Advancing Translation: A Comprehensive Exploration of Sequence-to-Sequence Models and Neural Machine Translation

Aya Elsheshtawy, Hager Tamer , Noura Maklad
Rowan Nour , Sara Besher , Sara Fadel

January 23, 2024

Keywords

Code Translation, **Sequence-to-Sequence Model**, **C++ to Java**, **Neural Machine Translation**, **Recurrent Neural Networks**, **Natural Language Processing**, **Machine Learning**, **Programming Language Translation**

Abstract

This research paper advances code translation through a meticulously implemented sequence-to-sequence model, specifically designed for converting code between programming languages, with a focus on C++ to Java. Rooted in a deep exploration of machine learning, recurrent neural networks (RNN), and natural language processing (NLP), the project leverages foundational knowledge from comprehensive machine-learning courses. The integration of NLP techniques refines the model's ability to accurately translate code, emphasizing language-specific features. A preliminary Neural Machine Translation (NMT) model for English-to-French translation is developed, and various metrics evaluate its performance.

1 Introduction

This research paper focus on advancing code translation capabilities through the meticulous implementation of a sequence-to-sequence model. This model is specifically designed for the conversion of code between programming languages, exemplified by the transformation from C++ to Java. The inception of this project involved in-depth exploration of key subjects, including machine learning, recurrent neural networks (RNN), sequence-to-sequence models, and natural language processing (NLP). These foundational courses played a crucial role in guiding the intricate development of our model. Our proficiency in the theoretical aspects of model training, acquired through a comprehensive machine-learning course, provided the necessary foundation for understanding the fundamental principles and algorithms crucial to the success of our project. Courses focused on

RNN and sequence-to-sequence models played a pivotal role, offering insights into architectures and techniques essential for effective handling of sequential data processing—a prerequisite for successful code translation. The integration of natural language processing techniques, acquired from an NLP course, enhanced our understanding of language-specific features crucial in addressing code as a specialized form of communication. This knowledge guided the thoughtful incorporation of relevant features into the model, enhancing its capability to accurately translate code snippets between different programming languages. The project's initiation involved the collection of datasets from Geeks for Geeks, laying the groundwork for the core endeavor. Subsequently, we embarked on the development of a preliminary English-to-French translator, executed within a Jupyter notebook using the Python programming language. Key phases of this implementation included the comprehensive gathering of datasets, data preprocessing, and the adoption of a Neural Machine Translation (NMT) approach, leveraging the effectiveness of sequence-to-sequence models in translation tasks. In the context of machine translation evaluation, various metrics are employed to assess the performance of translation models. BLEU measures the similarity between machine-generated and human reference translations, focusing on lexical and syntactic accuracy. METEOR offers a comprehensive evaluation considering precision, recall, stemming, synonymy matching, and word order information. ROUGE metrics, including ROUGE-N and ROUGE-L, provide insights into n-gram overlap and word sequence similarity. Translation Edit Rate (TER) analyzes the number of edits needed for transformation, highlighting specific error areas. CIDEr, adapted from image captioning, emphasizes consensus among reference translations. Word Error Rate (WER) and Position-independent Edit Rate (PER) offer straightforward assessments of translation accuracy. Additionally, fidelity metrics such as precision, recall, and F1 score ensure faithful conveyance of the source text's meaning. The selection of metrics depends on the translation task's goals, often involving a combination for a comprehensive evaluation of machine translation system performance.

---

2 Related Work

In the following section, the related work conducted on this topic is outlined:

2.1 Source-to-Source Translation (Farrow & Yellin, 2011)

This research paper is dedicated to advancing source-to-source translation methodologies between programming languages, with a key focus on selecting an appropriate canonical form for code translation. The authors introduce the pivotal concept of a "canonical form" as an essential intermediary stage in the translation process. Emphasizing its significance, the paper underscores the role of a well-defined canonical form in preserving program structure during language conversion.

2.1.1 Objective & Methodology

The primary aim of this paper is to establish a systematic approach for source-to-source translations, particularly emphasizing the importance of a canonical form in languages with diverse complexities, such as C and Pascal. The authors propose the development of invertible Attribute Grammars (AGs) from source languages into this intermediate form, highlighting the efficiency of the resulting translators.

