A Comparative Study of Machine Translation Models: Accuracy, Error Patterns, and Optimization Strategies

Group 4: Nastasa Denisa- Elena, Timur Jerčaks, Hitesh Satwani, Cecilia Mazzola, Viktoria Ivanova, Wenjie Liao, Jiang Haoting

Faculty of Science and Engineering, Maastricht University

Project 2-2

31st May 2025

Contents

1 Abstract

This report presents a comparative study of neural machine translation models, focusing on a traditional Seq2Seq LSTM-based architecture and two Transformer models: MarianMT and M2M100. The dataset we are using is the WMT14 English-German dataset and standard metrics such as BLEU, METEOR, BERTScore, COMET and Word Error Rate are used to evaluate translation quality. To assess robustness, we introduced controlled variations of 20% and 40% noise into the input data. Results show that Transformer-based models outperform the LSTM baseline in both accuracy and resilience to noisy inputs. MarianMT demonstrates the highest performance while M2M100 shows strong multilingual generalization. Fine-tuning improves performance, however requires careful monitoring to avoid overfitting especially on smaller or uncleaned datasets. The findings display key trade-offs between modern specialization, resource efficiency and translation validity . The study provides insights for selecting and adapting machine translation models in various contexts.

2 Introduction

Since the evolution of digital technologies, machine translation (MT) has become essential for multilingual communication in fields such as healthcare, education, and policy (**Koehn2017**). With the recent rise in AI development, models such as Seq2Seq and Transformer-based architectures have significantly improved translation accuracy and quality. Seq2Seq models, especially those based on LSTM (Long Short-Term Memory), made it possible to train translation models end-to-end (**Sutskever2014**). However, they still struggle with longer sentences and noisy or unstructured data. Transformer-based models, such as MarianMT and M2M100, outperform LSTMs ... outperform LSTMs (**Vaswani2017**). in many aspects due to their use of self-attention mechanisms (**Bahdanau2014**) and parallel processing capabilities. The drawback of these models, however, is their heavy reliance on large, clean datasets and significant computational resources.

In this study, we analyze the performance of LSTM-based and Transformer-based models using the WMT14 English-German dataset. We evaluate each model using standard metrics such as BLEU, METEOR, BERTScore, and COMET. Furthermore, we examine how each model handles noisy or misspelled inputs, and how data cleaning affects the performance of the LSTM model when dealing with imperfect input. Overall, this study aims to identify which models perform best in which circumstance and provide insights into their deployment in real-world machine translation applications. **While Seq2Seq and Transformer models have advanced machine translation, they still face important challenges in accuracy, robustness to noisy input, and adaptability across diverse real-world conditions. Addressing these limitations requires a systematic comparison of both architectures under controlled experimental settings.** To guide this investigation we formulated the following research questions:

- How do different Transformer architectures affect machine translation accuracy, and how adaptable are they to fine tuning on domain-specific data?
- How do Transformer-based models perform when they are exposed to noisy or perturbed input in machine translation tasks?
- How do variations in hyperparameter configurations impact the translation quality of Transformer-based machine translation models?
- How does applying data cleaning impact the performance of LSTM-based Seq2Seq Models in terms of translation accuracy?

3 Methods

3.1 Dataset

For this study we decided to use the WMT14 English-German dataset. This dataset contains approximately 4.5 million parallel sentence pairs derived primarily from news articles and commentary. To have more manageable experimentation, we split the dataset into three parts: training subset, which consists of the first 50,000 sentences, and test and validation, each containing 3,000 sentences.

For better understanding and manipulation of the data, we applied the LDA (Latent Dirichlet Allocation) clustering algorithm (`LdaModel`) to each subset. That showed six main topics within the dataset: *EU Commission Actions*, *EU Structure*, *Economics*, *Formal Debates*, *Global and Social Affairs*, and *Member States*. This analysis proved useful in identifying that the dataset primarily focuses on political and European Union-related content. As we can see in Figure 1, the topics are evenly distributed, with almost identical proportions between the training and test dataset,the same applies to the validation set. Later on these findings proved essential for guiding fine-tuning strategies, allowing the models to more effectively adapt to a domain-specific vocabulary.

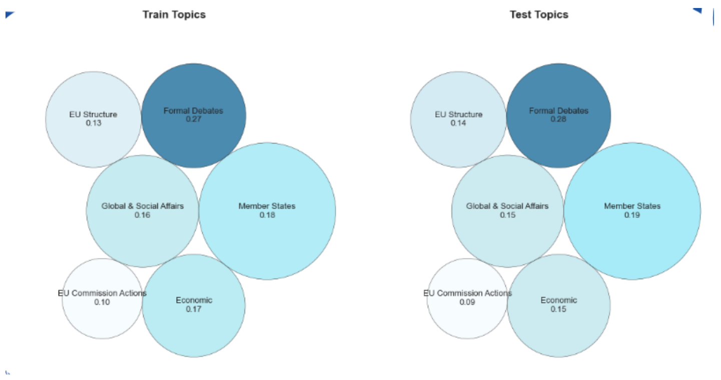


Figure 1: Train and Test splitting

3.2 Preprocessing

After having selected the dataset, the next step was preprocessing. For the Transformer models, MarianMT and M2M100, no cleaning was applied, as they are pre-trained. In contrast, the preprocessing approach was important when conducting our research for the LSTM-based model. Our main objective was investigating the impact of data cleaning on its performance.

To achieve this, the LSTM model was trained and evaluated in two different conditions, one using clean data and another using uncleaned, raw data. In the first case, preprocessing consisted of converting all text to lowercase, while preserving named entities, removing commas from numbers, but keeping them in case of decimals. Unnecessary punctuation was removed, but kept in cases such as possessive forms ("John's") and abbreviations ("U.S.A"). Additionally, control characters and duplicate sentences were removed, and Unicode normalization was applied to ensure uniform text representation. Following that, Byte-Pair Encoding was used to compress the sentences. In the second case, the dataset was used in its original form. This setup allowed us to measure how sensitive the LSTM architecture is to noise in raw, unprocessed texts.

3.3 Models

For our analysis we selected three neural machine translation models: a traditional Seq2Seq LSTM-based architecture and two Transformer models, MarianMT and M2M100. Each architecture offers unique advantages and limitations, motivating our comparative analysis of their performance.

The LSTM-based Seq2Seq model has a bidirectional LSTM encoder which captures context from past and future tokens. In addition to that, it has a unidirectional decoder. This architecture is computationally lighter and performs better in low resource scenarios.

MarianMT models are typically based on the standard Transformer architecture (Vaswani et al., 2017), featuring an encoder-decoder structure with self-attention and feed-forward layers. They are often optimized for efficient training and inference. Many available MarianMT models are **bilingual**, specifically trained for a single language pair. MarianMT was chosen due to its widespread adoption and proven strong performance on numerous translation benchmarks, particularly for specific language pairs like English-German. Its availability through platforms like Hugging Face facilitates ease of use and reproducibility. The focus on bilingual models allows for a deep dive into the capabilities of a system highly specialized for the target task.

