32 %
SciELO	433,695	441,326	421,146	409,191	24,504	5.65 %
Total	16,460,500	16,186,811	14,784,419	11,396,688		

Table 3.3: Sizes of the files generated when applying Bifixer and the hardrules, total of TUs in each step, and the total of TUs filtered with each percentage respectively

As explained in the preparation of data for the classic model, the two columns of

Bifixer represent, on one hand, the fixing of each corpus, and on the other, the removal of the duplicates. Then, the hardrules column after filtering all the corpora through them, and lastly the total of TUs filtered. In this case, the cleanest corpus would be TedTalks, while the noisiest would be UNPC.

After all the cleaning, all the corpora is combined, then around 10 millions pair of sentences are selected randomly.

3.2.2. Word frequency files

When training a AI full model, a word frequency file is also needed, but not the same way as the classic model. Bicleaner classic uses two word frequency files of the source and target language. However, Bicleaner AI, only needs the word frequency of the target language in order to use synthetic word frequency based noise when training. The corpus use for this is the one generated in the last section, with ∼10m TUs.

As done for the classic model, this command will generate the word frequency file for the target language, in this case, Spanish. The explanation of the command is explained in the section 3.1.3 since the retrieval of this word frequency file is done the same way.

3.2.3. Training the model

3.2.3.1. Training with my cleaned data

After having all the data cleaned and the creation of the word frequency file, a Bicleaner AI full model can be trained. In contrast to the classical model, which utilizes a single corpus for training, Bicleaner AI uses the entirety of all preprocessed and cleaned corpora for the training.

Two corpus will be needed for the training of this model. On one hand, ∼ 600k-700k pair of sentences are used for the training, and ∼ 2k for the development.

In this file is the command for the training of the model, with the following parameters:

- `-classifier_type` to indicate the neural network architecture of the classifier, in this case, XLMR.
- `-batch_size` for the training. The value used for the model was 128, but because the GPU only has 10GB, this was an issue. Therefore, the batch size used was 16.
- `-steps_per_epoch` to indicate how many steps each epoch of the training is going to have, in this case, 3000 steps per epoch.
- `-epochs` to indicate the number of epochs the training will have, in this particular instance, 15 epochs.
- `-patience` to indicate hen to stop training when validation has stopped improving after this number of epochs, here 3 was used as patience.
- `-m` to indicate the directory were the metadata, classifier and SentencePiece models will be saved.

- `-s` and `-t` for the source and target language, English and Spanish.
- `-F` to indicate the word frequency file for the target language, found here
- `-parallel_train` to indicate the file containing the parallel corpus to train the classifier.
- `-parallel_valid` to indicate the file containing the parallel corpus for development.
- `-lm_file_sl` and `-lm_file_tl` to indicate the source and target language model output files.

This file shows the output throughout the training process, with the following tracking of these values:

- Loss: refers to the value of the loss function, which measures the discrepancy between the predicted outputs of the model and the actual labels. The goal of training is to minimize this loss, indicating a better alignment between predictions and true labels.
- F1: F1 score is a metric commonly used in classification tasks, which combines precision and recall. It measures the balance between the model's ability to correctly identify positive instances (precision) and its ability to capture all positive instances (recall). Higher F1 scores indicate better overall performance.
- MCC (Matthews Correlation Coefficient): MCC is a metric that assesses the quality of binary (two-class) classifications, taking into account true positives, true negatives, false positives, and false negatives. It ranges from -1 to 1, with 1 indicating a perfect prediction, 0 indicating random prediction, and -1 indicating total disagreement between predictions and true labels.
- val_loss: refers to the loss function value calculated on a separate validation dataset. This validation dataset is not used for training, but rather for evaluating the model's performance on unseen data. This data would be the corpus in the `-parallel_valid` parameter of the training.
- val_f1: represents the F1 score calculated on the validation dataset. It provides an estimate of the model's performance on unseen data, again with the data of `-parallel_valid`
- val_mcc: Matthews Correlation Coefficient calculated on the validation dataset (`-parallel_valid`). It assesses the model's performance in terms of binary classification accuracy on unseen data.

The training for 15 epochs lasted 6000 seconds (1h:40m). Figure 3.2 shows the progression of all of these values throughout the training. It can be easily seen that the model stabilizes soon. After finishing the epochs, also visible in this file, there is an output of the following following metrics for the test data:

- Precision: 0.841 - Precision measures the proportion of true positive predictions (correctly identified positive instances) out of the total predicted positive instances. A higher precision indicates a lower rate of false positives.
- Recall: 0.883 - Recall, also known as sensitivity or true positive rate, measures

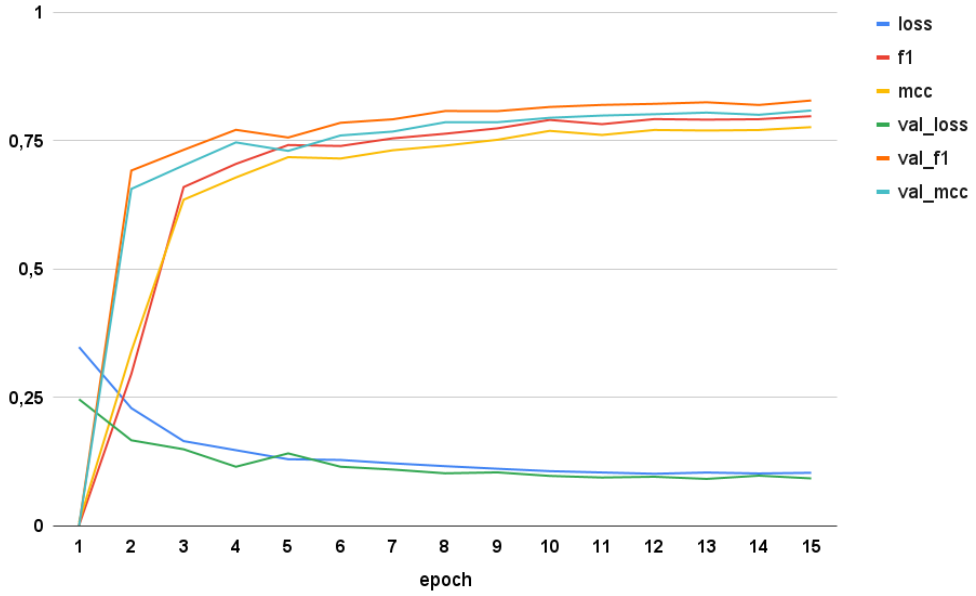


Figure 3.2: Progression of the training of the Bicleaner AI full model, with a XMLR as the neural network architecture of the classifier, using as data the cleaned corpora done in section 3.2.1

the proportion of true positive predictions out of the total actual positive instances. A higher recall indicates a lower rate of false negatives.

- F1 Score: 0.862 - F1 score is the harmonic mean of precision and recall. F1 score combines precision and recall into a single metric, where a higher value represents better overall performance.
- MCC: 0.846 - MCC is a metric commonly used in binary classification tasks that takes into account true positives, true negatives, false positives, and false negatives. Higher MCC values indicate better performance.

All of theses metrics are applied to the development corpus, as mentioned before, to the one provided by the parameter `-parallel_valid`. Overall, the training log values demonstrates a progressive improvement in the model's performance throughout the training process, with decreasing loss and increasing F1 score and MCC. The final performance metrics on the validation set indicate a well-performing XMLR classifier.

3.2.3.2. Training with the pre-trained model data

The process of this training will be exactly the same as the last section. The figure 3.4 shows the progress of the values throughout the training, and this file shows the actual output of the training.

For this model, the progression of the values is very similar to the last section, with the training of the model with my cleaned data of the same corpora. As said before, this cleaning was the default filtering, so the original filtering of the data to obtain the corpus used here, does not differ too much, given that the same set of corpora has indeed been used.

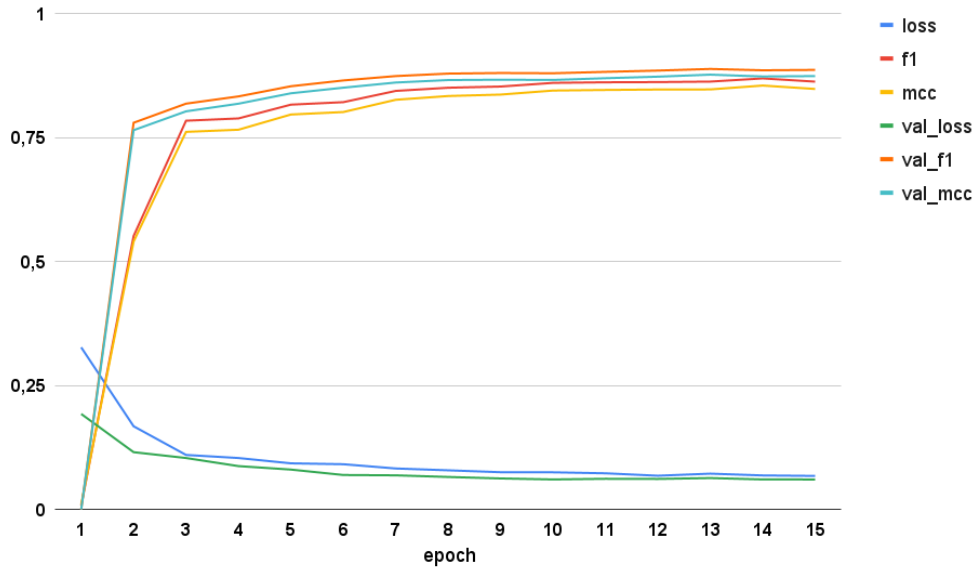


Figure 3.3: Progression of the training of the Bicleaner AI full model, with a XMLR as the neural network architecture of the classifier, using as data, the corpus that was used for the training of the pre-trained full AI model

3.2.4. Classification

All of the data trained and generated previously can be seen in the directories inside this directory of my repository.

After this, we will classify the corpus JW300 with both models. Both output classified files are stored in this two files (first and second).

Then the classified file from the pre-trained model is compared to the scores returned from the models we just trained. The scores are mostly almost the same with both of them. It would be impossible to have the exact scores, due to the fact that the subsets of corpus used for the training are selected randomly, there is no guaranty the filtering of the data is the same. Because of this, it is normal to have differ a lit of bit in the scores.

In the table 3.4 we can see some values from the classification. In the first column we have the pre-trained model scores, in the second column the score returned by the model previously trained with the data I cleaned, and the third, the model trained with the same data as the pre-trained one.

For the case of the first model (trained with my cleaned data), 91% of the scores of the TUs differ in 0,15 or less. For the case of the other model, a 90%. Therefore, due to this fact, one could assert that the models are very similar.

After achieving the same point as the pre-trained model of Bicleaner, the goal will be to better this, trying to improve the existing model.

EN	ES	Pre-trained	My data	Direct data
It has been said that temptations will certainly come to your door, but you are to blame if you invite them to stay for dinner.	Se ha dicho que las tentaciones ciertamente vendrán a la puerta de uno, pero que uno tiene la culpa si las invita a entrar y pasar un rato.	0,987	0,955	0,949
Shock Treatment	Hábito perjudicial	0,141	0,106	0,055
They appreciate that the overseers are trying to strike a proper balance between evangelizing, shepherding and teaching.	Comprenden que los superintendentes tratan de lograr un equilibrio apropiado entre evangelizar, pastorear y enseñar.	0,991	0,965	0,883
SOME EXPRESSIONS EXPLAINED	¿QUÉ SIGNIFICA?	0,338	0,301	0,235
Later, when Abraham and his three visitors were viewing Sodom from an elevated location, two left to visit the city.	Después, Abrahán y sus tres visitantes contemplaron a Sodoma desde un lugar alto, y dos de los visitantes partieron hacia aquella ciudad.	0,985	0,949	0,966
Here the assembly in the capital, Lomé, turned out to be of only one day's duration, but nevertheless a very successful one.	Aquí la asamblea en la capital, Lomé, resultó ser de solo un día de duración, pero de mucho éxito.	0,983	0,913	0,922
Especially so if it seems that your sibling—the oldest, the youngest, the best-behaved, or even the most disobedient—is in the spotlight all the time.	Tal vez llegues a sentirte como David cuando escribió: "Como alguien muerto y no en el corazón, he sido olvidado; he llegado a ser como un vaso dañado".	0,002	0,008	0,006
Generally it suffices to keep it at from 140 degrees to 160 degrees Fahrenheit.	Por lo general es suficiente mantenerlo de 60 a 71 grados centígrados.	0,899	0,761	0,646
Some women refuse to accept the headship of their husbands.	Otros cónyuges son víctimas de maltrato.	0,194	0,246	0,002
She teaches Sunday school and felt that this book would be a boost to her class. She said she would let me know later how many more copies she would be needing.	Como enseña en la escuela dominical, pensaba que este libro le sería útil para su clase y me comentó que más adelante ya me diría cuántos ejemplares más necesitaba.	0,972	0,873	0,799
And, in accord with God's own justice, he did not exempt from punishment even the king.	Y, en armonía con la propia justicia de Dios, no eximía de castigo ni siquiera al rey.	0,975	0,891	0,922
Such singing, accompanied by an instrument such as a guitar or a piano or by the Society's piano recordings, gives a spiritual tone to our gatherings.	Hacerlo, acompañados de un instrumento como la guitarra o el piano o utilizando las grabaciones de piano de la Sociedad, da un toque espiritual a nuestras reuniones.	0,716	0,698	0,727

Table 3.4: Output values of the classification of the JW300 corpus. The first column with the pre-trained model, the second column with the model I trained with the data that I cleaned and lastly the other model I trained with the direct data used for the training of the pre-trained model

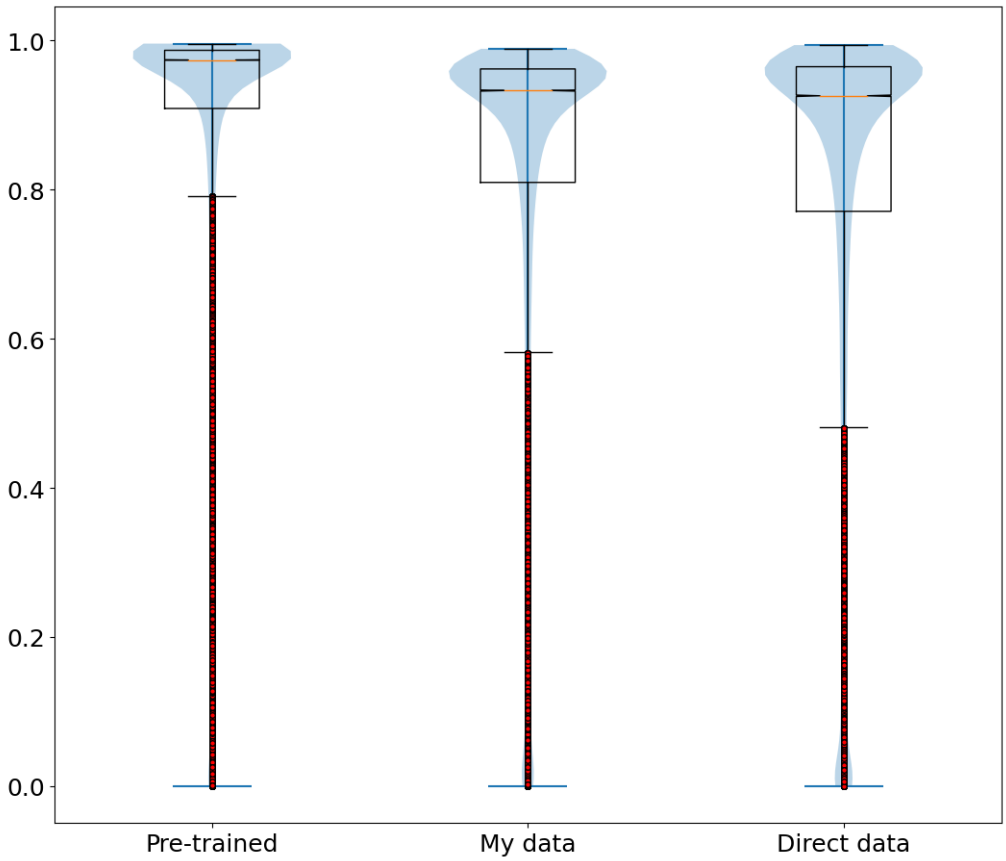


Figure 3.4: Violin+box plot of the classification scores of the JW300 corpus, with a XMLR as the neural network architecture of the classifier, using as data, the corpus that was used for the training of the pre-trained full AI model

3.3. Improvements on Bicleaner classic model

Upon reaching the starting point of matching the pre-trained Bicleaner classic model, the aim is to attempt its enhancement by modifying various training parameters and configurations, or even the data used for training.

