Universität Zürich[UZH] ETH zürich

Automatic Re-Generation of Sentences To Different Readability Levels

Master Thesis for obtaining the academic degree
Master of Informatics: Computing and Economics
from the Faculty of Business, Economics and Informatics
Institute of Informatics
University of Zurich

Author:
Andrianos Michail
Marticulation Number: 20-745-931

Referent: Prof. Dr. Martin Volk

Supervisors (ETH): Jiaoda Li,
Prof. Dr. Mrinmaya Sachan

Institute of Informatics

Submission Date: August 8, 2023

Abstract

The task of text simplification is to reduce the linguistic complexity of a text in order to make it more accessible. Previous work on text simplification has primarily focused on either a single level of simplification or multiple levels of simplification, but always with the goal of making the text simpler. In this work, we explore a related task: re-generating sentences to produce equivalent text that targets an audience at a different readability level, whether that level is simpler or more advanced. We formulate the problem as a sequence to sequence task and explore different methods of using the pre-trained T5 encoder-decoder model to perform the task. In particular, we investigate the use of the hyperformer++ Mahabadi et al. [2021] architecture to solve the task, and propose and evaluate custom variants of the architecture designed to maximize positive transfer between different transformation pairs. According to automatic metrics, our custom variant of hyperformer++ is able to compete with strong baselines while only storing a small fraction of parameters compared to updating the entire language model.

Zusammenfassung

Die Aufgabe der Textvereinfachung besteht darin, die sprachliche Komplexität eines Textes zu reduzieren, um ihn leichter zugänglich zu machen. Bisherige Arbeiten zur Textvereinfachung haben sich hauptsächlich auf eine oder mehrere Vereinfachungsebenen konzentriert, aber immer mit dem Ziel, den Text zu vereinfachen. In dieser Arbeit untersuchen wir ein verwandtes Problem: die Neuerstellung von Sätzen, um einen äquivalenten Text zu erzeugen, der sich an ein Publikum mit einem anderen Lesbarkeitsniveau richtet, unabhängig davon, ob dieses Niveau einfacher oder fortgeschrittener ist. Wir formulieren das Problem als Sequence-to-Sequence-Aufgabe und untersuchen verschiedene Methoden, um das vortrainierte T5-Encoder-Decoder-Modell zur Lösung der Aufgabe einzusetzen. Insbesondere untersuchen wir die Verwendung der hyperformer++ Architektur Mahabadi et al. [2021] zur Lösung der Aufgabe, schlagen geeignete Varianten dieser Architektur vor und evaluieren diese, um den positiven Transfer zwischen verschiedenen Transformationspaaren zu maximieren. Automatisierte Metriken zeigen, dass unsere angepasste Variante von hyperformer++ in der Lage ist, mit starken Baselines zu konkurrieren und dabei nur einen kleinen Teil der Parameter des Sprachmodells zu verwenden.

Acknowledgement

I would first like to thank the supervisors Jiaoda Li and Mrinmaya Sachan for their continuous and extremely valuable supervision. I would like to note my appreciation for the exchange of ideas on the topic and the beneficial proofreading by my colleague Ahmet Yavuz Uluslu. Last but not least, I would like to take a moment to thank Simon Clematide who helped me acquire skills, knowledge and interest in this field which were necessary for this project and for the many more to come!

Contents

List of Figures

List of Tables

1 Introduction

1.1 Motivation

In the past few years, advancements in Natural Language Processing (NLP) have opened new doors. Large Language Models (LLMs) are now capable of leveraging minimal task examples to perform as well as, or even better than, models specifically fine-tuned for particular tasks [Ray, 2023]. This breakthrough has gained worldwide attention with the increased public access to LLMs. Over the past ten years, through LLMs or even smaller models, there has been a tremendous progresses [Liu et al., 2019; Radford et al., 2019] of performance in many tasks, even tasks which it would have been unimaginable to be performed by non human intelligence. Even if multiple LLM's models have became publicly accessible through user interfaces or API's the size of these models prohibits their use on consumer devices, and as such, research on smaller and more compact models remains crucial.

However, there are also always negative consequences of such scientific progress. There have been numerous reporting of rouges utilising these models to achieve malicious purposes for their own benefit in the expense of others. As researchers, we should always instead strive to provide positive contribution to the community and thus find helpful contributions through the available tools. One such field is the technology for education which has been progressing towards great contributions such as Language Model study assistants and tutors, correcting tools for teachers and even digital teachers. Our research falls within that field and envisions to help in the following scenario:

As humans, we spend a large part (if not all) of our lives learning, either through the formal education system or simply searching for answers to questions that arise based on all the stimuli we are exposed to. Finding answers to questions has become very easy since the world wide web because it is publicly affordable, but we often stumble upon content that is inappropriate for our level of understanding. Sometimes middle school students are looking for simple answers to questions when the Internet is full of scholars' opinions on the topic. An example of such a discussion is within

mathematics is when you search "what is expected value" you somehow end up to a definition using integrals with respect to probability measures. But the problem does not end there; educated adults also search for answers to questions that are typically phrased for children, such as how to tie a hair bun.

But how could this problem be solved? Imagine a system which automatically rewrites the same content you are reading but instead make the content suitable to your reading level, whether that is simpler or more complicated.

This thesis is a tiny step towards that system. We focus on using Language Models (LM's) to transform sentences across different readability levels. Our study specifically explores sentence transformation across three English proficiency levels: "advanced", "intermediate", and "elementary".

1.2 Task Formalization

The goal of our task is to transform a given sentence of any readability level into a sentence that conveys the same information and meaning, but at a different readability level. This level may be simpler or more advanced.

Readability Level

The readability level of a text indicates how easy or difficult a reader finds it to comprehend the text [Vajjala, 2022]. Another way of formalising readability level it is that it is a measure of the complexity of the text. Texts with a low readability level, within our study called 'advanced', tend to use complex words and intricate sentence structures. These characteristics can make them difficult to read without a good knowledge of the language's vocabulary and grammar. In contrast, a highly readable, within our study called 'elementary', text uses simpler words and shorter sentences, making it easier to be understood for more readers. Another common interpretation of readability level is the expected level of proficiency in the natural language (often correlated with years of education) that a reader would need to understand the text.

Examples

Before we dive into the system requirements, let's first try to understand the task we pose to the models ourselves. Let's look at a couple of sample sentences (from the dataset we will later introduce) of different readability levels in Table 1.

The readability assement task is the automatic classification of the readability level

ID	Readability	Sentence
1	Advanced	Hearing that subtle click is harder if theres a lot of noise around you.
2	Advanced	Some Americans have taken to keeping their wealth secret.
3	Advanced	To be honest, I have never seen what my neighbours look like.
4	Intermediate	The leaves are said to provide energy and have medicinal qualities.
5	Intermediate	Sleep deprivation used to be a sign that you were busy and very much in demand.
6	Intermediate	Hair sample testing is also pointless, the guide says.
7	Elementary	He spends his spare moments at work reading news articles and books.
8	Elementary	The shop is uniting people here, Kalisa Migendo, a 24-year-old student, says.
9	Elementary	They found that it was not his baby and not her baby.
10	Guess it [1]	I have seen octopuses on boats escape through bilge pumps.

Table 1: Samples of short sentences (out of context) of different readability levels.

of a text using non parallel training data for different levels. As with many natural language processing tasks, the labels assigned to texts are subjective, influenced by the reader's background and previous experience. Personally, I was surprised to see the third sentence of Table 1 labeled 'advanced'. But there's a bigger question: even with the inherent subjectivity of labeling, would we be able to perform this task ourselves?