M2M100 is also a Transformer-based sequence-to-sequence model but is distinct in its **multilingual** capabilities. It is pre-trained to translate directly between any pair of 100 languages without relying on English as an intermediate pivot language (except for zero-shot translations not explicitly in training data). This is achieved through a shared encoder and decoder across all languages, often with language-specific tokens or embeddings to guide translation. Its ability to handle numerous language pairs, including low-resource ones, provides a contrast to the bilingual MarianMT. We aimed to assess how well a single, massive multilingual model performs on a specific high-resource pair (English-German) compared to a specialized model.

This selection allows for a focused evaluation of both specialized bilingual and broad-coverage multilingual Transformer models' capabilities on a substantial, commonly used dataset.

3.4 Training and Fine-Tuning

The LSTM model was trained entirely from scratch. For that purpose, in one scenario, we used the raw, uncleaned data, while in another we employed the preprocessed and BPE-encoded data.

For the MarianMT and M2M100, fine-tuning was used to enhance their adaptability to the WMT14 dataset. Unlike the LSTM, these models were pre-trained so we did not preprocess the data beforehand. During the fine-tuning process we monitored several evaluation metrics, including: BLEU, METEOR and BERTScore, allowing us to detect potential overfitting early on.

We used the information from the LDA clustering when fine-tuning the models. Domain-specific fine-tuning helped enhance the accuracy, fluency and overall translation quality of both Transformer models when processing contextually relevant content.

4 Implementation

4.1 Development Environment

The implementation of our models was conducted in Python 3.10 using a GPU-accelerated environment on Google Colab to ensure efficient training and evaluation. We structured our implementation across our 4 experiments and the code was modularized across separate notebooks for readability.

4.2 LSTM Seq2Seq Model

The LSTM-based Seq2Seq model was developed using PyTorch and was implemented with both clean and unclean data, the architecture consisted of a bidirectional LSTM encoder and unidirectional LSTM decoder with an integrated attention mechanism. Both the encoder and decoder used embedding layers with a fixed dimension of 300 and was initialized with pretrained GloVe embeddings.

Before training the model we cleaned the data by lower casing while preserving named entities, punctual normalization, character removal, duplicate filtering and sub word tokenization using Byte Pair Encoding (BPE). Sentences longer than 50 tokens were removed to ensure consistent input lengths and prevent extra padding.

4.3 Transformer Models Implementation

For the Transformer models, MarianMT and M2M100, we used the Hugging Face Transformers library. Fine-tuning was conducting using the Hugging Face API with a learning rate of 5e-5, batch size of 8 and with early stopping enabled. Alternatively to the LSTM model, the Transformer models were intentionally fine-tuned on uncleaned data to evaluate their robustness to noisy input.

4.4 Evaluation Pipeline

To evaluate translation quality and direct hyperparamter optimization, we used four metrics: BLEU, METEOR, BERTScore and COMET. Each metric helps provide a different perspective on translation performance.

- **BLEU**: Measures the n-gram overlap between a given reference and the generated translation, but often does not give enough value to semantically accurate rephrasings.
- **METEOR**: Uses stemming, synonym matching, and recall to handle variation in language
- **BERTScore**: Measures semantic similarity using embeddings from a pretrained BERT model, this allows it to provide a deeper assessment of whether the meaning of the sentence has been preserved.
- **COMET**: A model that has been trained on translation judgments passed by humans such as quality, balancing fluency and adequacy.

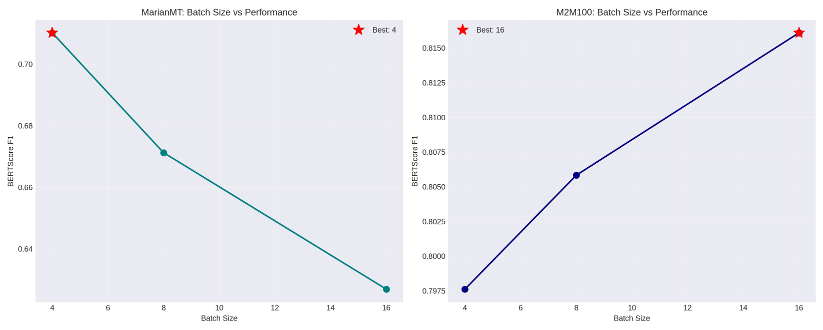


Figure 2: Batch Size Comparison

Although all four metrics were used during evaluation, **BERTScore F1 was selected as the primary metric for hyperparameter tuning**. The reason for this choice was the need to capture semantic adequacy and go a step beyond token overlap. This is an important consideration for modern Translation models like MarianMT and M2M100 which often create fluent by lexically diverse translations.

This decision is supported by the following observations:

- **Batch Size vs Performance (Figure 2):** BertScore F1 varied with batch size for MarianMT and M2M100. This sensitivity of the BERTScore to this hyperparameter shows that its an effective optimization target as it can spot real quality differences as the batch size changes.
- **Model Architecture Comparison (Figure 3):** BERTScore F1 was able to differentiate between model architectures of varying dept and size. This further confirms that BERTScore is semantically meaningful and can identify architechture-level differences.

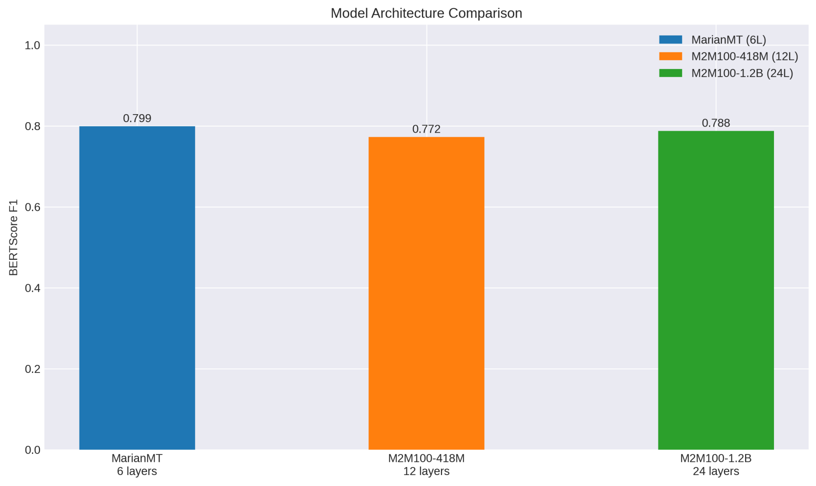


Figure 3: Model architecture comparison.

5 Experiments

5.1 The Most Accurate Evaluation Metric

After comparing the translations from the two models **MarianMT** and **M2M100** with different metrics and printing translation examples, we found that **BERT F1** is the most accurate evaluation metric. In general, we know that **BLEU** is based on n-grams, where if two translations use different words but with the same meaning it will have a lower value. While in the case of the **BERTScore** metric, the model transforms each word into a vector that represents its meaning in context. It can calculate the Precision (how much of the translated text is relevant to the reference), the Recall (how much of the reference was captured by the translation) and finally F1 that rewards accurate and complete translations.

Consider the following example:

- MarianMT: *Gutach: Mehr Sicherheit für Fußgänger*
- M2M100: *Gutach: Sicherheit für Fußgänger erhöht*

We noticed that for the **MarianMT** model we have a high precision but low recall, while for the **M2M100** model the sentence is more complete but less precise. This leads to high values for both the BERT Precision metric and the BERT Recall metric, but in our case **BERT F1** captures better the overall quality of the two models, which recognizes that both versions convey the same meaning even if with different words.