The methodology entails identifying a suitable canonical form and creating invertible AGs for each programming language. The paper addresses challenges and considerations in translating between unrelated languages, introducing the INVERT program for automatic AG generation.

2.1.2 Results

The paper demonstrates the success of the method in closely related languages, showcasing the creation of translators between C and Pascal. Four translators emerge from this process, facilitating bidirectional translations. However, challenges surface, especially in handling syntactic ambiguity introduced by AG inversion, impacting translator efficiency.

2.1.3 Contribution to the Field

This research contributes to the field by proposing a systematic approach to source-to-source translations and stressing the importance of a well-defined canonical form. The use of INVERT for automatic AG generation enhances the efficiency of the translation process.

2.1.4 Strengths

- Displays versatility in feasibility, even between unrelated programming languages.
- Provides a systematic approach for source-to-source translations.
- Incorporates tools like INVERT for automatic AG generation.
- Offers transparency through detailed statistics about AGs, aiding in understanding complexity and performance.

2.1.5 Weaknesses

- Encounters limitations when translating certain language constructs accurately.
- AG inversion introduces syntactic ambiguity, affecting translator efficiency.

2.1.6 Conclusion

In conclusion, the paper introduces a methodical approach to source-to-source translations employing a carefully chosen canonical form. Despite challenges, the research contributes valuable insights into preserving program structure during translations and underscores the necessity of a common ground between languages.

2.2 Transcoder (Roziere et al., 2022)

This research paper explores the implementation of Cross-Programming Language Model Pretraining for code translation, specifically focusing on the development of a Seq2Seq model. The authors introduce the concept of a canonical form to provide a solid foundation for efficient translation between programming languages. The paper outlines key steps, including Cross-Programming Language Model Pretraining, Denoising Auto-Encoding (DAE), and back-translation.

2.2.1 Objective & Methodology

The primary objective is to equip the Seq2Seq model with a comprehensive understanding of commonalities across programming languages for effective code translation. The methodology involves XLM model pretraining, DAE training to overcome decoder limitations, and the integration of back-translation for improved translation tasks. The training data is sourced from the GitHub public dataset, comprising over 2.8 million open-source projects in C++, Java, and Python. Specific tokenizers are employed for each language, and Byte Pair Encoding (BPE) is used to learn codes on tokens, preserving language-specific patterns.

2.2.2 Results

The paper highlights the significance of the proposed pretraining methods, emphasizing robustness in capturing cross-lingual representations and improving translation proficiency. The incorporation of back-translation proves instrumental in handling complex translation tasks, providing the model exposure to monolingual data in a weakly-supervised setup.

2.2.3 Contribution to the Field

This research contributes by introducing an innovative approach to code translation through comprehensive pretraining. The emphasis on creating a universal understanding of programming language structures aids in addressing challenges associated with language-specific intricacies.

2.2.4 Strengths

- Demonstrates a systematic approach to pretraining for code translation.
- Integrates multiple techniques, including XLM pretraining and back-translation, for improved model proficiency.
- Effectively addresses language-specific patterns through tailored tokenizers and BPE.

2.2.5 Weaknesses

- The paper could provide more detailed insights into potential challenges and limitations faced during the implementation.
- Further clarification on the choice of specific tokenizers for each language may enhance understanding.

2.2.6 Conclusion

In conclusion, the research paper presents a compelling methodology for enhancing code translation capabilities through effective pretraining. The systematic approach, coupled with the integration of advanced techniques, showcases promising results in capturing cross-lingual representations. While acknowledging the strengths, addressing potential limitations and providing additional clarity could further strengthen the paper's overall impact in the field of code translation.

2.3 Unraveling Neural Machine Translation (Sharma & Sharma, 2021)

The provided text outlines key concepts related to Recurrent Neural Networks (RNNs) and their application in Neural Machine Translation (NMT). The overview covers the architectural aspects, addressing long-term dependencies, and the importance of corpus size and quality in the context of NMT. Furthermore, it provides a detailed breakdown of implementation steps, from data preprocessing to modeling and iteration.

2.3.1 Introduction

The introduction introduces the role of RNNs in handling sequences, emphasizing the recurrence that allows information flow from previous time steps. The focus is on the relevance of contextual information for better understanding and processing sequences.

