Semi-Supervised Tree-to-Tree Translation for Functional Hierarchy Standardisation

Chayanika Gangopadhyay

Melinda Hodkiewicz[1], Tyler Bikaun[2]
1. Mechanical Engineering
2. Computer Science and Software Engineering
The University of Western Australia

Ronald Van Moere
CEED Client: Rio Tinto

Abstract

This project explores the application of Neural Machine Translation (NMT) to automate the standardisation of functional hierarchies in the mining sector. Asset hierarchies, crucial for maintenance management, often have inconsistencies due to manual creation and business acquisitions. Our approach employs NMT, treating unstandardised hierarchies as the source language and standardised hierarchies as the target language. We develop an annotation tool for data generation, supporting Subject Matter Experts to create source-target (paired) datasets. We then develop code for two models to handle the NMT: a baseline Long Short-Term Memory (LSTM) and a Tree-LSTM. Initial results suggest that model performance is related to the complexity of data structure with the LSTM model outperforming the Tree-LSTM model on a test of simple synthetic data. Work is ongoing to test these models on the annotated paired business data.

1. Introduction

Asset hierarchies provide a structured framework for organising assets in the mining sector, essential for maintaining equipment and infrastructure to ensure operational continuity (Jafari et al., 2014). These hierarchies typically consist of functional locations (FLOCs), forming what is known as a functional hierarchy. Functional hierarchies organise equipment and infrastructure based on operational roles and spatial relationships, ensuring that maintenance tasks can be targeted and efficiently managed.

The manual creation of asset hierarchies, often on an asset class by asset class, mine by mine, or commodity by commodity basis, has led to different hierarchies being used for the same asset type and asset make and model. This issue often occurs after business acquisitions, where each acquired group has years of maintenance history mapped to their own FLOC hierarchy. Mismatching FLOC hierarchies complicates the aggregation and comparison of maintenance data across different assets and sites, making it difficult to generate reports on asset performance and maintenance activities and ultimately impacting overall operational performance and profitability (Langan, 2013).

1.1 Objective

This project develops and evaluates an approach for mapping disparate hierarchies of the same asset to a standardised hierarchy using neural machine translation. The objective is to automate this mapping process. A key challenge in this project is the absence of off-the-shelf annotated data for the translation task.

1.2 Neural Machine Translation

Neural Machine Translation (NMT) is a subfield of machine learning that leverages neural networks to translate text from one language to another (Gupta & Kumar, 2021). A core component of NMT systems is the sequence-to-sequence (seq2seq) architecture. This architecture is designed to generate a variable-length sequence of tokens from a variable-length sequence of input data (Sriram et al., 2017). The process begins with encoding the input sequence into a fixed-dimensional vector representation known as the context vector, capturing the semantic meaning of the input. The decoder then utilises this context vector to generate the output sequence token by token, facilitating translation.