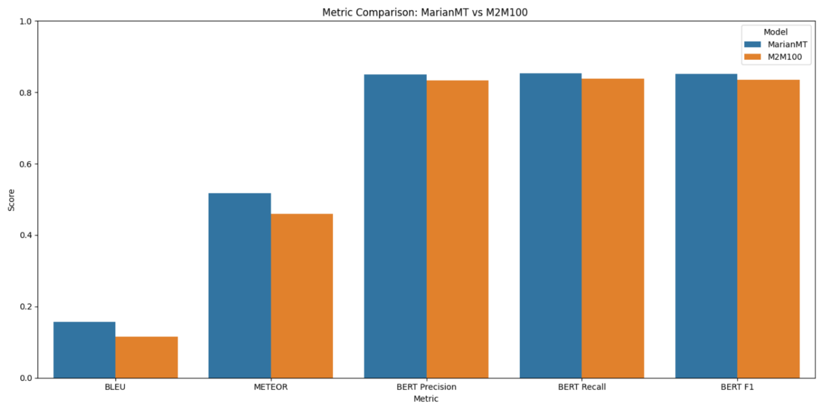


Figure 4: Metric Comparison

5.2 Baseline Transformers Performance

In this initial phase, we treated both pre-trained models as "black boxes." The models were loaded directly from the Hugging Face Hub and used for inference on the 3,000-pair test set without any prior training or fine-tuning. This step was designed to measure their out-of-the-box, zero-shot performance on our target domain.

5.3 Fine-Tunning

In our experiment, both the MarianMT and M2M100 models were fine-tuned under identical conditions to ensure **a fair comparison of their adaptability**. The key parameters were:

- **Training Data:** A 50,000 sentence-pair subset from the WMT14 en-de dataset.
- **Learning Rate:** A constant learning rate of 2×10^{-5}, as shown in the training graphs.
- **Epochs:** The training was conducted for 7 full epochs.

During this process, we continuously monitored training loss, validation loss, and a suite of performance metrics on a held-out validation set to observe the learning dynamics. The evolution of metrics during fine-tuning, presented in Figure 5 and Figure 6, reveals fundamentally different responses from the two models.

The **M2M100** model demonstrated a classic and highly successful adaptation pattern (Figure 5). Its validation loss followed a U-shaped curve, decreasing steadily to a minimum around the 4th epoch before starting to rise. This indicates that the model was effectively learning to generalize to new data. This optimal point is precisely mirrored by the performance metrics (BLEU, METEOR, BERTScore), which all peaked around the 4th epoch. This shows that the general-purpose model significantly benefited from specialization.

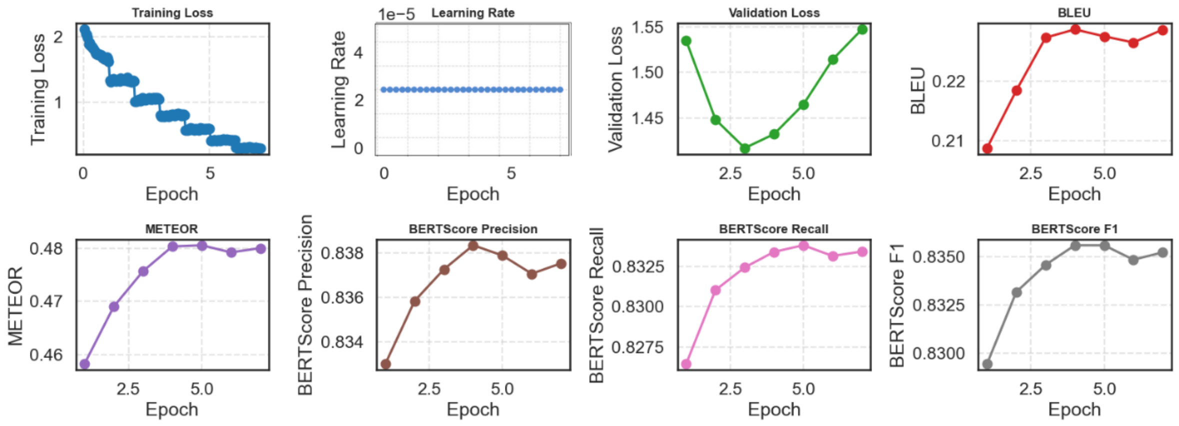


Figure 5: Training dynamics for the M2M100 model. The model shows successful adaptation, with performance metrics peaking around epoch 4, aligned with the minimum validation loss.

In stark contrast, the **MarianMT** model exhibited clear signs of immediate overfitting (Figure 6). While its training loss decreased consistently, its validation loss began to increase almost immediately. This negative trend was mirrored across all performance metrics, which peaked at the very beginning of the process (around epoch 2) and then steadily declined. This behavior suggests that the model was already so highly specialized for the language pair that further training on a moderately-sized dataset caused it to memorize the training data at the expense of its generalization capabilities.

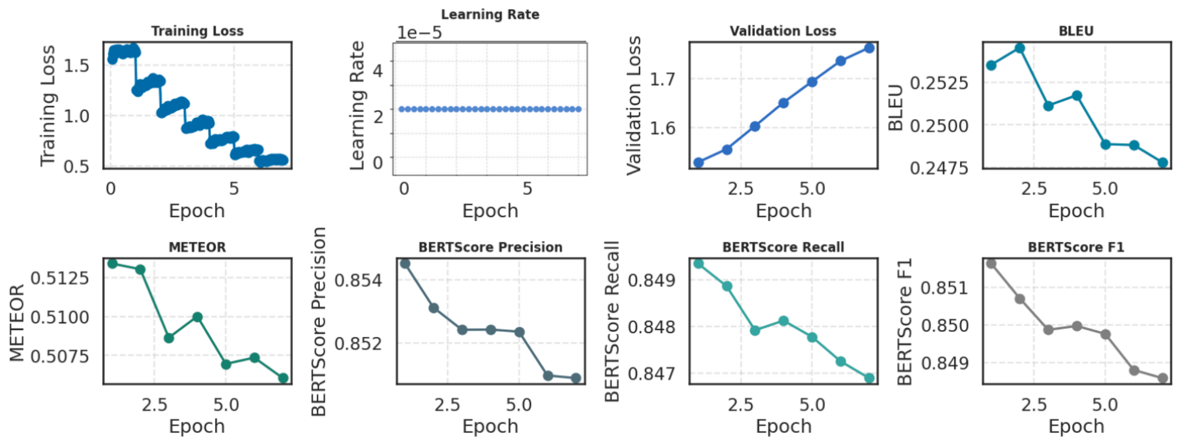


Figure 6: Training dynamics for the MarianMT model. The increasing validation loss and declining performance scores from an early stage are clear indicators of overfitting.

5.4 Noise Robustness Testing

To simulate realistic variability in natural language input, we had two methods. In the first one, we tried to introduce a **stochastic** noise model that perturbs a given text corpus, and secondly we experimented with **deterministic** sampling approach. The objective is to mimic natural errors such as typos, deletions, duplications, and semantic substitutions, which are common in generated real-world data.