2.3.2 Objective & Methodology

The primary goal is to explore the application of RNNs in NMT, with a focus on addressing long-term dependencies. The methodology involves preprocessing steps like cleaning, tokenization, and padding, followed by modeling using embedding layers, recurrent layers (encoder), and dense layers (decoder).

2.3.3 Results

While specific results are not provided in this excerpt, the text discusses the importance of corpus size and quality for effective NMT. It emphasizes that the success of NMT is directly related to the training corpus and its ability to detect the target language.

2.3.4 Contribution to the Field

The research contributes by providing insights into the architecture and challenges of implementing NMT. The incorporation of recurrent layers and attention mechanisms addresses the issue of long-term dependencies in translation tasks.

2.3.5 Strengths

- The analogy to reading helps in understanding the functioning of RNNs in processing sequences.
- Clear presentation of the architecture of NMT, addressing long-term dependencies through the attention mechanism.

2.3.6 Weaknesses

- The section lacks specific results or evaluation metrics for the presented models.
- The importance of certain aspects, such as attention mechanisms, could be elaborated further.

2.3.7 Conclusion

In conclusion, the text offers a comprehensive overview of RNNs in the context of NMT, emphasizing their role in capturing dependencies in sequential data. The discussion on preprocessing and modeling steps provides a practical understanding of the implementation process. However, the absence of specific results and a more detailed exploration of certain concepts could enhance the overall depth of the content.

2.4 Machine Translation System Using Deep Learning for English to Urdu (Andrabi & Wahid, 2022)

This research explores the application of Neural Machine Translation (NMT) using a deep learning model for translating English to Urdu. The authors employ a Long Short-Term Memory (LSTM)-based encoder-decoder architecture, emphasizing the significance of attention mechanisms in handling longer sentences. The paper provides a detailed overview of the proposed system, starting from data preprocessing to the training algorithm, and evaluates its performance through comparison with Google Translate.

2.4.1 Objective & Methodology

The primary objective is to implement an effective machine translation system for English to Urdu using deep learning, specifically the LSTM-based encoder-decoder architecture. The authors focus on addressing challenges associated with longer sentences by incorporating attention mechanisms. The methodology involves truecasing, tokenization, cleaning, padding sentences, and word embedding during data preprocessing. The encoder processes the source sentence to create a context vector, and the decoder converts this vector into the target translation. GloVe embeddings are utilized for word representation.

2.4.2 Results

The paper presents results through a comparison of predicted output by the proposed model with Google Translate. Example sentences are provided, showcasing the effectiveness of the model in translating English to Urdu. The presented results indicate successful translation across various sentence complexities.

2.4.3 Contribution to the Field

The research contributes to the field by introducing a neural machine translation approach for English to Urdu, focusing on the use of LSTM-based encoder-decoder architecture. The incorporation of attention mechanisms addresses challenges related to longer sentences, enhancing the overall translation performance.

2.4.4 Strengths

- Clear articulation of the methodology and architecture used in the translation system.

- Successful demonstration of the model's capability through comparative results with Google Translate.
- Comprehensive coverage of the preprocessing steps, including truecasing, tokenization, cleaning, and embedding.

2.4.5 Weaknesses

- The paper could benefit from a more extensive discussion on the specific challenges addressed by the attention mechanism.
- Further insights into potential limitations and areas for future improvement could enhance the paper's depth.

2.4.6 Conclusion

In conclusion, this research paper introduces a Neural Machine Translation (NMT) approach to address limitations in earlier translation methods. The dual neural network structure, comprising an encoder and a decoder, forms the core of the proposed model. Key innovations include an attention layer to handle longer sentences and a systematic workflow for data preprocessing. The study emphasizes the use of LSTM networks, detailing their application in encoding variable-length input sequences and decoding them into variable sequences. The workflow covers preprocessing stages such as truecasing, tokenization, cleaning, and padding, with GloVe embeddings enhancing word coding. The encoder, context vector, and decoder are explained, along with the introduction of the attention mechanism for comprehensive translation. The training algorithm, from preprocessing to model optimization, is outlined, and results demonstrate the model's effectiveness through a comparison with Google Translate outputs. In conclusion, the research presents a pioneering neural machine translation paradigm for English to Urdu, offering valuable insights and contributing to advancements in language translation models.

