National College of Ireland

# Transformer Framework for Language Translation in Low Resource Languages

MSc Research Project
MSc in Artificial Intelligence

Muhammet Gumus
Student ID: 23330562

School of Computing
National College of Ireland

Supervisor: Prof. Paul Stynes

**National College of Ireland**

**MSc Project Submission Sheet**

**School of Computing**

**Student Name:** Muhammet Gumus

**Student ID:** 23330562

**Programme:** MSc in Artificial Intelligence **Year:** 2024/2025

**Module:** Practicum Part 2

**Supervisor:** Prof. Paul Stynes

**Submission Due Date:** 11.08.2025

**Project Title:** Transformer Framework for Language Translation in Low Resource Languages

**Word Count:** 9532 **Page Count** 21

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project.
ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

**Signature:** Muhammet Gumus

**Date:** 09/08/2025

**PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST**

| | |
|---|---|
| Attach a completed copy of this sheet to each project (including multiple copies) | □ |
| **Attach a Moodle submission receipt of the online project submission,** to each project (including multiple copies). | □ |

| | |
|---|---|
| **You must ensure that you retain a HARD COPY of the project**, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer. | □ |

Assignments that are submitted to the Programme Coordinator Office must be placed into the assignment box located outside the office.

| **Office Use Only** | |
|---|---|
| Signature: | |
| Date: | |
| Penalty Applied (if applicable): | |

# Transformer Framework for Language Translation in Low Resource Languages

Muhammet Gumus
23330562

**Abstract**

Language translation refers to the process of converting written content from one natural language into another while preserving its meaning and intent. Low-resource languages are those that have a limited availability of resources compared to widely spoken languages. A transformer model is a type of neural network architecture used in machine learning, particularly for processing sequential data. While this relatively new architecture is a significant improvement over older machine translation systems, determining which model will consistently produce the highest quality translation for each sentence remains a persistent challenge. This research proposes a transformer framework that leverages multiple transformer-based translation models to detect the best candidate of translation on a per-sentence basis. The proposed framework combines general-purpose multilingual models, LoRA fine-tuned models on domain-specific datasets and a meta-learning component to select the best translation model for each sentence that predicts the most suitable model for each input. Experiments use English–Turkish TED2020, Tatoeba and Opus100 datasets and show practical and research value in efficient model selection for low-resource translation. Translation quality is evaluated using BLEU scores and reference-free signals from large language models, providing informative supervision for the meta-learner. Results show that the meta-learner system powered by transformer models can achieve accuracy up to 71 % with the ensemble strategy of ML models like RandomForest, XGBoost, and LightGBM on the limited datasets and budget.

# 1 Introduction

Transformer frameworks have become the foundation of modern machine translation systems, thanks to their ability to capture long-range dependencies and contextual meaning more effectively than previous architectures. However, these frameworks often struggle to maintain translation quality where parallel corpora and domain-specific data are limited in low-resource languages. This research investigates how transformer based frameworks take a role to enhance the translation quality in low-resource languages through sentence-level model selection. Despite the success of transformer-based models, determining which model will consistently yield the highest quality translation for each sentence remains a persistent challenge. This issue is particularly important when both general-purpose and fine-tuned models are available and must be considered dynamically. It presents a practical concern for translation technology practitioners who aim to balance translation quality and efficiency.

Accordingly, this study tries to find an answer the following research question: "*Can a meta learning based transformer framework effectively select the most suitable translation model for each sentence input in low resource language scenarios?*"

The aim of this research is to investigate to what extent a transformer framework automatically selects the most suitable translation model for each input sentence. To address the research question, the following specific sets of research objectives were derived:

1. Investigate the state of the art around transformer-based machine translation models in handling low-resource languages.
2. Design a transformer framework that integrates general purpose and LoRA fine tuned models into an adaptive selection system.
3. Implement a transformer framework scoring mechanism based on reference-free quality estimation using large language models to guide model selection.
4. Evaluate a transformer framework using both reference-based metrics such as BLEU and reference-free methods like GPT-based scoring to assess overall translation model selection performance.

The major contribution of this research is a novel transformer framework that combines a transformer model, statistical model and meta learner model to select the most suitable translation models. To be able to identify the optimal translation strategy, this research compares general-purpose models like google/mt5-small , facebook/nllb-200-distilled-200M, mBART models and including their LoRA fine-tuned versions.

The meta-learning element serves to dynamically select the optimal translation model for an input sentence. Meta learning elements such as sentence embeddings, linguistic features and comparative model scores are highly important for guiding the model selection. This strategy shows promise in effectively streamlining the translation process by ensuring only the most appropriate model is selected.

This paper discusses related work in Section 2, outlines the research methodology in Section 3, presents system design in Section 4, implementation details in Section 5, evaluation results in Section 6, and concludes with future directions in Section 7.

# 2 Related Work

The field of machine translation has undergone a significant transformation from rulebased approaches to deep learning methods. Thanks to these developments, translation quality has generally improved. However, the performance of translation systems for lowresource languages is still limited. This literature review examines both the structure of existing approaches and their effects in low-resource scenarios. The review is structured under five main headings: the evolution of translation systems, transformer architectures, challenges in low-resource languages, parametric efficient fine-tuning methods and evaluation approaches. Finally, meta-learning studies for translation model selection are also evaluated.

## 2.1 The Evolution of Machine Translation

Tang et al. (2020) use a denoising pre-training method to develop multilingual translation systems. The aim of this research is to increase the translation accuracy in the situation of parallel data lack. The method involves training a transformer model to predict the original form of corrupted input sentences. In this point, experiments have shown significant quality improvements in low-resource scenarios. Its power lies in its ability to utilize large scale monolingual data. On the other hand, it also comes with some negativities like high computational needs and long training time.

Xue et al. (2021) propose a multilingual, text-to-text transformer architecture called mT5. The goal is to enable strong transfer between different languages. The model is an adaptation of the T5 architecture to a wide range of languages. Positive results are obtained in tests conducted in multiple languages. Its advantage is its multilingual generalization success. However, performance may decrease in some languages when working with low-volume data.

## 2.2 Transformer Architecture and Multilingual Models

Transformer architecture is considered a groundbreaking structure in machine translation. Thanks to its attention mechanism, long-range correlations can be captured. This enables it to produce more meaningful translations.

The NLLB Team (2022) created a system that supports over 200 languages and its main focus is human centred translation approach. The purpose here is to provide high quality translation for low resource languages. The system follows and contains technical steps like balanced data sampling, tokenization and multilingual modelling. The system delivers results close to human-level quality in many languages. Its strong point is its extensive language coverage. However, the model size and infrastructure requirements are costly.

The mBART model (Tang et al., 2020) was developed using a pre training strategy based on automatically correcting damaged sentences. The main aim here is to enable multilingual translation without parallel data. Here, the method learns how to form correct sentences from noisy input using an encoder-decoder structure. The results show significant improvement, especially in rare languages. Its advantage is the ability to use extensive monolingual data. However, it requires precise hyper parameter tuning and has a long training time.

## 2.3 Challenges in Low-Resource Languages

Translation systems in low resource languages often fail due to insufficient and few data. This situation makes both model training and domain adaptation processes complex. On the other hand, using multilingual training data in a balanced structure is used by the NLLB Team (2022) to solve this problem. As a method, data sampling techniques and filtering mechanisms are used to strengthen the representation of rare languages. The results show an increase in quality in low-resource languages. However, high hardware requirements and complex data processing structures limit the application.

Kang et al. (2025) propose an approach called DaCoM. The goal is to produce lowresource and domain-specific (e.g., legal, medical) translation datasets. The method involves applying specialized data creation strategies that incorporate domain-specific terms and structures. The results show an increase in domain-specific translation quality. However, the method is time-consuming due to the manual data collection and processing process.