For the **stochastic model**, let a sentence consist of n words: $w_1, w_2, \ldots, w_n$. We define a random process that independently applies noise to each word w_i with probability p. This process is modeled as a sequence of independent Bernoulli trials, where for each word:

$$X_i \sim \text{Bernoulli}(p), \quad i = 1, \ldots, n$$

Here, $X_i = 1$ indicates that noise is applied to word w_i, and $X_i = 0$ otherwise. The total number of perturbed words in a sentence is therefore a binomially distributed random variable:

$$X = \sum_{i=1}^{n} X_i \sim \text{Binomial}(n, p)$$

The expected number of words to which noise is applied is given by:

$$\mathbb{E}[X] = np$$

In contrast to the stochastic method described previously, we also implement a **deterministic sampling approach**, where a fixed fraction of the words in a sentence are guaranteed to be perturbed. Given a sentence of n words, we define a noise fraction parameter $f \in (0, 1]$, and compute the number of words to perturb as:

$$k = \max(1, \lfloor f \cdot n \rfloor)$$

This ensures that at least one word is always noised, even for short sentences. The k words to be perturbed are selected uniformly at random without replacement from the set of all word indices $\{1, 2, \ldots, n\}$. This results in exactly k words undergoing noise injection, in contrast to the binomial variability present in the previous method.

For both methods, we applied the same type of noise, a single noise type is chosen randomly out of these: deletion of the word, insertion of punctuation, replacement of a character, deletion of a character, insertion of an additional character, shuffling of characters within the word, or the introduction of typographic errors (i.e., substituting characters with nearby keys on a standard keyboard layout). The deterministic method ensures consistent coverage of noise, applying it to a fraction f of the words. Consequently, the number of perturbed words is always fixed, enabling controlled experimentation and better comparability across samples. Unlike the Bernoulli model, where perturbation is probabilistic per word, this approach produces a fixed sample size drawn from a uniform distribution over the word positions.

5.5 Effect of Data Cleaning on LSTM

To investigate the impact of data cleaning on the performance of the LSTM-based Seq2Seq model , we conducted a controlled experiment where the the only difference between the two runs was the state of the training data(clean and uncleaned). The goal of this experiment was to assess whether reprocessing improves translation accuracy and model stability when training a Seq2Seq model.

The LSTM architecture for both runs was as follows: a bidirectional LSTM encoder, a unilateral LSTM decoder and an attention mechanism. Embedding layers were initialized with pretrained GloVe vectors, and training was performed using the Adam optimizer (with a learning rate of 0.001) with teacher forcing (with a ratio of 0.5)

IN the first run the model was trained on the uncleaned version of the WMT14 English-German dataset. This included inconsistent capitalization, control characters, punctuation and duplicate or close to duplicate sentence pairs. In the second run the same model was trained on a cleaned version of the dataset, this cleaning included:

- Lower casing, while ensuring to preserve named entities
- Normalizing Punctuation
- Control character removal
- De duplication of sentence pairs.
- Byte Pair Encoding (BPE) tokenization
- Filtering out longer sentences (¿50 tokens)

Both models were trained on a subset of 50,000 pairs from the original dataset for 10 epochs on the same train/validation/test splits. After each epoch the model performance was evaluated using BLUE, METEOR, and BERTScore(Precision, Recall, and F1). We focused more specifically on the BERTScore F1 as it provides a more deeper measure of semantics. Furthermore, training and validation loss were also tracked to monitor the convergence dynamics.

5.6 Hyperparameter Sensitivity

To identify optimal hyperparameter configurations, we conducted extensive grid search experiments across six key parameters: beam size, temperature, top-p, max length, repetition penalty, and length penalty.

Our analysis revealed that beam size emerged as the most critical parameter. As shown in Figure 7, M2M100 performance plateaued at 0.827 when beam_size ≥ 3, while beam_size $= 1$ resulted in significantly lower scores (0.79-0.82 depending on other parameters).

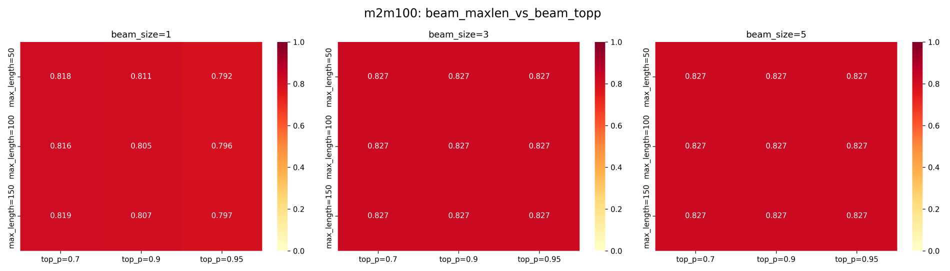


Figure 7: Impact of beam size on M2M100 performance across different top-p and max length combinations

Temperature also showed strong influence on performance. Figure 8 demonstrates that high temperature (1.3) severely degraded performance, with BERT F1 dropping to as low as 0.802 for MarianMT when beam_size $= 1$.

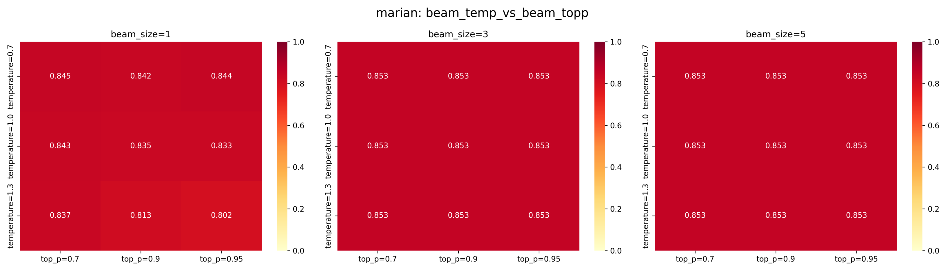


Figure 8: Effect of temperature on MarianMT performance with varying beam sizes

6 Results

6.1 Baseline Transformers Performance

To establish a performance baseline, we evaluated two pre-trained transformer models: MarianMT and M2M100. The evaluation was conducted on a randomly selected subset of 3,000 sentence pairs from the WMT14 dataset. We assessed the models' performance using several standard machine translation metrics: BLEU, METEOR, and BERT-based scores (Precision, Recall, and F1-Score) without any prior fine-tuning on the target domain.

The results are presented in Figure 9.

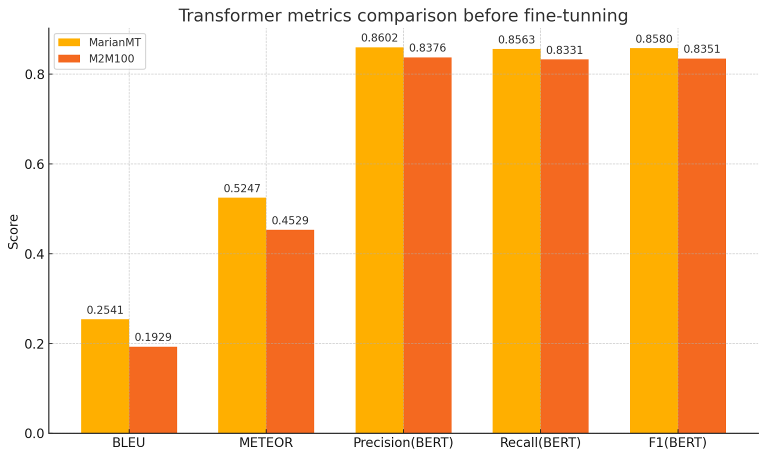


Figure 9: Comparison of baseline performance metrics for MarianMT and M2M100 models before fine-tuning.