Given a limited set of examples, our initial human strategy would likely involve identifying commonalities in vocabulary usage. Given enough samples, this strategy proves to be very effective. This is evidenced by the continued success of frequency-based classifiers, which remain a challenging benchmark in the field, even for advanced language models.

In Table 2, we present some of the pairs from the fine-tuning data used for our models. For a model to effectively transform text from level X to level Y, it must recognize the unique characteristics of level X and understand how to construct an equivalent text that fits level Y. It is important to note that the model does not have access to the surrounding context of a sentence. Therefore, it is unrealistic to expect it to predict replacements like it 'to the mirrors' during inference, as seen in the last example.

From Table 2 we can also observe the Flesch-Kincaid Grade Level (FKGL) readability metric (introduced later), where each value difference is equal to roughly one grade level. We can observe a great deal of inconsistency, as two of the advanced-elementary pairs have less than one grade level difference, while the first example

Parallel Pairs
Advanced - Hearing that subtle click is harder if theres a lot of noise around you. Elementary - It is harder to hear that click if theres a lot of noise around you. (Source - Target) FKGL Difference: 0.8
Advanced - You need a 15-minute distance and typical off-the-shelf drones have about that distance. Elementary - You need a 15-minute distance, and typical drones have about that distance. (Source - Target) FKGL Difference: 0.4
Advanced - This is, instead, approximated by peer support such as online discussion forums. Intermediate - Instead, they get peer support, such as online discussion forums. (Source - Target) FKGL Difference: 3.1
Advanced - You dont realize how safe Vienna is until you head abroad, said Hartlauer. Intermediate - You dont realize how safe Vienna is until you go abroad, said Hartlauer. (Source - Target) FKGL Difference: 0.0
Intermediate - To be honest, I have never seen what my neighbours look like. Elementary - I have never seen what my neighbours look like. (Source - Target) FKGL Difference: 2.3
Intermediate - You could put it on top of shopping malls. Elementary - You could put the mirrors on top of shopping malls. (Source - Target) FKGL Difference: -1.6

Table 2: Parallel short sentences of the OneStop dataset (out of context) of different readability pairs.

of advanced-intermediate calculates a three-year grade level difference between the parallel sets.

While the model learns based on patterns present in the training set, many training pairs involve single or double word replacements. However, there are also examples, especially in the advanced-to-elementary pair, where the underlying sentence structure changes, similar to the first example in the table. Therefore, we expect that a model trained on all available fine-tuning data (as opposed to just pairwise data) would perform more diversely.

Necessary Properties

In order for our sentence re-generation system to be viable and useful for consumer use, it should satisfy the following constraints:

1. Require only one set of language model parameters across all transformation tasks.
2. Preserve the original meaning while generating grammatically correct and fluent sentences.

Constraint Relaxation

During both the training and inference phases, the model will have access to information about the readability level of both the original and desired target text.

1.3 Hypotheses

In this research, we aim to use a contextual parameter generation approach [Platanios et al., 2018], specifically through the adapter generation variant as implemented in Hyperformer++ [Mahabadi et al., 2021]. Our goal is to address our task within a multitasking environment using an architecture designed to maximize the models ability to benefit from emergent patterns in the training across different transformation pairs. While exploring different configurations for this architecture, we examine the following hypotheses:

- Initializing the supplementary input representation (later referred to as task embeddings) with information about the readability level, as opposed to random initialization, leads to improved performance.
- Separating the supplementary inputs for the Encoder and Decoder components of the model facilitates greater transfer to the more difficult transformation pairs.
- Disabling parameterization of a meaningful supplementary input representation is expected to enhance model performance.

In addition to testing our proposed approach, we compare its performance to a wide range of baseline models. We hypothesize that the performance of the models will be ranked as follows:

- Models that share parameters between transformation pairs are expected to consistently perform better than their pairwise equivalents in the more difficult transformation pairs.
- Models that have dedicated parameters for a transformation pair are expected to perform better in smaller models.

1.4 Thesis Outline

This thesis progresses through a logical sequence of chapters, each focusing on one or more key topics related to our study. These topics are:

- **Background**: Provides a background on T5 and discusses related research in the area of parameter efficiency and its applicability to our study.
- **Datasets & Evaluation Metrics**: Details the data sets used and outlines the metrics used to evaluate the performance of our model.
- **Methodology**: Describes the architecture of our proposed model, any variations explored, and the results of the experiments that led to our proposed architecture.
- **Results**: Presents and discusses the results of our experiments compared to the baselines.
- **Limitations & Future Work**: Acknowledges the limitations of the present work, and outlines directions for future research, for example within a multilingual setting.
- **Conclusions**: Summarises our contributions within this study and concludes what we have learned.

2 Background

To provide a brief overview of the study of related tasks, there have been a plethora of studies on text simplification, including a series of Shared Tasks by PAN [Ermakova et al., 2023, 2022], where language models are explored for both extractive and generative approaches. Similarly, automatic readability assessment (the classification of the readability level of a text) has also seen its fair share of studies [Vajjala, 2022] using language models such as BERT [Deutsch et al., 2020], RoBERTa [Lee et al., 2021] and T5 [Lee and Lee, 2023]. However, the aforementioned tasks do not have a many-to-many sequence-to-sequence nature, so we need to get creative and design our own baselines inspired by a popular sequence-to-sequence task, machine translation. We formalize our choice of baselines in Chapter 5.

Kew and Ebling [2022] take a completely different approach to a task similar to ours, text simplification at different levels. This method uses FUDGE [Yang and Klein, 2021], a lightweight binary classifier for each target simplification level, to control the complexity of the generated text during inference. This is a very interesting alternative idea as within their approach, they can perform the multiple level simplification relying solely on off-the-shelf language models and a simple lightweight classifier, which is both practical and efficient.

To justify the idea behind our proposed multitask setting, we examine recent advances in the field. An extensive study in the NLP community that used a single model for transfer learning among different tasks is the introductory paper on the T5 architecture [Raffel et al., 2020]. The T5 model, an encoder-decoder-transformer architecture, treats each task as a sequence-to-sequence problem. This text-to-text framework provides a straightforward method for training a single model on a set of tasks using a uniform loss function and decoding procedure. The study also provided evidence that the T5 model can sometimes adequately perform some text-to-text tasks in a zero-shot setting. For our study, we chose the T5 model as the pre-trained model of interest, and all of our solutions use the English pre-trained T5.

One particular finding in the T5 study caught our interest, and we would like to explore its implications in more depth. The authors found that the traditional pro-

cess of unsupervised pre-training followed by fine-tuning on single or a few related tasks, where all model parameters are updated, outperforms any form of mass multitask learning, regardless of multiple mixing ratio settings. In contrast, Mahabadi et al. [2021] find in their results that multitask learning is beneficial to single task performance.

In another study, the authors of MUPPET [Aghajanyan et al., 2021] found that incorporating a multi-task "pre-fine-tuning" phase consistently improved RoBERTa's performance on various tasks, especially those with sparse training data. The keys to the success of this pre-tuning process were the use of a large number of tasks (15+), combined with heterogeneous batching[1] and a scaled loss function across different task heads. These conflicting results leave us with an open question: "What characteristics of tasks and multitask settings cause multitask to improve the performance of single tasks?"

In a different line of research focused on parameter efficiency, Li and Liang [2021] employ a technique they call "prefix tuning". This method essentially adds a small set of parameters (they study 2% or 0.1%) to each layer. During fine-tuning, these parameters are updated while the remaining model parameters remain frozen, essentially acting as a continuous prompt. This technique is competitive with full model fine-tuning, while remaining parameter efficient, since only these parameters have to be switched between tasks.