Chronopoulou (2023) tests low cost fine-tuning methods for adapting models in environments with limited data. The some specific methods like AdapterFusion and LoRA used that include PEFT techniques. The results show that domain adaptation is possible with limited data. However, the methods may not yet be fully effective in some tasks.

These studies explain the fundamental issues encountered in low-resource languages and domains. They also theoretically support the necessity of the automatic selection and domain adaptation system presented in the thesis.

## 2.4 Parameter-Efficient Fine-Tuning (PEFT) Approaches

It is difficult to train large language models directly in low-resource scenarios. Therefore, PEFT methods have come to the fore in recent years. These techniques adapt the model to the target task by updating only a small portion of it.

Hu et al. (2021) developed a method called LoRA (Low-Rank Adaptation). The goal is to customize large language models by fine-tuning them with fewer parameters. In this method, fixed model weights are preserved; only low-rank matrices are trained. This way, model performance is maintained while the number of parameters is significantly reduced. Its strengths come from its efficiency and modularity. However, selecting the appropriate matrix size often requires too many tries.

Pfeiffer et al. (2021) present the AdapterFusion approach. The goal is to enable reuse by combining adapter modules trained for different tasks. As a method, separate adapter layers are added for each task and then dynamically combined. This provides flexibility for transfer learning. The method yields successful results; however, the system can become complex, and the combination process requires careful attention.

Ben Zaken et al. (2022) propose the BitFit method. In this method, only bias terms are trained. The goal is to adapt the model's behavior with small changes. The method is very simple and fast and it is particularly effective when working with small data sets. However, when it comes to the deep tasks performance may be limited.

## 2.5 Translation Quality Evaluation: Traditional Metrics and LLMSupported Approaches

The success of machine translation systems must be measured using accurate evaluation methods. While traditional metrics have been used for this purpose for a long time, LLM-based reference-free evaluation methods have come to the fore in recent years.

Papineni et al. (2002) introduced the BLEU score to measure translation quality based on n-gram overlap. The method is based on structural similarity between the system output and the reference text. The results provide meaningful comparisons at the surface level. Its advantage is that it is fast and automatic. However, it cannot evaluate deep features such as word order and semantic accuracy.

Snover et al. (2006) developed the TER (Translation Edit Rate) metric. The goal is to measure how close the system translation is to the reference text in terms of editing. The method calculates the required additions, deletions, and rearrangements. It provides a more accurate evaluation. However, it is still dependent on the reference and may penalize different expressions.

Zhang et al. (2020) introduce the BERTScore metric to measure semantic similarity. The goal is to compare the semantic overlap between the reference and the system output using BERT-based embeddings. The method considers the similarity between word vectors. The results produce scores that are more consistent with human judgment. However, the computational cost is high due to the model being weight-based.

Zaidan and Callison-Burch (2011) highlight the importance of human judgment. The aim is to observe the differences between professional and amateur translation. Human evaluations and crowdsourcing are used as methods. The results reveal that automatic metrics are insufficient in some cases. Although the method is reliable, it requires human resources.

Kocmi et al. (2023) investigate the evaluation success of LLMs. The aim is to test whether LLMs can evaluate like humans. Example translation pairs were presented to GPT models, and quality scores were requested. The findings show that LLMs produce results very close to human judgments. The advantage is that they can evaluate without needing a reference. However, model biases remain an issue.

Liu et al. (2023) present the G-Eval system. Text generation with GPT-4 is evaluated. The aim is to provide both scoring and explanatory quality analyses. The method relies on obtaining qualitative and quantitative output from the model with explicit instructions for translation examples. The results produce insights that are similar to traditional metrics but more detailed. However, the reliability of the method still depends on model behavior.

Gu et al. (2018) combine evaluation and selection processes with meta-learning approaches. The goal is to make quality predictions that can be used in model selection with limited data. The model learns from past outputs to select the best system. The method directly links translation quality signals to model selection. Even though the results are positive, meta-learning structures remain complex and dependent on the data.

MQM-APE (2023) performs error estimation in translation output using LLM. The goal is to automate the post editing process. LLM can be used for error detection. However, success varies across language pairs.

Liu et al. (2023) in "LLMs Are Not Scorers" discuss some of the weaknesses of LLMs in terms of evaluation. The aim is to test whether these systems can provide stable scores. The findings show that scores are context-sensitive and variable. This situation can create security vulnerabilities in LLM-based systems.

Yin et al. (2023) propose Dialect-guided LLM evaluation. The method provides evaluation without reference in low-resource languages. It involves guiding GPT with structured prompts. The results are consistent with classical metrics. This demonstrates that LLMs can play an effective role in the translation evaluation process.

## 2.6 Translation Model Selection with Meta-Learning

Current translation systems are often limited to the use of fixed models. Meta-learning techniques are emerging as a way to select the most appropriate model for different sentence types.

Gu et al. (2018) aims to improve translation performance by using meta learning in low resource scenarios. The goal is to develop systems that can quickly adapt with a small number of examples. As a method, a meta-learner model is trained using a task set consisting of translation tasks. The results have been particularly successful in adapting to new language pairs. Its strength lies in its rapid adaptation; however, it requires a large number of pre-tasks. Finn et al. (2017) introduce the model-agnostic meta-learning (MAML) algorithm. The goal is to prepare the learner for new tasks. The method optimizes the model's initial parameters in a way that allows for quick updates. This method can be used in selecting a translation model.
Its advantage is that it can be applied to different model types but its weakness is the high computational load.

The literature review reveals that current systems have limited success in low-resource scenarios. Traditional evaluation methods are often insufficient, and reference dependency creates problems in measuring true quality. Additionally, there is a lack of an effective and dynamic structure for selecting the most appropriate translation from among the outputs of

different models. In this context, while the methods proposed by previous studies offer important contributions, they fail to fully address current needs. Therefore, developing a meta-learning-based model selection system that combines parametric efficient models with LLM-based evaluation has become crucial. This research aims to fill this gap and improve translation quality in low-resource languages.

# 3 Research Methodology

The main objective of this study is to compare the performance of different large language models in English Turkish translation tasks and to develop a meta-learner system that can predict the most appropriate translation model for a new sentence using these outputs. For this purpose, the methodological process is divided into five stages. These are sequentially data collection, data preprocessing, fine-tuned model training, meta-learner model training, and evaluation methods. Each stage has been carefully structured to support the validity of the experimental design and enhance the interpretability of the results.

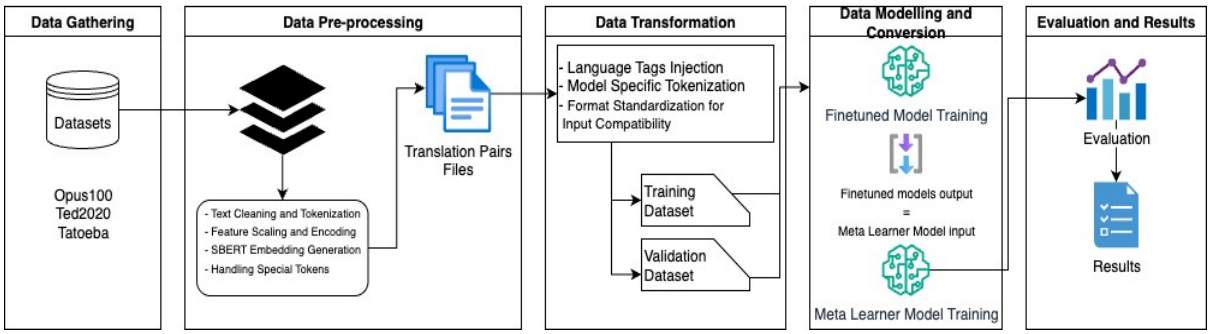



**Figure 1.** Research Methodology

## 3.1 Data Gathering

In this study, three main parallel datasets were obtained from public sources and used. They are Opus-100, TED2020 and Tatoeba. They were obtained from publicly available sources. They consist of English Turkish translation pairs covering various topics such as daily conversations, short sentences and educational contents. These datasets approximately contain half a million to one million English Turkish examples. However, due to limited computational power and cost constraints, a subset of the first 100,000 sentence pairs from each dataset was selected for use in the model training process. This selection's main purpose was providing meaningful representation while saving GPU time costs.