The results clearly indicate that before any fine-tuning, the **MarianMT model consistently outperformed the M2M100 model** across all evaluated metrics. For instance, MarianMT achieved a BLEU score of 0.2541 compared to M2M100's 0.1929, and its BERT-based F1-Score was 0.8580 versus 0.8351 for M2M100.

This suggests that monolingual models like MarianMT, which are often specialized for a particular language pair, tend to offer higher initial accuracy on in-domain data. In contrast, massive multilingual models like M2M100, while versatile, may require fine-tuning to achieve competitive performance on a specific task.

6.2 Impact of Fine-Tuning

Following the baseline evaluation, we fine-tuned both the MarianMT and M2M100 models to adapt them to our specific domain. The fine-tuning process was conducted under **identical conditions** for both models to ensure a fair comparison:

- **Training Data:** A training set of 50,000 sentence pairs from the WMT14 dataset.
- **Training Duration:** The process was run for a fixed 7 epochs.

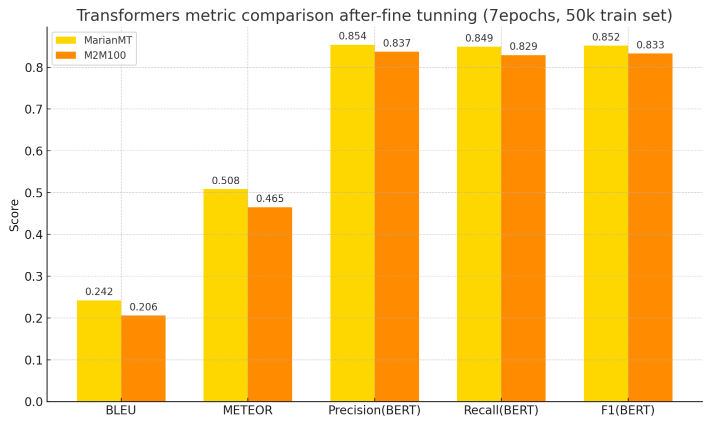


Figure 10: Performance metrics comparison after fine-tuning both models.

After fine-tuning, the models were re-evaluated on the same 3,000-sentence test set. Figure 10 illustrates the final performance, while Table 1 provides a comprehensive comparison of scores before and after the fine-tuning process.

Table 1: Comparison of Model Performance Before and After Fine-Tuning.

	MarianMT		**M2M100**	
Metric	**Baseline**	**Fine-Tuned**	**Baseline**	**Fine-Tuned**
BLEU	0.2541	0.2420	0.1929	0.2060
METEOR	0.5247	0.5080	0.4529	0.4650
Precision (BERT)	0.8602	0.8540	0.8376	0.8370
Recall (BERT)	0.8563	0.8490	0.8331	0.8290
F1-Score (BERT)	0.8580	0.8520	0.8351	0.8330

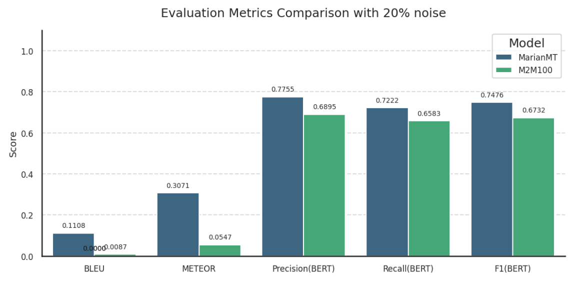
The analysis presented in Table 1 reveals a crucial distinction in how each model responded to fine-tuning.

The **MarianMT** model, which already demonstrated strong baseline performance, **showed limited gains and even a slight degradation** in its scores after fine-tuning. For example, its BLEU score decreased from 0.2541 to 0.2420, and its F1-Score dropped from 0.8580 to 0.8520. This indicates that for a model already highly specialized for a language pair, further training on a moderately-sized in-domain dataset may offer diminishing returns.

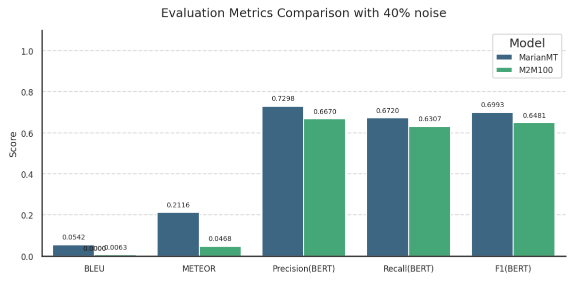
Conversely, the **M2M100** model, despite its lower absolute scores, showed **clear improvement in key translation metrics**. Its BLEU score increased from 0.1929 to 0.2060, and its METEOR score rose from 0.4529 to 0.4650. This demonstrates that general-purpose multilingual models possess a greater capacity to benefit from domain-specific fine-tuning, as it allows them to adapt their broad knowledge to a specialized task.

6.3 Noise Robustness

To evaluate the impact of noise on translation quality, we applied two noise levels, 20% and 40%, to the input data and compared the resulting model translations. These levels were chosen to simulate realistic noise commonly encountered in user-generated or noisy textual data. As illustrated in Figures 11 and 12, both noise injection methods yielded comparable results, with model performance remaining within a similar range across methods.



(a) 20% noise

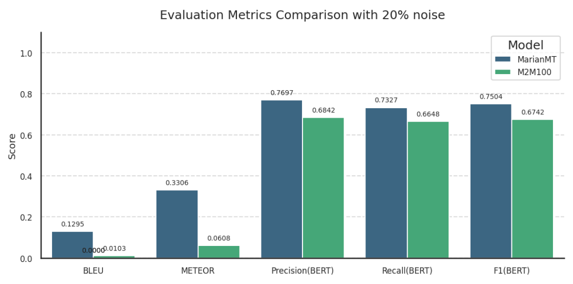


(b) 40% noise

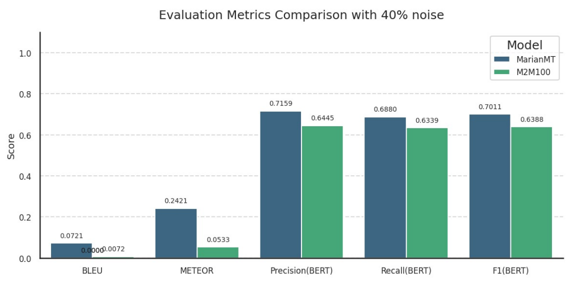
Figure 11: Comparison of noise levels using the probabilistic method

To better understand how noise influences translation quality, we visualized the performance of the models using line plots based on the BERT F1 score. This metric influences BERT's contextual embeddings to compare the semantic similarity between the model output and the reference translation. Unlike traditional string-based metrics, BERT F1 can better capture changes in meaning, making it particularly suitable for evaluating robustness to noise. As shown in the plots 14 and 13, the fixed-percentage noise method consistently resulted in slightly lower BERT F1 scores compared to the probability-based method. This decline is expected, as fixed-percentage noise guarantees that a larger portion of the sentence is perturbed, whereas the probabilistic method introduces noise more sparsely. These trends validate the assumption that increased or systematic noise degrades translation quality more significantly, and the use of BERT F1 provides a more nuanced view of this degradation.