In a simpler parameter-efficient approach, previous studies such as Pfeiffer et al. [2020] use adapters [Houlsby et al., 2019] – a small bottleneck neural network attached to the end of each transformer block – as an alternative to updating the entire set of model parameters to learn a new task.

A subsequent study [Ansell et al., 2021] used a shared hyperparameter generation network to generate the weights of the adapters for each language. This network uses representations of sparse typological vectors consisting of 289 binary linguistic features as inputs for the corresponding language. The hyperparameter generation network is trained using a masked language modeling objective over different languages, allowing parameters to be shared between languages with similar linguistic features. This approach serves as the primary inspiration for our research. We build on this concept by investigating whether a simple representation of the readability level of a source and target sentence, given as input to a hyperparameter generation network, can generate weights for adapters to effectively control the readability level of the generated sentences.

[1] batches that contain equal samples from all fine-tuning tasks

3 Datasets & Evaluation Metrics

In this chapter, we first analyze the available datasets for our task, including those that are not usable for our specific case. We then introduce traditional readability features used to observe the generation of our models and apply this analysis to our datasets. Finally, we explain how we evaluate the performance of our model and how we utilize external language models to ensure fluency and preservation of meaning within the generated sentences.

3.1 OneStop English

The OneStop English dataset is a publicly accessible text readability dataset containing texts at three distinct readability levels: advanced, intermediate, and elementary. The creation of this corpus involved sourcing article and was later rewritten by teachers to suit the three readability levels [Vajjala and Lučić, 2018].

Additionally, the dataset contains aligned sentences extracted through a one-to-all comparison of sentences between the aligned texts. The comparison was done using cosine similarity, and sentences falling in between 0.7 and 0.95 were extracted as aligned sentences [Vajjala and Lučić, 2018]. Table 3 shows the quantity of available sentence pairs[1] across these three levels:

Level	Level	Train Set	Validation Set	Test Set
Advanced	Intermediate	1710	222	222
Advanced	Elementary	1757	210	199
Intermediate	Elementary	1328	167	179

Table 3: Available aligned parallel sentences in the OneStop English Dataset. Note: data is bidirectional

[1]We later found out there were 1000 additional missing pairs from our advanced-intermediate training set, likely due to a corrupted file. This accident inadvertently helped to balance our dataset and therefore we decided to keep this reduced set of pairs.

3.2 Newsela

The Newsela dataset is a collection of 1130 news articles that has been rewritten to four different grade levels by editors at Newsela[2], a company that produces reading materials for pre-college classroom use [Xu et al., 2015]. These reading levels are denoted as follows: 'Simple0' indicates the original text, 'Simple1' refers to the most complex rewritten version, while 'Simple4' represents the most simplified version of the text. For more information and examples of the Newsela dataset please refer to [Xu et al., 2015].

Further work by Jiang et al. [2020] created a manual alignment of sentences for different levels of simplification. From these annotations, we extracted (using code from Kew and Ebling [2022]) aligned sentences that spanned across three specific levels, to make our experiments similar to those conducted on the onestop English. In choosing these three levels—Simple0, Simple1, and Simple3—our primary consideration was the size of the resulting dataset. Table 4 shows the resulting dataset. However, no matter what we experimented with this dataset, the models were unable to train, hence we do not report results from Newsela in this thesis. This might be due to the very limited amount of samples or possibly due to the sentences being already fairly simple as also shown through the readability metrics in Table 7.

Level	Level	Train Set	Validation Set	Test Set
Simple0	Simple1	951	120	256
Simple0	Simple3	330	31	98
Simple1	Simple3	554	65	175

Table 4: Number of available aligned sentences for the selected simplification levels. Note: data is bidirectional

3.3 Non-Parallel Readability Datasets

There exists datasets with non-parallel data that are valuable in the field, but not directly applicable to our specific study. One such dataset is the WeeBit dataset [Vajjala and Meurers, 2012]. This dataset consists of articles from two sources: the WeeklyReader[2], which contains educational articles aimed at Years 2, 3, 4 and Senior, and the BBC-Bitesize[2], which contains articles divided into four grade levels

[2] https://newsela.com

[2] http://www.weeklyreader.com

[2] http://www.bbc.co.uk/bitesize

(KS1, KS2, KS3, GCSE). Together, these two sources provide a diverse readability dataset. However, since the dataset is not parallel neither by text nor by sentences across levels, it falls outside the scope of our study.

3.4 Evaluation/Observation Metrics: Traditional Readability Features

In this study, we use a set of traditional statistical readability features taken from the Martinc et al. [2021] study. The simplest of these is Average Sentence Length (ASL), but there are others that take into account factors such as word length and word difficulty. These features are defined in the survey as follows:

The Gunning Fog Index [Gunning, 1952] (GFI) estimates the years of formal education a person needs to understand the text on the first reading. It is calculated with the following expression:

$$GFI = 0.4 \left(\frac{\text{totalWords}}{\text{totalSentences}} + 100 \frac{\text{longWords}}{\text{totalSentences}} \right) \tag{3.1}$$

where *longWords* are words longer than 7 characters. Higher values of the index indicate lower readability.

Flesch Reading Ease (FRE) [Kincaid, 1975] assigns higher values to more readable texts. It is calculated in the following way:

$$FRE = 206.835 - 1.015 \left(\frac{\text{totalWords}}{\text{totalSentences}} \right) - 84.6 \left(\frac{\text{totalSyllables}}{\text{totalWords}} \right) \tag{3.2}$$

The values returned by the Flesch-Kincaid Grade Level (FKGL) [Kincaid, 1975] correspond to the number of years of education generally required to understand the text for which the formula was calculated. The validity of this metric has been critised in a recent study. The formula is defined as follows:

$$FKGL = 0.39 \left(\frac{\text{totalWords}}{\text{totalSentences}} \right) + 11.8 \left(\frac{\text{totalSyllables}}{\text{totalWords}} \right) - 15.59 \tag{3.3}$$

However, FKGL has its limitations. Tanprasert and Kauchak [2021] have shown experimentally that FKGL can be easily increased by very basic postprocessing techniques. As also stated by the authors, it remains a valuable form of understanding the behaviour of generative models.

Another readability formula that returns values corresponding to the years of education required to understand the text is the Automated Readability Index (ARI) [Smith and Senter, 1967]:

$$ARI = 4.71\left(\frac{\text{totalCharacters}}{\text{totalWords}}\right) + 0.5\left(\frac{\text{totalWords}}{\text{totalSentences}}\right) - 21.43 \tag{3.4}$$

In our study, we calculate[3] the average readability features for the validation set every 1000 steps during our model's fine-tuning phase. These averages help indicate the model's generative direction. While these metrics are relatively simplistic and solely based on the lexical level, we supplement them with reference-based evaluation metrics to provide a more comprehensive overview.

3.5 Readability Features Analysis of Datasets

To further understand the nuances of our parallel sentences, we perform an average readability metric analysis within the parallel sentences of the onestop and newsela datasets. The results of this analysis are presented separately for each split: Table 5 for the training set, Table 6 for the validation set, and Table 7 for the test set.