During the testing phase, an initial pre-test set was created by randomly selecting 100 examples from each dataset. This phase was conducted to establish the evaluation infrastructure and test the metrics. In the next phase, this test set was expanded for a more stable evaluation, and final test sets were created by randomly selecting 1,000 samples (3 x 1,000) from each dataset. This made performance comparisons more reliable while preserving data diversity.

## 3.2 Data Pre-Processing

The data pre-processing process was carried out systematically to ensure that language models were trained properly and to increase the reliability of evaluation metrics. First, English to Turkish sentence pairs obtained from the TED2020, Tatoeba and Opus100 datasets were converted into a common structure and data formats were unified. At that point, sentence alignments were checked and incorrect matches were eliminated. After that, rows containing

missing (NaN) values were removed from all datasets. Additionally unnecessary whitespace characters, HTML like special symbols and formatting issues were normalized to clean the text. Extra characters at the beginning or end of sentences were removed using text related operations such as strip().

Tokenization processes were performed separately for each model according to its own structure. For example, mT5 models use SentencePiece-based subword tokenization, mBART models work with byte-level BPE (BBPE) architecture and NLLB performs language-aware processing through its own custom tokenizer structure. Taking these differences into account, each model's tokenizer was called in its own local format to ensure model-input compatibility.

Special tokens (e.g., <extra_id_0>, <pad>, <s>, </s>) in the outputs of some models may appear before or after the text. Since these tokens can affect the score calculations of classic automatic metrics such as BLEU and ChrF++, as well as GPT-3.5-based evaluation systems, they were manually removed from the texts prior to evaluation. This cleaning process enabled more accurate formal and semantic comparisons.

At the end of this preprocessing process, all training and test examples were cleaned, aligned, and restructured to meet the input requirements of each model. This restructuring represents a critical step in terms of both the consistency of model comparisons and the feeding of the meta-learner system.

## 3.3 Finetuned Model Training

The translation models used in this study are three transformer-based architectures that have been pre-trained on large multilingual datasets and are available as open source: mT5small, NLLB-200-distilled, and mBART-50. The main purpose of selecting these models was to enable the comparison of different architectural structures and to test the effectiveness of models with more compact structures in low-resource scenarios. The mT5-small model, in particular, is notable for having approximately 300 million parameters and relatively low computational costs. This model was selected with the aim of observing whether small-scale transformer models can provide sufficient performance in translation tasks. The mBART model was used in this study with the mBART-50 configuration instead of the large version, which kept the total number of parameters at a more manageable level. The NLLB model, also offered by Facebook, was chosen in its more efficient NLLB-200-distilled-600M version.

Each model was fine-tuned separately on three different datasets: TED2020, Tatoeba, and Opus100. This resulted in a total of nine different models. Each model was trained only with its own target dataset, and no mixing of datasets was performed. This approach aims to establish a direct learning relationship between the model and the dataset and prevent crossinteraction.

The training processes were carried out on the Google Colab Pro+ platform in a GPUsupported environment. The training process was limited to only 1 epoch. This limitation was determined in order to efficiently use computational resources, reduce time costs, and evaluate the transfer learning capacity at a basic level. The AdamW algorithm was used for optimization processes. Hyperparameters were kept close to their default values to ensure stable learning of the model.

The translation process may require some special prefix for some models like mT5small.In here, the translation direction was specified with an prefix. It contains sentence like “translate English to Turkish: ”. This sentence can be customized to other source and target languages. This structure ensures that the encoder part is directed toward the target language. In the NLLB-200 and mBART-50 models, the target language was determined using

forced_bos_token_id. This application played an important role in directing the translation outputs to the correct language. The obtained models were stored along with their outputs under a standard file and directory structure, thus providing a consistent and reproducible structure for the evaluation processes to be carried out in the subsequent stages.

## 3.4 Meta Learner Model Training

The main goal of the meta-learner system developed in this study is to predict the model that will produce the highest quality translation based solely on the source sentence. This system aims to provide an automatic and low-cost selection mechanism that can be integrated into multi-translation model infrastructures. To create a labeled dataset for the meta-learner, a total of 3,000 sentences were randomly selected from the TED2020, Tatoeba, and Opus100 datasets; each sentence was translated using fine-tuned mT5, NLLB, and mBART models. These translation outputs were scored using traditional metrics like BLEU, ChrF++, METEOR and TER. In addition to them, fluency and adequacy evaluations supported by GPT-3.5. The model that provided the highest overall quality score for each sentence was designated as the "best_model_label" and used as the target variable for classification.

The dataset includes the source sentence (source), model output (generated_translation), reference translation (reference), traditional metric scores such as BLEU and ChrF++, and structural and semantic features such as GPT-based gpt_score, gpt_fluency_score, gpt_adequacy_score,best_model_label,source_length_words, and length_ratio_source_target. Additionally, 768-dimensional vectors based on SBERT, which provide a semantic representation of the source sentences, were used as input features.

As classification algorithms, deep learning-based structures as well as Random Forest, XGBoost and ensemble combinations of these algorithms were tested. These models were preferred because they provide high accuracy,interpretability and giving opportunity of generalizing effectively with a small number of examples. In parameter selection, the following hyperparameter settings were used: for Random Forest, n_estimators=200, max_depth=None, class_weight='balanced'; for XGBoost, max_depth=6, learning_rate=0.1, n_estimators=150, scale_pos_weight. Due to the different scales of the input variables, all numerical data were normalized using MinMaxScaler. Stratified shuffle split was applied during the training process and a weighted classification strategy was adopted to decrease the effect of class imbalances.

At the code level, basic neural network tests were also performed with MLPClassifier, and the results were found to be similar in performance. However, decision tree-based methods were preferred as they provided more stable results in terms of both training time and overall accuracy. The entire modeling process was performed on a GPU-supported environment on Google Colab Pro+.

## 3.5 Evaluation and Results

In this study, the outputs of both fine-tuned translation models and meta-learner classification models were analyzed using a multidimensional evaluation process. The metrics used are divided into two groups. They are traditional automatic metrics and reference-free evaluation techniques that are based on large language models (LLMs). All evaluation metrics BLEU, CHRF, TER, METEOR, BERTScore and GPT based fluency/adequacy are defined in this section and are referenced in later chapters to avoid repetition.

Traditional metrics include scores such as BLEU, ChrF++, METEOR and TER. In here, BLEU is used for the calculation of n-gram overlaps between the translation and the reference

text. On the other hand, ChrF++ used for getting a more detailed evaluation at the character level. Additionally, METEOR score is used to obtain a more semantically rich measurement by taking into account word roots, synonyms and word order. Finally, TER reflected the translation errors based on the editing processes required to convert the model output into the reference. All these scores were calculated in the Python environment using the sacrebleu and evaluate libraries.

However, due to the lack of contextual understanding in traditional metrics, a referencefree evaluation process based on GPT-3.5 has also been developed. Each translation output was scored from 1 to 5 under the headings of fluency and adequacy and the average of these two scores was defined as gpt_score. This method acts as a powerful complement in measuring idiomatic structures, syntactic rules and context-dependent meaning.