Looking further into why the BERT F1 scores remain relatively high at 100% noise (0.60 for MarianMT and ∼0.50 for M2M100), we analyzed corrupted input examples. Despite severe character-level corruption, many keywords or fragments remained partially intact.

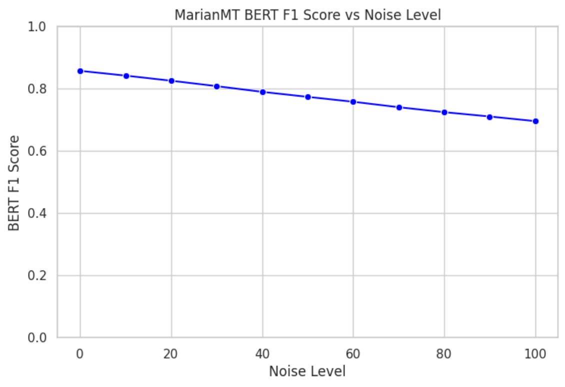


(a) 20%

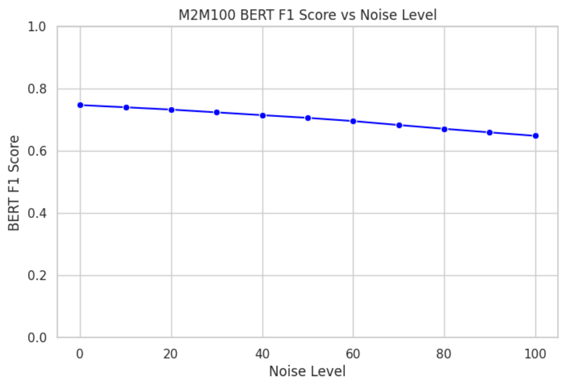


(b) 40%

Figure 12: Comparison of noise levels using the fixed-percentage method

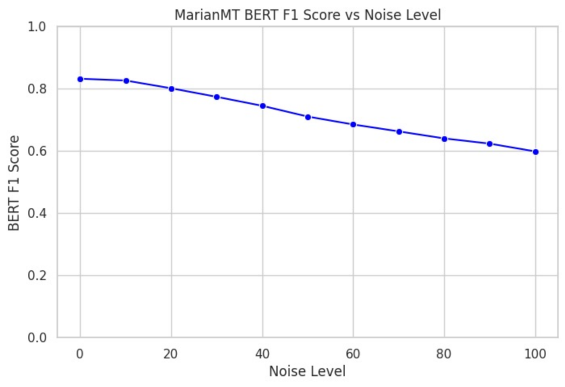


(a) BERT F1 score for MarianMT

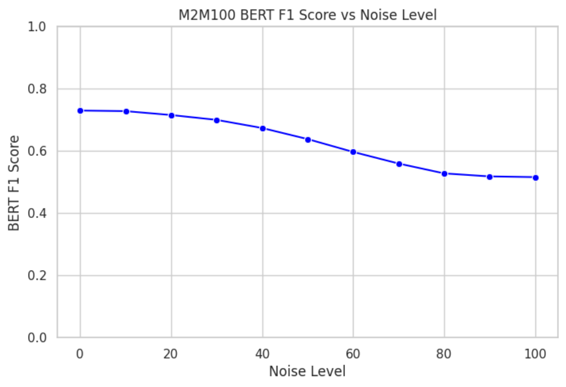


(b) BERT F1 score for M2M100

Figure 13: Performance of models under probabilistic noise injection



(a) BERT F1 score for MarianMT



(b) BERT F1 score for M2M100

Figure 14: Performance of models under fixed-percentage noise injection

For instance, the noised version of the sentence *"The French version contains the word 'délit', whereas this is written as 'crime' in the English version."* becomes: `"T!he Fr.ench ve,rsion containsc thea word 'd!élit' swheera !this asw crim ni rjr Eng!lish."` Although unreadable to humans, this version still retains enough semantic clues, such as partial words like `"Fr.ench"`, `"contains"`, and `"Eng!lish"`, for the model to infer the likely meaning.

For example, MarianMT produces the output: *"nxk F.rench vers; ion enthalten yvis, in der Erwägung, daß es sich hierbei um 'Verbrechen' handelt."* Here, the model successfully recovers part of the intended semantics, translating `"crime"` as *"Verbrechen"*, though it hallucinates a formal legal tone not present in the original. This suggests that the encoder-decoder attention mechanism is able to map corrupted fragments to latent representations that are still semantically plausible.

M2M100, by contrast, produces: *"nxk F.rench vers;ion enthält!s yvis statt; thi ist ritten 'Verbrechen'. wmfkudg"*. This output shows lower fluency and grammaticality but still includes the correct German word for `"crime"` (*"Verbrechen"*), demonstrating that key lexical items were recoverable even under noise.

Another illustrative example is the corrupted input for the sentence *"As you know, Madam President, this does not have the same meaning in French."*, which becomes:`";you know?, Preside?nt, do.es !not gsbw s;ame nrsbomh French.?"` In this case, MarianMT political content, generating: *"Der Präsident. — Das Wort hat die Fraktion der Europäischen Demokraten."* This output is semantically unrelated to the original, indicating that the model resorted to prior knowledge when input signals were insufficient. Conversely, M2M100 output: *"Du weißt?, Präsident?nt.es nicht gsbw s;ame nrsbomh französisch?"* This suggests a more literal copying of the corrupted input with less semantic reconstruction, resulting in low-quality translation but reduced hallucination.

Since the BERT F1 score relies on contextual embeddings rather than exact string matching, it can identify these partially corrupted segments as semantically similar to their clean counterparts. For instance, BERT embeddings for `"Eng!lish"` and `"English"` are still likely to lie close

6.4 Effect of Data Cleaning on LSTM

The results of this experiment show that data cleaning causes significant improvement in the performance and stability of LSTM-based Seq2Seq models.