3.3.1. Using AI model data to train classic model

First, since the data in the Bicleaner AI model is much more diverse than that of the classical model, we will begin by training the classical model using these values. This way, both the classical model and the AI model will be trained on the same data. This corpora was formed combining the following: TildeMODEL, JW300, Europarl, TedTalk, OpenSubtitles, GlobalVoices, NewsCommentary, UNPC and SciELO.

As usual, the initial step involves pre-processing and cleaning all the data. However, this step will be skipped since we will be utilizing the corpora from the AI model, which detailed instructions for this process can be found in section 3.2.1. All of these TUs, separated and the single file combined, are stored here.

After the cleaning process, in the case of a classical model, the next step was to create probabilistic dictionaries and word frequency files. However, for the AI model, only

the target language word frequency file is required. Therefore, it will be necessary to create probabilistic dictionaries for both languages and the source language word frequency file. This will be done in the same manner as explained in sections 3.1.2 and 3.1.3. Both dictionaries and both word frequency files are store in this directory.

Regarding Bicleaner classic, the training corpus used was NewsCommentary. In this scenario, a different approach will be employed, similar to that of the AI model. Specifically, a random selection of approximately 100k sentence pairs will be made from the combined corpus, combining all the datasets, found here.

As explained in section 3.1.4, the training of a classic model is the execution of the command `bicleaner-train`. The training will be conducted with the same parameters as those used in the previous section, with the only modification being the corpus used.

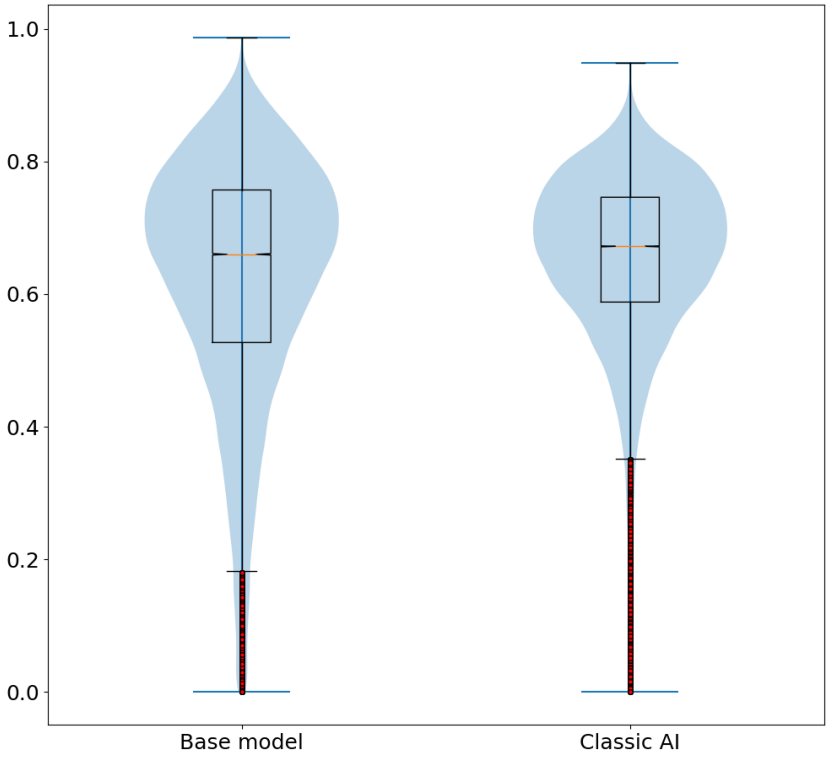


Figure 3.5: Violin+box plot of the classification scores of the JW300 corpus of the classic model trained with the data of the AI model, on the right. On the left the base classic model created in section 3.1.

Figure 3.5 shows the output values of the classification scores of the JW300 corpus of the classic models, one being the base model created before and the other one the one trained with the new data. It shows how more TUs pass the threshold, in the case of the base model around 77% sentences are above 0.5, while the classic AI model has around 88% TUs going over that threshold. The table 3.5 show some examples where the model with the AI data outperforms the base model.

With this change of data, having the model trained with a wider set of corpus, we managed to make the classic model a little bit less strict, which is better due to some unimportant restrictions that the base model has that make accurate TUs fail.

EN	ES	Base model	Classic AI
WHEN I was a child, clergymen prayed fervently for peace, but when the second world war broke out, they prayed for victory.	EN AQUELLOS años, los ministros religiosos rogaban a Dios que reinara la paz; pero al estallar la segunda guerra mundial, empezaron a rogar por la victoria.	0.427	0.754
Since a woman who is pregnant is more likely to hemorrhage, heeding warning signals assumes a still greater importance.	Como las mujeres embarazadas tienen más posibilidades de sufrir hemorragias, el prestar atención a los avisos del cuerpo adquiere aún más importancia.	0.480	0.748
According to Gerhard Friedrich, “runaway slaves who were caught used to be branded in their foreheads.	Según Gerhard Friedrich, “a los esclavos fugitivos que detenían los marcaban en la frente con hierro candente.	0.487	0.814
And vital to obedience is godly fear, yes, fear of displeasing God.	Y para obedecer es vital que tengamos temor piadoso, sí, temor de desagradar a Dios.	0.377	0.704
These large areas of “conflicting regional interests” were often the consequence of military conquest, since kings were invariably military leaders.	Dichas zonas extensas con ‘intereses regionales contrapuestos’ solían obedecer a conquistas, pues los reyes siempre eran caudillos militares.	0.423	0.712
If the criminal willingly conforms to punitive orders and exhibits changes in his attitude and behavior, a judge or president may choose to pardon him by lessening his sentence or totally forgiving his sentence.	En algunos países, los jueces y algunos funcionarios de alto rango tienen potestad para conmutar la pena a un delincuente —o incluso indultarlo— si este acepta su castigo y demuestra buena conducta.	0.087	0.654
In order to make ends meet, some mothers turn to prostitution and sell illegal drugs or encourage their daughters to do so.	Para subsistir, algunas se prostituyen y trafican con drogas, o empujan a sus hijas a tales actividades.	0.247	0.574
Our power of reason also tends to shy away from things that seem hopelessly vague and undefinable.	La razón también tiende a descartar todo lo que parece totalmente vago e indefinible.	0.407	0.690
When the first tremor subsided, they rushed outside and saw each other. Together they ran to higher ground.	Cuando pasó el terremoto, salieron corriendo de sus casas, se encontraron en la calle y huyeron a un lugar alto.	0.303	0.632
Indeed, kind parents strive to discern what discipline works best for each of their children.	Sin duda, los padres bondadosos tratan de descubrir cuáles son las medidas que funcionan mejor con cada hijo.	0.487	0.674
Others favor retaining the three-line form, making the middle one slightly longer.	Otros prefieren conservar la forma de tres versos, pero alargan un poco el segundo.	0.337	0.710
She used to bring substantial profit to her masters by fortune-telling.	Era una esclava, y ganaba mucho dinero para sus dueños, adivinando.	0.250	0.566
Later on, though, I began to feel that maybe I had been a little extreme.	Pero luego me puse a pensar si no me habría ido al otro extremo.	0.390	0.706

Table 3.5: Differences in output values of the classification of the JW300 corpus, comparing the base model I trained and the same classic model but trained with the data used for the AI model.

Having this new and wider corpora could help in areas like:

- Increased coverage: could allow the model to encounter a more diverse range of linguistic patterns, sentence structures, and vocabulary.
- Improved robustness: the model could gain exposure to different genres, domains, and writing styles.
- Enhanced contextual understanding: could provide the model with a richer contextual understanding of language.

All of these points could be some of the reasons why the model with the wider corpus has better performance.

3.3.2. Types of classifier

One of the parameters that facilitates the training of a classic Bicleaner model is the choice of classifier. Up until now, the Extremely Randomized Trees (Extra Trees) classifier has been examined, which is the default option. However, this is not the sole choice available. The following types of classifiers can be chosen:

- SVM (Support Vector Machines) [36]: SVM is a supervised learning algorithm used for classification. It constructs hyperplanes to separate different classes in high-dimensional data. SVM uses the `SVC` implementation. It is configured with parameters like gamma and C for optimal separation and probability estimation, respectively.
- NN (Nearest Neighbor) [37]: The Nearest Neighbor classifier is a non-parametric algorithm used for classification. It assigns a new instance to the class of its nearest neighbors in the training data. In Bicleaner, the `KNeighborsClassifier` is used, where the number of neighbors (k) is set to 5. This allows Bicleaner to classify instances based on the similarity to their nearest neighbors.
- NN1 (Nearest Neighbor 1): NN1 is a variant of the Nearest Neighbor classifier that considers only the closest neighbor for classification. It also uses the `KNeighborsClassifier` with k=1, namely, only uses its nearest neighbor
- AdaBoost (Adaptive Boosting) [38]: AdaBoost is an ensemble learning method that combines multiple weak classifiers to create a strong classifier. In Bicleaner, the `AdaBoostClassifier` is used with 100 weak classifiers.
- Random Forest [39]: Random Forest is an ensemble learning technique that builds multiple decision trees and combines their predictions. In Bicleaner, the `RandomForestClassifier` is employed with 200 decision trees.
- Extra Trees [40]: Extra Trees, also known as Extremely Randomized Trees, is another ensemble learning method similar to Random Forest. In Bicleaner, the `ExtraTreesClassifier` is utilized, constructing 200 extra randomized decision trees.

Next, different models of Bicleaner Classic will be trained, using all of these classifiers, using this file. A little modification had to be made to the code of the Bicleaner tool, due to the fact that the function `svm.SVC()` for the SVM classifier was outdated. In the official code the parameter `n_jobs` was still being used for that function, but it is deprecated, so it was deleted for the training.

Classifiers	Training time	Classification time
Adaboost	230s ∼ 3min 50s	66s ∼ 1min 6s
NN1	200s ∼ 3min 20s	113s ∼ 1min 53s
NN	199s ∼ 3min 19s	114s ∼ 1min 54s
SVM	4870s ∼ 1h 21min 10s	242s ∼ 4min 2s
Random Forest	2124s ∼ 35min 24s	84s ∼ 1min 24s
Extra Trees	1349s ∼ 22min 29s	100s ∼ 1min 40s

Table 3.6: Execution times of the training and classification (100k TUs) of the different Bicleaner classic models with the different classifiers.

Table 3.6 shows the different executions times for the training and classification for the different classifiers. Even though these times could potentially change depending the characteristics of the computer where it was executed, there are significant differences between them . For instance, the training of the SVM takes significantly more time than the rest, while the Adaboost, NN1 and NN takes very little time. On the other hand, for the classification of the SVM also take more than the rest, while the rest do not have that much of a difference.

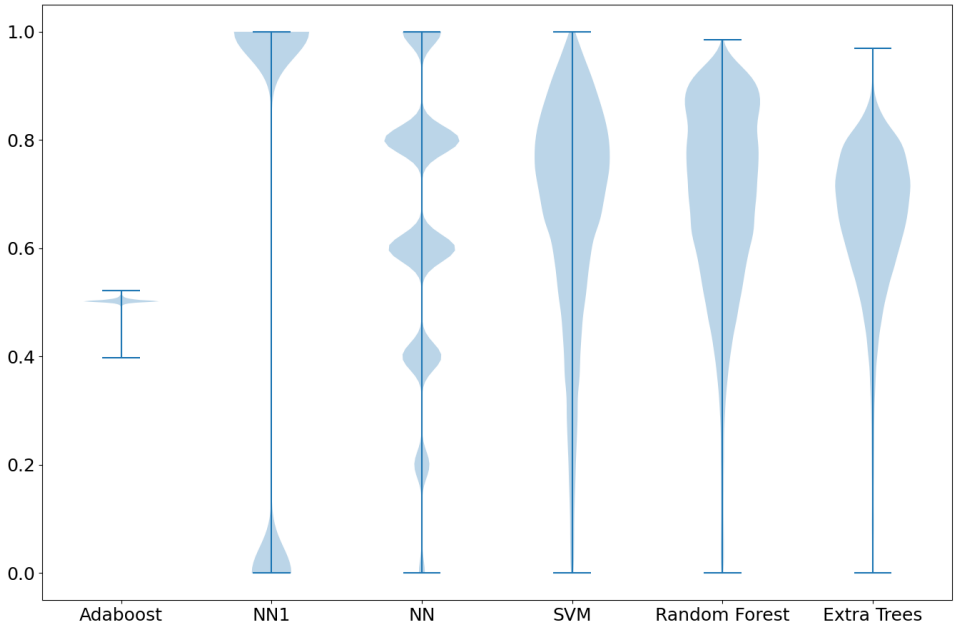


Figure 3.6: Violin plot of the classification scores of the JW300 corpus of the model trained with each classifier.

Figure 3.6 shows all of the classification scores of the corpus JW300 of each model. The classifier SVM and Random Forest work very similar to the default Extra Trees. However, Adaboost, NN and NN1 return scores very different. Individually each of them tells us the following:

- Adaboost: It returns all the scores in the range of 0.4 - 0.53, which indicates that the model is consistently uncertain about the classification of the instances and is unable to assign higher or lower probabilities confidently. The training indicate that the threshold for this model is 0.5, which around 78k out of the 100k sentences pairs pass that threshold.

- NN1: It returns, for all the scores, 0.0 or 1.0, nothing in between, this can be because NN1 inherently tends to produce binary outcomes since it assigns instances to the class of their closest neighbor. The training indicates that the threshold here is 0.1, and around 65k sentences pairs have the score of 1.0, so those are the TUs that pass that threshold.
- NN: It returns only the values 0.0, 0.2, 0.4, 0.6, 0.8 and 1.0, happens similar to the NN1, but instead of having a binary outcome, because the number of neighbors here is 5, the scores will cluster of these values since, again, it assigns instances to the class of their closest 5 neighbors. Here, the threshold is 0.3, and around 91k TUs go over this threshold.
- SVM, Random Forest and Extra Trees: These are more the expected values of the default (extra trees) classifier, which go from 0 to 1, with everything in between. In the case of SVM the threshold is set to 0.4, and for Random Forest and Extra Trees to 0.5. Around 89k, 85k and 88k TUs pass those thresholds respectively.