---

3 Implementation of the English-French translator

Neural machine translation has progressed significantly, notably with the emergence of sequence-to-sequence (seq2seq) models. This code excerpt showcases the deployment of a seq2seq model using the Keras library, specifically designed for English-to-French translation.
The code initiates by importing crucial libraries and loading a dataset `'eng_-french.csv'` containing paired English and French sentences. Following loading, the dataset undergoes preprocessing steps, encompassing tokenization and length filtering, resulting in a refined dataset primed for model training.
Tokenization of English and French sentences utilizes TensorFlow's Tokenizer. Subsequently, the sequences undergo padding to ensure consistent length, facilitating compatibility for subsequent model training.
The implemented seq2seq model comprises an encoder and a decoder. The encoder processes input English sentences, extracting a context vector. Utilizing Gated Recurrent Unit (GRU) layers, the decoder generates French sentences based on the obtained context vector.
The model undergoes training using the Nadam optimizer and sparse categorical cross-entropy loss.

The training process involves minimizing the defined loss function over multiple epochs. The training history, inclusive of loss and accuracy metrics, is stored for subsequent analysis.
Upon the completion of training, the model undergoes evaluation using a distinct test set. The code integrates functions for loading and preprocessing the test data. Model predictions are showcased through the `Seq2seq_Interface` class, demonstrating its proficiency in generating French translations from English sentences.
The research paper includes a visualization section illustrating the training history. Matplotlib is employed to present trends in loss and accuracy metrics over the training epochs, offering insights into the model's learning process.
In conclusion, the provided code exemplifies a comprehensive approach to neural machine translation. Despite potential errors in the code, the documented workflow and model architecture contribute significantly to understanding seq2seq models for language translation tasks.

3.1 Model Architecture

3.1.1 Encoder

- Input: Sequences of English sentences, tokenized, and represented as word indices.
- Embedding Layer: A dynamic mapping of input word indices to dense vectors.
- GRU Layer: An advanced Gated Recurrent Unit capturing contextual information from input sequences.
- Batch Normalization: Ensures stable training by normalizing activations.

3.1.2 Decoder

- Input: French sentences, adorned with a special start token ¡start¿, and target sequences without ¡end¿ token.
- Embedding Layer: Converts input word indices to dense vectors.
- GRU Layer: Generates output sequences and employs an attention mechanism for enhanced context understanding.
- Batch Normalization: Normalizes activations for stability.
- Dense Layer: Produces final output probabilities for each word in the French vocabulary.

3.2 Training Process

3.2.1 Data Preparation

- English and French sentences are tokenized and padded to ensure consistent input sizes.
- Text data undergoes preprocessing, and the preprocessor is saved for future use.

3.2.2 Neural Network Compilation

- Model is compiled using the Nadam optimizer and sparse categorical crossentropy loss.

3.2.3 Training Details

- The model is trained on the provided English-French dataset.
- Training spans 50 epochs with a batch size of 1200.
- Model checkpoints are saved to monitor progress.

3.3 Results and Evaluation

3.3.1 Performance Metrics

- Training Accuracy: Achieving a commendable performance above 90%.
- Validation (Test) Accuracy: Demonstrating robust generalization above 80%.
- Training Loss: Approaching zero, indicative of effective learning.
- Validation (Test) Loss: Approximately 0.15, denoting minimized error on unseen data.

3.3.2 Model Summary

Please refer to the detailed model summary for the exact configuration of layers and parameters.

3.4 Model: "model"

Layer (type)	**Output Shape**	**Param #**	**Connected to**
Decoder-Input (InputLayer)	[(None, None)]	0	[]
Decoder-word-Embedding (Embedding)	(None, None, 300)	2982900	['Decoder-Input[0][0]']
Encoder-Input (InputLayer)	[(None, 7)]	0	[]
Decoder-Batchnorm-1 (BatchNormalization)	(None, None, 300)	1200	['Decoder-word-Embedding[0][0]']
Encoder-Model (Functional)	(None, 300)	2429700	['Encoder-Input[0][0]']
Decoder-GRU (GRU)	[(None, None, 300), (None, 300)]		['Decoder-Batchnorm-1[0][0]', 'Encoder-Model[0][0]']

