Natural Language Processing
Project 2.2: English-Italian Machine Translation

This project examines machine translation in NLP, focusing on advanced neural network architectures. Using the "European Parliament Proceedings Parallel Corpus 1996-2011" we evaluate the effectiveness of recurrent neural networks (RNNs) for text translation. We explore various preprocessing and vectorization techniques to improve translation quality and assess the impact of attention mechanisms on model performance. Our systematic experimentation and analysis aim to advance machine translation methods and their applications in multilingual contexts.

1 INTRODUCTION

In this section, we will outline our overall approach for the project, detailing its structure and the sequential steps undertaken to achieve our objectives, which will be elaborated upon in this report.

Project's Structure. In this section, we outline our overall project approach, describing its structure and the sequential steps we undertook to meet our objectives, which will be detailed further in this report.

Project Structure. The project is organized into the following files and directories:

- **dataset** The dataset includes the "European Parliament Proceedings Parallel Corpus 1," featuring English-Italian and German-English text pairs. It is used to train and evaluate our machine translation models.
- **logs** This directory contains logs for evaluating various model configurations and setups. It includes sample translations with source and target languages, along with evaluation metrics for each configuration. All evaluation results that are discussed in the report can be found here.
- **plots** This directory contains plots that provide insights into the data, including model training results and various attention maps generated by attention mechanisms.
- **analyze.py** This script analyzes the data to extract various insights before any training occurs, saving them as plots.
- **train.py** Trains various model configurations, evaluates them, and generates corresponding logs and plots.
- **dataset.py** This script contains a class that encapsulates the data and is responsible for implementing various pipelines for maniuplating the data such as pre- processing and vectorization.
- **models.py** Defines and outlines the models used in this project, including RNN-based architectures such as LSTMs for encoding and decoding languages, as well as attention layers.
- **requirements.py** This script defines all the required packages to be able to run this project.

All the code is available in the GitHub repository here.

.

Project's Approach and Outline. In this project, we followed these steps to achieve the goal of extracting meaningful insights from the data and evaluating various models on it.

(1) Extracting general insights from the data
(2) implementing different pre-processing pipelines
(3) implementing different vectorization pipelines
(4) Train LSTM models for English-Italian machine translation (both directions) using various configurations and evaluate their performance.
(5) Train character-based LSTM models for English-Italian machine translation (both directions), encoding and decoding individual characters instead of whole tokens, using various configurations and evaluating their performance.
(6) Implement attention mechanisms in both the encoder and decoder components of the LSTM model, and assess the resulting improvements in translation performance.
(7) Perform German-to-Italian translation using English as a pivot language, evaluate performance after both the initial and secondary translations, and assess the associated challenges.

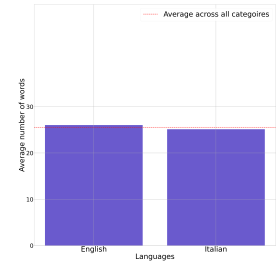
2 EXTRACTING GENERAL INSIGHTS FROM THE DATA

In this section, we will explore key insights derived from translating the English-Italian dataset of the "European Parliament Proceedings Parallel Corpus" and discuss their implications for the subsequent phases of the project.

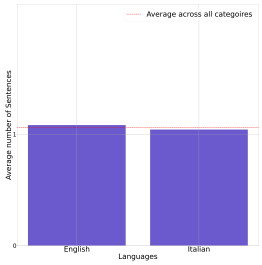
Average Number of words. As illustrated in Figure 1a , both languages exhibit a similar average number of words per sample, suggesting comparable levels of expressiveness. This uniformity implies that the translation models do not need to account for significant discrepancies in sentence length between the two languages. Additionally, the similarity in average word count might contribute to balanced training data, potentially leading to more consistent translation quality across both languages.

Average Number of sentences. As depicted in Figure 1b, the average number of sentences per sample is similar across both languages, with approximately one sentence per sample. This uniformity suggests that the translation task may be relatively straightforward, as fewer sentences per sample generally simplify the translation process. A higher number of sentences typically introduces additional complexity, so this low average could contribute to more manageable translation challenges.