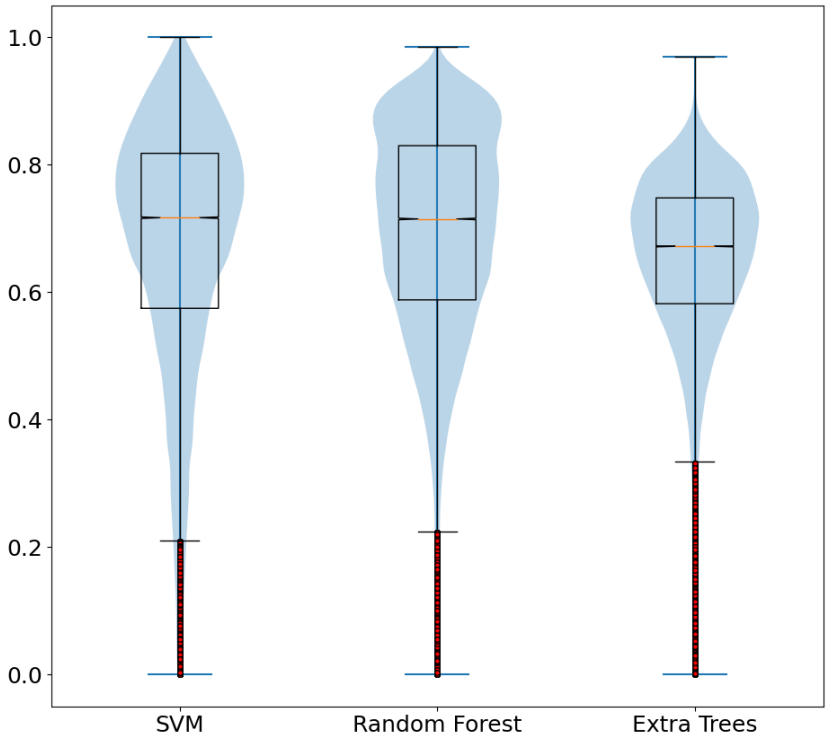


Figure 3.7: Violin+box plot of the classification scores of the JW300 corpus of the model trained with the SVM, Random Forest and Extra Trees classifiers.

Figure 3.7 shows the classification scores of the corpus JW300 of only the models with the SVM, Random Forest and Extra Trees classifiers. It shows how the Extra Trees gives less bad scores which its more accurate. Even tough SVM has the most TUs over the threshold (because it is 0.4), there are lost of them with inaccurate values. And between Random Forest and Extra Trees, the latter works better while in some case Random Forest misbehaves.

Table 3.7 shows some of the multiple examples where SVM and Random Forest cut off some accurate TUs, while Extra Trees does not. Because all this, the default version of Bicleaner, Extra Trees, will stay as the chosen one.

EN	ES	SVM	Random Forest	Extra Trees
When World War I ended on November 11, 1918, the world rejoiced.	El mundo se alegró cuando la I Guerra Mundial llegó a su fin el 11 de noviembre de 1918.	0.318	0.463	0.648
A tapioca crepe with coconut and condensed milk filling	Crepe de tapioca rellena de leche condensada y coco	0.382	0.417	0.606
Applying God's Word is especially helpful.	Obedecer la Palabra de Dios es especialmente útil.	0.328	0.485	0.644
So it was that in March 1948, I went to Brooklyn.	Así que en 1948 me mudé a Brooklyn.	0.151	0.482	0.600
The Bible urges parents to spend time giving their children spiritual training.	La Biblia insta a los padres a dedicar tiempo a la instrucción espiritual de sus hijos.	0.372	0.331	0.536
Liberation Theology and the Bible	La teología de la liberación y la Biblia	0.165	0.440	0.648
House slaves had little recourse if they were treated cruelly by their owners.	Los esclavos de una casa no tenían mucho de lo cual valerse si sus amos los trataban cruelmente.	0.320	0.315	0.584
Illustrate how fine manners in the ministry often bring good results.	Ilustre el hecho de que el desplegar buenos modales en el ministerio da buenos resultados.	0.341	0.490	0.560
Thus in Africa, where tribalism is a divisive factor, the use of Swahili serves to unite people.	Por eso en el África, donde el espíritu de tribu es un factor divisivo, el uso del swahíli sirve para unir a las personas.	0.361	0.480	0.628
Once an osprey was observed to sink his talons into a larger fish than he could handle.	Se observó en una ocasión a un quebrantahuesos que clavó sus garras en un pez que era más grande de lo que podía levantar.	0.273	0.490	0.542
Impossible?	¿Le parece imposible?	0.095	0.480	0.650
Weeks of shopping in preparation for Christmas is nerve-racking.	El salir de compras semana tras semana en preparación para las Navidades destroza los nervios.	0.267	0.495	0.528
In one trailer court two hundred trailers were swept away.	En un parque para casas rodantes doscientas casas rodantes fueron arrastradas.	0.202	0.465	0.562
What Is There Worth Reading?	¿Qué hay que valga la pena leer?	0.310	0.427	0.638
What will help us to endure all things?	¿Qué nos ayudará a aguantar todas las cosas?	0.386	0.445	0.526
Death of a Loved One	La muerte de un ser amado	0.318	0.492	0.576
She had a five-year-old son.	Ella tenía un hijo de cinco años.	0.398	0.383	0.624
So I put all my things in order before leaving, returning the book to my fellow nun.	Así es que antes de irme puse todas mis cosas en orden y le devolví el libro a mi compañera monja.	0.307	0.482	0.524

Table 3.7: Differences in output values of the classification of the JW300 corpus, comparing the models with the classifiers SVM, Random Forest and Extra Trees. In particular, some of the examples where SVM and Random Forest misbehaves while Extra Trees goes over the threshold.

3.3.3. Different ways of generating noisy samples

During the training of a Bicleaner model, since the version 0.13, Bicleaner itself generates these noisy samples. But instead of letting generate itself, there are two ways that we could generate this noise. To generate this noisy samples itself, Bicleaner uses methods to manipulate the sentences within a given range, apply frequency-based modifications, shuffling, or word removal. Therefore in this section, the objective will be to see if generating the noise manually with these two ways makes a difference.

- Extraction of noisy sentences from an existing corpus with heuristic rules, that is, using those TUs that did not pass the filters of the hardrules during the cleaning of the data. In the sixth step of the cleaning of the corpora, instead of keeping those that passed the hardrules (that is those with a 1), here those with the value 0 are going to be kept, this is all of the sentences pairs that did not passed the hardrules. From all of these TUs, 100k are randomly selected to use in the training, stored here.
- Build synthetic noise. Bicleaner includes this file to generate noise. This simply shuffles the characters within each input corpora, resulting in sentences with random words. Again, 100k are randomly selected to use in the training, stored here

The training of these models are performed the same way, only adding the parameter `-wrong_examples_file` to indicate the parallel noisy corpus, as this file shows.

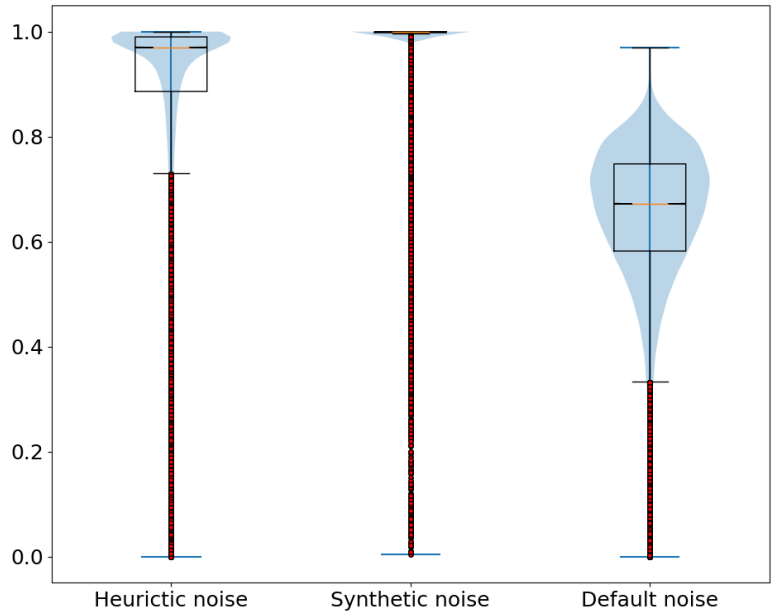


Figure 3.8: Violin+box plot of the classification scores of the JW300 corpus of the model trained with the different noise samples generated with the heuristics, synthetic and default classifiers.

Firstly, the classification scores of the corpus JW300 are presented in Figure 3.8. Like mentioned before, the models used here involve three approaches: (1) extracting noise from the existing corpus using heuristic rules, (2) generating synthetic noise, and (3) allowing Bicleaner to generate noise itself.

The figure indicate that the models behave quite different. Firstly, if you train with synthetic noise (2) generated by the script, the wrong sentences inserted into training will always be completely random phrases, therefore if you later pass Bicleaner Classic in inference, anything that resembles a normal phrase, it will always give you high values (close to 1). Secondly, when trained on this heuristic noise (1), the sentences that are tagged as bad ones are, on average, much more obvious bad sentences. So, when you classify a corpus that contains, in addition to these types of sentences (obvious noise), other phrases that are not like them (not so obvious noise), the classifier finds them better, giving them high scores.

Overall, for both cases of manually generating the noise, the classification is mostly biased, because they only contain a specific type of noise. Therefore, it would makes to let Bicleaner generate its own noise, due to that it was probably done this way because of the impediments of manually generating would have.

3.3.4. Curriculum learning

Curriculum learning is a training methodology inspired by the educational concept of curriculum, where learning progresses from simple concepts to more complex ones [41].

In curriculum learning, the training data is presented to the model in a meaningful order that gradually increases the difficulty of the examples. The goal is to facilitate the learning process by providing a structured and guided curriculum for the model to follow. By starting with easier examples and gradually introducing more challenging ones, curriculum learning aims to improve learning efficiency, generalization, and overall performance.

In the context of training a classic bicleaner model, curriculum learning involves presenting the training data to the model in a carefully designed order that gradually increases the difficulty of the examples. Specifically, the training data is sorted based on sentence length, arranging them from the simplest to the most challenging. By structuring the curriculum in this way, the model initially encounters shorter and simpler sentences, allowing it to grasp basic patterns and linguistic features. As the training progresses, the model is gradually exposed to longer and more complex sentences, which require a deeper understanding of syntax, semantics, and contextual dependencies.

By utilizing curriculum learning with a curriculum based on sentence length, the objective would be that the model can learn incrementally from simpler to more complex sentence structures, resulting in improved performance and a deeper understanding of the underlying linguistic patterns and features.

Although in this specific case, the desired objective of improving the performance of the classic Bicleaner model through the use of curriculum learning based on sentence length is not achieved, this lack of improvement is understandable because some characteristics that Extra Trees have that could after this:

- Lack of Sequential Dependency: Extra Trees typically do not rely on sequential dependencies in the data. They make decisions based on individual instances or subsets of features independently, rather than considering the order in which the examples are presented. As a result, the concept of a curriculum, where the

difficulty of examples is gradually increased, may not be applicable or necessary.

- Feature Independence: Extra Trees is an ensemble method that combines multiple decision trees, where each tree is trained on a random subset of features. The independence of features in each tree further reduces the relevance of curriculum learning, as the model already incorporates diversity and randomness during training.

The training and result model of this are stored here. And the difference here was the use of the same training data but sorted by sentence length.

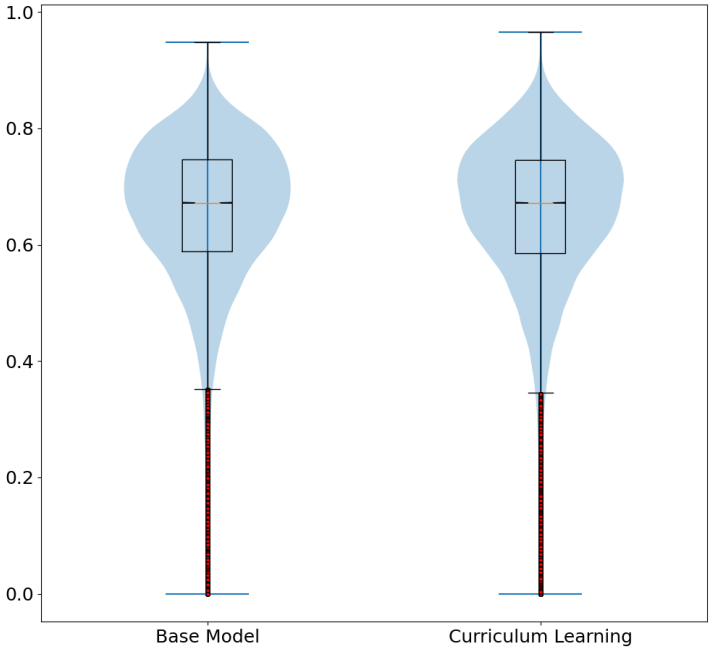


Figure 3.9: Violin+box plot of the classification scores of the JW300 corpus of the base model and the model trained using curriculum learning.

Figure 3.9 shows the classification scores of the corpus JW300 comparing the base model and the model trained using the method curriculum learning, where the train corpus was sorted by sentence length. And as predicted, there is not much change. Around 95% of scores have a difference of less than 0.1, which is a normal variation, but not significant.

Concluding, for a classic Bicleaner model, this method of training is not necessary, due to the characteristics of the Extra Trees which makes it ineffective.

3.3.5. Enabling different training parameters

During the training of a classic model, various parameters can be enabled or disabled to customize the behavior and performance of the model. Some of them are:

- `-treat_oovs`: This parameter allows for the special treatment of out-of-vocabulary (OOV) words in the model's training process. By enabling this parameter, the model adjusts its behavior to handle OOV words differently. It includes specific steps to calculate the probabilities of correspondences between OOV words and

source language (SL) words, taking into account the source and target dictionaries. This parameter can be useful in scenarios where the presence of OOV words significantly impacts the model's performance.

- `-disable_features_quest`: This parameter serves to disable less important features during training. When enabled, it allows the model to focus on the most relevant features while disregarding features that may have a lesser impact on the model's overall performance. By selectively disabling these features, training can be expedited, and the model's efficiency can be improved. This parameter is particularly useful when training time is a crucial factor and when it is determined that certain features contribute minimally to the model's effectiveness.
- `-qmax_limit`: This parameter controls the number of maximum target words taken into account during training, sorted by length. By default, the value is set to 20. However, it can be adjusted to different values, depending on the specific requirements of the task. By limiting the number of target words considered, the training process can focus on a subset of words that are deemed most relevant or influential. This parameter is beneficial in situations where longer target sentences have a disproportionate impact on the model's performance, and it is desirable to prioritize the handling of shorter sentences.

For this case, these parameters are not likely to make a significant difference in achieving the objective of creating a model capable of classifying any corpus. The primary goal in such a scenario is to develop a model that exhibits robust and generalized classification performance across a wide range of inputs, regardless of their specific characteristics. For the case of the `-treat_oovs`, it can be useful for addressing specific challenges related to OOV words, it does not fundamentally alter the model's ability to classify any corpus comprehensively. For `-disable_features_quest`, it would important to consider all available features to capture the diverse patterns and linguistic characteristics present in different types of text. And lastly, `-qmax_limit`, which restricting the consideration of target words based on their length may limit the model's ability to generalize across different sentence lengths and could lead to biased performance.

In summary, when the objective is to create a model that can classify any corpus effectively, it is essential to prioritize a comprehensive and inclusive approach during training, considering all words, features, and sentence lengths. By doing so, the model can learn to capture the rich and varied characteristics of different texts and achieve the desired classification performance on a wide range of inputs.

In light of the mentioned statements, we shall proceed to verify that, despite the circumstances, the models pertaining to this case will remain virtually unaltered. The trained models and the classification outputs are stored here.

Figure 3.10 shows the classification scores for the JW300 corpus, comparing the base model with the different models trained with the previously mentioned parameters `-treat_oovs`, `-disable_features_quest` and `-qmax_limit`, for the latter, the values 10 and 40 were used (half and double the default value 20). Each of them has almost the same number of TUs passing the threshold, `-disable_features_quest` has around 87% and the rest 88%, so practicly the same.