Classic classification metrics such as precision, recall, and F1-score were also used to evaluate the classification performance of the meta-learner model. These scores provided discrete evaluation on a class-by-class basis while also balancing overall performance. The metrics were calculated using the scikit-learn library in the Python environment and the effectiveness of the classification process was measured in terms of both accuracy and balance.

To ensure that the evaluation process could be applied consistently across all data, all output files were formatted according to a standard structure, enabling each score to be used comparatively in model performance analysis. This approach allowed for a multifaceted analysis of translation quality and model selection by combining both quantitative and qualitative criteria.

# 4 Design Specification

The proposed transformer framework architecture combines individual fine-tuned translation models, semantic embedding techniques, a meta-learner classification model and an evaluation module as shown in Fig. 2. The components of this architecture include the data processing pipeline, base translation models, the meta-learner, and the evaluation and selection module, which are described in detail in sections 4.1, 4.2, 4.3, and 4.4, respectively.

## 4.1 Data Processing and Feature Engineering

The first component of the translation system proposed in this study is the data processing pipeline. This structure ensures that raw data is processed and made suitable for model training. Architecturally, this layer consists of three main sections: data set acquisition, linguistic preprocessing, and semantic vector creation.

In the first stage, three different multilingual datasets were used: TED2020, Tatoeba, and Opus100. These datasets were organized under the system's data folders. Each dataset is kept independent within itself. This allowed separate model training to be performed for each one, and the model's sensitivity to contexts was observed.

After the data is collected, the linguistic preprocessing process begins. This process is carried out using special scripts written in Python. Open source libraries such as transformers, datasets, and tokenizers were used in the Python environment. These steps include:

- Texts are read in UTF-8 format and stripped of special characters.
- According to the structure of tokenizers that needs to be used for model, sentences are splitted. For example, SentencePiece was used for mT5, and Byte-Pair Encoding (BPE) was used for mBART and NLLB.
- Appropriate language codes are added to each sentence pair (e.g., en_XX → tr_TR).

- Length and character checks are performed. E.g. very long sequences are filtered or limited.

After preprocessing, semantic vectors are created for each sentence. For this process, the sentence-transformers library and the Sentence-BERT (SBERT) model were used. SBERT outputs convert sentences into vectors of fixed length. These vectors are used in the system's subsequent stages for model evaluation and selection processes.

All these processes are managed within the system using a structured file system. Separate folders have been created for each dataset and model type, containing tokenized data, SBERT outputs, and training inputs. This structure ensures the system's reproducibility and modularity.

## 4.2 Fine-Tuned Model Architecture

The models used in this system are transformer based architectures that have been pre trained with large amounts of multilingual data. These models have been fine-tuned specifically for target data sets. Additionally, they are used to achieve better performance in low-resource languages. At the same time, this adaptation aims to make the models more suitable for specific fields.

The three main models used are mT5-small, mBART and NLLB-200-distilled. The mT5-small model is developed by Google. It is the multilingual version of the T5 architecture and offers low computational costs thanks to its small size. The another model is mBART is developed by Facebook that has an encoder decoder architecture. This is suitable for multilingual translation tasks. The NLLB-200-distilled model is a scaled-down version of a large model published by Meta that supports over 200 languages.

Each model was trained on three separate datasets: TED2020, Tatoeba, and Opus100. A total of nine different models were created. During the training process, each model was trained only on its own dataset, and no mixing of datasets was performed. This structure allowed for a clearer performance measurement by ensuring that the models learned only in their own contexts.

Model training was conducted within certain standards. The training period was limited to only one epoch, and the AdamW algorithm was preferred for optimization.
Hyperparameters were set to a learning rate of 2e-4 and a batch size of 8, which were selected to ensure stable learning of the model. The all training processes were carried out on the Google Colab Pro+ platform in a GPU-supported environment also sometimes just CPU.

The model input structures were also adjusted to suit the tokenizer structures used. For the mT5 model, the input sentences were explicitly directed with the command “translate English to Turkish:”. In the mBART and NLLB models, the translation direction was determined with the forced_bos_token_id parameter. This structure ensured that the correct target language was directed.

In addition to each model, a LoRA (Low-Rank Adaptation) layer was applied. This method allows only low-rank matrices to be updated instead of all model parameters. As a result, the models were adapted with fewer computational resources and gained flexibility without increasing the number of parameters.

Finally, the trained models are stored in the system under a standard folder structure. Each model output is recorded in an organized manner for future evaluation and classification processes.

## 4.3 Meta Learner Classification Model

The meta learner classification system developed in this study aims to predict the most successful translation among those produced by fine-tuned translation models. This classification process relies only on automatic metrics and contextual features.Here, reference sentences are not used. In the first stage, the translation outputs for each fine tuned model were evaluated, and the most successful model was determined using classical automatic metrics such as BLEU, CHRF, METEOR and TER.In addition to them GPT based fluency and adequacy scores were added. Reference sentences were used as a basis for this evaluation. Thus, the "best_model_label" information was obtained for each example.

The meta-learner model was trained based on these labels, but no reference sentences were used during training. The model used only automatic scores, structural information about text lengths and ratios, categorical information such as model identity and dataset, and SBERT embedded representations of source sentences as input. In this way, the system was developed as a classifier for predicting the best model without the need for references.

During the labeling process, "general_model_label," which is a generalized form of "best_model_label," was used to improve the model's performance and reduce the imbalance between classes. For example, detailed labels such as "nllb-200M_opus100" were simplified to "nllb." This simplification enabled the model to perform more stable and meaningful classification, reducing the impact of classes with few examples.

In the classification process, basic ensemble algorithms such as Random Forest and XGBoost were first tested. Then, different models such as LightGBM, CatBoost, Support Vector Machines (SVM) and multi-layer artificial neural networks were also applied to compare performance. Each was trained separately and evaluated based on metrics such as accuracy and classification performance. A manual ensemble approach was applied using a soft voting mechanism based on the outputs of all these models. This method created a more stable decision mechanism by averaging the probability distributions between models. Thus, a robust selection system based on multiple model views was obtained under unsupervised conditions.

## 4.4 Evaluation Metrics and Output Analysis

The definitions and calculation details of all evaluation metrics are provided in Section 3.5. This section focuses on how these metrics were applied to evaluate the performance of the developed models and the meta-learner classifier.

In calculating these metrics, model outputs were directly compared with target sentences. The evaluation results were used to analyze the quality of both individual models and translations selected by the meta-learner. In addition to that, basic metrics such as confusion matrix, accuracy and f1-score were also presented for classification success.

During the visualization of the outputs, heatmaps, accuracy graphs, and class distribution tables were used to clearly highlight performance differences. These analyses helped to understand which models the meta-learner preferred more frequently and to what extent these preferences were consistent with the evaluation metrics.

# 5 Implementation

## 5.1 Evaluation Metrics and Output Analysis

All application stages of this study were carried out using the Python programming language. Google Colab Pro+ was chosen as the development environment, and virtual

machines equipped with NVIDIA T4 GPUs were used on this platform. T4 GPUs provided sufficient speed and memory capacity for the training and inference processes of transformer based large language models. This GPUs ensure that operations were performed efficiently.

In the model evaluation processes, GPT based scorers were also used to analyze translation quality in greater depth. In this context, GPT-4, GPT-4o, and GPT-3.5 Turbo models were tested during the testing phase, but due to practical limitations such as cost and API access time, the main evaluation metrics were primarily conducted using the GPT-3.5 Turbo model. During these processes, OpenAI API integration was provided to obtain fluency and adequacy scores for each translated sentence.