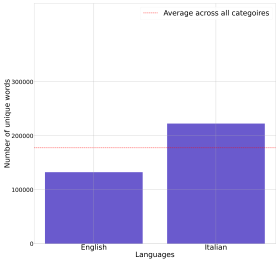
Number of unique words. As illustrated in Figure 1c, there is a substantial difference in the number of unique words between the Italian and English translations, with Italian exhibiting a notably higher count. This disparity suggests increased complexity in encoding and decoding Italian compared to English, as the model must account for a broader range of vocabulary in Italian. Consequently,



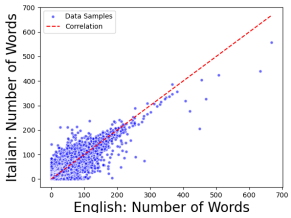
(a) Average number of words per language



(b) Average number of sentences per language



(c) Number of unique words per language



(d) Correlation Between Text Lengths in English and Italian

Fig. 1. Various plots

this higher diversity of unique words may pose additional challenges in translation accuracy and model performance.

Correlation of Text Lengths Across Languages. We analyze the correlation between English and Italian translation lengths (number of words) to determine if longer English texts correspond to longer Italian texts. The covariance matrix in Table 1 shows a strong correlation, indicating that variations in text length are closely related between the two languages. Longer English texts generally align with longer Italian texts, suggesting consistent translation lengths. Figure 1d further supports this, revealing a high correlation with some outliers, but they are relatively few. These insights enhance our understanding of the translation process and the effect of text length on translation quality.

Table 1. Covariance Matrix of Text Lengths

	English Text Length	Italian Text Length
English Text Length	235.20	216.68
Italian Text Length	216.68	223.84

Important Words in each language. We identified the most significant words in each category using the TF-IDF (Term Frequency-Inverse Document Frequency) metric. By averaging the TF-IDF scores for each word across both English and Italian datasets, we assessed their importance and plotted the results for each category,

(a) Important words in English

(b) Important words in Italian

Fig. 2. Most Important Words

as shown in Figure 2. The analysis reveals that both languages highlight similar key terms, such as "commission," "president," "Europe," and "union," indicating their importance in both languages.

3 PRE-PROCESSING PIPELINES

For machine translation, we have developed several preprocessing pipelines to enhance the training and processing of input texts. Given our dataset's large size of 1,909,115 samples, we will filter out any samples that may be distorted or poorly processed. This section outlines the key preprocessing methods we have implemented to ensure the quality and effectiveness of the data.

Lowercasing. Lowercasing converts all text to lowercase, ensuring consistent handling of words regardless of their original case.

Removing Invalid Samples. We have removed empty samples and their corresponding entries, as they do not contribute value during training or testing. Additionally, we excluded samples containing XML tags (starting with "<") to avoid introducing spurious correlations in the translation process.

Tokenization. Tokenization is a key preprocessing step in machine translation. We chose Byte Pair Encoding (BPE) for its ability to handle frequent and rare words by merging common byte pairs, thus efficiently managing vocabulary size and capturing subword information. We trained a BPE tokenizer on our English and Italian datasets with a vocabulary size of 10,000 for each language, balancing coverage and efficiency. Special tokens, "<s>" for start and "</s>" for end, were added to help the model identify sequence boundaries, improving its handling of training and translation.

Other Pre-processing Methods. We also considered other preprocessing methods, such as removing stop words and punctuation.

However, we decided against these techniques because stop words and punctuation can provide essential meaning and context. Removing them could hinder the translation process, as the model needs to generate these elements to produce meaningful and accurate translations.

4 VECTORIZATION PIPELINES

We have developed vectorization pipelines to convert preprocessed text into vector representations, enabling effective machine translation.

Word Embeddings. We trained two separate Word2Vec [Mikolov et al. 2013] models—one for English and one for Italian—each to generate semantic word embeddings. This approach captures the meanings and relationships between words within a continuous vector space, which is crucial for effective machine translation. Each Word2Vec model was trained using negative sampling with 50 negative samples, over 10 epochs, and with a context window of 5. This configuration aimed to produce meaningful and high-quality embeddings for both languages.

Trained Embeddings. We also implemented learned embeddings as a method of vectorization, where the embeddings are weights or parameters within the model that are learned end-to-end during training. This approach allows the model to develop its own embeddings tailored specifically to the task, offering greater flexibility and adaptability. Although this method is more complex, it enables the model to better fit the task requirements by learning context-specific embeddings.

In the following sections, we will evaluate both vectorization methods to compare fixed embeddings with learned embeddings and determine which approach performs better.

5 EVALUATION METRICS

Evaluating machine translation models can be challenging, as traditional metrics such as accuracy, precision, and recall are not well-suited for this complex task. Instead, we use two evaluation metrics that address different aspects of machine translation performance.