Level - Level	**GFI**		**FRE**		**FKGL**		**ARI**	
	Left	Right	Left	Right	Left	Right	Left	Right
OneStop Parallel Sentences								
Advanced - Intermediate	14.22	13.21	55.74	59.35	11.72	10.77	14.48	13.28
Advanced - Elementary	11.96	10.53	64.22	69.61	9.65	8.36	11.86	10.39
Intermediate - Elementary	12.06	10.99	62.99	67.10	9.80	8.82	12.00	10.83
Newsela Parallel Sentences								
Level0 - Level1	10.93	10.79	64.12	64.78	8.55	8.46	10.79	10.70
Level0 - Level3	8.88	7.86	71.60	76.64	6.61	5.77	8.21	7.34
Level1 - Level3	9.07	8.12	69.95	74.56	6.86	6.03	8.65	7.83

Table 5: Average readability features across parallel sentences in the training set. Lower scores in GFI, FKGL, and ARI imply greater readability, whereas for FRE, a higher score indicates more readable text.

[3]using the textstat package https://pypi.org/project/textstat/

Level - Level	GFI		FRE		FKGL		ARI	
	Left	Right	Left	Right	Left	Right	Left	Right
OneStop Parallel Sentences								
Advanced - Intermediate	14.39	13.27	55.11	59.15	11.94	10.91	14.68	13.47
Advanced - Elementary	12.93	11.33	61.26	68.04	10.45	8.89	12.89	11.00
Intermediate - Elementary	11.58	10.60	65.82	69.91	9.20	8.32	11.21	10.30
Newsela Parallel Sentences								
Level0 - Level1	11.13	11.00	64.03	65.32	8.79	8.68	11.24	11.11
Level0 - Level3	8.40	7.06	71.40	78.21	6.32	5.32	7.89	7.15
Level1 - Level3	9.91	7.65	65.06	75.50	7.61	5.83	9.33	7.65

Table 6: Average readability features across parallel sentences in the validation set. Lower scores in GFI, FKGL, and ARI imply greater readability, whereas for FRE, a higher score indicates more readable text.

By carefully observing the average readability features within the OneStop English dataset in Table 7, we can find what each value represents in an American school system, and by doing so we find minor disagreements between the different metrics. This partial disagreement is to be expected since each metric measures readability through a different formula.

For the "elementary" sentences, the ARI metric suggests they are appropriate for fifth graders, while other metrics suggest they would be suitable for eighth to ninth graders.

For the next level, "intermediate", the sentences are deemed appropriate for seventh graders according to the ARI. In comparison, other metrics rate them as suitable for tenth graders.

For our most difficult level, "advanced", ARI suggests they these sentences are suitable for ninth graders, while other metrics suggest they would be appropriate for students between eleventh and twelfth grade.

Upon examining the readability features of the dataset in more details, there are a few more key observations to be made:

- The readability features of the Newsela parallel sentences are generally better, indicating higher readability. This could be due to inherent differences in the complexity and structure of the texts in these different resources.

Level - Level	GFI		FRE		FKGL		ARI	
	Left	Right	Left	Right	Left	Right	Left	Right
OneStop Parallel Sentences								
Advanced - Intermediate	13.98	12.84	56.20	60.74	11.55	10.54	14.06	12.78
Advanced - Elementary	12.27	10.53	62.93	70.38	9.97	8.28	12.41	10.46
Intermediate - Elementary	12.03	10.74	63.15	68.42	9.69	8.52	12.01	10.55
Newsela Parallel Sentences								
Level0 - Level1	11.67	11.37	62.04	63.68	9.14	8.87	11.44	11.21
Level0 - Level3	8.37	7.64	72.83	77.52	6.30	5.45	7.98	7.08
Level1 - Level3	8.58	8.00	72.20	75.43	6.43	5.84	8.37	7.76

Table 7: Average readability features across parallel sentences in the test set. Lower scores in GFI, FKGL, and ARI imply greater readability, whereas for FRE, a higher score indicates more readable text.

- Minor discrepancies in the average features exist across the train, validation, and test sets, but these are negligible.
- Within each pair, the features are consistent with the associated readability level.

However, there are inconsistencies when comparing the same readability levels across different sentence pairs in both datasets. Take, for example, the average Flesch-Kincaid Grade Level (FKGL) of the 'advanced' sentences in the 'advanced-intermediate' pair, which is 11.72. Surprisingly, this average is much lower, at 9.65 (almost a two school grades drop), in the advanced-elementary pair. This discrepancy is significant enough to cause the 'advanced' sentences in the 'advanced-elementary' pair to be considered more readable than the 'intermediate' sentences in the 'advanced-intermediate' pair, according to these readability features. This inconsistency suggests that these readability features may not fully capture the quality and behavior of the generative models.

3.6 Evaluation Metrics: Reference-Based

Reference-based evaluation metrics for generative models are used to evaluate the quality of the generated output by comparing it to a reference text or set of reference texts. These metrics quantify the similarity between the generated output and the

reference output(s), where a higher score indicates a higher similarity and therefore better performance of the generative model. As one might anticipate, automatic reference-based evaluation has faced criticism for not adequately capturing all improvements or subtleties of generative models. Recent studies highlight SARI as the leading metric for evaluating text simplification. The SARI[4] score is an average of F1 scores based on three operations relative to the reference text: added n-grams, kept n-grams, and deleted n-grams. In addition to its popularity, a recent study experimentally demonstrated that minor changes to the text have a negligible effect on the metric [Tanprasert and Kauchak, 2021]. These characteristics, along with the inconsistencies in the average readability features within our dataset[5], led us to choose SARI as our primary metric.

3.7 Evaluation Metrics: Language Models Based Quality Assurance

Much of the recent literature has used BERTScore Zhang et al. [2019], a scoring system utilising BERT [Kenton and Toutanova, 2019] to evaluate the quality of text produced by generative models. Inspired by this recent trend, within this study we we utilise inference Encoder Transformer models to ensure the quality of the text generated through the methods we are about to introduce fluency and meaning preservation checks

Fluency

In the recent research on paraphrase generation [Krishna et al., 2020], the researchers used a Large RoBERTa model, fine-tuned using the CoLA corpus [Warstadt et al., 2019], to check the fluency of the generated sentences. We consider this methodology appropriate and use the same model[6] to evaluate the fluency of our own sentences generation. To ensure that the models adhere to the generation of fluent sentences, we observe the percentage of generated sentences that are classified as fluent throughout the learning curve. The results of the fluency test are relatively stable, consistently classifying over 95% of the generated sentences to be fluent.

[4]The acronym stands for System output Against Reference and against the Input sentence.

[5]Note that there is a consistent 0.27 ± 0.03 SARI for the basic baseline of copying the input into predictions which is often the starting behaviour of models.

[6]accessible at huggingface

Meaning Preservation

Preserving the inherent meaning and information of the original sentence is a critical factor when re-generating sentences. In natural language, even the slightest alteration, whether it is the addition, deletion, or modification of a single word or conjunction, can drastically change and potentially invert the intended message of the original sentence. Therefore, a methodology that can understand and compare the meaning of the generated sentence against the input sentence is crucial. To address this, we fine-tune an Encoder-Transformer model (arbitrarily choosing MUPPET RoBERTa [Aghajanyan et al., 2021] for its demonstrated limited sample generalizability) to validate that the input and predicted sentences are accurate paraphrases (True Paraphrases).

While there are pre-existing models fine-tuned for this purpose, our project requires a model that is ideally suited to our specific data. Since the parallel sentences derived from the onestop dataset often have minimal differences due to the high cosine similarity alignment methodology, we aim to train our model to refrain from classifying two sentences as paraphrases based solely on their lexical proximity. To accomplish this, we combine sentence pairs from the onestop dataset [Vajjala and Lučić, 2018] with adversarial paraphrase sentence pairs from Nighojkar and Licato [2021] to create a comprehensive fine-tuning set. The adversarial paraphrase sentence positive pairs are characterized by their lexical and syntactic divergence, as indicated by low BLEURT scores. Further details about the selected fine-tuning data can be found in Table 8.