When the model is trained on unclean data (Figure 15), the model showed unstable behavior in multiple evaluation metrics:

- BERTScore F1 decreased from 0.57 to 0.54 across epochs, showing a loss of semantic accuracy while training.
- BLEU and METEOR changed significantly between epochs without a clear trend show it was increasing.
- Validation loss plateaued after an initial drop, showing that the model was struggling to generalize from noisy input

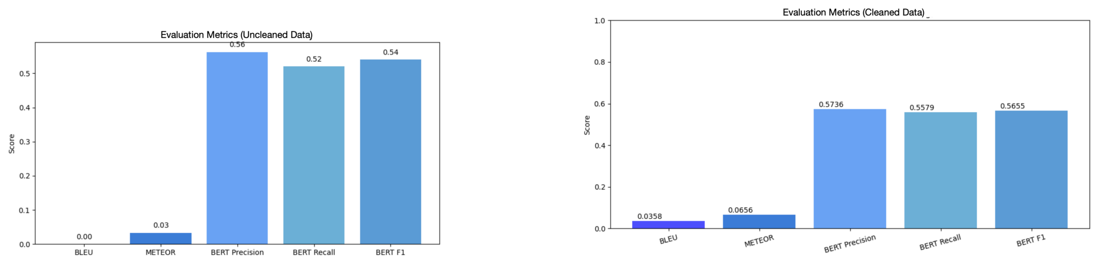


Figure 15: Comparison of training metrics

In contrast, the model that was trained on cleaned data (Figure 15 saw consistently significantly improved results:

- As shown in Figure 15, BERTScore F1 improved steadly and remained higher than it was in the uncleaned model
- BLEU and METEOR scores were more stable and showed more clear positive trends across every epoch
- Training and validation loss decreased more smoothly as seen in Figure 16

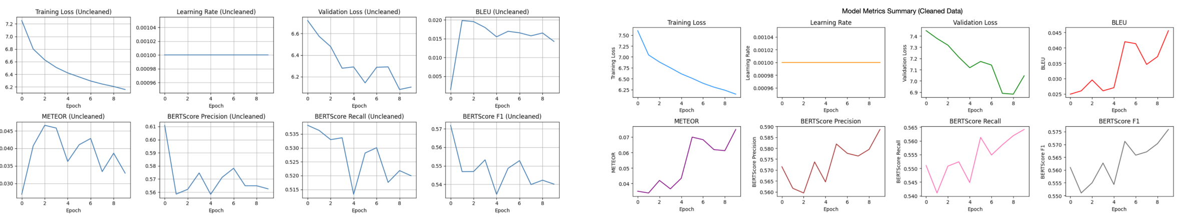


Figure 16: Comparison of training metrics

These findings confirm that traditional LSTM architechtures are highly dependent and sensitive to the the input quality. Data cleaning plays an important role in allowing these models to learn accurate translation mappings and avoid instability during training. While Transformer-based models are more robust to noise and uncleaned data, LSTM-based Seq2Seq models benefit from the extra preprocessing.

6.5 Hyperparameter Combinations

Based on our comprehensive hyperparameter analysis, we identified optimal configurations for both models through systematic evaluation of parameter interactions using grid search across 11 different parameter combination scenarios.

6.5.1 Parameter Interaction Analysis

Figure 17 illustrates the interaction between beam size, length penalty, and repetition penalty for M2M100. The heatmaps reveal that with beam_size ≥ 3, the model achieves consistent performance around 0.826-0.829 BERT F1 score, with only marginal variations based on penalty parameters. This plateau effect suggests that M2M100 exhibits robust performance once the beam size threshold is met.

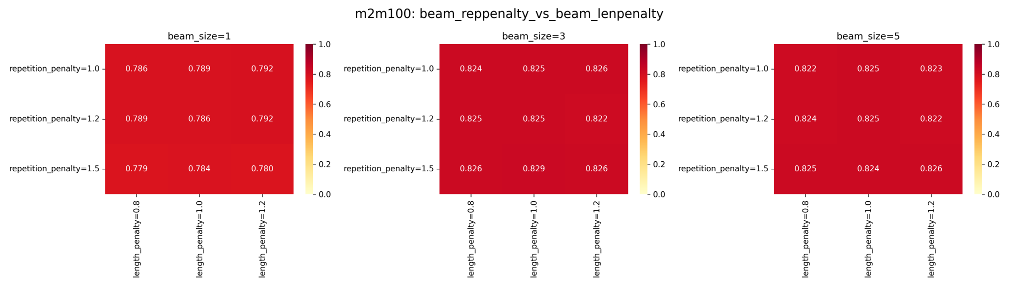


Figure 17: M2M100 performance heatmap showing interactions between beam size, length penalty, and repetition penalty

For MarianMT, Figure 18 demonstrates similar patterns but with notably higher overall scores (0.851-0.854), indicating the specialized bilingual model's greater stability and performance ceiling. The model shows less sensitivity to penalty parameters when beam_size ≥ 3.

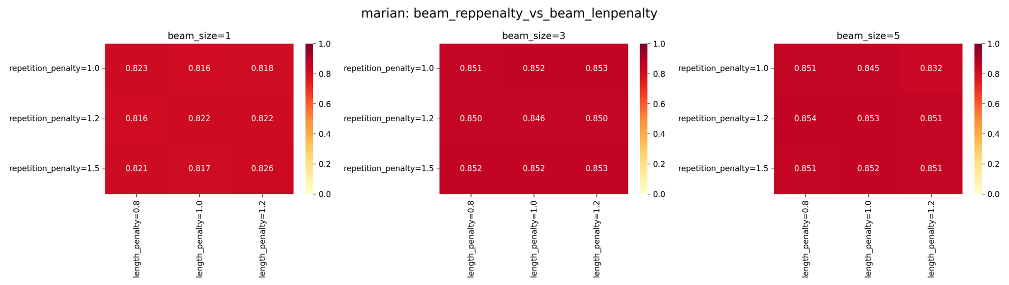


Figure 18: MarianMT performance heatmap showing parameter interactions with consistently high performance

6.5.2 Temperature and Sampling Parameter Interactions

A critical finding from our analysis is the strong negative interaction between temperature and sampling parameters. Figure 19 shows that when temperature increases to 1.3, performance degrades significantly across all top-p and repetition penalty combinations, with BERT F1 scores dropping as low as 0.745 for M2M100.

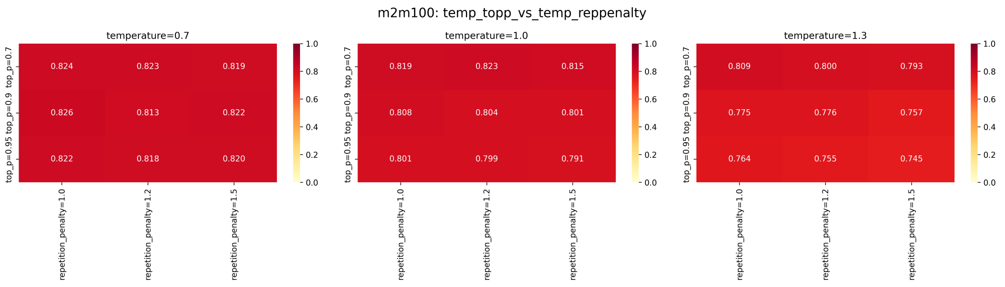


Figure 19: Impact of temperature on M2M100 performance across different top-p and repetition penalty settings

6.5.3 Optimal Configurations

Through our extensive experimentation, we identified the following optimal hyperparameter configurations:

Table 2: Optimal hyperparameter configurations for both models

Parameter	**MarianMT**	**M2M100**
beam_size	5	3
temperature	0.7	0.7
top_p	0.7	0.7
max_length	150	150
repetition_penalty	1.0	1.5
length_penalty	0.8	1.0
BERT F1 Score	**0.853**	**0.827**

6.5.4 Key Findings

Our hyperparameter optimization revealed several important insights:

1. **Beam Size as Threshold Parameter**: Both models exhibit a clear performance threshold at beam_size = 3, beyond which improvements are marginal. This suggests practitioners can achieve near-optimal performance with beam_size = 3 for M2M100, potentially reducing computational cost by 40% compared to beam_size = 5.