The Bilingual Evaluation Understudy (BLEU) score assesses machine translation quality by comparing generated translations to reference translations, focusing on n-gram precision. To address variations in translation length, BLEU includes a brevity penalty, with scores ranging from 0 to 1, where higher values indicate better quality. We use BLEU with a smoothing function from the NLTK library to manage cases with missing n-grams. The formula for BLEU is: $\text{BLEU} = \text{BP} \times \exp\left(\sum_{n=1}^{N} w_n \log p_n\right)$ where BP is the brevity penalty, p_n is the precision of n-grams, and w_n is the weight for each n-gram, with smoothing applied to handle zero counts.

Perplexity. Perplexity measures how well a probability model predicts a sample. In machine translation, it evaluates how effectively the model predicts the target translation given the input. Lower perplexity indicates better performance, as it signifies that the model assigns higher probabilities to the correct translations. The perplexity is the inverse probability of the target sequence, normalized by the length of the sequence. The formula for perplexity is: $\text{Perplexity} = \left(\prod_{i=1}^{N} p(w_i|w_{i-1}, \ldots, w_{i-n+1})\right)^{-\frac{1}{N}}$ where N is the number of words in the target sequence, and $p(w_i|w_{i-1}, \ldots, w_{i-n+1})$ is the probability assigned by the model to the i-th word given the previous words.

Length-Metric Correlation. Length-metric correlation examines how the evaluation metric's score relates to the length of the input sequences, measured in tokens. This analysis reveals how the model's performance varies with different sequence lengths, highlighting any tendencies or issues with shorter or longer texts. Understanding this correlation helps in assessing whether text length impacts translation quality and guides improvements to address performance inconsistencies.

6 LONG SHORT-TERM MEMORY (LSTM)

For machine translation, various model architectures can be employed, including different types of Recurrent Neural Networks (RNNs) such as vanilla RNNs, and Long Short-Term Memory (LSTM) networks [Vennerød et al. 2021]. We have chosen LSTMs because they are more effective at handling long-term dependencies compared to traditional RNNs. This capability makes LSTMs more likely to produce high-quality translations by capturing complex patterns and relationships over extended sequences.

Model Architecture and Hyperparameters. The model architecture comprises two unidirectional LSTMs: one serving as the encoder and the other as the decoder. The encoder processes the vectorized input sequentially, generating hidden and cell states. These states are then passed to the decoder, which generates sequences until it reaches either the maximum sequence length of 512 tokens or the end token "</s>".

Model Hyperparameters. For both the encoder and decoder LSTMs, we used an embedding size of 200 for both fixed and learned embeddings. The hidden layer size is set to 500 neurons, which corresponds to the size of the hidden and cell states. Additionally, the decoder takes the output of the LSTM and applies a fully connected layer to map this output to the vocabulary size, enabling the prediction of probabilities for each token.

Training the Model. We trained all configurations with a learning rate of 0.001, as recommended in similar scenarios. Despite the general recommendation for more epochs, we limited training to 10 epochs to efficiently experiment with different configurations. We employed the Adam optimizer and used cross-entropy loss with softmax as the training criterion. We split the data into 80% for training and 20% for testing (unseen data) across all configurations.

Table 2. Evaluation Metrics for Different Translation Configurations

Translation	BLEU	Perp.	BLEU-Len. Cov.	Perp.-Len. Cov.
en→it (Fixed)	0.0567	38.01	-0.4159	184.92
en→it (Learned)	0.0621	38.68	-0.4562	154.64
it→en (Fixed)	0.0689	37.37	-0.4531	237.02
it→en (Learned)	0.0769	39.49	-0.5377	246.99

Evaluating the Model. In Table 2, we present the evaluation metrics for different configurations of translations: English-to-Italian (en → it) and Italian-to-English (it → en), using both fixed embeddings (trained Word2Vec embeddings) and learned embeddings (model parameters). The metrics include BLEU scores, Perplexity (Perp.), and the covariance between these metrics and sequence length.

We observe that the Italian-to-English translations consistently show better metrics compared to English-to-Italian translations across all configurations. This may be due to the fact that English has fewer unique words compared to Italian, increasing the likelihood of generating the correct token in the English translations.