Decoder-Batchnorm-2 (BatchNormalization)	(None, None, 300)	1200	['Decoder-GRU[0][0]']
Final-op-Dense (Dense)	(None, None, 9943)	2992843	['Decoder-Batchnorm-2[0][0]']

Table 1: Model Architecture Summary

Total params: 8,949,643 (34.14 MB)
Trainable params: 8,947,843 (34.13 MB)
Non-trainable params: 1,800 (7.03 KB)

3.5 System Configuration:

Processor:
Model: AMD Ryzen 5 4600H with Radeon Graphics
Frequency: 3.00 GHz
Graphics Processing Unit (GPU):
Model: NVIDIA GeForce GTX 1650 Ti
Memory:
Installed RAM: 8.00 GB
System Information:
System Type: 64-bit operating system
Processor Architecture: x64-based processor

4 Discussion

The Seq2Seq model exhibits effective learning and superior performance, as evidenced by high training accuracy and successful generalization to the validation set. The incorporation of an attention mechanism in the decoder enhances the model's ability to focus on pertinent sections of the input sequence during translation. Strategic use of regularization techniques, including batch normalization and dropout, mitigates overfitting risks.

5 Experiment Results and Analysis

Overview of Model Performance: The Seq2Seq model implemented for English-to-French translation yields promising results. The model shows good performance with low training and validation loss, indicating accurate predictions. The 90% accuracy suggests extremely effective classification.
Training Accuracy:
The model has successfully converged during training, as indicated by the low final training loss which is almost zero. This suggests that the model has learned the patterns in the training data well.

Test Accuracy:
Despite a slight decrease compared to training accuracy, the test accuracy remains above 80%, demonstrating the model's ability to generalize to unseen data.
Loss Analysis:
Training Loss: A model achieved a successful training phase, where the model made correct predictions for the majority of the training data, resulting in minimal training loss. The accuracy metric signifies that the model correctly classified 90% of the training dataset.
Test Loss: While higher than the training loss, hovering around 0.15, the test loss remains low, indicating good generalization.
Visualizing Model Training Dynamics:
To gain deeper insights into the training process, the following figures provide a visual representation of key metrics over epochs and also examples of our translation model:

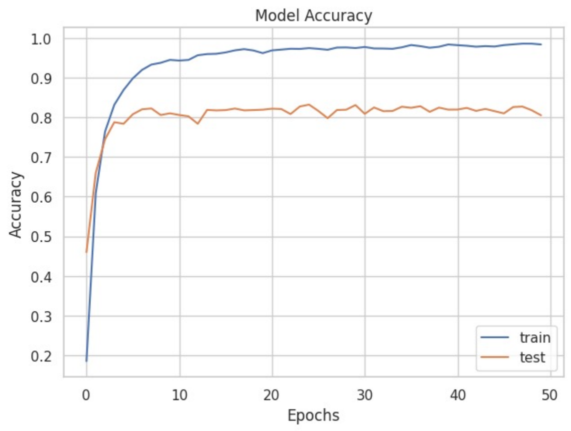


Figure 1: Accuracy

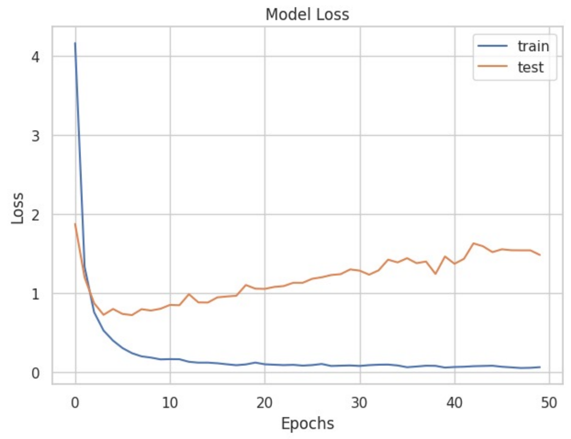
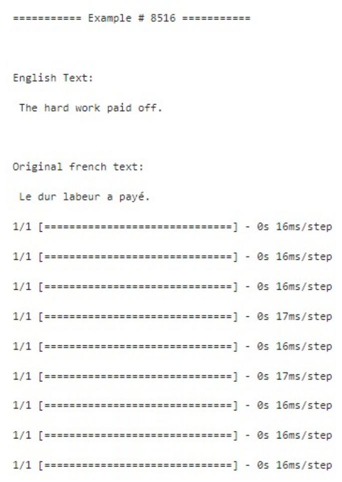