With this, we can conclude what we expected, that the use of these parameter for a

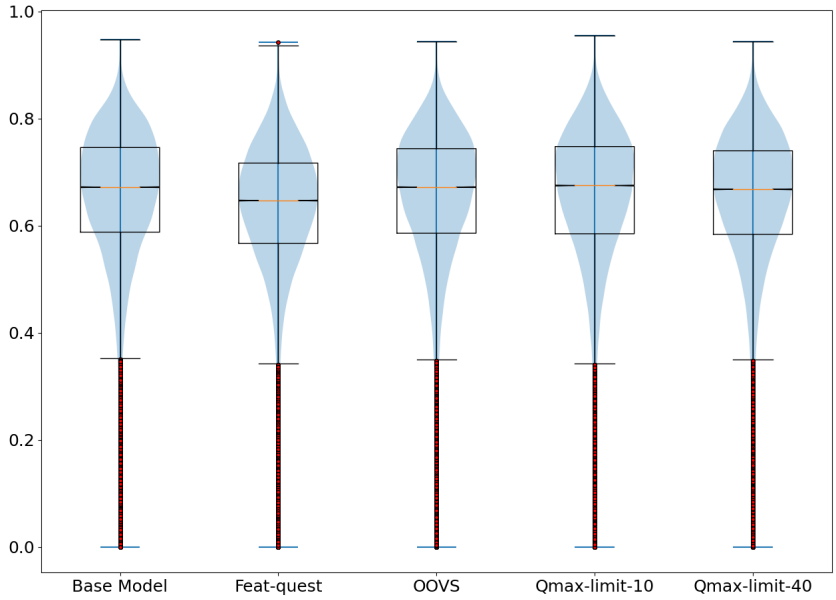


Figure 3.10: Violin+box plot of the classification scores of the JW300 corpus of the base model and the models trained enabling different parameters.

general use of a model is unnecessary. These flags could be useful for specific cases like mentioned before.

3.3.6. Cleaning a new corpus with different hardrules

Like explained in previous sections, Bicleaner-hardrules is used for a filtering step to clean the data that is going to be used for the training of a model. The previously used corpus was cleaned with the initial version of hardrules, like explained in its respectively section. For now we will use the latter version where the list of hardrules is the following:

- `no_empty`: Removes empty texts.
- `not_too_long`: Specifies the maximum allowed length for a text.
- `not_too_short`: Specifies the minimum allowed length for a text.
- `length_ratio`: Controls the ratio between the length of the text and the average length of the sentences within the text.
- `no_identical`: Removes duplicate texts.
- `no_literals`: Removes texts containing specific literal phrases or symbols.
- `no_only_symbols`: Removes texts that consist only of symbols.
- `no_only_numbers`: Removes texts that consist only of numbers.
- `no_urls`: Controls the removal of texts containing URLs.
- `no_breadcrumbs`: Removes texts containing breadcrumbs.
- `no_glued_words`: Removes texts with glued words, where spaces are missing between words.

- `no_repeated_words`: Specifies the maximum number of allowed repeated words in a text.
- `no_unicode_noise`: Removes texts with Unicode noise or non-textual Unicode characters.
- `no_space_noise`: Removes texts with space noise or extraneous space characters.
- `no_paren`: Removes texts containing parentheses.
- `no_escaped_unicode`: Removes texts with escaped Unicode characters.
- `no_bad_encoding`: Removes texts with bad encoding.
- `no_titles`: Removes texts that resemble titles.
- `no_number_inconsistencies`: Controls the removal of texts with number inconsistencies.
- `no_script_inconsistencies`: Controls the removal of texts with script inconsistencies.
- `no_wrong_language`: Specifies the maximum allowed number of language detection mistakes in a text.
- `no_porn`: Removes texts containing explicit adult content.
- `lm_filter`: Applies a language model-based filtering.

Some of them were changed from the default values (presented here) to new ones, to see if they could enhance the model:

- `length_ratio`: From 2.0 to 3.0, with the objective of allowing for more flexibility in sentence lengths, potentially capturing longer and more complex sentences.
- `no_identical`: False, in some cases, short sentences could be written the same way, specially if it is names or anglicism.
- `no_literals`: From `["Re:","", "%s", "", "+++", "***", '=\"']` to False. This modification aims to retain any specific literals that may hold significance in the translations.
- `no_breadcrumbs`: False, this change allows for the preservation of breadcrumbs that may provide useful information or context in the translated text.
- `no_repeated_words`: From 1 to 3. This adjustment accounts for cases where repetition is intentional or stylistic, avoiding unnecessary filtering of sentences that may be grammatically correct despite word repetition.
- `no_number_inconsistencies`: True. This change ensures that sentences containing inconsistent or conflicting numerical information are filtered out.

All of this hardrules were applied to almost the same corpora. In the case of classic version some corpus were used and in the case of AI another ones, overlaping in some of them. For this, not only a new set of hardrules are being tested but a new set of corpus. A new set is created from all of the corpora of classic and AI

	Initial	Bifixier 1	Bifixier 2	Hardrules	Total filtered	% filtered
DGT	5,127,624	5,146,877	3,249,531	1,907,495	3,220,129	62.80 %
EUbookshop	5,215,515	5,330,194	4,688,614	3,208,656	2,006,859	38.48 %
Europarl	2,009,073	2,032,947	1,958,596	1,913,494	95,579	4.76 %
GlobalVoices	693,544	701,260	687,571	626,601	66,943	9.65 %
JW300	1,646,545	1,578,767	1,462,451	1,300,596	345,949	21.01 %
NewsCommentary	238,511	239,856	239,746	229,485	9,026	3.78 %
OpenSubtitles	5,000,000	4,565,267	3,769,593	3,116,220	1,883,780	37.68 %
SciELO	433,695	441,326	421,146	387,372	46,323	10.68 %
TildeMODEL	3,775,496	3,783,461	3,388,253	3,054,302	721,194	19.10 %
UNPC	4,000,000	4,095,380	3,998,365	594,086	3,405,914	85.15 %
TedTalk	210,181	223,919	222,669	205,778	4,403	2.09 %
Total	28,350,184	28,139,254	24,086,535	16,544,085		

Table 3.8: Sizes of the files generated when applying Bifixer and the new hardrules, total of TUs in each step, and the total of TUs filtered with each percentage respectively.

(DGT, EUbookshop, Europarl, GlobalVoices, JW300, NewsCommentary, OpenSubtitles, SciELO, TildeMODEL, UNPC and TedTalk). This way we have an even wider set with the objective of having a better coverage.

Table 3.8 shows the new process of filtering each of the corpus, with the total of filtered sentences and the percentage that this represents, respectively. With new filtering, the noisiest corpus is DGT, while the cleanest is TedTalk.

The same way as before, with this corpora, firstly a set of around 10M is randomly selected to generate the dictionaries and word frequency files. Secondly, a set of around 100k is also randomly selected to use as training data. The model and all its files is stored here.

Figure 3.11 shows the classification scores of the model trained with this new filtered corpus compared to the base model and the one trained directly with the Bicleaner AI corpus. As it shows, there is not much of a difference from the already made improvement of using the AI corpus. The change in those hardrules are not very significant in the total of the whole corpus, because in its majority it is used for the dictionaries and word frequency files. What is significant is the use of a wider corpus compared to the base model.

With all of this said, and seeing there is not much of a difference, the new corpus filtered here will be the one used for the final classic model. It is proven that a wider corpus helps for this model and knowing that this new corpora has a few more corpus that the AI data, it is a little bit wider that the last trained.

3.4. Improvements on Bicleaner AI full model

Like for the classic mode, after reaching the initial stage of aligning with the pre-existing Bicleaner AI model, the objective is to enhance it through adjustments to different training parameters and setups. Starting of from the final chosen corpus used for the Bicleaner classic model, so both both use the same data.

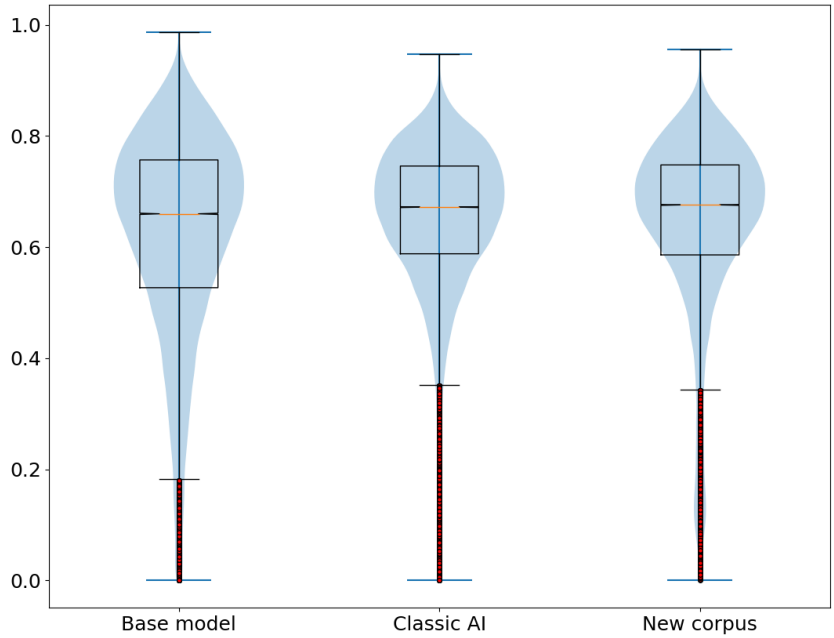


Figure 3.11: Violin+box plot of the classification scores of the JW300 corpus of the base model, the model trained in section 3.3.1 with the Bicleaner AI data and the model with this new filtered corpus.

3.4.1. Increasing epochs

The number of epochs that Bicleaner AI has as the default value is 10. However for all the previous training, 15 epochs was set, due to the fact that the pre-trained model was trained with 15 epochs and the first objective was to reproduce that model.

Knowing that the training has the parameter `-patience`, to stop training when validation has stopped improving after that number of epochs. The default number for this is 3, so knowing that the training will stop on its own, the number of epochs is increased to 50, to visualized when the training stabilizes and stops. The model is stored here.

Figure 3.12 shows the progression of the training of this model, and how even tough it was set to 50 epochs, the model stopped on its own in the epoch 24.

	Precision	**Recall**	**F1**	**MCC**
Base model	0.841	0.883	0.862	0.846
50 epochs	0.878	0.862	0.870	0.855

Table 3.9: Test metrics of the base model and the one trained with 50 epochs, however it stopped in 24.

After the training, the model returned the test metrics shown in table 3.9 compared to the base model. The model trained for 50 epochs (stopped at epoch 24) generally performed slightly better than the base model trained for 15 epochs in terms of precision, F1 score, and MCC. However, the base model had a higher recall. It's worth noting that the performance difference is relatively small.

Even tough the improvement is small, the number of epochs will be increased to 20

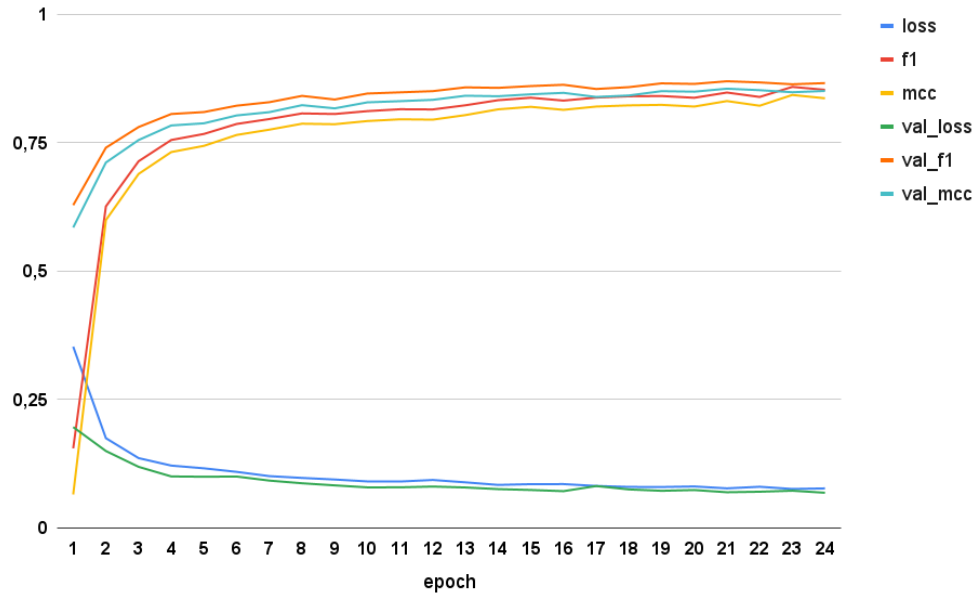


Figure 3.12: Progression of the training of the Bicleaner AI full model with 50 epochs.

to let the models stabilizes a bit more during the training.

3.4.2. Noisy samples parameters

For the case of Bicleaner AI, the noise generation process includes various types of noise, and can be found in this part of the training. Each of them have in their sub-level the arguments used to controlled each of them.

- Sentence Modifications: This technique involves modifying the sentences in the dataset to create variations, specifically, capitalization, by modifying it in sentences randomly, the code aims to introduce variations and avoid biases related to missing starting capital letters, ensuring a more diverse and realistic generation of negative samples.
- Positive and Negative Sample Generation: Positive samples represent valid, correctly aligned sentences, while negative samples are created by misaligning sentences.
  - `-pos_ratio`: Ratio of positive samples used to oversample on validation and test sets (default: 1).
  - `-rand_ratio`: Ratio of negative samples misaligned randomly (default: 3).
- Frequency-based Noise: This technique introduces noise by replacing words in a sentence based on their frequency. Words with higher frequencies are more likely to be replaced, resulting in a misaligned sentence. Controlled by argument
  - `-freq_ratio`: Ratio of negative samples misaligned by replacing words by frequence (default: 3).
  - `-min_freq_words`: Minimum words to replace per sentence in freq noise (default: 1).

- Word Omission: This technique randomly omits words from a sentence, leading to a misalignment between source and target sentences.
  - `-womit_ratio`: Ratio of negative samples misaligned by randomly omitting words (default: 3).
  - `-min_omit_words`: Minimum words to omit per sentence in omit noise.
- Fuzzy Matching: Fuzzy matching involves finding similar words or phrases in the dataset and replacing them with different words. This introduces noise and misalignment between sentences.
  - `-fuzzy_ratio`: Ratio of negative samples misaligned by fuzzy matching (default: 0).
- Neighboring Sentence Misalignment: This technique misaligns sentences by considering the context of neighboring sentences. It can involve reordering sentences or swapping phrases between adjacent sentences.
  - `-neighbour_mix`: If use negative samples misaligned by neighbourhood (default: False).

Almost all of these parameter are enabled and being used to generate noise, all but the fuzzy matching and neighboring sentence misalignment. Two approaches will be tested, first, these two will be enabled to let Bicleaner generate the noise with all its forms. And secondly, we will change the values of those that are enabled by default.

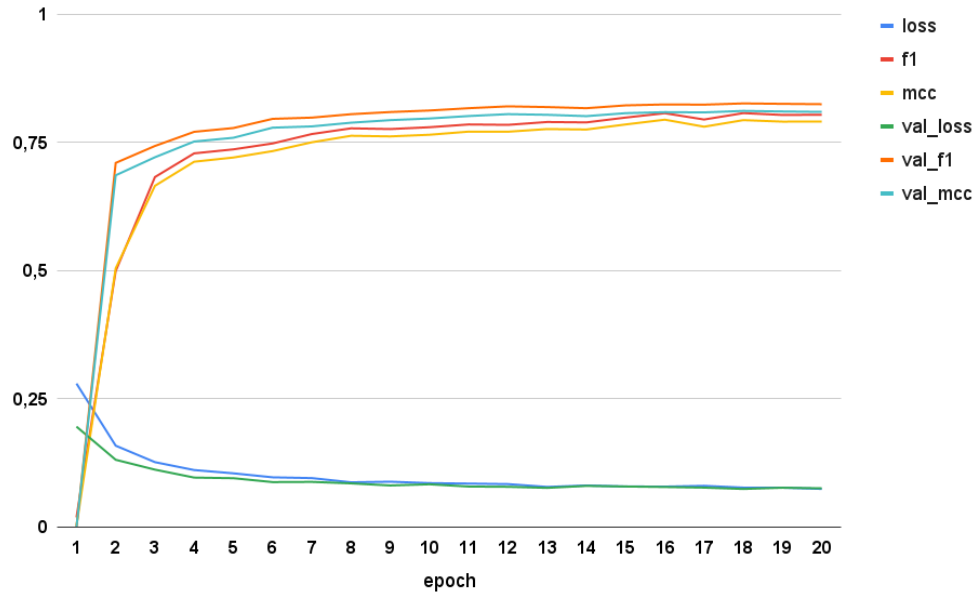


Figure 3.13: Progression of the training of the Bicleaner AI full model with all forms of generating noise enabled.