SBERT (Sentence-BERT) based embedding vectors were used in semantic similarity calculations and in creating the input features of the meta learner structure. For this purpose, the paraphrase-multilingual-MiniLM-L12-v2 model, which has a high multilingual sentence representation capability, was selected. Thanks to this model, the semantic representations of both the source and target sentences were extracted, providing the classification model with stronger contextual features.

Numerous open-source Python libraries were used in the implementation of the project. The Transformers and PEFT libraries were used for loading pre-trained transformer models and low cost adaptation techniques (especially LoRA). The Datasets library played a role in the management and loading of processed data sets. Sentence-transformers were used to generate embeddings with SBERT models. The evaluate and openai libraries were integrated for the calculation of evaluation metrics and external service integration. Scikit-learn, LightGBM, CatBoost, SVC and tensorflow keras packages were used for the training and comparative testing of meta learner models. The visualization, analysis and reporting of the obtained results were performed using the matplotlib and seaborn libraries.

A standard folder structure was used to manage all outputs and experiments in a consistent and reproducible manner. Thanks to this structure, model outputs, evaluation scores and classification results were archived systematically, making every stage of the process traceable.

## 5.2 Data Preparation and Conversion

The data used for model training in this project was obtained from three different multilingual parallel translation datasets. As mentioned before, they are TED2020, Tatoeba and Opus100. These datasets are widely used in research on low resource languages and include the English to Turkish language pair. Each dataset is presented in two separate files. One in English and one in Turkish with ".en" and ".tr" extensions. Thanks to these parallel structures, there is a Turkish translation corresponding to each English sentence. This feature provides data that can be used directly for supervised learning processes.

The text files used in the project were read as DataFrame structures using the Python's pandas library. This structure made easy the processing, filtering and visualization of data in an orderly and organized manner. As a result, the content which was reorganized to be suitable for model training and being processing quickly.

The data was cleaned in the first stage. During this process, empty rows, entries containing incorrect characters, and sentences that were too short or too long were removed. Sentences between 3 and 100 words in length were used. This limitation was imposed to prevent the model from breaking down in sentences that were too short or too long and to keep the training process balanced. In addition, due to the imbalance in the data distribution in some data sets, English-Turkish translation pairs were randomly sampled.

Tokenization was performed for each model according to the tokenizer structure. For the mT5-small model, explicit commands such as “translate English to Turkish:” were added to the input texts. In the NLLB-200-distilled and mBART-50 models, the input and target language codes were determined using the forced_bos_token_id parameter. This step is critical for the models to correctly predict the target language. The tokenizer classes of the Hugging Face Transformers library were configured according to the structure of each model. At this point, google/mt5-small, facebook/nllb-200-distilled-600M and facebook/mbartlarge-50-many-to-many-mmt tokenizers were used respectively.

The tokenized data has been saved in the datasets. DatasetDict format and organized so that it can be reused in training processes and called again with the load_from_disk() function. Each model has been matched with only a specific dataset, and no mixing between datasets has been performed. This choice purposes to establish a direct learning relationship between the model and the dataset and ensure that the model adapts only to its own data context.

In addition, the necessary data formats for LoRA-based fine-tuning have been created. These formats are configured to update only a small portion of the model parameters. Following all these processes, the data sets were organized using a standard folder structure, and a consistent file structure was prepared for training each model. This structure was designed to ensure the reproducibility of experiments and provide easy access to outputs in subsequent evaluation stages.

## 5.3 Finetuned and Meta Learner Model Training Process and Outputs

In this study, model training was carried out at two separate levels: fine-tuning Transformer-based translation models individually and training a meta classification model that learns from the outputs of these models. Model and algorithm selections were made carefully at both levels, and approaches that would be effective in low-resource scenarios were preferred.

In the first stage, three different Transformer models pre-trained on large multilingual data were selected: mT5-small, mBART-50 and NLLB-200-distilled. These models were chosen because of their multilingual support and their ability to operate with low computational resources. For example, mT5-small provides sufficient translation performance with 300 million parameters while offering low hardware requirements. The model mBART-50 offers broad language coverage with its multilingual encoder-decoder architecture.On the other hand, the NLLB-200-distilled model is a version optimized by Meta for low-resource languages and includes nearly 600 million parameters like mBART. Comparative evaluation of these models has also enabled inter-architecture performance analysis.

Each model was trained on three separate datasets.Training was limited to a single epoch in order to measure the transfer learning effect and reduce costs. The learning rate was 2e-4, the batch size was 8 and AdamW was used as the optimizer. The training process was carried out using Hugging Face's Seq2SeqTrainer class. In addition to these, Low Rank Adaptation (LoRA) layer was added to all models. This structure efficiently uses memory by updating only the low-rank subsets of model parameters and provides effective adaptation with small data. This method, implemented with the get_peft_model and LoraConfig classes, offers a lighter solution than classic full fine-tuning.

Model inputs were adapted according to the tokenizer structure. For mT5-small, input sentences are directed with explicit statements such as “translate English to Turkish:”. In the mBART-50 and NLLB-200 models, the target language direction is specified using the

forced_bos_token_id parameter. Each model output is stored in a standard index structure, enabling model-based analysis in subsequent evaluation steps.

The meta learner model developed in the second stage learns from the scores obtained from the outputs of individual models. In this stage, model outputs were evaluated using BLEU, CHRF, METEOR, TER, and GPT-based scores without the need for reference sentences; these scores were presented to the model as input along with SBERT embeddings.
In particular, the semantic vectors created with the SBERT (paraphrase-multilingualMiniLM-L12-v2) model ensured that the semantic context at the sentence level was transferred to the model.

The labels were derived from best_model_labels determined based on the prediction performance of the models; however, these labels were simplified to reduce model diversity and ensure class balance (e.g. “nllb-200M_opus100” → “nllb”). This generalization has enabled a more balanced classification by reducing the number of classes and has had a direct positive impact on performance.

Different algorithms were tried for meta model training. Initially, Random Forest and XGBoost were used. The following algorithms by experiments were LightGBM, CatBoost, Support Vector Machines (SVM) and deep learning based neural networks. This multi approach provided the opportunity to leverage the strengths of different classifiers. Here, the best performance was achieved with an ensemble strategy using the soft voting method. The performance of each model was measured separately and the most successful combination was determined.

The algorithm selections were made by some criteria like classification power, learning time and interpretability. For example, CatBoost stood out with its ability to process categorical variables directly, while the neural network model offered the ability to learn effectively from multidimensional vectors. Overall, this process resulted in a flexible and powerful classification system in which model outputs were evaluated based on data.

Finally, the different training strategies were tested while creating the meta learner model. The results obtained with the basic data and parameters used in the first stage were not satisfactory. Therefore, different data filtering strategies, class generalization approaches, and more complex classifiers were tried to improve the model's learning performance. Additionally, incorporating semantic features such as SBERT and GPT-based quality scores into the model resulted in noticeable improvements in classification accuracy. The effects of all these variations and their comparative results will be discussed in detail in the next section titled Evaluation.

## 5.4 Production and Visualization of Evaluation Results

The end of the model training, evaluation of the outputs obtained by calculating accuracy rates and presenting these results graphically. The performance of each model was analyzed using confusion matrices, accuracy scores and classification reports.

During the evaluation process, metrics were calculated using the classification_report and accuracy_score functions. They were prepared based on predictions made on the test data. Additionally, confusion matrices were got by using the libraries like seaborn and matplotlib. These visuals show the level of accuracy of the model's predictions in each class.

In addition, ensemble methods were compared with individual models. In these comparisons, the individual performances of models such as CatBoost, LightGBM, SVM and

Neural Network were analyzed side by side with the results of a manual ensemble model created using the soft voting approach. Graphical representations showed that the ensemble structure increased the overall accuracy rate.