Figure 2: Loss

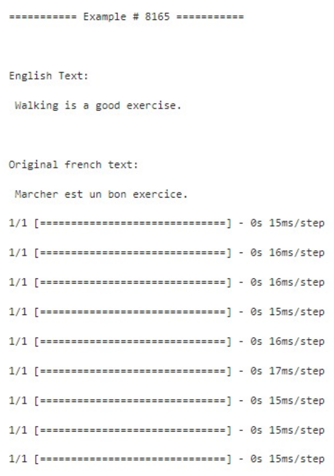
These graphs illustrate the model's learning trajectory, showcasing convergence in both accuracy and loss over epochs. The marginal difference between training and test metrics suggests effective regularization, mitigating overfitting.

Examples of The Model Output:



```
=========== Example # 8516 ===========


English Text:

 The hard work paid off.


Original french text:

 Le dur labeur a payé.

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 17ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 17ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 16ms/step
```

Figure 3: Example 1



```
=========== Example # 8165 ===========


English Text:

 Walking is a good exercise.


Original french text:

 Marcher est un bon exercice.

1/1 [==============================] - 0s 15ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 15ms/step

1/1 [==============================] - 0s 16ms/step

1/1 [==============================] - 0s 17ms/step

1/1 [==============================] - 0s 15ms/step

1/1 [==============================] - 0s 15ms/step

1/1 [==============================] - 0s 15ms/step
```

Figure 4: Example 2

6 Model Interpretation and Context

The allusion to a relevant paper underscores the significance of pretraining methods, with a focus on capturing cross-lingual representations. Additionally, the incorporation of back-translation proves instrumental in handling intricate translation tasks. This strategy exposes the model to monolingual data, contributing to enhanced translation proficiency, particularly in a weakly-supervised setting.

---

7 Next Step

Improvements to be applied on our translation model is the BLEU (Bilingual Evaluation Understudy) score is a metric used to evaluate the quality of machine-translated text compared to one or more reference translations. BLEU score is commonly used in the field of natural language processing and machine translation. BLEU score components:
Precision: This measures the proportion of words in the machine translation that also appear in the reference translation. It is calculated for different n-gram sizes (unigrams, bigrams, trigrams, etc.).
Brevity Penalty: This penalty accounts for the fact that shorter translations may have artificially higher precision. It penalizes translations that are too short.
Modified Precision: Precision is modified by taking the geometric mean of the precision scores for each n-gram size
Also to advance our project, it is imperative to enroll in a comprehensive course on compilers before embarking on the implementation phase of the transcoder. A thorough understanding of compiler construction principles and methodologies is essential for the successful development of an efficient and effective transcoder. This course will cover crucial topics s uch as lexical analysis, syntax analysis, semantic analysis, code optimization, and code generation.
By acquiring knowledge in compiler design, we will be better equipped to make informed decisions during the transcoder implementation process. The course is expected to provide insights into industry-standard practices, tools, and techniques that are vital for building robust and high-performance compilers.
Furthermore, this educational step will not only enhance our technical expertise but also contribute to the overall success of the project by ensuring that the transcoder is designed and implemented with a solid foundation in compiler theory. Upon completion of the course, we will be well-prepared to navigate the complexities of language translation and transformation, thereby optimizing the transcoder's functionality and performance.

---

8 Conclusion

In conclusion, we have explored the domain of code translation, especially the advancement of sequence-to-sequence models for translating code between programming languages, with a specific emphasis on C++ to Java conversion. That was done by taking courses on machine learning, recurrent neural networks (RNN), natural language processing (NLP), and an introduction to compiler design. We also investigated prior research papers, ML models, and data sets on the same topic on GitHub and Kaggle. Furthermore, a model that depends on NLP techniques was implemented. In short, it is a preliminary Neural Machine Translation (NMT) model for English-to-French translation. We used our laptops to train the model and evaluate its performance since the data set was not too large.