Learned embeddings, while more complex and potentially better at capturing data nuances, require more extensive data to ensure meaningful representations for all tokens. Although learned embeddings achieved higher BLEU scores, indicating better sequence generation for shorter texts, they performed worse than fixed embeddings for longer sequences. This is evident from the covariance metrics, which show that learned embeddings are less effective for longer sequences compared to fixed embeddings. Additionally, fixed embeddings have lower perplexity.Perplexity is more sensitive to longer sequences, suggesting fixed embeddings are more robust to longer sequences, while learned embeddings perform better with shorter sequences.

7 CHARACTER-BASED LSTM

In previous configurations, we employed Byte Pair Encoding (BPE) for tokenization. In this section, we explore a character-based LSTM, which performs tokenization at the character level. In this approach, each character is encoded independently, and the model generates one character at a time during decoding.

The primary advantage of this method is the significantly smaller vocabulary size, typically between 200 and 300 characters, compared to the 10,000-token vocabulary used with BPE. However, a major drawback is that character tokens lack semantic meaning, which can hinder the model's ability to capture contextual information. Additionally, the resulting sequences of tokens are much longer, which can pose challenges for the model's training and performance.

Table 3. Evaluation Metrics for Different Character-Based Translation Configurations

Translation	BLEU	Perp.	BLEU-Len. Cov.	Perp.-Len. Cov.
en → it (Fixed)	0.2370	2.49	1.0596	-0.3316
en → it (Learned)	0.2776	2.48	2.3229	-0.5304
it → en (Fixed)	0.2357	2.54	-0.1329	0.6469
it → en (Learned)	0.2595	2.50	1.0199	0.9953

Evaluating the Model. The character-based models show significantly lower BLEU scores compared to the BPE-tokenized models, as observed in Table 3. This is likely due to the longer sequence lengths inherent in character-based models, which complicate the LSTM's ability to encode and decode sequences effectively. Additionally, the lack of semantic meaning in individual characters further impairs translation quality. Despite their lower BLEU scores, character-based models exhibit relatively low perplexity, attributed to their smaller character vocabulary and reduced complexity in choosing characters. However, this lower perplexity does not correlate with better translation quality. The covariance between BLEU score and sequence length is positive for character-based models, indicating that longer sequences lead to slightly improved BLEU scores, although still lower overall. In contrast, BPE models show a negative covariance, suggesting longer sequences generally yield lower BLEU scores. Overall, while character-based models are simpler and require less memory, their translation quality is inferior to BPE models, demonstrating a trade-off between model simplicity and translation effectiveness.

8 ATTENTION MECHANISM

Attention mechanisms have revolutionized natural language processing by enabling models to weigh the importance of different tokens in a sequence relative to each other. This approach allows tokens to attend to all other tokens and assign varying levels of attention based on context, resulting in more contextually nuanced embeddings. Initially introduced in the context of neural machine translation and popularized by the Transformer model [Vaswani et al. 2023], attention mechanisms can also be integrated into RNN-based architectures. In this project, we employ a multi-head attention mechanism, inspired by the Transformer, to enhance our LSTM-based model. We implement attention in three phases: first, source tokens use attention to interact with both preceding and succeeding tokens; second, the decoder leverages attention to align encoded source tokens with decoded target tokens, using a query from the target and keys and values from the source; and third, masked attention is applied within the target sequence to prevent future tokens from influencing the current token's prediction. This multi-faceted attention approach aims to improve the contextual understanding and overall performance of the translation model.

Model and Training Hyperparameters. For the multi-head attention mechanism, we use a single layer of attention with 4 attention heads. The hidden size of the fully connected layer following the attention mechanism matches the hidden size of 500 used throughout the model, while the output embeddings are consistent with the embedding size of 200. This multi-head attention setup is inspired by the attention mechanisms implemented in the Transformer architecture. All other hyperparameters and training configurations, including the learning rate, optimizer (Adam), loss function (cross-entropy with softmax), and the number of training epochs, remain consistent with the previous configurations used in the project.

Table 4. Evaluation Metrics for Different Translation Configurations using attention mechanism

Translation	BLEU	Perp.	BLEU-Len. Cov.	Perp.-Len. Cov.
en→it (Fixed)	0.0540	22.95	-0.4607	97.67
en→it (Learned)	0.0742	25.89	-0.5333	103.13
it→en (Fixed)	0.0615	22.86	-0.5462	143.05
it→en (Learned)	0.0738	26.20	-0.6082	163.49

Evaluating the Model. As shown in Table 4, the application of the attention mechanism has led to significant improvements in the evaluation metrics. Both BLEU and perplexity scores have improved, especially perplexity, indicating a substantial enhancement in the generation of sequences. Another significant insight is the change in metric-length covariance: the BLEU-length covariance is higher, and the perplexity-length covariance is lower, reflecting better performance of the attention mechanism for longer sequences. This improvement is due to the attention mechanism's ability to attend to all past tokens, thereby enhancing contextualized embeddings. In contrast, the standard LSTM model, which processes tokens sequentially, may suffer from forgetting, leading to less effective sequence encoding.