In conclusion, the model outputs were analyzed in detail both textually and graphically, and the comparative successes of different algorithms were visualized and reported.

# 6 Evaluation

## 6.1 Experiment 1 - First Experiment

In this first experimental study, 100 examples were taken from each data set and the training process was started with a total of 300 parallel data. These data consist of source and reference segments belonging to English and Turkish language pairs and were used as direct input-output examples in the training of the models.

In this experiment, meta-learner models developed based on the outputs of fine-tuned translation models were tested. The aim here was to design a classifier that could predict which translation model would perform better on a given example. In this context, Random Forest and XGBoost algorithms were trained separately as meta-learners. Each was trained on scores and textual features associated with translations produced by previously trained models, and the best translation model was used as the label.

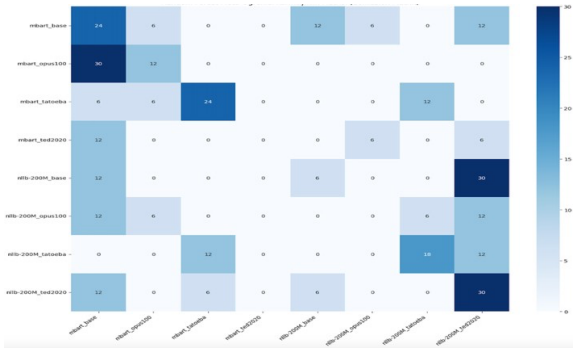
The evaluation phase was completed by getting accuracy, precision, recall, f1-score and confusion matrix visualizations.They were used to assess the prediction performance of both models. The following results were obtained:

- The XGBoost model achieved 20% accuracy. This model showed low F1 and recall scores for some labels in particular. The overall average F1 score was 18%. Some labels were found to be unclassifiable.
- The Random Forest model performed better with 32% accuracy. The average F1 score was 27%, with particularly successful results for labels such as “mbart_base” and “nllb-200M_ted2020.”

The confusion matrices of the two models are shown in Figure 2 and Figure 3. In particular, it has been observed that some models are often confused with each other, such as mbart_base and nllb_200M_base, where there is significant uncertainty.

The first reason for the limited performance of the models is the small size of the dataset used. Additionally, models like mT5-small could not be correctly classified by the metalearner due to their low performance. This situation created a significant disadvantage in terms of data representation and reduced the overall success of the meta-learner.

In conclusion, this experiment demonstrated the necessity of increasing the data size and the importance of making model labels more representative. With this motivation, the data volume was increased in the next experiment.



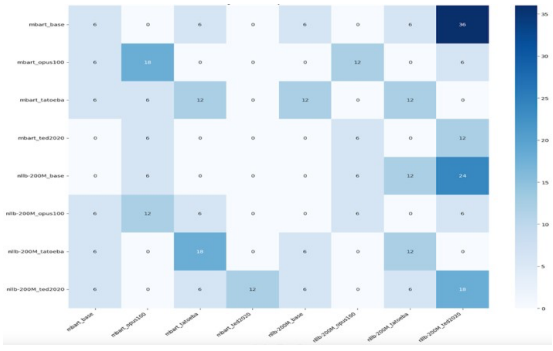


**Figure 2.** Random Forest Confusion Matrix

**Figure 3.** XGBoost Confusion Matrix

| | | | | |
|---|---|---|---|---|
| accuracy | | | 0.32 | 354 |
| macro avg | 0.28 | 0.29 | 0.27 | 354 |
| weighted avg | 0.30 | 0.32 | 0.30 | 354 |

**Figure 4.** Random Forest Scores

| | precision | recall | f1-score | support |
|---|---|---|---|---|
| mbart_base | 0.17 | 0.10 | 0.12 | 60 |
| mbart_opus100 | 0.38 | 0.43 | 0.40 | 42 |
| mbart_tatoeba | 0.25 | 0.25 | 0.25 | 48 |
| mbart_ted2020 | 0.00 | 0.00 | 0.00 | 24 |
| nllb-200M_base | 0.00 | 0.00 | 0.00 | 48 |
| nllb-200M_opus100 | 0.20 | 0.17 | 0.18 | 36 |
| nllb-200M_tatoeba | 0.25 | 0.29 | 0.27 | 42 |
| nllb-200M_ted2020 | 0.18 | 0.33 | 0.23 | 54 |
| accuracy | | | 0.20 | 354 |
| macro avg | 0.18 | 0.20 | 0.18 | 354 |
| weighted avg | 0.18 | 0.20 | 0.19 | 354 |

**Figure 5.** XGBoost Scores

## 6.2 Experiment 2 - Performance Evaluation on Large Data Sets Using Optimized Meta-Learner Models

In the second experiment phase, a larger data set was used compared to the first experiment. A total of 3,000 examples were collected from each dataset to create a dataset of 3000 examples. These parallel language data were prepared in English and Turkish in source and reference formats. Meta-learner classification was performed on the predictions generated by fine-tuned models.

In this experiment, Random Forest, XGBoost and a combination of these two, Ensemble VotingClassifier, were used as meta-learner models. Hyperparameter optimization was performed for each model and 5-fold cross-validation was applied. Similarly to previous experiment score calculations were made according to the best_model_label tag and no generalized labeling strategy was applied in this experiment too.

At that point the Random Forest model's the best performance with an accuracy rate of measured 41.85%.On the other hand, XGBoost model ranked second with an accuracy rate of 37.23%. Finally, the Ensemble VotingClassifier approach demonstrated a performance similar to XGBoost but more balanced with an accuracy rate of 37.73%.

Here, confusion matrices clearly show which classes the models confuse. For example, some models tend to make a mistake about correct variants selection.Such as instead of choosing mbart_tatoeba, frequently nllb_tatoeba were selected. This may be due to fine-tuned models exhibiting very similar performance and the meta model struggling with differentiation.

This experiment demonstrated that increasing the model volume and sample size improves accuracy. However, the high overlap between classes generally indicates a need for fine-tuned models to be trained more balanced or for the meta learner to be fed with more advanced features.

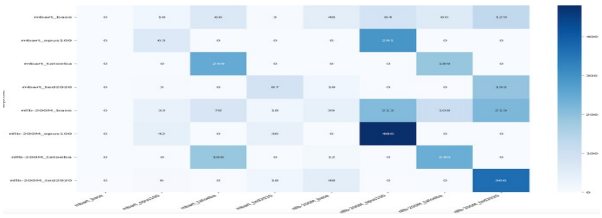


**Figure 6.** Random Forest Confusion Matrix

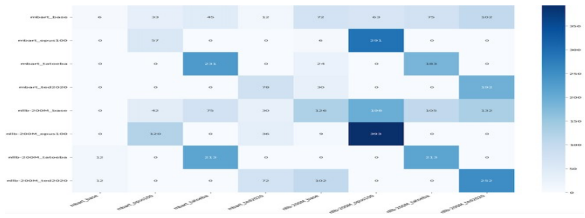


**Figure 7.** XGBoost Confusion Matrix

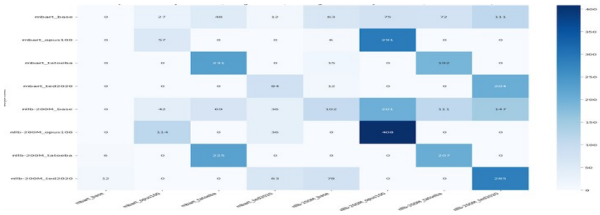


**Figure 8.** Voting Classifier (Ensemble) Confusion Matrix

## 6.3 Experiment 3 - General Labeled Meta-Learner Training and Evaluation

In the third experiment phase, a more generalized labeling approach was adopted instead of the fine-tuned model names used in previous experiments. In this context, models such as mT5-small, mBART and NLLB-200 were reduced to three main categories and named mt5, mbart and nllb, respectively. This strategy was preferred because it was observed that model name specific variations created uncertainty in the classifiers in previous experiments. General labeling aims to enable the model to evaluate examples from different data sources but belonging to the same model family in a more holistic manner.