Data Source	Sentence Pairs	Paraphrase Label	Sourcing Method
OneStop	1000	True	Balanced Sample
Adversarial	2433	True	Human Written
Adversarial	1313	False	Human Written
Adversarial	2000	False	T5 Generated
Adversarial	2000	True	T5 Generated
Total Samples	5433T/3313F	True/False	Concatenation of above

Table 8: Paraphrase model fine-tuning dataset

We fine-tune our model for a maximum of 10 epochs, following the standard fine-tuning parameters of RoBERTa [Liu et al., 2019]. After carrying out three experimental iterations of fine-tuning, we found that extending it beyond 2 epochs did not lead to any noticeable improvement in validation performance or loss. Consequently, we decide to keep the checkpoint reached at the end of the second epoch.

The performance of our model is compared to RoBERTa fine-tuned on the adversarial dataset [Nighojkar and Licato, 2021] as a paraphrase detector, across two different datasets. When applying it to the onestop parallel sentences (where the label always indicates true paraphrases), our model shows a 1% (total 96%) increase in accuracy. Similarly, when applied to the test set of the adversarial dataset [Nighojkar and Licato, 2021], our model exhibits a 0.7% (total 77%) improvement in accuracy. Hence, we conclude that our model is adequate to ensure our generation models maintain the meaning of the original text.

4 Methodology

In this chapter, we first introduce hyperformer++ (first on a high level and then formally), the core architecture underlying our proposed approach. We then present and explain the idea behind our architectural modifications and provide experiments to demonstrate their effectiveness.

4.1 Hyperformer++

Hyperformer++ is an architecture that allows parameter-efficient multi-task training of the T5 Encoder Decoder model as introduced in past work [Mahabadi et al., 2021]. There are a few key components we need to define sequentially to explain the architecture:

Adapters

Adapters are small, trainable bottleneck neural networks attached to the end of each frozen transformer block. They serve as an alternative to updating the entire set of model parameters to learn a new task [Houlsby et al., 2019].

Hyperparameter Generation Networks

A hyperparameter generation network, or hypernetwork, is a compact neural network that generates the weight of another network [Ha et al., 2017]. Its primary function is to generate the weights and biases of another part (or whole) of a larger neural network, based on (sometimes) arbitrary inputs. The hypernetwork can be trained through the backpropagation signal during the standard fine-tuning process of the larger model that utilizes these weights and biases.

Adapter Generation in Hyperformer++

Hyperformer++ places an uninitialized adapter block at the end of each transformer layer. During the fine tuning and inference phases, the hypernetwork is given parameters representing the task, the *layer id*, and the *adapter position*[1] as a concate-

[1]This categorization of parameters signals to the hypernetwork whether the adapter parameters

nated input representation. The hypernetwork then uses these inputs to compute a representation that is passed through linear layers to determine and instantiate all parameters of the adapter module for the corresponding layer and block. This process is also illustrated in a diagram in Figure 1.

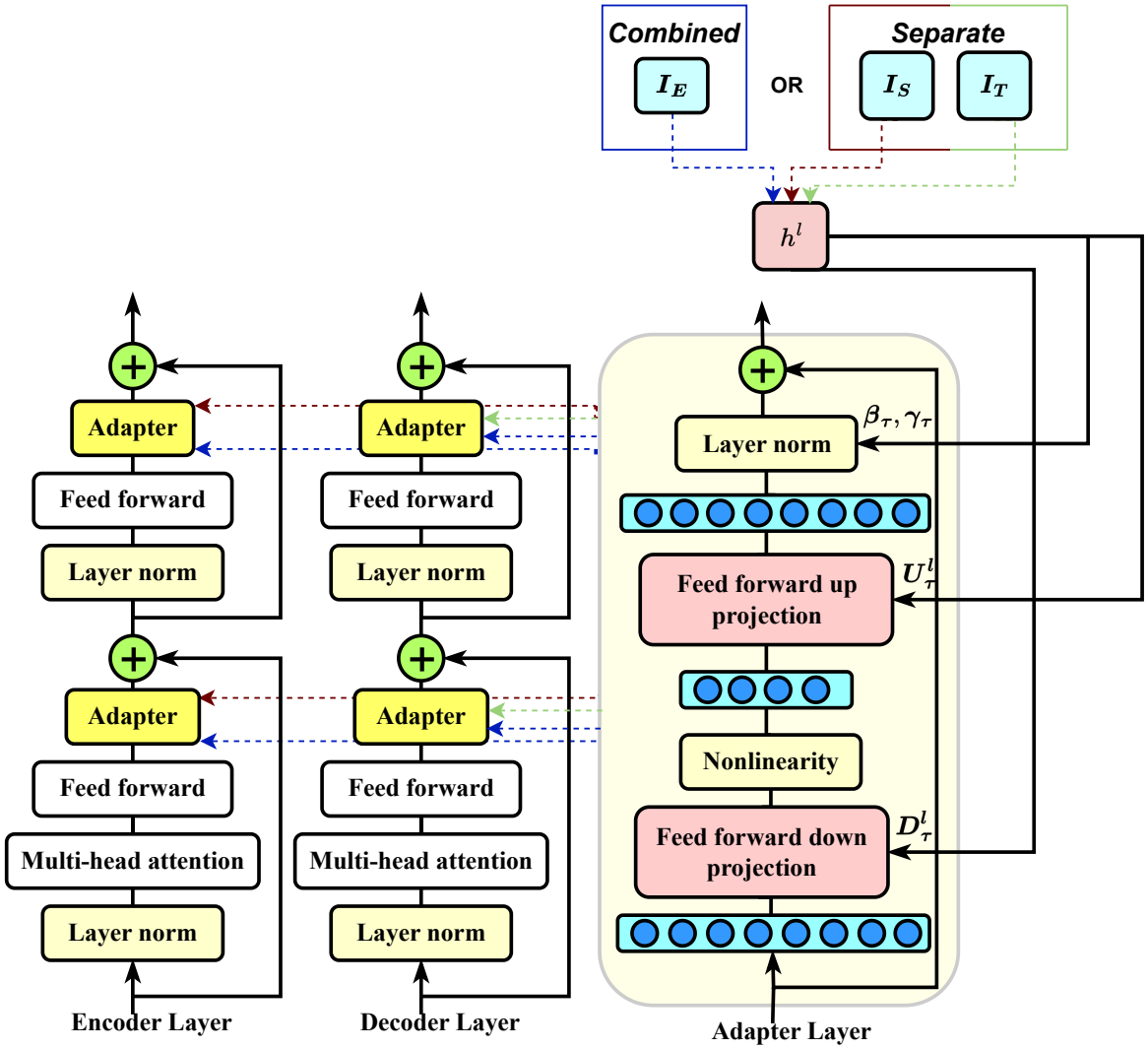

Figure 1: The hyperformer++ architecture variants. The dotted blue lines represent the original (also called combined) variant which generates both the Encoder and Decoder adapters from the same task representation. Within the separate variant, the dark red dotted line represents the source readability which is used for the generation of the adapters of the Encoder Layers, whilst the light green dotted line represents the target readability which is used for the generation of the adapters of the Decoder Layers.

to be generated will be used at the end of the feedforward layer or at the end of the attention layer.