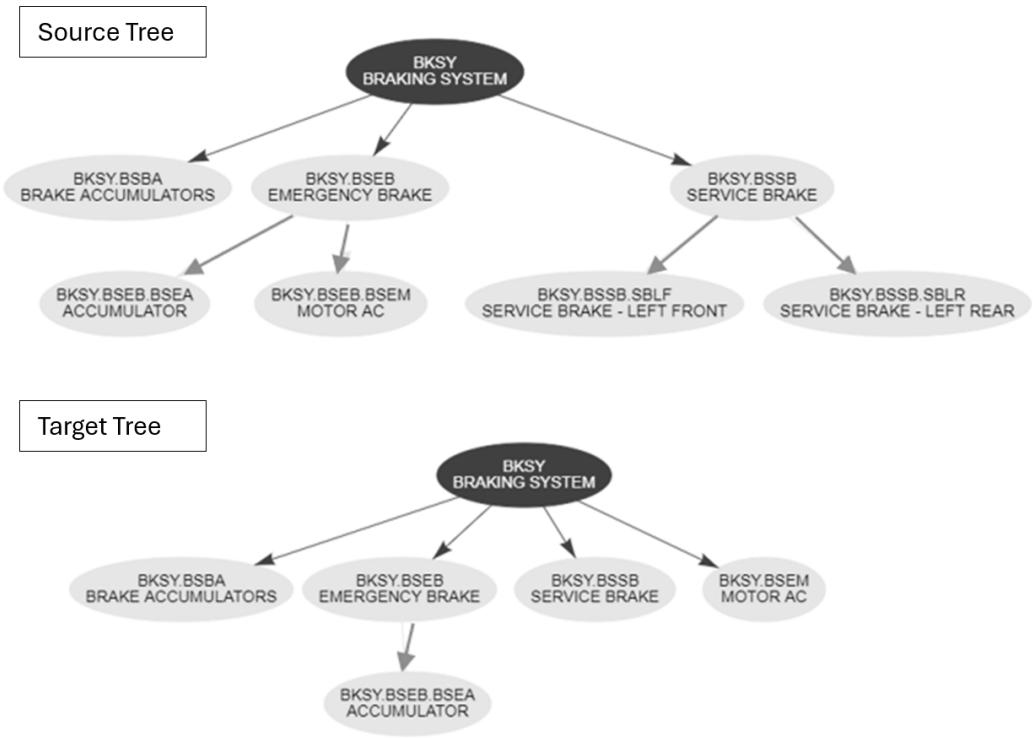

Figure 1 Example Tree-to-Tree Translation.

Similarly, our task of converting an unstandardised hierarchy to its standardised form can be seen as a translation task (see Figure 1). Here, the unstandardised hierarchy is the "source language" containing varied and inconsistent asset classifications. The standardised hierarchy is the "target language" representing a uniform and consistent classification system. The translation process involves understanding the structure of the unstandardised hierarchy, extracting relevant information, and mapping it to the standardised format.

Unlike sequential data, asset hierarchies are organised in a tree-like structure with assets and sub-assets nested within each other, as shown in Figure 1. Standard neural networks combine

the task of learning hierarchical rules with aligning hierarchical sequences. This all-in-one approach can reduce performance when handling such structured data (Chen et al., 2018). Tree-based models explicitly represent the hierarchical structure, allowing for a more effective separation of these tasks. Instead of processing information sequentially like regular neural networks, they work their way up from the bottom of the tree, combining information from child nodes at each level to create a representation of the entire structure. This allows them to better capture relationships and dependencies present in hierarchical data.

Literature on tree-to-tree translation often assumes having sufficient paired data for training. This project is novel in exploring the potential of tree-to-tree translation on hierarchical data in the mining domain where training data is limited. Adapting semi-supervised approaches from existing work, as described in section 2.4, to our task of tree-to-tree translation can be a promising direction for overcoming this challenge.

2. Process

The FLOC Mapping Annotation Tool (FLMAT) described in Section 2.1 was created to generate source-target pairs as training data for our translation model. The original tree-based Long Short-Term Memory (Tree-LSTM) translation model by Chen et al. (2018) did not include code so we developed our own implementation based on Section 3.2 of their paper. To evaluate the performance of our custom model, we created synthetic data as described in Section 2.2. The process is shown in Figure 2.