In terms of data representation, SBERT embedding values were used for each example, and numerical features such as character and word length were also included in the model. In particular, SHAP analyses revealed that the source_length_words and source_length_chars features have a high impact on classification decisions. This finding indicates that the structural length of examples is decisive for classification performance. Additionally, some embedding dimensions contributed moderately and supported model decisions.

When examining the confusion matrices of the models, it was observed that the mt5 category was generally difficult for all models to recognize. The classification success between mbart and nllb varied depending on the model. In all models, the mt5 class yielded the weakest results with precision and recall values close to zero.

| Model | Accuracy | mbart F1 | mt5 F1 | nllb F1 |
|---|---|---|---|---|
| RandomForest | 0.604 | 0.25 | 0.0 | 0.73 |
| XGBoost | 0.5923 | 0.4 | 0.0 | 0.69 |
| LightGBM | 0.6 | 0.42 | 0.0 | 0.71 |
| MLP | 0.5491 | 0.44 | 0.0 | 0.63 |
| Ensemble | 0.6073 | 0.42 | 0.0 | 0.71 |

**Figure 9.** Experiment 3 - Ensemble Scores

This experiment proved that data increasing and label clustering increased model performance by approximately 20% compared to previous experiments. However, the failure of all models for the mt5 class indicates that this model cannot be distinguished from others based on SBERT embeddings. This results show that the mt5 model may not have been represented sufficiently during the fine tuning process.

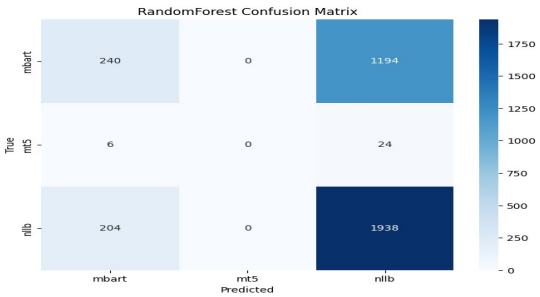



**Figure 10.** E3-RF Confusion Matrix

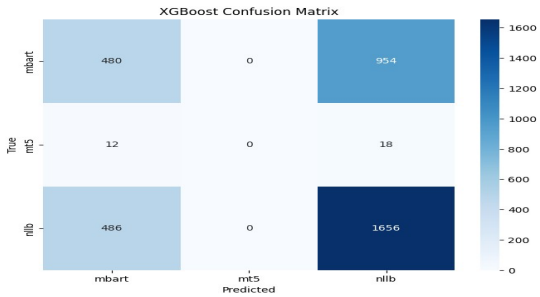



**Figure 11.** E3-XGBoost Confusion Matrix

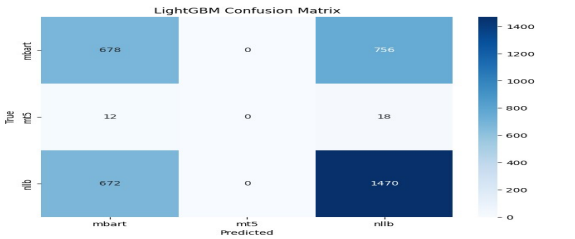



**Figure 12.** E3-LightGBM Confusion Matrix

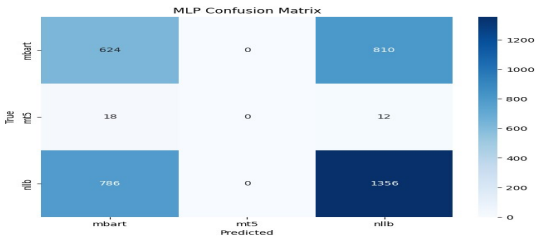



**Figure 13.** E3-MLP Confusion Matrix

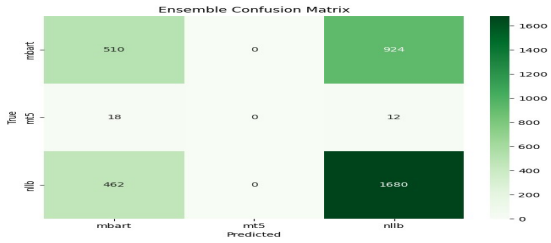



**Figure 14.** E3-Ensemble Confusion Matrix

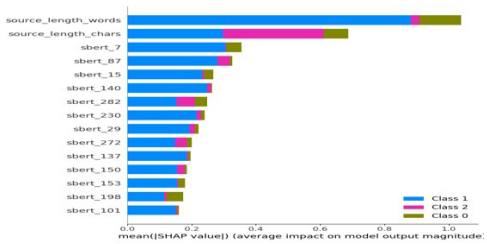



**Figure 15.** SHAP - Feature Importance

## 6.4 Experiment 4 - Multi-Label Classification Experiment

In this stage, unlike previous single classification approaches, a multi-label classification method was adopted. In this method, it was assumed that each example could be assigned more than one correct label. “Chat GPT scores” were used for model labeling, and models with scores of 3 or higher or the highest score were labeled under a new best_model_list column. Thus, a label structure was created so that each row would have the correct output of multiple models.

During the modeling process, the MultiOutputClassifier class was used to enable each model to produce separate predictions for all labels. The classification algorithms like Random Forest, XGBoost, LightGBM, Logistic Regression, Extra Trees, Gradient Boosting were trained separately.

The evaluation metrics obtained were performed using F1-score (micro and macro) and Jaccard scores. While the micro F1-score reflects the total success rate of all examples, the macro F1-score provides a more fair overall measurement by considering the balance between classes. The Jaccard score evaluates the intersection and union of predicted and actual labels.

When we compare the models performance, the LightGBM model placed at the top with the highest micro F1 score with 0.7193. This model also ranked first in terms of Jaccard scores and was evaluated as the best model in terms of overall performance. On the other hand, with the 0.4374 F1 macro score the Logistic Regression model showed a relatively high performance It is the demonstration of remarkable performance in terms of maintaining class balance. This experiment showed that the multi-labeling strategy can improve classification accuracy and that co-labeling models with high conversion quality can be beneficial.

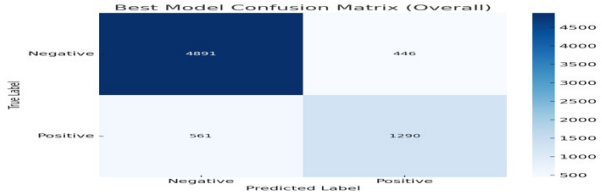



**Figure 16.** E4 - Best Model Confusion Matrix

This distribution shows that the model trained with LightGBM was able to successfully distinguish the negative class, while errors were higher in the positive class. However, remarkably success was also observed in the class of positive. When we look at the overall picture, it shows that the model was able to develop a balanced learning structure.