Hyperformer++ formalisation

To formally introduce hyperformer++ and our variants within our task, we use the same notation and closely follow the original paper [Mahabadi et al., 2021], which first introduces the general hyperformer architecture and then hyperformer++. Consider a multi-task learning problem[2], where we are given the data from a set of transformation pairs $\{\mathcal{D}_\tau\}_{\tau=1}^{T}$, where T is the total number of transformations and $\mathcal{D}_\tau = \{(\mathbf{x}_\tau^\mathbf{i}, y_\tau^i)\}_{i=1}^{N_\tau}$ shows the training data for τ-th task with N_τ samples. We assume we are also given a large-scale pretrained language model $f_\theta(.)$ parameterized by θ that computes the output for input $\mathbf{x}_\tau^\mathbf{i}$. Standard multi-task fine-tuning minimizes the following loss on the training set:

$$\mathcal{L}(\theta, \{\mathcal{D}_\tau\}_{\tau=1}^{T}) = \sum_{\tau=1}^{T} \sum_{(\mathbf{x}_\tau^\mathbf{i}, y_\tau^i) \in \mathcal{D}_\tau} w_\tau l\left(f_\theta(\mathbf{x}_\tau^\mathbf{i}), y_\tau^i\right).$$

where l is typically the cross-entropy loss, and w_τ shows the sampling weight for τ-th task. The goal of hyperformer is to finetune the pretrained model in a multi-task learning setup efficiently, while allowing sharing information across tasks and at the same time, enabling the model to adapt to each individual task.

The key idea of hyperformer++, depicted in Figure 1, is to learn a parametric task embedding $\{\mathbf{I}_\tau\}_{\tau=1}^{T}$ for each task, and then feed these task embeddings to hypernetworks parameterized by ν that generate the task-specific adapter layers [Houlsby et al., 2019]. The authors insert adapter modules within the layers of a pretrained model, making the final model of $\mathcal{X}_\nu(\mathbf{x}_\tau^\mathbf{i}, \theta, \mathbf{I}_\tau)$ parameterized by ν that computes the output for input $\mathbf{x}_\tau^\mathbf{i}$. During training, hypernetwork parameters are trained ν, task embeddings $\{\mathbf{I}_\tau\}_{\tau=1}^{T}$, and layer normalizations in $f_\theta(.)$, while the rest of the pretrained model parameters θ are fixed:

$$\mathcal{L}(\nu, \{\mathbf{I}_\tau\}_{i=1}^{T}, \{\mathcal{D}_\tau\}_{\tau=1}^{T}) = \sum_{\tau=1}^{T} \sum_{(\mathbf{x}_\tau^\mathbf{i}, y_\tau^i) \in \mathcal{D}_\tau} w_\tau l\left(\mathcal{X}_\nu(\mathbf{x}_\tau^\mathbf{i}, \theta, \mathbf{I}_\tau), y_\tau^i\right),$$

The hypernetworks capture the shared information across transformation pairs in a multi-task learning model enabling positive transfer between related pairs, while adapters are reducing negative interference, encapsulating task-specific information.

We consider simple linear layers as hypernetworks that are functions of input task embeddings $\mathbf{I}_\tau$. As part of plain hyperformer, the authors introduce these hypernetworks in each layer of the transformer. They define hypernetwork $h_A^l(.)$ that

[2]In our case, each task is a transformation pair

generates task conditional adapter weights ($\mathbf{U}^{\mathbf{l}}_{\tau}$, $\mathbf{D}^{\mathbf{l}}_{\tau}$):

$$(\mathbf{U}^{\mathbf{l}}_{\tau}, \mathbf{D}^{\mathbf{l}}_{\tau}) := h^{l}_{A}(\mathbf{I}_{\tau}) = \left(\mathbf{W}^{\mathbf{U}^{\mathbf{l}}}, \mathbf{W}^{\mathbf{D}^{\mathbf{l}}}\right)\mathbf{I}_{\tau},$$

where $\mathbf{W}^{\mathbf{U}^{\mathbf{l}}} \in \Re^{(d\times h)\times t}$ and $\mathbf{W}^{\mathbf{D}^{\mathbf{l}}} \in \Re^{(h\times d)\times t}$ are the respective hypernetwork parameters.

They additionally define the hypernetwork $h^{l}_{LN}(.)$ that computes the layer normalization parameters:

$$(\gamma^{\mathbf{l}}_{\tau}, \beta^{\mathbf{l}}_{\tau}) := h^{l}_{LN}(\mathbf{I}_{\tau}) = \left(\mathbf{W}^{\gamma^{\mathbf{l}}}, \mathbf{W}^{\beta^{\mathbf{l}}}\right)\mathbf{I}_{\tau},$$

where $\mathbf{W}^{\gamma^{\mathbf{l}}} \in \Re^{h\times t}$ and $\mathbf{W}^{\beta^{\mathbf{l}}} \in \Re^{h\times t}$.

Having a separate hypernetwork for each transformer layer introduces a lot of parameters and the authors propose to share hyperparameters across all layers to reach the hyperformer++ architecture. Reapplying the same hypernetwork across all layers introduces weight sharing across target parameters, which may not be diserable. Therefore, to allow for a flexible parameterization of task conditional adapters/layer normalization, for a transformer of L layers, the authors introduce a set of *layer id* embeddings $\mathcal{I} = \{\mathbf{l_i}\}^{L}_{i=1}$, and *adapter position* embeddings $\mathcal{P} = \{\mathbf{p_j}\}^{2}_{j=1}$, which specify the position of adapter layers in each transformer block (after the attention layer or feed-forward layer), which are used as additional inputs to the hypernetworks. For simplicity, we consider $\mathbf{l_i} \in \Re^{t}$, $\mathbf{p_j} \in \Re^{t}$, and $\mathbf{z}_{\tau} \in \Re^{t}$. This is followed by a concatenation of $(\mathbf{z}_{\tau}, \mathbf{l_i}, \mathbf{p_j})$ to a similar task projector network h'_{I} as shown before:

$$\mathbf{I}_{\tau} = h'_{I}(\mathbf{z}_{\tau}, \mathbf{l_i}, \mathbf{p_j}),$$

which is then followed by a shared layer normalization to compute final task embeddings $\mathbf{I}_{\tau} \in \Re^{t}$ to the hypernetwork. This way, the hypernetwork is able to produce distinct weights for each task, adapter position, and layer of a transformer. Furthermore, layer id and adapter position embeddings are parameters that are learned via back-propagation, allowing the authors to train the whole model end-to-end conveniently.

In later Section 4.2 when we introduce our separate architectural variant, what we essentially do is to train two separate sets of task, *layer id* $\mathcal{I} = \{\mathbf{l_i}\}^{L}_{i=1}$, and *adapter position* $\mathcal{P} = \{\mathbf{p_j}\}^{2}_{j=1}$ embeddings, and projector networks, one that is given as input to the hypernetwork during the generation of adapters for the encoder layers and

one during the generation of adapters for the decoder layers.

Generation Procedure

The generation procedure plays a crucial role in the performance of generative tasks. To maintain consistency and comparability, we use a uniform procedure across all of our experiments. Specifically, we adopt the standard procedure for generation tasks as used in Mahabadi et al. [2021], which has two primary components: a $topk = 50$ sampling method and a four-beam search.

The $topk = 50$ is a parameter used in the sampling method that determines the pool of words from which the next word is selected. This means that at each step during generation, the model considers only the top 50 words with the highest probabilities and selects the next word from this pool. This approach helps to promote diversity in the generated output while still limiting the selection to relatively likely options.

Beam search is a search algorithm used in many natural language processing tasks to improve the quality of the output. The number of "beams" refers to the number of alternative sequences that the algorithm maintains at each step. In our case, we maintain four beams, or four alternative sequences. This allows the model to explore multiple high-probability paths through the search space, increasing the likelihood of producing higher quality output.