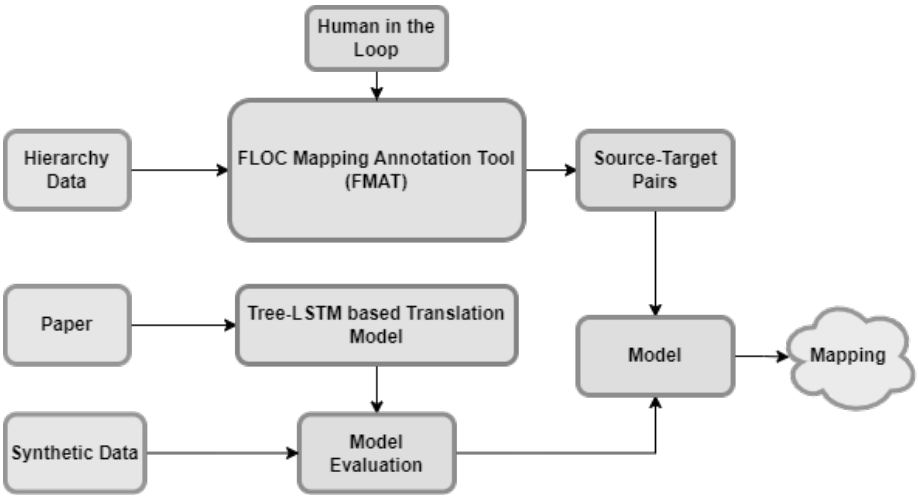

Figure 2 Process map of the tree-to-tree translation process.

2.1 FLOC Mapping Annotation Tool (FLMAT)

One of the challenges in our task is the lack of paired data mapping unstandardised hierarchies to their standardised forms. To address this, we developed the FLOC Mapping Annotation Tool (FLMAT), a human-in-the-loop system that enables Subject Matter Experts (SMEs) to visualise different hierarchies for the same asset and create a standardised hierarchy from them. FLMAT was built using Streamlit, a Python library that simplifies user-interface development. As the user interacts with the tool, paired data and transformations are created, matching unstandardised hierarchies with their standardised forms.

FLOC codes defining the same functional location in different hierarchies are often inconsistent e.g. "Braking system" might be labelled as BRAK in Hierarchy A and as BKSY in Hierarchy B. Standardising these codes is essential before training the model, as inconsistencies can cause the model to misinterpret identical FLOCs as distinct entities, leading to inaccurate translations. FLMAT allows users to change FLOC codes, and these changes are saved in the background. After multiple uses of the tool, we will have a dataset of paired hierarchies and transformations that can be used as a rule base for FLOC code standardisation. We use a two-step approach: first, we apply a rule-based system to convert FLOC codes into their standardised forms. Next, we use NMT to work with trees that contain the standardised FLOC codes.

2.2 Training Data

The paired data produced by FLMAT serves as the training data for our model. The FLMAT output is organised into two lists of dictionaries—one for the source trees and one for the target trees. Corresponding positions in these lists represent pairs, forming each training instance. Each dictionary represents a hierarchy where the keys are FLOC codes (nodes). These hierarchies include attributes such as parent, work order count (WO count), and total cost. The FLOC code acts as the unique node label within each hierarchy, with the parent attribute defining the parent-child relationships. The work order count reflects the node's activity level, while total cost indicates the financial impact of the associated work orders.

To obtain preliminary results and evaluate the model architecture, we generated synthetic data resembling the output of FLMAT. A Python script was used to create 100 data pairs, allowing us to test various model configurations. The synthetic target trees were generated using a simple rule—removing leaf nodes with less than 40 work orders—providing an initial benchmark for model performance. Figure 3 illustrates an example of a synthetic data pair.