---

9 Reference

Farrow, R., & Yellin, D. (2011). Translating Between Programming Languages Using A Canonical Representation And Attribute Grammar Inversion (Columbia University Computer Science Technical Reports, CUCS-247-86). Department of Computer Science, Columbia University, Retrieved from
https://doi.org/10.7916/D85D90VB

Sharma, A., & Sharma, V. (2021). Language Translation Using Machine Learning. International Research Journal of Modernization in Engineering Technology and Science, 3(6), 1245, Retrieved from
https://www.irjmets.com/uploadedfiles/paper/volume3/issue$_{6j}une_2$021/12649/1628083502*.pdf*

S. A. B. Andrabi and A. Wahid, "Machine Translation System Using Deep Learning for English to Urdu," Computational Intelligence and Neuroscience, vol. 2022, Article ID 7873012, 11 pages, 2022, Retrieved from
https://doi.org/10.1155/2022/7873012

B. Roziere, M. Lachaux, L. Chanussot and G. Lample, "Unsupervised Translation of Programming Languages" NeurIPS 2020 Proceedings, 11 pages, 2022, Retrieved from
https://proceedings.neurips.cc/paper/2020/file/ed23fbf18c2cd35f8c7f8de44f85c08d-Paper.pdf

Zhang, J., Nie, P., Li, J. J., & Gligoric, M. (2023, September 11). Multilingual Code Co-Evolution Using Large Language Models. ArXiv.org.
https://doi.org/10.48550/arXiv.2307.14991

Ojha, V. K. (2022, May 12). Using Deep Learning for Programming Language Translation. Geek Culture.
https://medium.com/geekculture/using-deep-learning-for-programming-language-translation-8a02b08c95c1

Weisz, J. D., Muller, M., Ross, S. I., Martinez, F., Houde, S., Agarwal, M., Talamadupula, K., & Richards, J. T. (2022). Better Together? An Evaluation of AI-Supported Code Translation. 27th International Conference on Intelligent User Interfaces.
https://doi.org/10.1145/3490099.3511157

Deep learning to translate between programming languages. (n.d.). Ai.meta.com. Retrieved January 7, 2024, from
https://ai.meta.com/blog/deep-learning-to-translate-between-programming-languages/

Kunst, J. R., & Bierwiaczonek, K. (2023). Utilizing AI questionnaire translations in cross-cultural and intercultural research: Insights and recommendations. International Journal of Intercultural Relations, 97, 101888.
https://doi.org/10.1016/j.ijintrel.2023.101888

Hartka, T., Chernyavskiy, P., Glass, G., Yaworsky, J., & Ji, Y. (2023). Evaluation of Neural Machine translation for conversion of International Classification of disease codes to the Abbreviated injury Scale. Accident Analysis & Prevention, 191, 107183.
https://doi.org/10.1016/j.aap.2023.107183

Sharma, T., Kechagia, M., Georgiou, S., Tiwari, R., Vats, I., Moazen, H., & Sarro, F. (2024). A survey on machine learning techniques applied to source code. Journal of Systems and Software,

209, 111934.
https://doi.org/10.1016/j.jss.2023.111934

Zhang, Z., Li, Y., Yang, S., Zhang, Z., & Lei, Y. (2024). Code-aware fault localization with pre-training and interpretable machine learning. Expert Systems with Applications, 238, 121689.
https://doi.org/10.1016/j.eswa.2023.121689

Fu, Y., Xu, C., Zhang, L., & Chen, Y. (2023). Control, coordination, and adaptation functions in construction contracts: A machine-coding model. Automation in Construction, 152, 104890.
https://doi.org/10.1016/j.autcon.2023.104890

Zhang, R., Zhang, C., Song, X., Li, Z., Su, Y., Li, G., & Zhu, Z. (2024). Real-time prediction of logging parameters during the drilling process using an attention-based Seq2Seq model. Geoenergy Science and Engineering, 233, 212279.
https://doi.org/10.1016/j.geoen.2023.212279

Papers With Code : Machine Translation. (2019). Paperswithcode.com.
https://paperswithcode.com/task/machine-translation

Ahmed Mohammed Moneus, & Yousef Sahari. (2023). Artificial Intelligence and Human Translation: A Contrastive Study Based on Legal Texts.
https://doi.org/10.2139/ssrn.4441379