As table 3.10 shows, the base model outperformed the model trained here in terms of precision, recall, F1 score, and MCC. This suggests that the additional noise introduced by fuzzy matching and neighboring sentence misalignment may have negatively impacted the model's performance on the test set. However, it's worth noting that the difference in performance between the two models is relatively small. But

	Precision	Recall	F1	MCC
Base model	0.841	0.883	0.862	0.846
Fuzzy and neighboring	0.817	0.833	0.825	0.810

Table 3.10: Test metrics of the base model and the default ways of generating noise plus the fuzzy matching and neighboring sentence misalignment.

taking this into account together with the fact that the training of this model takes significantly longer (14765 second $\sim$ 4h 6min 5s) than the base model (8421s $\sim$ 2h 20min 21s), it makes understanding the fact that the fuzzy matching and neighboring sentence misalignment are disabled by default, because makes the training way longer and the performance it is not improved.

After seeing not enabling these two parameters, we will change the values of the enabled parameters if see if those make a difference. The following changes will be made:

- Increase in positive samples (`pos_ratio`): By increasing this, the model will have more exposure to positive instances. This can help the model learn the patterns and features associated with positive samples more effectively, potentially improving its ability to classify positive instances accurately.
- Decrease in randomly misaligned negative samples (`rand_ratio`): Random misalignment can sometimes introduce irrelevant or confusing patterns, making it harder for the model to learn the true underlying patterns. By reducing this ratio, the model can focus more on learning meaningful patterns from the positive and other types of negative samples.
- Decrease in randomly omitted words (`womit_ratio`) and minimum words to omit (`min_omit_words`): The model will encounter fewer instances where words are randomly removed from the target sentence. This can help preserve the integrity of the original sentences and prevent excessive loss of information.
- Increase in frequency-based misaligned negative samples (`freq_ratio`) and minimum words to replace (`min_freq_words`): By increasing the minimum number of words to replace per sentence, the model will have a higher likelihood of encountering negative samples where words are replaced based on their frequency. This can introduce more varied and realistic noise in the training data, helping improve the model's robustness and ability to handle different types of noise.

Overall, these changes aim to have a balance between introducing informative noise for training and preserving the integrity of the original sentences, making the training process focus on relevant patterns, reduce irrelevant noise, and provide the model with more varied examples, potentially improving its performance and generalization capabilities.

	Precision	Recall	F1	MCC
Base model	0.841	0.883	0.862	0.846
Noise paramaters	0.892	0.938	0.914	0.889

Table 3.11: Test metrics of the base model and the default ways of generating noise but changing the default values.

Table 3.11 show an improvement with the changed values. These improvements

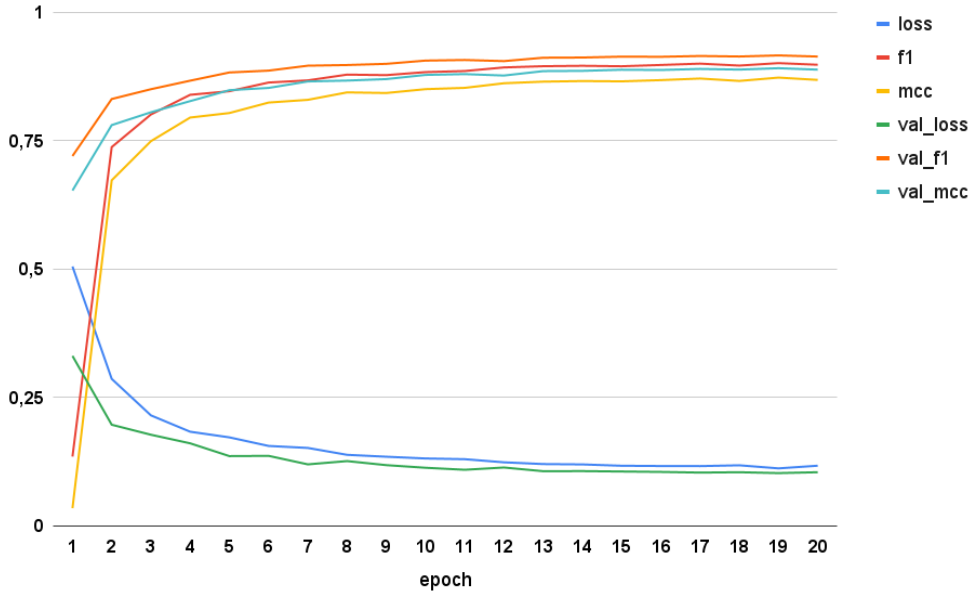


Figure 3.14: Progression of the training of the Bicleaner AI full model changing the values of the enabled parameters used to generate the noise.

suggest that the changes in the noise parameters, such as adjusting the ratios of positive and negative samples, word omission, and word replacement, contribute to enhancing the model's performance in terms of precision, recall, F1 score, and MCC.

Because all of this, the parameters of fuzzy matching and neighboring sentence misalignment are not going to be enabled, while the presented changes of the rest of parameters will be used, in order to improve to model.

3.4.3. Curriculum learning

This method is already explained in section 3.3.4, and the same way is going to be used here. A Bicleaner full AI model will be trained with the train data sorted by sentence length. This way the model can start learning simpler sentences first.

Like mentioned before, this method was not effective for Bicleaner classic. However, when it comes to Bicleaner AI, which uses a fine-tuned XLMR, curriculum learning could be more effective. Fine-tuned XLMR models are highly capable of learning from diverse and complex data. They can benefit from curriculum learning by gradually increasing the difficulty of the training examples. This said, the base model is already pretty good, still, we will give it a try. Despite the already satisfactory performance of the base model, we will proceed with an attempt to further improve its effectiveness. The model is stored here.

	Precision	**Recall**	**F1**	**MCC**
Base model	0.841	0.883	0.862	0.846
Curriculum learning	0.908	0.935	0.921	0.913

Table 3.12: Test metrics of the base model and the model trained using curriculum learning.

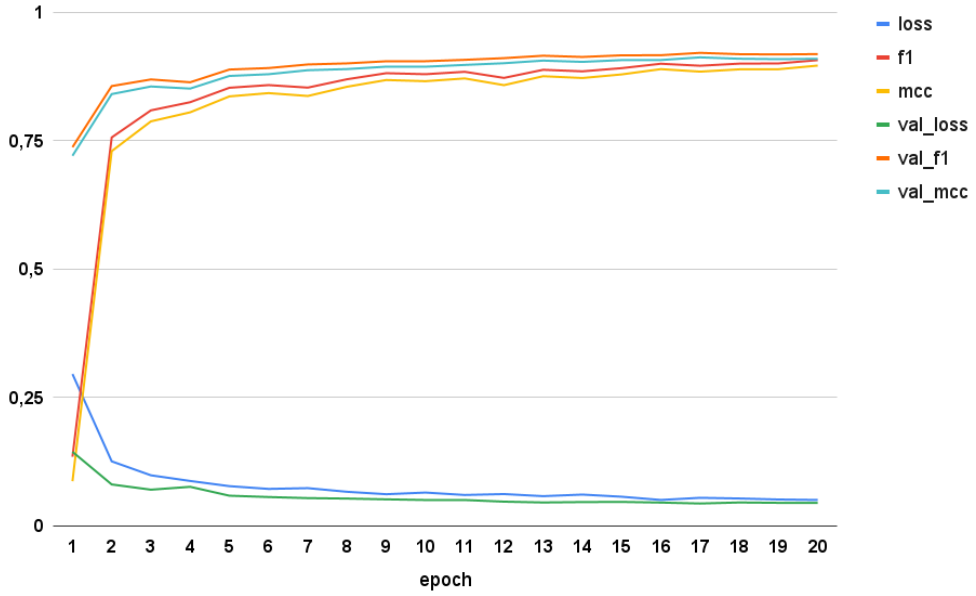


Figure 3.15: Progression of the training of the Bicleaner AI full model trained using curriculum learning.

Table 3.12 show improvements that demonstrate that training the model using curriculum learning, where the data is presented in a structured manner from shorter to longer sentences, enhances the model's performance in terms of precision, recall, F1 score, and MCC. Therefore, and like it was suspected, this way of structured learning helps the XLMR model of Bicleaner to improve the performance.

3.4.4. Hyperparameters tuning

After changing the Bicleaner training parameters, some changes are going to be made to the source code of the tool, specially here, where the hyperparameters of the model are set. Bicleaner does not allow you to change the majority of these parameters externally when training. Because of these, some changes in the code will be made there, to help tune this hyparameters.

3.4.4.1. Activation function

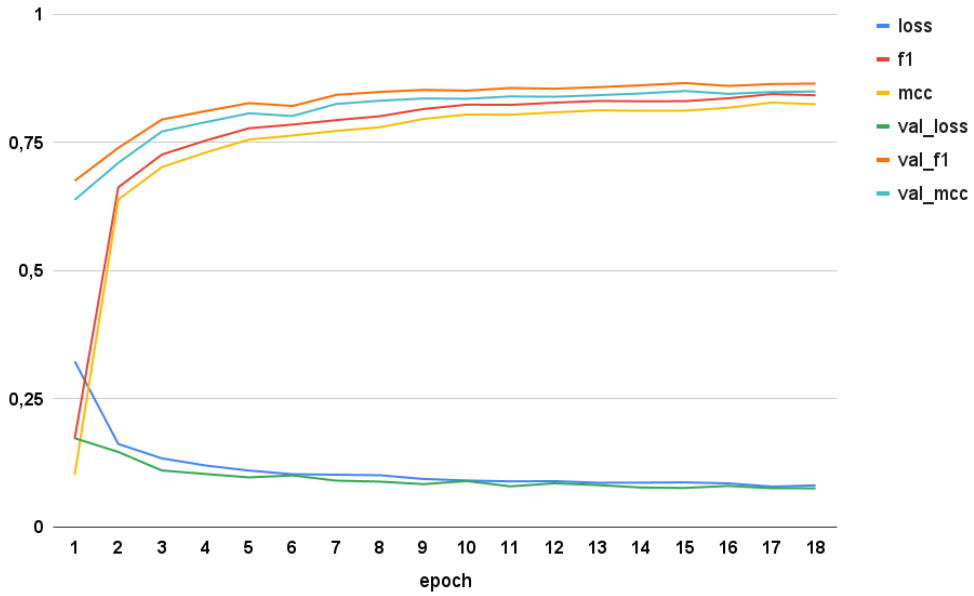
Bicleaner AI full model uses ReLu as the activation function. ReLu [42] is a simple activation function that outputs the input value if it is positive and zero otherwise. It essentially "activates" the neuron when the input is positive. It is commonly used in deep learning models due to its computational efficiency and ability to mitigate the vanishing gradient problem.

Two more activation function will be tried to compare them with the default, to see if it could improve the performance:

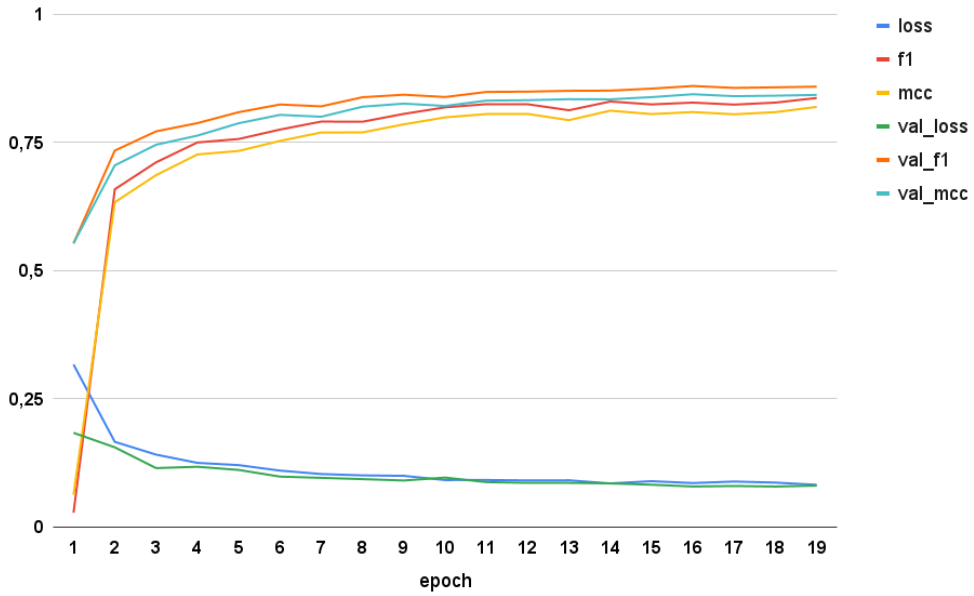
- Tanh: is an activation function that squashes input values to a range between -1 and 1. It is like a scaled and shifted version of the sigmoid function. It is suitable for tasks where negative values are meaningful. It can model complex non-linearities in the data and capture negative correlations.

- GeLu [43]: is an activation function that approximates the Gaussian cumulative distribution function. It smoothly transitions from linearity for negative inputs to non-linearity for positive inputs. It has shown promising performance in deep learning models, especially in natural language processing tasks. It introduces non-linearity while maintaining smoothness, which can be beneficial for complex data patterns.

The models for both activation functions are stored here.



(a) Tanh



(b) GeLu

Figure 3.16: Progression of the training of the Bicleaner AI full model, using different activation functions.

	Precision	Recall	F1	MCC
ReLu	0.841	0.883	0.862	0.846
Tanh	0,837	0,886	0,856	0,841
GeLu	0.842	0.879	0.860	0.844

Table 3.13: Test metrics of the base model using ReLu as the activation function and two models trained with Tanh and GeLu as activation functions.

Comparing the metrics shown in table 3.13, we can observe that the models trained with ReLU and GELU as the activation functions perform relatively better than the one trained with Tanh. The ReLU and GELU models have similar performance across most metrics, while the Tanh model slightly lags behind in terms of precision, F1 score, and MCC.

Overall, based on these metrics from the test set, both ReLU and GELU can be considered as effective activation functions for the Bicleaner AI model, as they yield similar or slightly better performance compared to Tanh. So the final activation function will be ReLu given that is the default one and the other two do not show and improvement.

3.4.4.2. Learning rate

The learning rate determines the step size at which the model updates its parameters during training. A well-tuned learning rate can significantly impact the convergence speed and final performance of the model. It could help with accelerating convergence, balancing exploration and exploitation or enhancing generalization performance.

Bicleaner AI uses as default learning rate 2×10^{-6}, which will be reduced to 1×10^{-6}. Having a smaller learning rate could be beneficial, because it could lead to more stable training by preventing large updates to the model's parameters or even help improve the generalization performance of the model. It can reduce the model's reliance on specific training examples and encourage it to learn more robust and generalized representations that generalize well to unseen data.

	Precision	Recall	F1	MCC
Base model (2×10^{-6})	0.841	0.883	0.862	0.846
Learning rate 1×10^{-6}	0.792	0.869	0.828	0.809

Table 3.14: Test metrics of the base model trained with the default learning rate 2×10^{-6} and the model trained with a smaller learning rate (1×10^{-6}).