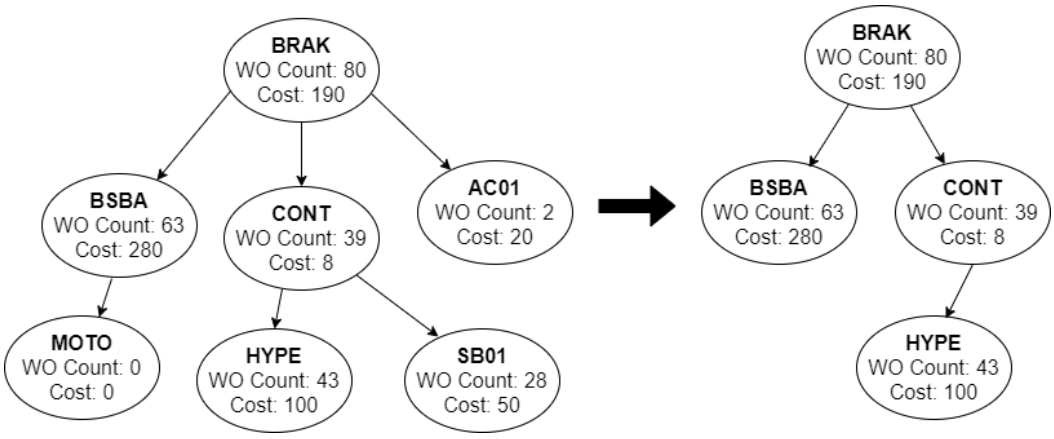

Figure 3 Synthetic data pair.

2.4 Models

Generating large volumes of annotated data using SMEs is time-consuming, so FLMAT is only used to create enough data for training the translation model in a semi-supervised setting. To compliment this, we implement a semi-supervised approach that leverages both paired and unpaired data to enhance model performance. This approach involves encoding both paired and unpaired data into latent representations, which capture essential features. Unpaired data is then decoded back to its original form to help the model learn patterns without explicit labels, while labelled data is used to guide direct translations into the target domain. This strategy improves

the model's performance and ability to generalise, even with limited paired data (Zhao et al., 2020).

Using the semi-supervised framework, we test two architectures for the translation task: a baseline LSTM model and a Tree-LSTM model. LSTM networks are well-suited for capturing sequential data and long-range dependencies, making them effective for translation tasks (Gupta & Kumar, 2021). In contrast, the Tree-LSTM model is specifically designed for tree-structured data and implemented following the methodology from Chen et al. (2018). By comparing these models, we aim to assess the relative effectiveness of LSTM and Tree-LSTM architectures for handling hierarchical data mapping.

3. Results and Discussion

3.1 Baseline Model Results

The LSTM-baseline model and the Tree-LSTM model were first trained without the semi-supervised approach to establish initial performance benchmarks. The training data, consisting of 100 pairs, was divided into a training set (80 pairs) and a validation set (20 pairs). Figure 4 shows the training and validation loss as well as the validation accuracy for the baseline model. From this, we can observe that the model was learning effectively, as indicated by the decreasing loss and increasing validation accuracy during training. After training, we tested the model on a test set (10 pairs).

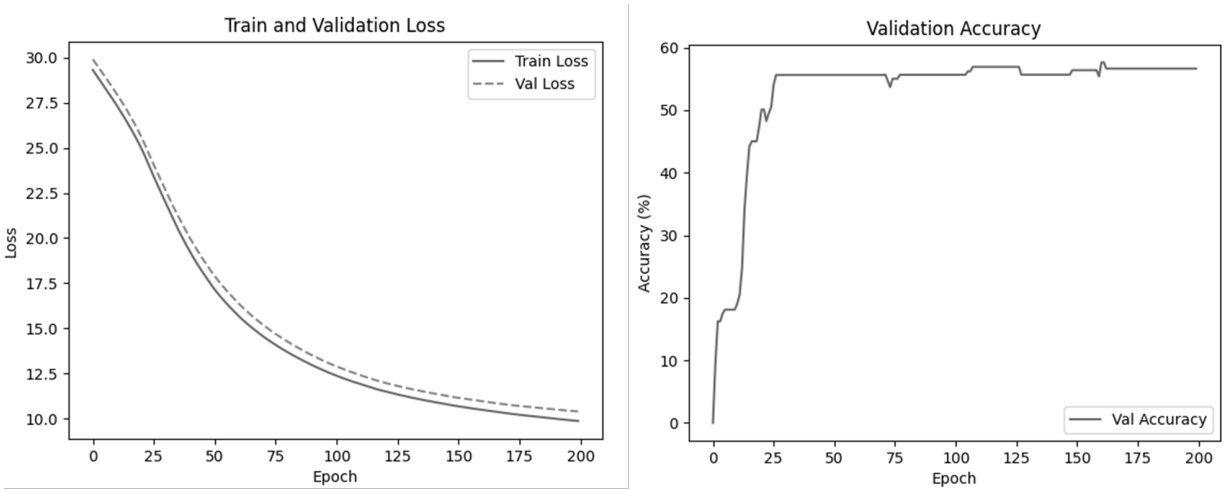

Figure 4 Loss and validation accuracy during training.