4.2 Hyperformer++ Architecture Variations

Architectural Variation Assessment Experiments Setup

While hyperformer++ provides a solid foundation for multi-task transfer learning, in this chapter we investigate modifications aimed at maximizing the suitability of this architecture for performing the task. To this end, we compare the performance on the validation set over 65,000 fine-tuning steps (homogeneous batches each consisting of 32 sentences) in terms of SARI (primary metric), loss, and the average readability features of the predictions. Our exact hyperparameter setup and learning curves that do not provide interesting new insights are placed in the Appendix A.

Ordinal Readability Initialization

Hyperformer++ initializes the task embedding (the input representing the task E) with 64 numbers sampled from a standard normal distribution. We propose initialising this embedding with a simple ordinal representation of the readability transformation pair. The motivation behind this initialization is to aid the model to identify the ordinal nature of the readability level. We formalise our proposed representation in the following way:

Let $N(64)$ be the total dimension of the task embedding, and let $R(r)$ represent the function that maps readability levels to values, defined as follows:

$$R(r) = \begin{cases} \frac{N}{3}(21) & \text{if } r = \text{Advanced} \\ \frac{N}{4}(16) & \text{if } r = \text{Intermediate} \\ \frac{N}{8}(8) & \text{if } r = \text{Elementary} \end{cases}$$

Given a source readability level src_readability and a target readability level tgt_readability, we map these levels to scores s and t using (r):

$$s = R(\text{src\_readability}), \quad t = R(\text{tgt\_readability})$$

The source readability vector S and the target readability vector T are then independently defined as:

$$S[i] = 1 \text{ if } i \leq s \text{ else } 0 \quad \text{for } i = 1, 2, ..., \frac{N}{2}$$

$$T[i] = 1 \text{ if } i \leq s \text{ else } 0 \quad \text{for } i = 1, 2, ..., \frac{N}{2}$$

Finally, the resulting task embedding vector E is formed by concatenating S and T:

$$E = [S, T]$$

Here, the task embedding vector E is a binary vector of dimension N which uniquely represents the transformation task from source to target readability level.

Example: Advanced to Elementary

$$s = 21, t = 8$$

$$S[i] = [1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

$$T[i] = [1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]$$

$$E = [S, T]$$

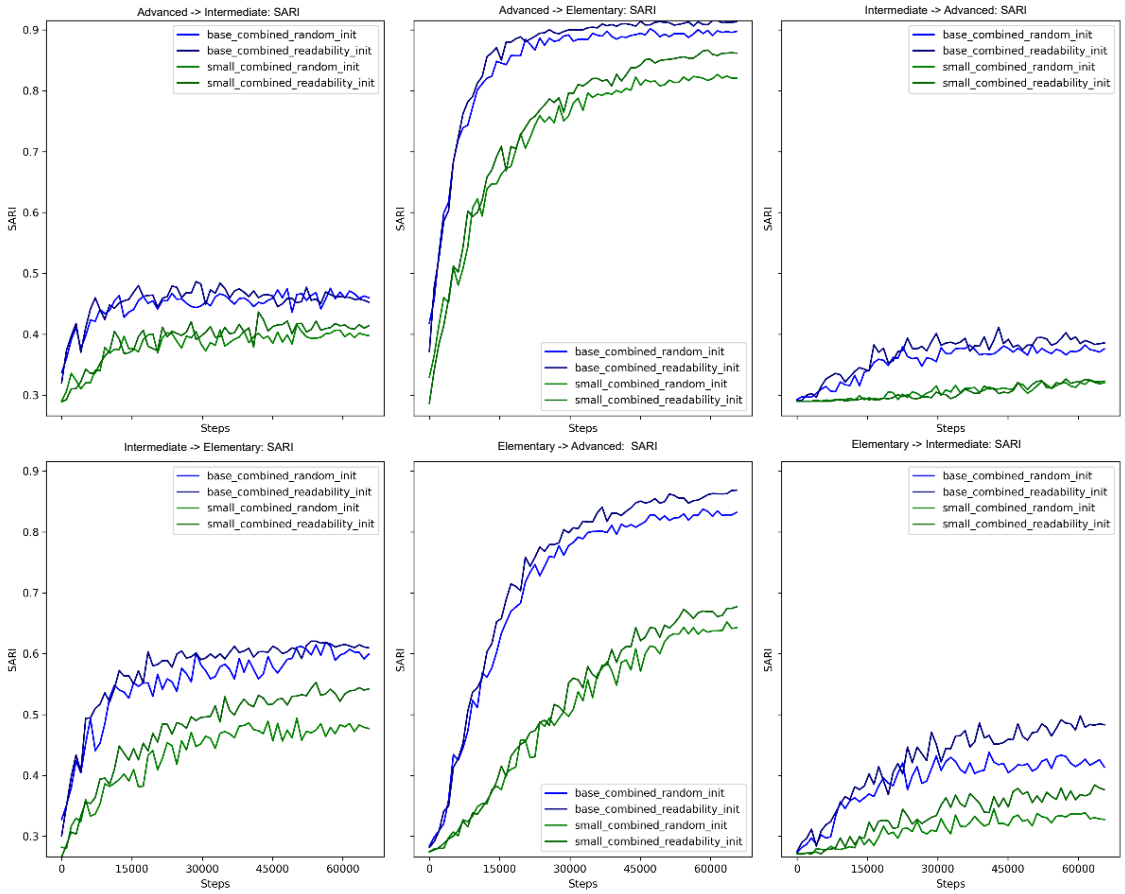

Figure 2: Learning curve of SARI of the random/readability initialization in the combined architecture