This experiment demonstrates that the multi labeled classification approach can provide more accurate predictions. Especially in datasets with complex label structures, compared to single-labeled models. Also, the LightGBM has proven the applicability of this approach with its high micro F1-score and successful classification rates.

| Model | F1-Score (Micro) | F1-Score (Macro) | Jaccard Score (Micro) |
|---|---|---|---|
| RandomForest | 0.6968 | 0.3638 | 0.5347 |
| XGBoost | 0.7166 | 0.4178 | 0.5584 |
| LightGBM | 0.7193 | 0.4142 | 0.5616 |
| LogisticRegression | 0.6902 | 0.4374 | 0.5269 |
| ExtraTrees | 0.6743 | 0.3244 | 0.5087 |
| GradientBoosting | 0.7119 | 0.4608 | 0.5526 |

**Figure 17.** E4 - Model Score Table

## 6.5 Discussion

The completed four different experiments in this study exposed the strengths and weaknesses of the developed approach. In the first experiment, the low number of data points severely limited the classification performance of meta learner models. Models such as Random Forest and XGBoost struggled to distinguish between classes and low performing models in particular could not be classified correctly. This situation demonstrated the direct effect of insufficient data.

In the second experiment, the amount of data was increased and a little bit improvements in the performance of the model were observed. However, the problem of mixing similar model variants such as mbart_tatoeba and nllb_tatoeba persisted. This revealed that when fine-tuned models produce similar outputs, the meta-learner struggles to distinguish between them.

In the third experiment, to reduce this problem, model labels were reduced to more general clusters . Additionally, SBERT based semantic features were added too. This strategy gave more balanced results and contributed to reducing confusion between models. Nevertheless, the mt5 class could not be classified in most models with low F1 scores. This indicates that the mt5 model is not sufficiently represented and is difficult to distinguish from others.

In the final experiment, as a main change multi-labeling approach was used. In helped to each examples ability to be assigned to more than one correct model. This method better reflected the uncertainty in model selection and increased accuracy rates. The LightGBM model showed the best performance and it became successful about predictions in the negative class. As a result, the multi labeling approach provided a more realistic and flexible solution.

When all experiments were evaluated together, it was easily understood that data volume, label structure and feature engineering were significant factors in model success. In future studies, it may be recommended to use more diverse data sources, support samples with quality scores and work with more advanced meta learner structures. However, the use of GPT models in the evaluation process and the processing of large datasets have caused significant computational costs. In particular, the process of obtaining GPT based interpretations and creating SBERT embeddings required high GPU power, which limited the scope of some

experiments. In such projects, access to powerful hardware and cost management emerge as external factors that directly affect success.

This study proposes a meta learning based system that can select the most appropriate sentence based translation model in low resource language pairs. This approach, which has been addressed in a limited number of studies in the literature, has been extended to a broader scope through the joint evaluation of datasets from different domains (TED Talks, general translation datasets, etc.) and the integration of models fine tuned with LoRA with general purpose multilingual models. The variety of the data sets used has increased the ability of the system to adapt different content types. In this respect, the study has made a important contribution both methodologically and at the application level.

In response to the research question , the developed meta learner based model selection mechanism has provided a stable increase in quality beyond individual models. Supported by BLEU, CHRF, TER, METEOR scores and reference free LLM based evaluation scores, the system achieved an accuracy rate of up to 71% even under conditions of limited computational resource and data. This result demonstrates that sentence based model selection can significantly improve translation quality in low resource languages and may serve as a viable strategy for future research.

# 7 Conclusion and New Research Direction

This study proposes a meta-learning-based model selection system that combines general-purpose multilingual models with domain-specific datasets fine-tuned using the LoRA method to improve translation quality in low-resource languages. The use of different domain datasets together has increased the system's ability to adapt to content diversity.

The meta-learner, supported by BLEU, CHRF, TER, METEOR and reference-free LLMbased scores, achieved up to 71% accuracy using ensemble methods such as Random Forest, XGBoost, and LightGBM. The findings and results demonstrate that sentence level model selection significantly may help the choosing of best candidate translation model candidates in low resource languages.

Future researches could focus on testing the system on larger datasets, broader language pairs and comparing different meta learning. In addition to them, different labeling strategies and diversifying reference free evaluation methods could be the another search areas. Lastly, optimizing the balance between performance and latency through integration with real time translation scenarios could be a interesting and valuable area for new researches.

# References


Liu, Y., Gu, J., Goyal, N., Li, X., Edunov, S., Ghazvininejad, M., Lewis, M. and Zettlemoyer, L. (2020) 'Multilingual Denoising Pre-training for Neural Machine Translation', arXiv preprint arXiv:2001.08210

Xue, L., Constant, N., Roberts, A., Kale, M., Al-Rfou, R., Siddhant, A., Barua, A. and Raffel, C. (2021) 'mT5: A massively multilingual pre-trained text-to-text transformer', *arXiv preprint*. Available at: https://arxiv.org/abs/2010.11934

NLLB Team (2022) 'No Language Left Behind: Scaling Human-Centered Machine Translation', *arXiv preprint*. Available at: https://arxiv.org/abs/2207.04672

Kang, J., Tak, K., Choi, J., Kim, M. and Jang, J. (2025) 'DaCoM: Strategies to Construct Domain-specific Low-resource Language Machine Translation Dataset', *COLING 2025*. Available at: https://aclanthology.org/2025.coling-industry.53

Xu, L., Xie, H., Qin, S.-Z.J., Tao, X. and Wang, F.L. (2023) 'Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment', arXiv preprint. Available at: https://arxiv.org/abs/2312.12148

Hu, E.J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, L. and Chen, W. (2021) 'LoRA: Low-Rank Adaptation of Large Language Models', *arXiv preprint*. Available at: https://arxiv.org/abs/2106.09685

Pfeiffer, J., Rücklé, A., Vulić, I., Gurevych, I. and Ruder, S. (2021) 'AdapterFusion: Non-Destructive Task Composition for Transfer Learning', *arXiv preprint*. Available at: https://arxiv.org/abs/2005.00247

Ben Zaken, E., Ravfogel, S. and Goldberg, Y. (2022) 'BitFit: Simple Parameter-efficient Fine-tuning for Transformer-based Masked Language-models', *arXiv preprint*. Available at: https://arxiv.org/abs/2106.10199

Papineni, K., Roukos, S., Ward, T. and Zhu, W. (2002) 'BLEU: a Method for Automatic Evaluation of Machine Translation', *ACL*, pp. 311–318.

Snover, M., Dorr, B., Schwartz, R., Micciulla, L. and Makhoul, J. (2006) 'A Study of Translation Edit Rate with Targeted Human Annotation', *AMTA*.

Zhang, T., Kishore, V., Wu, F., Weinberger, K.Q. and Artzi, Y. (2020) 'BERTScore: Evaluating Text Generation with BERT', *ICLR*. Available at: https://arxiv.org/abs/1904.09675

Zaidan, O.F. and Callison-Burch, C. (2011) 'Crowdsourcing Translation: Professional Quality from NonProfessionals', *NAACL-HLT*.

Kocmi, T. and Federmann, C., 2023. Large Language Models are State-of-the-Art Evaluators of Translation Quality. arXiv preprint arXiv:2302.14520.

Liu, P., Yu, W., Zhang, Y., Qian, H., Huang, M. and Neubig, G. (2023) 'G-Eval: NLG Evaluation using GPT-4 with Better Human Alignment', *arXiv preprint*. Available at: https://arxiv.org/abs/2303.16634

Gu, J., Wang, S. and Zhao, J. (2018) 'Meta-Learning for Low-Resource Neural Machine Translation', *arXiv preprint*. Available at: https://arxiv.org/abs/1808.08437

Finn, C., Abbeel, P. and Levine, S. (2017) 'Model-Agnostic Meta-Learning for Fast Adaptation of Deep Networks', *ICML*. Available at: https://arxiv.org/abs/1703.03400

Liu, P. et al. (2023) 'LLMs Are Not Scorers: Exploring LLMs' Ability to Score NLG Outputs', *arXiv preprint*. Available at: https://arxiv.org/abs/2305.13285