Attention Map. One of the key advantages of using the attention mechanism is the ability to visualize attention weights as maps, which reveal how different tokens attend to one another. This visualization helps interpret the contextual relationships learned by the model. We averaged the attention weights across all four attention heads to generate these maps. Our findings indicate that the attention maps for models with learned embeddings are more interpretable compared to those with fixed embeddings. We will present a sample attention map and provide an interpretation of its insights.

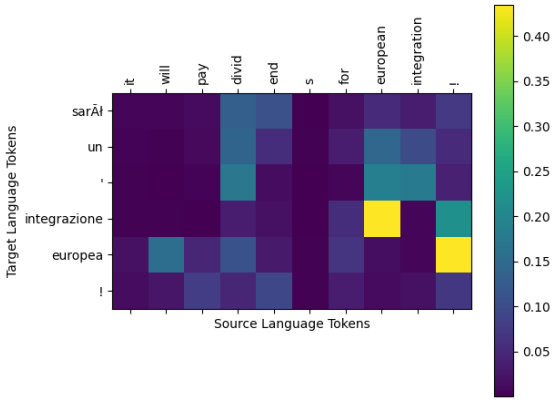


Fig. 3. Attention Map

In Figure 3, we present the attention map illustrating how the LSTM model employs the attention mechanism for translating from the source language. This mechanism, used between the encoder and decoder, allows output tokens to focus on different input tokens to predict the next word. For instance, the attention map shows that the target token *integrazione* gives significant attention to the word *european*, aiding in predicting the next word, *europea*. Additionally, the token *europea* attends to the token *"!"*, which helps it correctly predict this punctuation mark and improves the overall sequence decoding by leveraging context from previous tokens.

9 PIVOT TRANSLATION

We explored a pivot translation approach by translating from German to Italian via English as the pivot language. This method streamlines multilingual translation by using a single pivot language for intermediate translations. However, it requires two translation models, which might impact performance. Given that the LSTM model with learned embeddings and attention mechanisms yielded the best results, we first trained a German-to-English model on the same corpus. We then used this model in conjunction with an English-to-Italian model, both configured identically, to perform the German-to-Italian translation.

Table 5. Evaluation Metrics for German-to-English (Learned Embeddings)

Translation	BLEU	Perp.	BLEU-Len. Cov.	Perp-Len. Cov.
de→en (Learned)	0.0780	27.33	-0.7251	194.98

As shown in Table 5, the German-to-English translation performed relatively well. However, since the translation datasets for German-to-English and English-to-Italian are not aligned quantitatively, a detailed analysis of the German-to-Italian pivot translation is not feasible. Nonetheless, qualitatively judging the samples reveals a significant drop in translation quality due to the use of two models for translation. For example:

> **German:** es ist nicht anders möglich, als dem Präsidenten der Kommission Erfolg zu wünschen; im anderen Falle wird die Kommission in Mißkredit geraten, geschwächt werden, und ihre Befugnisse werden sich verringern.
>
> **English-Pivot:** it is not possible to achieve the commission's powers, and they will be able to achieve their own and they will be in the commission and the commission.
>
> **Italian:** non è possibile raggiungere la commissione e non si sono impegnati e non saranno in grado di non e non.

10 CONCLUSION

Machine translation is a complex task that demands sophisticated approaches. In this project, we utilized an LSTM-based architecture for encoding and decoding text and experimented with various configurations. The integration of attention mechanisms provided a significant performance boost, demonstrating their critical role in improving translation quality by enabling the model to focus on relevant parts of the input sequence. This improvement highlights the importance of attention mechanisms and the need for more advanced methods to tackle the intricacies of machine translation effectively

REFERENCES

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient Estimation of Word Representations in Vector Space. arXiv:1301.3781 [cs.CL]

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2023. Attention Is All You Need. arXiv:1706.03762 [cs.CL] https://arxiv.org/abs/1706.03762

Christian Bakke Vennerød, Adrian Kjærran, and Erling Stray Bugge. 2021. Long Short-term Memory RNN. arXiv:2105.06756 [cs.LG] https://arxiv.org/abs/2105.06756