2. **Temperature Sensitivity**: Maintaining temperature at 0.7 is crucial for both models. Higher temperatures (1.0-1.3) lead to significant performance degradation, particularly when combined with lower beam sizes.

3. **Model-Specific Penalty Preferences**: M2M100 benefits from higher repetition_penalty (1.5) compared to MarianMT (1.0), likely due to its multilingual nature requiring stronger constraints against repetitive outputs.

4. **Stability Differences**: MarianMT demonstrates greater parameter stability with less variation across different configurations, reflecting its specialized training for the English-German pair.

These findings provide clear guidelines for practitioners: prioritize beam size optimization, maintain conservative temperature settings, and adjust penalty parameters based on the specific model architecture and use case requirements.

7 Discussion

Our results provide evidence to support the fact that Transformer-based models significantly outperform LSTM-based Seq2Seq models in machine translation tasks. MarianMT achieved the highest accuracy on clean data but showed a sensitivity to overfitting while fine-tuning and M2M100 adapted better during fine-tuning and showed a superior robustness to noisy input. These findings suggest that while highly specialized models like MarianMT are powerful, multilingual models like M2M100 offer great flexibility and robustness in real-world applications where the quality of input and domain can differ.

Furthermore, M2M100 maintained the highest BERTScore F1 under both probabilistic and fixed percentage noise, this shows that its training across diverse languages and domains contributed heavily to its ability to handle imperfect input. Although MarianMT performed well in clean settings it was more susceptible to performance degradation when noise was introduced.

Our hyperparameter sensitivity analysis showed us that Transformer models require careful tuning of parameters such as beam size, temperature and length penalties to achieve the best translation quality. BERTScore F1 was the most effective metric for guiding this tuning process as it captured the small improvements in semantic accuracy that more surface level metrics like BLEU might have missed.

The importance of data cleaning for LSTM models was particularly evident in our experiments. Without cleaning the LSTM model produced more unstable results with reduced semantic accuracy and limited generalization. After applying a comprehensive data clean the models stability and accuracy improved substantially. This shows that traditional LSTM architectures remain extremely sensitive to input quality.

Overall, our findings suggest that while Transformer-based models are most robust and flexible, they still need careful fine tuning to avoid overfitting or sensitivity to specific parameters. LSTM models, although performing worse in terms of raw performance improved substantially through data preprocessing. In a real-world application the model choice should depend on the target domain, data quality and available resources.

8 Conclusion

This project presented a comparative analysis of LSTM-based Seq2Seq and Transformer-based models for machine translation, focusing on translation accuracy, robustness and adaptability under varying data conditions. Our results show that Transformer models specifically MarianMT and M2M100, outperform LSTM models. While MarianMT achieved the highest baseline accuracy, M2M100 showed better adaptability during fine tuning and maintained stronger performance under noise. In contrast, the LSTM based model was highly sensitive to input quality with data cleaning significantly improving its stability and accuracy. Our hyperparameter experiments showed that Transformer models need careful tuning to get the most optimal results, with BERTScore F1 being the most reliable metric to guide this process.

To conclude, our findings show that while modern Transformer models are extremely effective for real-world machine translation, their performance is highly dependent on fine-tuning and parameter selection. Meanwhile, traditional Seq2Seq architectures can benefit from preprocessing, drastically improving their performance. These insights can help inform real-life model selection and adaption for machine translation based applications.

References

[Vaswani et al.(2017)Vaswani, Shazeer, Parmar, Uszkoreit, Jones, Gomez, Kaiser, & Polosukhin] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems* (NeurIPS). https://arxiv.org/abs/1706.03762

[Bahdanau et al.(2014)Bahdanau] Bahdanau, D. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*. https://arxiv.org/abs/1409.0473

[Liu et al.(2020)Liu, Ott, Goyal, Du, Joshi, Chen, Levy, Lewis, Zettlemoyer, & Stoyanov] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., & Stoyanov, V. (2020). Multilingual denoising pre-training for neural machine translation. *arXiv preprint arXiv:2001.08210*. https://arxiv.org/abs/2001.08210

[Kudo & Richardson(2018)Kudo, & Richardson] Kudo, T., & Richardson, J. (2018). SentencePiece: A simple and language-independent subword tokenizer and detokenizer for neural text processing. *arXiv preprint arXiv:1808.06226*. https://arxiv.org/abs/1808.06226

[Stahlberg(2019)Stahlberg] Stahlberg, F. (2019). Neural machine translation: A review and survey. *arXiv preprint arXiv:1912.02047*. https://arxiv.org/abs/1912.02047

[Luong et al.(2015)Luong, Pham, & Manning] Luong, T., Pham, H., & Manning, C. D. (2015). Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (pp. 1412–1421). Association for Computational Linguistics.

[Papineni et al.(2002)Papineni, Roukos, Ward, & Zhu] Papineni, K., Roukos, S., Ward, T., & Zhu, W.-J. (2002). BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics* (pp. 311–318). Association for Computational Linguistics.

[Elliott et al.(2016)Elliott, McKeown, & Others] Elliott, D., McKeown, M. K., & Others. (2016). Multi30k: A multilingual image description dataset for cross-lingual image captioning. In *Proceedings of the European Chapter of the Association for Computational Linguistics (EACL)* (pp. 17–24).

[Sennrich et al.(2016)Sennrich, Haddow, & Birch] Sennrich, R., Haddow, B., & Birch, A. (2016). Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics* (pp. 1715–1725). https://www.aclweb.org/anthology/P16-1162/

[Koehn & Knowles(2017)Koehn, & Knowles] Koehn, P., & Knowles, R. (2017). Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872*. https://arxiv.org/abs/1706.03872

[WMT(2014)WMT] WMT. (2014). WMT 2014 English-to-German dataset. http://www.statmt.org/wmt14/translation-task.html

[Sutskever et al.(2014)Sutskever, Vinyals, & Le] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in Neural Information Processing Systems* (NeurIPS).

[Lee & Park(2019)Lee, & Park] Lee, J., & Park, J. (2019). Spelling error correction in neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 2946–2952). Association for Computational Linguistics.

[Michel & Neubig(2018)Michel, & Neubig] Michel, P., & Neubig, G. (2018). MTNT: A testbed for machine translation of noisy text. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing* (pp. 543–553). Association for Computational Linguistics.

[Belinkov & Bisk(2018)Belinkov, & Bisk] Belinkov, Y., & Bisk, Y. (2018). Synthetic and natural noise both break neural machine translation. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

[HuggingFace(2025)HuggingFace Datasets] HuggingFace. (2025). HuggingFace datasets 2025. https://huggingface.co/datasets/wmt/wmt14/tree/main

[HuggingFace(2025)HuggingFace Transformers] HuggingFace. (2025). HuggingFace Transformers 2025 library. https://github.com/huggingface/transformers