Based on these results in table 3.14, it appears that the base model trained with the default learning rate performs better than the model trained with a smaller learning rate across multiple evaluation metrics. This could be because the optimization landscape of the Bicleaner AI model may be relatively flat or have fewer local minima, allowing for faster convergence with a larger learning rate. In such cases, a smaller learning rate may result in slower convergence or getting stuck in suboptimal solutions.

For all this, in order not to keep directly with the default value and since in this case a lower learning rate does not help, a higher learning rate will be tested to see if this will improve performance.

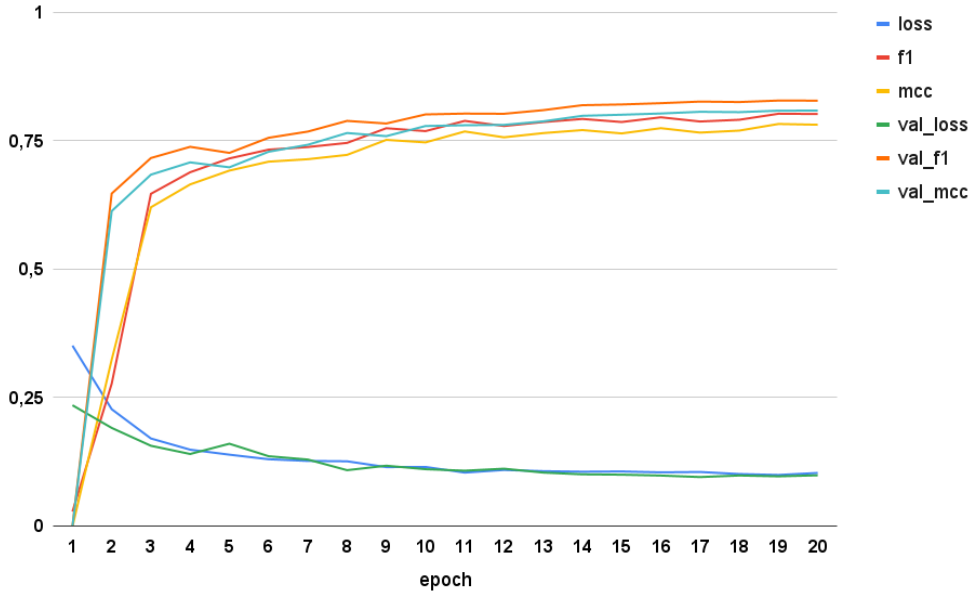


Figure 3.17: Progression of the training of the Bicleaner AI full model with a smaller learning rate.

3.4.4.3. Units of the hidden layer

For Bicleaner AI, which is a fine-tuned XLMR model, uses the base model of XLM-RoBERTa, which has a total of 12 hidden layers. Bicleaner AI has the default value that each hidden layer has 2048 units. A larger number of this is going to be tried, due to fact that could be improve performance. A larger `n_hidden` value allows the model to have more parameters and, therefore, a higher capacity to represent complex relationships in the data. With more hidden units, the model can learn more intricate patterns and capture finer details, potentially improving its ability to discriminate between noise and clean data. Specially, the default value will be increased to 4096 to see if it could improve performance. The model is stored here.

	Precision	**Recall**	**F1**	**MCC**
Base model (2048)	0.841	0.883	0.862	0.846
N_hidden 4096	0.885	0.881	0.888	0.869

Table 3.15: Test metrics of the base model using 2048 units in the hidden layers, and the model trained using 4096 units.

As table 3.15 shows, the model with increased n_hidden (4096) demonstrates improved performance across multiple evaluation metrics, including precision, F1 score, and MCC. This suggests that the larger n_hidden value allows the model to capture more complex relationships in the data, leading to better discrimination between noise and clean data.

In conclusion, having a larger number of units in the hidden layer has improved the base model, so the hyperparameter `n_hidden`, representing the number of unit each hidden layer has, will be increased to 4096.

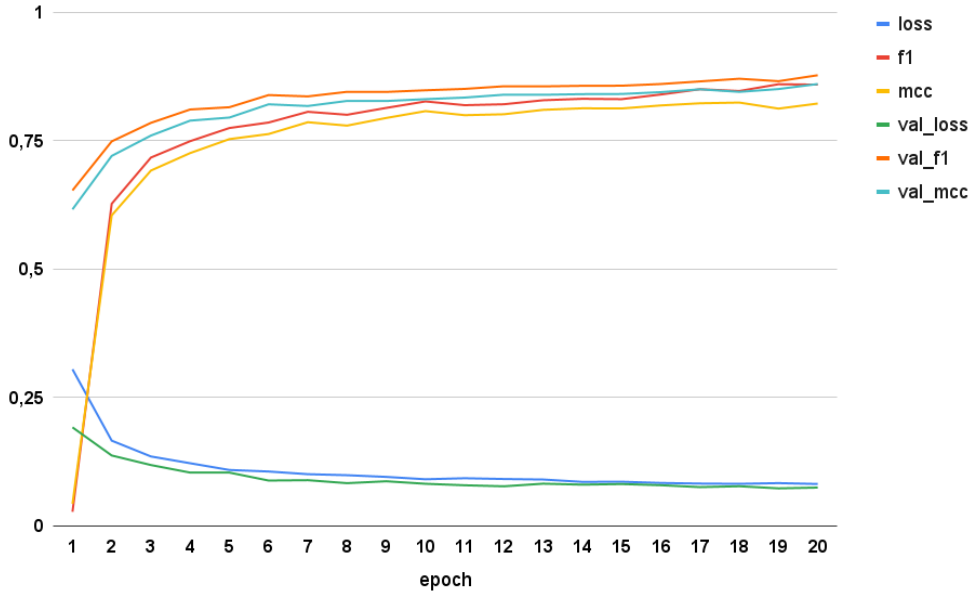


Figure 3.18: Progression of the training of the Bicleaner AI full model with 4096 units in the hidden layers.

3.4.4.4. Decay rate

The decay rate refers to the rate at which the learning rate decreases during the training process. It is commonly used in optimization algorithms to gradually reduce the learning rate over time. The default value of the decay rate is set to 0.1, this means that the learning rate will decrease by 10% at each step during training.

In the given context where the Bicleaner model stabilizes quickly and achieves convergence within a small number of epochs, using a larger decay rate may not necessarily improve the model's performance. A larger decay rate would result in a more rapid decrease in the learning rate over time. However, if the model already converges quickly within a few epochs, applying a larger decay rate may cause the learning rate to decrease too quickly, preventing the model's ability to fine-tune its parameters effectively.

In such cases, a smaller decay rate might be more suitable. This allows the model to continue refining its parameters for a longer duration, which can help capture additional nuances in the data and further improve performance. The 0.1 default decay rate will be decreased to 0.01 to see if it makes a difference. The model is stored here.

	Precision	**Recall**	**F1**	**MCC**
Base model (0.1)	0.841	0.883	0.862	0.846
Decay rate 0.01	0.828	0.878	0.852	0.835

Table 3.16: Test metrics of the base model using 0.1 as the decay rate, and the model trained using a decay rate of 0.01.

Based on these metrics from the test set in table 3.16, it appears that reducing the decay rate from 0.1 to 0.01 did not lead to an improvement in the model's performance. The model trained with the lower decay rate showed slightly lower precision,

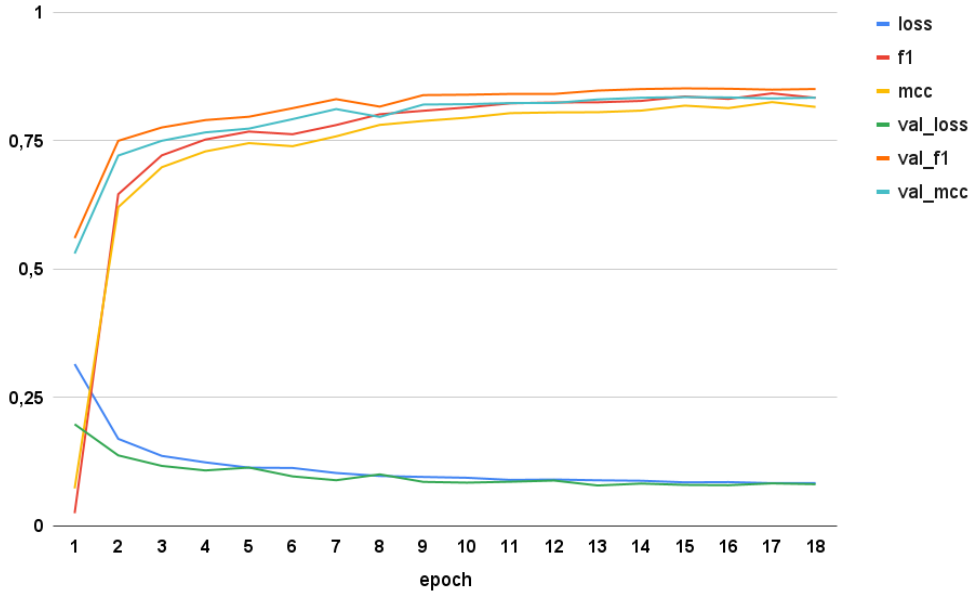


Figure 3.19: Progression of the training of the Bicleaner AI full model with a decreased decay rate of 0.01.

F1 score, and MCC compared to the base model. Because of this, the decay rate will stay as it is, the default value 0.1.

Chapter 4

Results

In this study, we firstly aimed to improve the performance of the Bicleaner classic model by exploring different approaches. We focused on several areas, including using AI model data to train the classic model, exploring different types of classifiers, investigating different ways of generating noisy samples, curriculum learning, enabling different training parameters, and cleaning a new corpus with different hard rules. Through these investigations, we sought to enhance the model's ability to classify parallel text units (TUs) across various domains and in particular for corpus English-Spanish.

Firstly, we examined the use of AI model data to train the classic Bicleaner model. By leveraging the Bicleaner AI model data, which had been trained on a large and diverse corpus, we aimed to incorporate its knowledge and improve the classic model's performance. We found that the use of this wider corpora made the model less strict and it did lead to improvements in the classic model's classification accuracy.

Next, we explored the impact of different types of classifiers on the performance of the classic Bicleaner model. We evaluated six classifiers: Adaboost, NN1, NN, SVM, Random Forest, and Extra Trees. The experimental results revealed valuable insights into their performance. Considering the performance analysis and computational efficiency, we conclude that the Extra Trees classifier emerged as the most favorable choice for the classic Bicleaner model. It exhibited comparable performance to other classifiers while requiring less training time compared to SVM. Therefore, we recommend utilizing the Extra Trees classifier as the default option in the classic Bicleaner model.

Furthermore, we investigated different ways of generating noisy samples to augment the training data. By introducing noise in the form of artificial errors and variations, we aimed to improve the model's robustness and generalization. However, the experimental results demonstrated that manually generating noisy samples did not lead to significant improvements in the classic model's performance, and how Bicleaner itself generates noise during the training which worked better.

Additionally, we explored curriculum learning as a potential training methodology to enhance the classic Bicleaner model. However, due to the characteristics of Extra Trees, which do not rely heavily on sequential dependencies and feature independence, the concept of curriculum learning based on sentence length did not make significant improvements in the model's performance. Consequently, we concluded

that for the classic Bicleaner model, curriculum learning was not necessary and did not provide substantial benefits.

We also investigated the effects of enabling different training parameters on the model's performance. However, the experimental results showed that these parameters had minimal impact on the model's ability to classify any corpus effectively. While they might be useful for specific cases and scenarios, for the general use of the classic Bicleaner model, a comprehensive and inclusive approach during training, considering all words, features, and sentence lengths, proved to be more effective.

Finally, we explored the option of cleaning a new corpus with different hardrules. We modified the existing set of hard rules to account for specific linguistic characteristics and domain-specific challenges. This approached, like happened with the AI, help significantly, due to the use of a wider corpora, which included a broader range of domains, proved to be more beneficial in achieving improved classification accuracy.

Based on the findings from our investigations, we concluded that the most significant improvement in the performance of the classic Bicleaner model came from using a wider corpus. The inclusion of additional corpora, encompassing diverse domains and languages, provided a better coverage and enhanced the model's ability to classify TUs accurately. Therefore, we recommend using the new corpus, filtered or not, with the modified hard rules and comprising a wider range of text sources, as the final model for the classic Bicleaner.

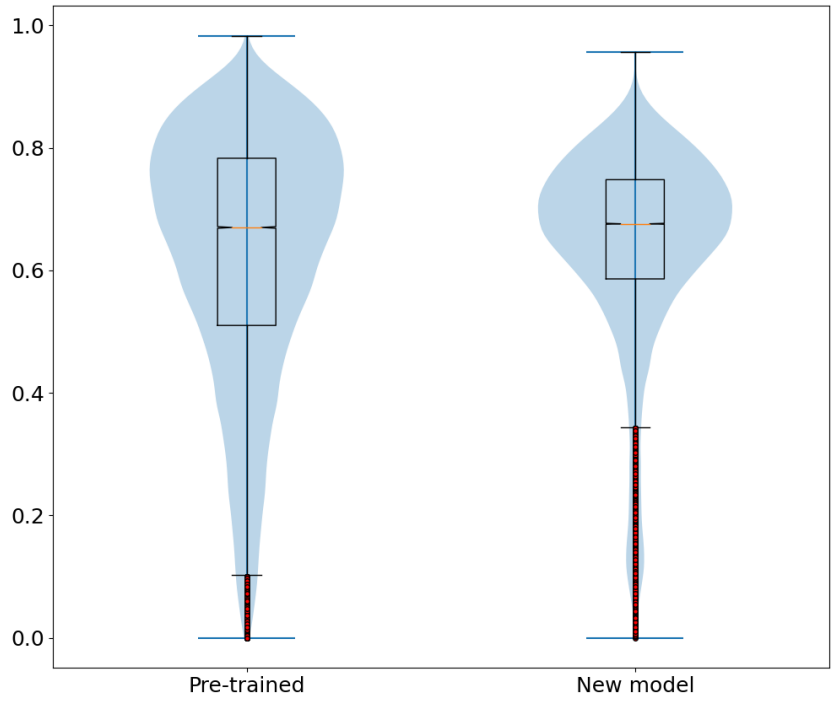


Figure 4.1: Classic Bicleaner. Violin+box plot of the classification scores of the JW300 corpus of the pre-trained with the final model presented using the wider corpus.

Figure 4.1 shows how the new Bicleaner model is less strict allowing to give better scores to a lot of the sentences that the pretrained model does not. Some of this sentences can be seen in the table 4.1.