The model training and validation loss profile is as desired, but the final loss is still high. However, the validation accuracy of 60% is encouraging. The next step is to test a larger training set.

3.2 Tree-LSTM Model Results

The results from the tree-LSTM model did not match those from the baseline LSTM model. The tree-LSTM achieved an average accuracy of 30% on the test set. This outcome could be attributed to the complexity of the tree-LSTM model relative to the simplicity of the synthetic data (only a maximum of 10 nodes for any tree, see Figure 3). Tree-LSTMs are designed to capture intricate hierarchical relationships and dependencies within tree structures. However, if the patterns in the synthetic data are relatively simple, a more complex model might not

necessarily yield better performance. Instead, it can be overfit to noise or irrelevant features in the data, rather than learning the straightforward underlying patterns. This can cause the model to perform worse compared to simpler models that are better suited to the data's complexity.

The actual data used for the task will be more complex, which might better leverage the capabilities of the tree-LSTM model. To improve performance, we can also consider starting with more hyperparameter tuning and experimenting with different configurations to find the optimal setup for the model.

4. Conclusions and Future Work

This project has demonstrated that Neural Machine Translation (NMT) has the capability to translate hierarchical data effectively. Our baseline LSTM model performs better on simple hierarchical translations compared to the Tree-LSTM model, indicating that the simplicity of the data plays a significant role in model performance. From the experiments conducted, we can see that tree-to-tree translation can translate from a source tree to a target tree. Once applied to actual data, we aim to evaluate the effectiveness of automating the standardisation process and determine the scalability of our approach for mining data. To complete this project, we need to generate more annotated functional hierarchy data, test both the LSTM-baseline and Tree-LSTM models in a semi-supervised setting, and apply the trained models to hierarchies for different asset types to assess generalisation capability and scalability. Additionally, future work arising from this project includes exploring the use of tree transformers and incorporating other node attributes such as FLOC descriptions to improve model performance and accuracy. These steps, although beyond the current project's scope, are crucial for further advancements in the field.

5. Acknowledgements

I would like to express my sincere gratitude to my supervisors, Melinda Hodkiewicz and Tyler Bikaun, and my client mentor, Ronald Van Moere, for their expert guidance, feedback, and ongoing support throughout this project. Furthermore, I am grateful to the staff at the CEED office, Jeremy Leggoe and Kimberlie Hancock, for providing this incredible opportunity. Additionally, I acknowledge the sponsorship of this project by Rio Tinto through the AMplify project work, which made this endeavour possible.

6. References

Chen, X., Liu, C., & Song, D. (2018). Tree-to-tree neural networks for program translation. *Advances in neural information processing systems*, *31*.

Gupta, M., & Kumar, P. (2021). Robust neural language translation model formulation using Seq2seq approach. *Fusion: Practice and Applications*, *5*(2), 61-67.

Jafari, M., Parlikad, A., Robazetti-Concho, L., & Jafari, B. (2014). Review of asset hierarchy criticality assessment and risk analysis practices.

Langan, P. (2013). Finding the hidden value in asset hierarchy validation: make the effort now to save time and money down the line. *Plant Engineering*, *67*(9), 81-84.

Sriram, A., Jun, H., Satheesh, S., & Coates, A. (2017). Cold fusion: Training seq2seq models together with language models. *arXiv preprint arXiv:1708.06426*.

Zhao, T., Tang, X., Zhang, X., & Wang, S. (2020, October). Semi-supervised graph-to-graph translation. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (pp. 1863-1872).