EN	ES	Pre-trained	New model
Since a woman who is pregnant is more likely to hemorrhage, heeding warning signals assumes a still greater importance.	Como las mujeres embarazadas tienen más posibilidades de sufrir hemorragias, el prestar atención a los avisos del cuerpo adquiere aún más importancia.	0.237	0.776
According to Gerhard Friedrich, "runaway slaves who were caught used to be branded in their foreheads.	Según Gerhard Friedrich, "a los esclavos fugitivos que detenían los marcaban en la frente con hierro candente.	0.473	0.792
And vital to obedience is godly fear, yes, fear of displeasing God.	Y para obedecer es vital que tengamos temor piadoso, sí, temor de desagradar a Dios.	0.456	0.664
These large areas of "conflicting regional interests" were often the consequence of military conquest, since kings were invariably military leaders.	Dichas zonas extensas con 'intereses regionales contrapuestos' solían obedecer a conquistas, pues los reyes siempre eran caudillos militares.	0.369	0.680
If the criminal willingly conforms to punitive orders and exhibits changes in his attitude and behavior, a judge or president may choose to pardon him by lessening his sentence or totally forgiving his sentence.	En algunos países, los jueces y algunos funcionarios de alto rango tienen potestad para conmutar la pena a un delincuente —o incluso indultarlo— si este acepta su castigo y demuestra buena conducta.	0.193	0.586
In order to make ends meet, some mothers turn to prostitution and sell illegal drugs or encourage their daughters to do so.	Para subsistir, algunas se prostituyen y trafican con drogas, o empujan a sus hijas a tales actividades.	0.323	0.644
Our power of reason also tends to shy away from things that seem hopelessly vague and undefinable.	La razón también tiende a descartar todo lo que parece totalmente vago e indefinible.	0.427	0.652
When the first tremor subsided, they rushed outside and saw each other. Together they ran to higher ground.	Cuando pasó el terremoto, salieron corriendo de sus casas, se encontraron en la calle y huyeron a un lugar alto.	0.243	0.634
Indeed, kind parents strive to discern what discipline works best for each of their children.	Sin duda, los padres bondadosos tratan de descubrir cuáles son las medidas que funcionan mejor con cada hijo.	0.480	0.640
Others favor retaining the three-line form, making the middle one slightly longer.	Otros prefieren conservar la forma de tres versos, pero alargan un poco el segundo.	0.153	0.700
She used to bring substantial profit to her masters by fortune-telling.	Era una esclava, y ganaba mucho dinero para sus dueños, adivinando.	0.070	0.584
Later on, though, I began to feel that maybe I had been a little extreme.	Pero luego me puse a pensar si no me habría ido al otro extremo.	0.380	0.696
It will be as if they stood still, not functioning as light bearers, but letting Jehovah's radiant missiles put on a display of an illuminating power.	Será como si estuvieran estáticos, sin funcionar como portadores de luz, permitiendo más bien que los radiantes mísiles de Jehová desplieguen su poder iluminador.	0.313	0.724

Table 4.1: Differences in output values of the classification of the JW300 corpus, comparing the pre-trained model and the final model trained using a wider corpora.

With all of this said, Bicleaner classic still has impediments, this model relies on predefined rules and features to classify TUs as clean or noisy. While this approach can achieve some accuracy, it may struggle to handle the complexities of language that often lead to errors and noise in translations. Traditional classifiers like Extra Trees are not specifically designed to address the subtleties of language and may face challenges in accurately identifying and correcting noisy TUs.

For this reason the Bicleaner AI version was developed, to overcome these limitations and enhance the effectiveness of noise detection and correction. And even tough the Bicleaner AI model was already good, a study was also made to try and improve the Bicleaner AI model performance.

In the pursuit of enhancing the performance of the Bicleaner AI full model, several approaches were explored. Firstly, the number of epochs was increased from 10 to 50 to observe the model's progression and ensure stability. However, the training process stopped on its own in epoch 24. Although the increase in epochs did not yield significant improvements, it provided insights into the training process and helped determine the optimal stopping point.

Next, the noise generation parameters were adjusted to introduce more informative noise while preserving the integrity of the original sentences. Parameters such as positive and negative sample ratios, frequency-based noise, and word omission were modified. And even tough enabling the fuzzy matching and neighboring sentence misalignment did not help, by fine-tuning the rest of parameters, the model demonstrated improved performance in terms of precision, recall, F1 score, and MCC. These adjustments allowed the model to better distinguish between noise and clean data, leading to enhanced alignment accuracy.

Furthermore, curriculum learning was employed to gradually increase the difficulty of the training examples. The training data was sorted by sentence length, enabling the model to learn simpler patterns first and then go about more complex patterns. This approach proved to be effective, as it significantly enhanced the model's performance in terms of precision, recall, F1 score, and MCC. Curriculum learning provided the model with a structured learning experience, improving its ability to handle various complexities in the data.

After evaluating different hyperparameters, it was found that increasing the number of hidden units from 2048 to 4096 led to improved performance across multiple evaluation metrics. This change allowed the model to capture more complex relationships in the data, resulting in better discrimination between noise and clean data.

However, some approaches did not lead to substantial improvements. Modifying the activation function, learning rate, and decay rate did not result in notable improvements compared to the base model. The default values for these hyperparameters proved to be effective in achieving a better performance.

In conclusion, through a exploration of various training parameters and setups, significant improvements were achieved in the Bicleaner AI full model. The incorporation of curriculum learning, adjustments to noise parameters, and an increase in the number of hidden units led to enhanced alignment accuracy and improved performance in terms of precision, recall, F1 score, and MCC. These findings provide valuable insights for optimizing and fine-tuning the Bicleaner AI full model, contributing to its effectiveness.

	Precision	Recall	F1	MCC
Base model	0.841	0.883	0.862	0.846
Noise parameters	0.892	0.938	0.914	0.889
Curriculum learning	0.908	0.935	0.921	0.913
N_hidden	0.885	0.881	0.888	0.869

Table 4.2: Bicleaner AI (full). Comparison of base model versus each approach model individually with test data. The tuning of the parameters to generate the noise during training, using curriculum learning and increasing the number of units in the hidden layers.

Firstly, in table 4.2 we can see again the improvements each of the approaches made individually. Even though, the three of them improved the base model, they did not make a difference the same way. Firstly, the use of curriculum learning made the most difference by improving significantly all of the metrics. Secondly, the adjustments of the noise parameters, improved all metrics, but significantly the recall and F1. And lastly, even tough the improvement of the increase of units in the hidden layers is not a great as the other, it made still an improvement of the model.

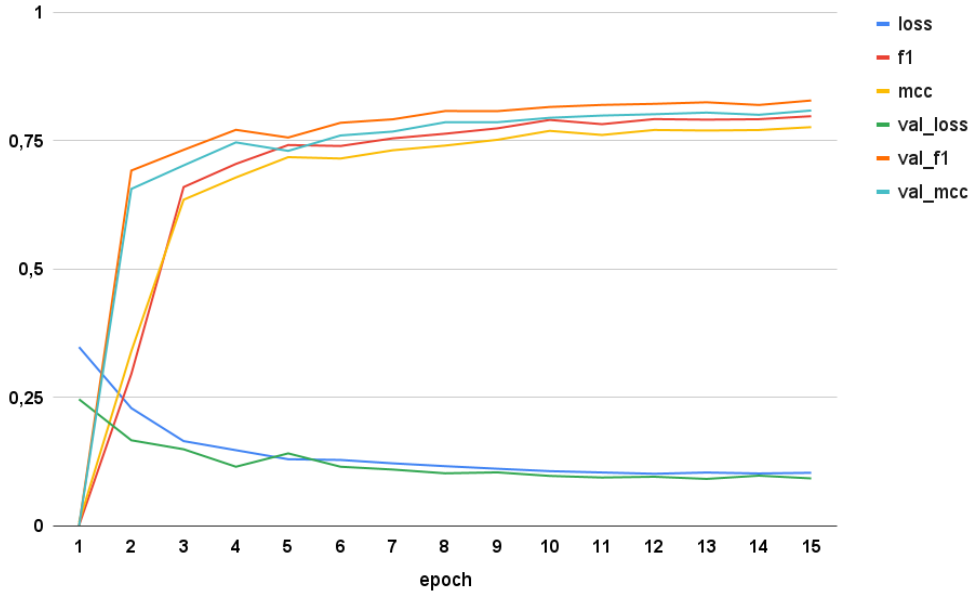
The final model will be created combining the three beneficial approaches (stored here), using curriculum learning, increasing the units in the hidden layer to 4096 and increasing the values of the noise parameters. Because of this combination the NVIDIA GeForce RTX 3080 GPU (10GB) were it is executed, the normally batch size used until now cannot be handle, so the batch size had to be reduced to 8 in order to train this final model.

Figure 4.2 shows the comparison of the progression of the training of both the base model and this new model presented. And even though the base model only has 15 epochs, it does not reach to the new model values. Apart from the epochs, there is a significant difference between, were this new model is notably better.

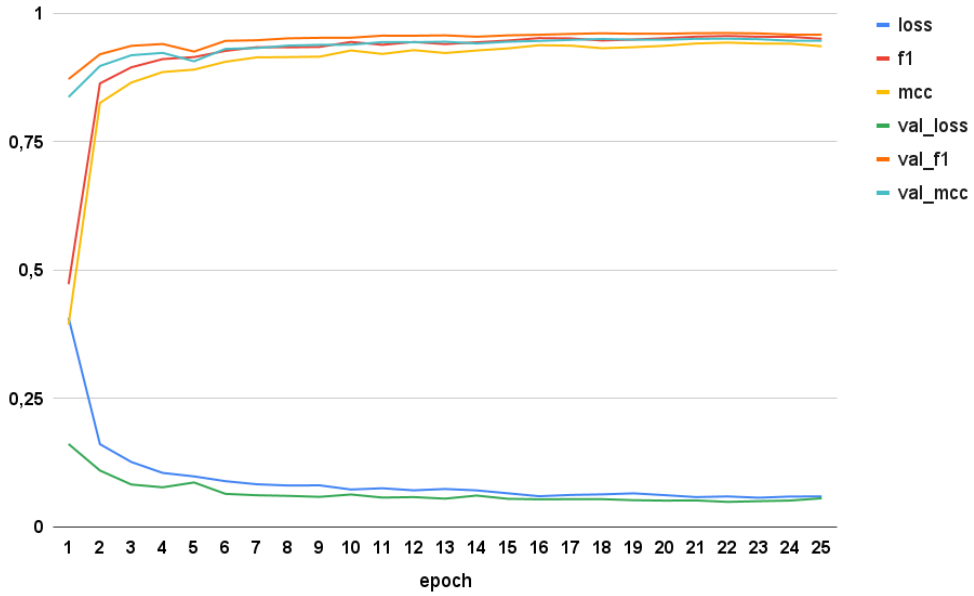
Combining all the approaches and as the figure showed, table 4.3 also shows this significant improvement, were all four metrics increased notably compared to the base model metrics.

To ensure that these are confident values, the training will be executed a total of 10 times, to get accurate average values.

- Precision: 0.956, 0.946, 0.959, 0.957, 0.952, 0.952, 0.957, 0.955, 0.957, 0.946
  - Average: 0.953
  - Variance: 0.00002
  - Standard deviation: 0.004
- Recall: 0.962, 0.967, 0.961, 0.966, 0.967, 0.963, 0.964, 0.970, 0.963, 0.970
  - Average: 0.965
  - Variance: 0.00001
  - Standard deviation: 0.003
- F1: 0.959, 0.956, 0.960, 0.961, 0.959, 0.958, 0.960, 0.962, 0.960, 0.958
  - Average: 0.959



(a) Base model



(b) New model

Figure 4.2: Progression of the training of the Bicleaner AI base model compared to the new presented model.

- Variance: 0.000002
- Standard deviation: 0.001

- MCC: 0.947, 0.943, 0.948, 0.950, 0.948, 0.945, 0.949, 0.951, 0.948, 0.946
  - Average: 0.947
  - Variance: 0.000005

- Standard deviation: 0.002

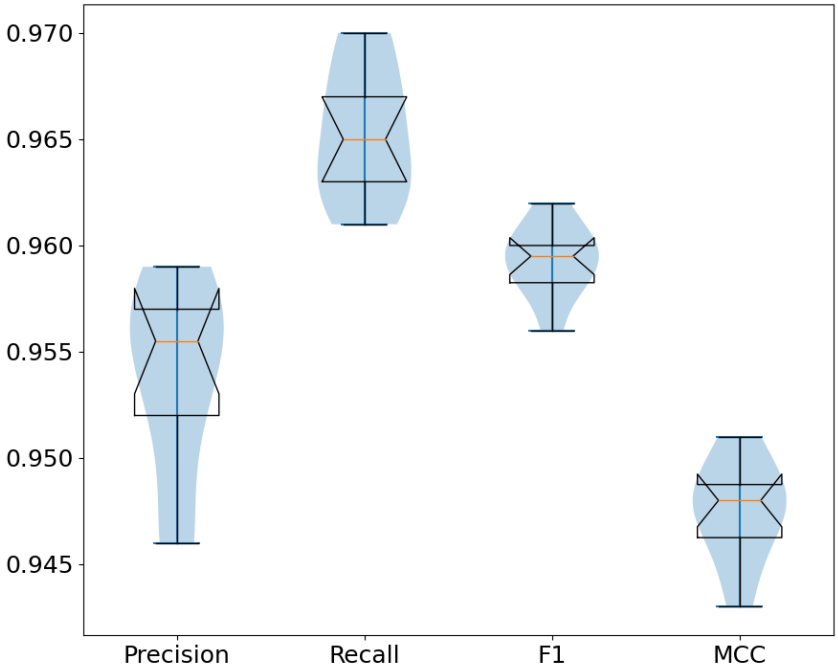


Figure 4.3: Violin+box plot graphs of the values of each metric from a total of 10 trainings of our final model.

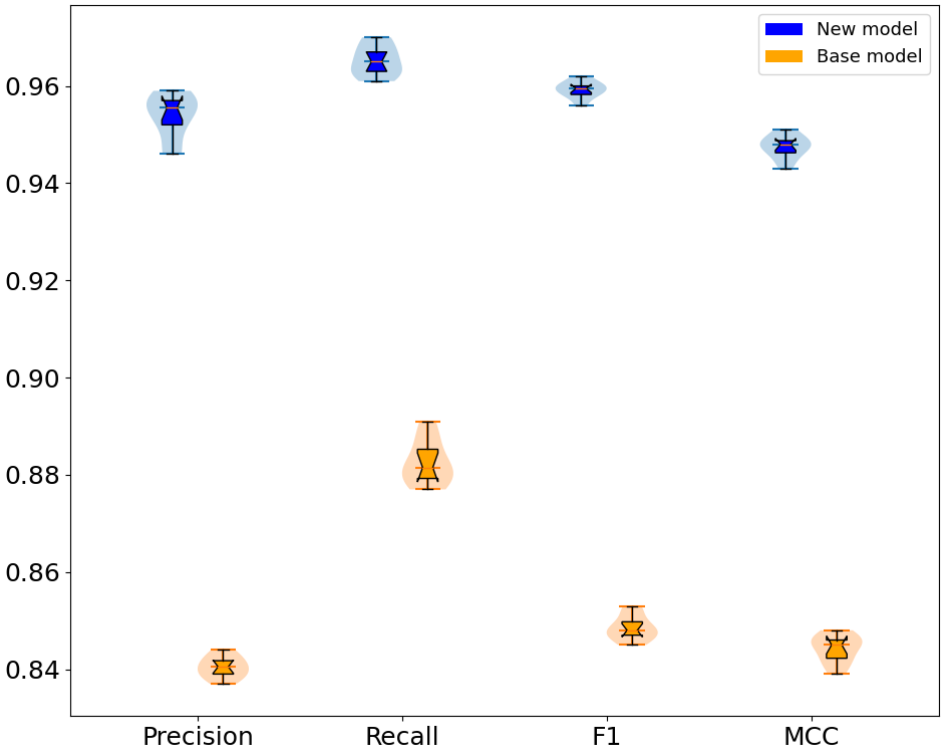


Figure 4.4: Violin+box plot graphs of the values of each metric from a total of 10 trainings of both, comparing the base model with our final model.

Figure 4.3 shows the range of values gotten from training this final model a total of 10 times. It shows there is not much of a variance for any of them, having only a range of 0.013 for the wider one, precision. Also executing the base model other 10 times to get also margin of error and compare the values to the new model in the figure 4.4.

	Precision	**Recall**	**F1**	**MCC**
Base model	0.841±0.0007	0.883±0.0014	0.862±0.0008	0.846±0.0009
New model	0.953±0.0015	0.965±0.0010	0.959±0.0005	0.947±0.0007

Table 4.3: Bicleaner AI (full). Comparison of base model versus (our) new model with test data. Our model enhances all the metrics.

Lastly, table 4.4 shows differences in the output classification were the pre-trained model misbehaves while out new presented model does not. In the first have of the table we can see some example were the pre-trained model returns scores under the threshold when it should not, and the other half of the table the contrary, it give scores over the threshold when they are not accurate translations.

EN	ES	Pre-trained	Final model
(Ex. 16:27) Therefore, God's message through Moses was: "How long must you people refuse to keep my commandments and my laws?	16:27) Por lo tanto, el mensaje de Dios por medio de Moisés fue: "¿Hasta cuándo tendrán ustedes que rehusar guardar mis mandamientos y mis leyes?	0.463	0.889
A population curb, of course.	Un freno demográfico, naturalmente.	0.446	0.814
Thus, the only lasting solution involves a radical change, which God himself purposes to accomplish.	De modo que la única solución duradera implica un cambio radical, cambio que Dios mismo se propone realizar.	0.470	0.886
A charming silent film entitled A Trip Down Market Street enthralls viewers with its revealing display of life as it was in San Francisco, U.S.A., at the turn of the 20th century.	La película muda titulada A Trip Down Market Street (Paseo por la calle Market) presenta cómo era la vida en San Francisco (Estados Unidos) a principios del siglo XX.	0.482	0.648
What Causes Left-Handedness?	¿Por qué se es zurdo?	0.433	0.750
April-May	abril-mayo	0.316	0.996
David wrote: "Ask, you people, for the peace of Jerusalem.	David escribió: "Pidan la paz de Jerusalén.	0.290	0.820
SYRIA	SIRIA	0.419	0.894
He has made even sluices for the rain, and he brings forth the wind from his storehouses."—Jeremiah 10:12, 13.	Ha hecho hasta conductos para la lluvia, y saca el viento de sus almacenes" (Jeremías 10:12, 13).	0.439	0.828
But what goes right in those that succeed?	¿Cuál es su secreto?	0.723	0.073
(Philippians 4:8) Doing this may not be easy, but with effort it can be done.	Como dice la Biblia, debemos centrarnos solo en las cosas que sean "de seria consideración" y "justas", adoptando ideas positivas y dejando de lado las negativas (Filipenses 4:8).	0.723	0.269
So she continued to worship at the tabernacle and pray to Jehovah, telling him how she felt.	Aunque no podía cambiar su situación, ella confiaba en que Jehová la consolaría.	0.631	0.354
(1 Thessalonians 2:14; 5:11) They helped their brothers and sisters to keep faithful to Jehovah by meeting together and being close friends.	Ellos tuvieron que aguantar persecución cruel, así que fue necesario que se dieran ayuda y ánimo para mantenerse fieles (1 Tesalonicenses 2:14; 5:11). ¿Cómo lo hacían?	0.665	0.344
Why the increase?	¿Cuál fue la razón?	0.711	0.350
Is there anything sinful about sexual relations between husband and wife?—Proverbs 5:15, 18, 19.	(Proverbios 5:15, 18, 19.)	0.654	0.403
But by then the U.S. figure had risen to almost 1 car for every 2 persons.	Actualmente Alemania y Luxemburgo cuentan con 1 vehículo por cada 2 habitantes.	0.753	0.345
(Deut. 7:1) How did this affect the Israelites?	Por el contrario, a medida que se establecían entre las "siete naciones" que ocupaban el territorio, el contacto regular con sus habitantes los llevó a entablar relaciones amistosas con ellos (Deu. 7:1). ¿Cómo les afectó aquello?	0.743	0.200

Table 4.4: Differences in output values of the classification of the JW300 corpus, comparing the pre-trained model and our final model.

Chapter 5

Conclusions

I would like to express my sincere gratitude to all those who have contributed to the completion of my master's thesis. First and foremost, I am immensely grateful to Tudor, from Prompsit, company that developed the Bicleaner tool. His invaluable support, guidance, and expertise were instrumental in my research journey. Tudor's assistance in understanding the intricacies of the tool and his prompt responses to my queries significantly contributed to the successful implementation of my work.

I would also like to extend my appreciation to the Ontology Engineering Group (OEG) at Universidad Politécnica de Madrid. Their generosity in providing access to the high-performance machine was crucial for carrying out the resource-intensive computations required for my experiments, their resources greatly facilitated the execution of my objectives.

Furthermore, I want to express my gratitude to my tutor, Mariano. His guidance, encouragement, and insightful feedback throughout the entire process were invaluable. Mariano's expertise and mentorship played a pivotal role in shaping the direction and quality of my research. His willingness to share his knowledge and provide constructive criticism have been instrumental in my personal and academic growth.

Finally, I am grateful to all the individuals who have provided assistance, feedback, and encouragement throughout my master's thesis journey. Their support, whether intellectual, emotional, or practical, has been immeasurable and deeply appreciated.

I am truly honored to have had the privilege of working with such exceptional individuals and organizations. Their contributions have been fundamental to the successful completion of my master's thesis, and I am very grateful for their unwavering support.

Chapter 6

References

[1] 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 on Association for Computational Linguistics*, ser. ACL '02, Philadelphia, Pennsylvania: Association for Computational Linguistics, 2002, pp. 311–318. DOI: 10.3115/1073083.1073135. [Online]. Available: https://doi.org/10.3115/1073083.1073135.

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

[3] N. Mathur, T. Baldwin, and T. Cohn, *Tangled up in bleu: Reevaluating the evaluation of automatic machine translation evaluation metrics*, 2020. arXiv: 2006.06264 [cs.CL].

[4] X. Song, T. Cohn, and L. Specia, «Bleu deconstructed: Designing a better mt evaluation metric», *Int. J. Comput. Linguistics Appl.*, vol. 4, pp. 29–44, 2013.

[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*, Ann Arbor, Michigan: Association for Computational Linguistics, Jun. 2005, pp. 65–72. [Online]. Available: https://aclanthology.org/W05-0909.

[6] T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi, *Bertscore: Evaluating text generation with bert*, 2020. arXiv: 1904.09675 [cs.CL].

[7] M. Przybocki, K. Peterson, S. Bronsart, and G. Sanders, «The nist 2008 metrics for machine translation challenge—overview, methodology, metrics, and results», *Machine Translation*, vol. 23, no. 2/3, pp. 71–103, 2009. (visited on 04/25/2023).

[8] 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*, ser. HLT '02, San Diego, California: Morgan Kaufmann Publishers Inc., 2002, pp. 138–145.

[9] M. Popović, «ChrF: Character n-gram F-score for automatic MT evaluation», in *Proceedings of the Tenth Workshop on Statistical Machine Translation*, Lisbon, Portugal: Association for Computational Linguistics, Sep. 2015, pp. 392–395. DOI: 10.18653/v1/W15-3049. [Online]. Available: https://aclanthology.org/W15-3049.

[10] M. Snover, B. Dorr, R. Schwartz, L. Micciulla, and J. Makhoul, «A study of translation edit rate with targeted human annotation», in *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, Cambridge, Massachusetts, USA: Association for Machine Translation in the Americas, Aug. 2006, pp. 223–231. [Online]. Available: https://aclanthology.org/2006.amta-papers.25.

[11] P. Koehn, H. Khayrallah, K. Heafield, and M. L. Forcada, «Findings of the WMT 2018 shared task on parallel corpus filtering», in *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, Belgium, Brussels: Association for Computational Linguistics, Oct. 2018, pp. 726–739. DOI: 10.18653/v1/W18-6453. [Online]. Available: https://aclanthology.org/W18-6453.

[12] P. Koehn, F. Guzmán, V. Chaudhary, and J. Pino, «Findings of the WMT 2019 shared task on parallel corpus filtering for low-resource conditions», in *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 54–72. DOI: 10.18653/v1/W19-5404. [Online]. Available: https://aclanthology.org/W19-5404.

[13] G. Ramírez-Sánchez, J. Zaragoza-Bernabeu, M. Bañón, and S. Ortiz-Rojas, «Bifixer and bicleaner: Two open-source tools to clean your parallel data.», in *Proceedings of the 22nd Annual Conference of the European Association for Machine Translation*, Lisboa, Portugal: European Association for Machine Translation, Nov. 2020, pp. 291–298, ISBN: 978-989-33-0589-8.

[14] V. M. Sánchez-Cartagena, M. Bañón, S. Ortiz-Rojas, and G. Ramírez-Sánchez, «Prompsit's submission to wmt 2018 parallel corpus filtering shared task», in *Proceedings of the Third Conference on Machine Translation, Volume 2: Shared Task Papers*, Brussels, Belgium: Association for Computational Linguistics.

[15] M. Esplà-Gomis, V. M. Sánchez-Cartagena, J. Zaragoza-Bernabeu, and F. Sánchez-Martínez, «Bicleaner at WMT 2020: Universitat d'alacant-prompsit's submission to the parallel corpus filtering shared task», in *Proceedings of the Fifth Conference on Machine Translation*, Online: Association for Computational Linguistics, Nov. 2020, pp. 952–958. [Online]. Available: https://aclanthology.org/2020.wmt-1.107.

[16] P. Geurts, D. Ernst, and L. Wehenkel, «Extremely randomized trees», *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr. 2006. DOI: 10.1007/s10994-006-6226-1. [Online]. Available: https://doi.org/10.1007/s10994-006-6226-1.

[17] A. P. Parikh, O. Täckström, D. Das, and J. Uszkoreit, *A decomposable attention model for natural language inference*, 2016. arXiv: 1606.01933 [cs.CL].

[18] 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)*, Doha, Qatar: Association for Computational Linguistics, Oct. 2014, pp. 1532–1543. DOI: 10.3115/v1/D14-1162. [Online]. Available: https://aclanthology.org/D14-1162.

[19] A. Conneau, K. Khandelwal, N. Goyal, *et al.*, «Unsupervised cross-lingual representation learning at scale», in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online: Association for Computational Linguistics, Jul. 2020, pp. 8440–8451. DOI: 10.18653/v1/2020.acl-main.747. [Online]. Available: https://aclanthology.org/2020.acl-main.747.

[20] Y. Liu, M. Ott, N. Goyal, *et al.*, *Roberta: A robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[21] G. Lample and A. Conneau, *Cross-lingual language model pretraining*, 2019. arXiv: 1901.07291 [cs.CL].

[22] T. Kudo and J. Richardson, *Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing*, 2018. arXiv: 1808.06226 [cs.CL].

[23] T. Kudo, *Subword regularization: Improving neural network translation models with multiple subword candidates*, 2018. arXiv: 1804.10959 [cs.CL].

[24] D. Chicco and G. Jurman, «The advantages of the matthews correlation coefficient (mcc) over f1 score and accuracy in binary classification evaluation», *BMC Genomics*, vol. 21, Jan. 2020. DOI: 10.1186/s12864-019-6413-7.

[25] J. "Zaragoza-Bernabeu, G. Ramírez-Sánchez, M. Bañón, and S. Ortiz Rojas, «"bicleaner AI: Bicleaner goes neural"», in *"Proceedings of the Thirteenth Language Resources and Evaluation Conference"*, "Marseille, France": "European Language Resources Association", Jun. 2022, "824–831". [Online]. Available: https://aclanthology.org/2022.lrec-1.87.

[26] M. Aulamo, S. Virpioja, and J. Tiedemann, «OpusFilter: A configurable parallel corpus filtering toolbox», in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, Association for Computational Linguistics, Jul. 2020, pp. 150–156. DOI: 10.18653/v1/2020.acl-demos.20. [Online]. Available: https://www.aclweb.org/anthology/2020.acl-demos.20.

[27] M. Aulamo, U. Sulubacak, S. Virpioja, and J. Tiedemann, «OpusTools and parallel corpus diagnostics», in *Proceedings of The 12th Language Resources and Evaluation Conference*, European Language Resources Association, May 2020, pp. 3782–3789, ISBN: 979-10-95546-34-4. [Online]. Available: https://www.aclweb.org/anthology/2020.lrec-1.467.

[28] R. Vázquez, U. Sulubacak, and J. Tiedemann, «The University of Helsinki submission to the WMT19 parallel corpus filtering task», in *Proceedings of the Fourth Conference on Machine Translation (Volume 3: Shared Task Papers, Day 2)*, Florence, Italy: Association for Computational Linguistics, Aug. 2019, pp. 294–300. DOI: 10.18653/v1/W19-5441. [Online]. Available: https://aclanthology.org/W19-5441.

[29] P. Golik, P. Doetsch, and H. Ney, «Cross-entropy vs. squared error training: A theoretical and experimental comparison», in *Interspeech*, 2013.

[30] A. J. Bowers and X. Zhou, «Receiver operating characteristic (roc) area under the curve (auc): A diagnostic measure for evaluating the accuracy of predictors of education outcomes», *Journal of Education for Students Placed at Risk (JESPAR)*, vol. 24, pp. 20–46, 2019.

[31] W. P. Fox, «Undergraduate understanding of the sum of squared errors in modeling with least squares», *PRIMUS*, vol. 2, pp. 375–385, 1992.

[32] H. Bozdogan, «Model selection and akaike's information criterion (aic): The general theory and its analytical extensions», *Psychometrika*, vol. 52, pp. 345–370, 1987.

[33] V. és véletlen számítás and matematikai statisztikák, «Bayesian information criterion», *The SAGE Encyclopedia of Research Design*, 2022.

[34] V. Siivola, T. Hirsimaki, and S. Virpioja, «On growing and pruning kneser–ney smoothed n-gram models», *Audio, Speech, and Language Processing, IEEE Transactions on*, vol. 15, pp. 1617–1624, Aug. 2007. DOI: 10.1109/TASL.2007.896666.

[35] R. Östling and J. Tiedemann, «Efficient word alignment with Markov Chain Monte Carlo», *Prague Bulletin of Mathematical Linguistics*, vol. 106, pp. 125–146, Oct. 2016. [Online]. Available: http://ufal.mff.cuni.cz/pbml/106/art-ostling-tiedemann.pdf.

[36] M. Bhatt, V. Dahiya, and A. K. Singh, «Supervised learning algorithm: Svm with advanced kernel to classify lower back pain», *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, pp. 17–19, 2019.

[37] T. Seidl, «Nearest neighbor classification», in *Encyclopedia of Database Systems*, L. LIU and M. T. ÖZSU, Eds. Boston, MA: Springer US, 2009, pp. 1885–1890, ISBN: 978-0-387-39940-9. DOI: 10.1007/978-0-387-39940-9_561. [Online]. Available: https://doi.org/10.1007/978-0-387-39940-9_561.

[38] R. E. Schapire, «Explaining adaboost», in *Empirical Inference: Festschrift in Honor of Vladimir N. Vapnik*, B. Schölkopf, Z. Luo, and V. Vovk, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 37–52, ISBN: 978-3-642-41136-6. DOI: 10.1007/978-3-642-41136-6_5. [Online]. Available: https://doi.org/10.1007/978-3-642-41136-6_5.

[39] L. Breiman, «Random forests», *Machine Learning*, vol. 45, no. 1, pp. 5–32, Oct. 2001, ISSN: 1573-0565. DOI: 10.1023/A:1010933404324. [Online]. Available: https://doi.org/10.1023/A:1010933404324.

[40] P. Geurts, D. Ernst, and L. Wehenkel, «Extremely randomized trees», *Machine Learning*, vol. 63, no. 1, pp. 3–42, Apr. 2006, ISSN: 1573-0565. DOI: 10.1007/s10994-006-6226-1. [Online]. Available: https://doi.org/10.1007/s10994-006-6226-1.

[41] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, «Curriculum learning», vol. 60, Jun. 2009, p. 6. DOI: 10.1145/1553374.1553380.

[42] A. F. Agarap, *Deep learning using rectified linear units (relu)*, 2019. arXiv: 1803.08375 [cs.NE].

[43] D. Hendrycks and K. Gimpel, *Gaussian error linear units (gelus)*, 2023. arXiv: 1606.08415 [cs.LG].