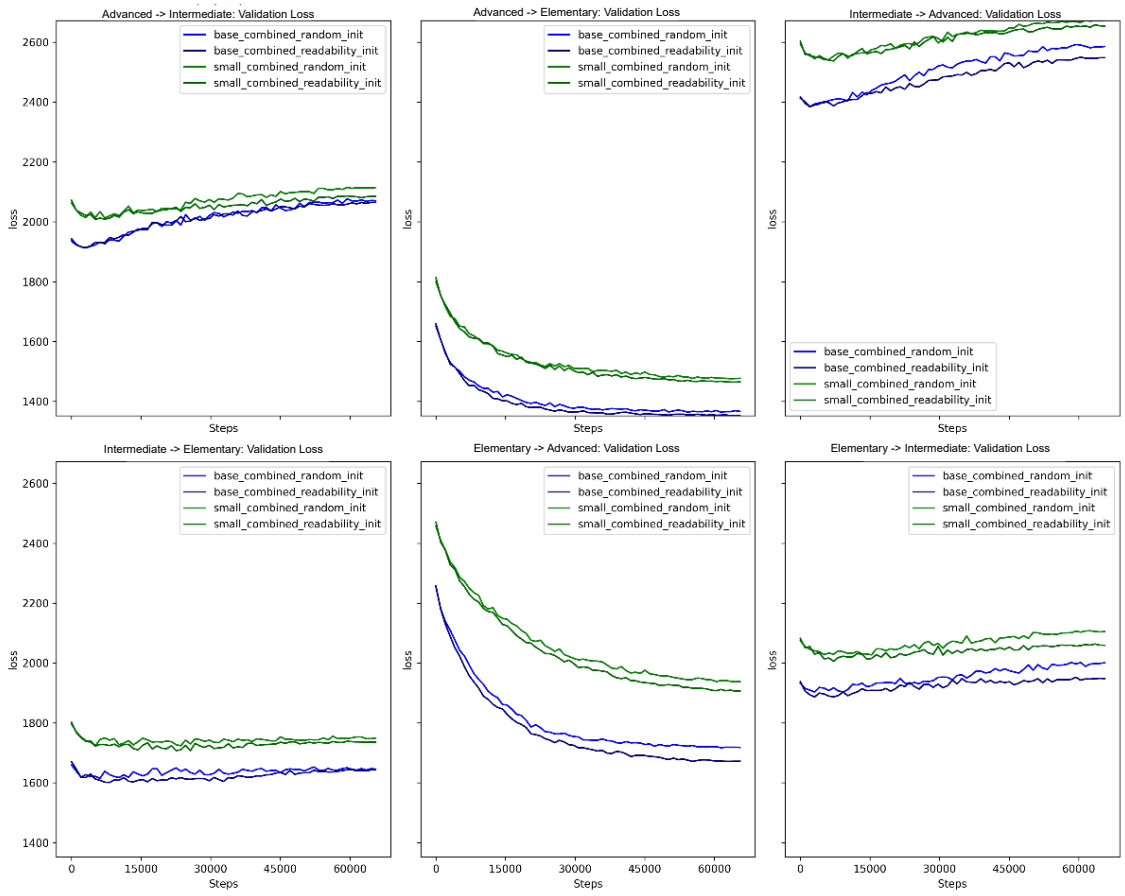

Figure 3: Learning curve of validation loss of the random/readability initialization in the combined architecture

To understand whether this readability initialization is beneficial, we explore this experimentally. However, before we dive deeper into the readability initialization, some general observations need to be made. Figure 2 illustrates the SARI on the validation set over the course of the model's training. A key observation is that the advanced-elementary (bidirectional) pair is able to achieve much higher SARI scores than the other four pairs. This is also reflected in Figure 3, where the validation loss is decreasing for the advanced-elementary pair, while it is either stable or increasing for the other pairs. The hypothesis for this pattern is that the signal between adjacent pairs is not strong enough to train a pairwise model, which we later confirm through the results of the baselines at Section 5.2. This might be because the changes in the training data are too minor for the model to be able to train on, or the data samples are simply not meaningful enough.

Comparing the SARI performance of our proposed readability initialization with

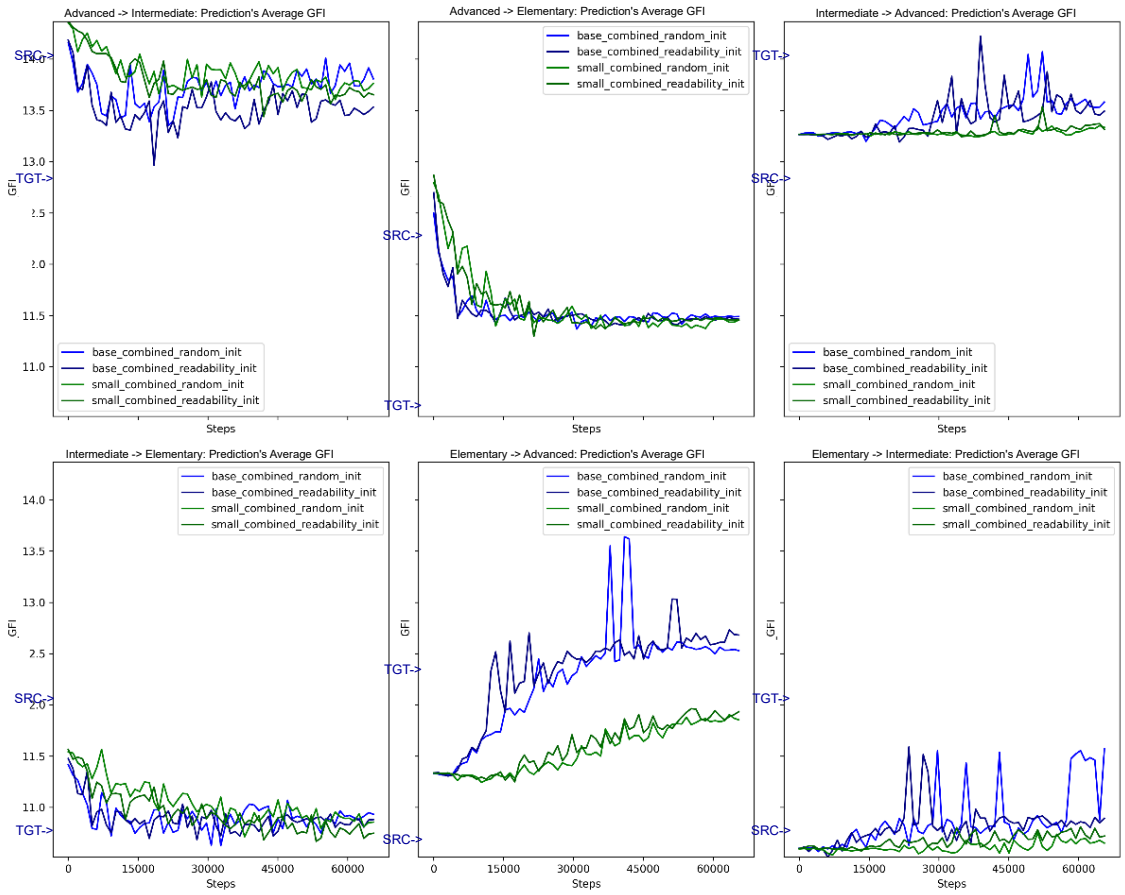

Figure 4: Learning curve of average predictions' GFI of the random/readability initialization in the combined architecture

random initialization, in Figure 2, we observe that the readability initialization is beneficial throughout the learning curve. This observation is also consistent when we look at the average GFI of the predictions in Figure 4, as the readability initialization more closely matches the average GFI of the target sentences. However, we should note the occasional sharp spikes in most of the readability features. These are likely due to inconsistent checkpoints resulting from the bottlenecks of the limited parameters allowed for updates. In the following experiments, unless explicitly stated otherwise, task embeddings will be initialized with the readability level described above.

Separate Task Embeddings Instances for Encoder and Decoder

Inspired by recent advances in machine translation using decoupled encoders and decoders [Philip et al., 2020], we propose the use of separate instances of task embeddings to represent the encoder and decoder transformer stack. These are then given as inputs to the hyperparameter generation network when generating adapters for the corresponding transformer layers.

This allows us to adapt our definition of readability initialization to be a separate embedding E, consisting only of either the source readability level S or the target readability level T. We expect that this approach will also allow for greater positive transfer between different transformation pairs.

For instance, in the case of the decoder, the decoder task embedding is initialized with identical values for both the advanced-elementary and intermediate-elementary transformation pairs. These values are updated independently during training by the shared hyperparameter generation network, and thus may diverge as a result. However, it is important to note that this method does not completely decouple the encoder and decoder. This idea is also shown in the diagram in Figure 1 labeled 'Separate'.

Similar to the last section, we start our empirical examination by looking at the SARI score shown in Figure 5. We observe small performance improvements in the advanced-elementary pair, with more notable improvements in the lower performing pairs. These improvements are particularly evident in the base model size. This observation supports our hypothesis that our task embedding separation technique allows for improved positive transfer to lower performing pairs. Similar to our previous analysis, we examined the average readability features of the predictions and observe a slight improvement over the original initialization, as illustrated by the FKGL in Figure 6.

To verify that our separate instances of task embeddings also benefit from readability initialization, we compare their SARI scores to their randomly initialized counterparts in Figure 7. The figure clearly illustrates the benefits of readability initialization in all evaluation pairs and is further supported by the learning curve of Figure 8.

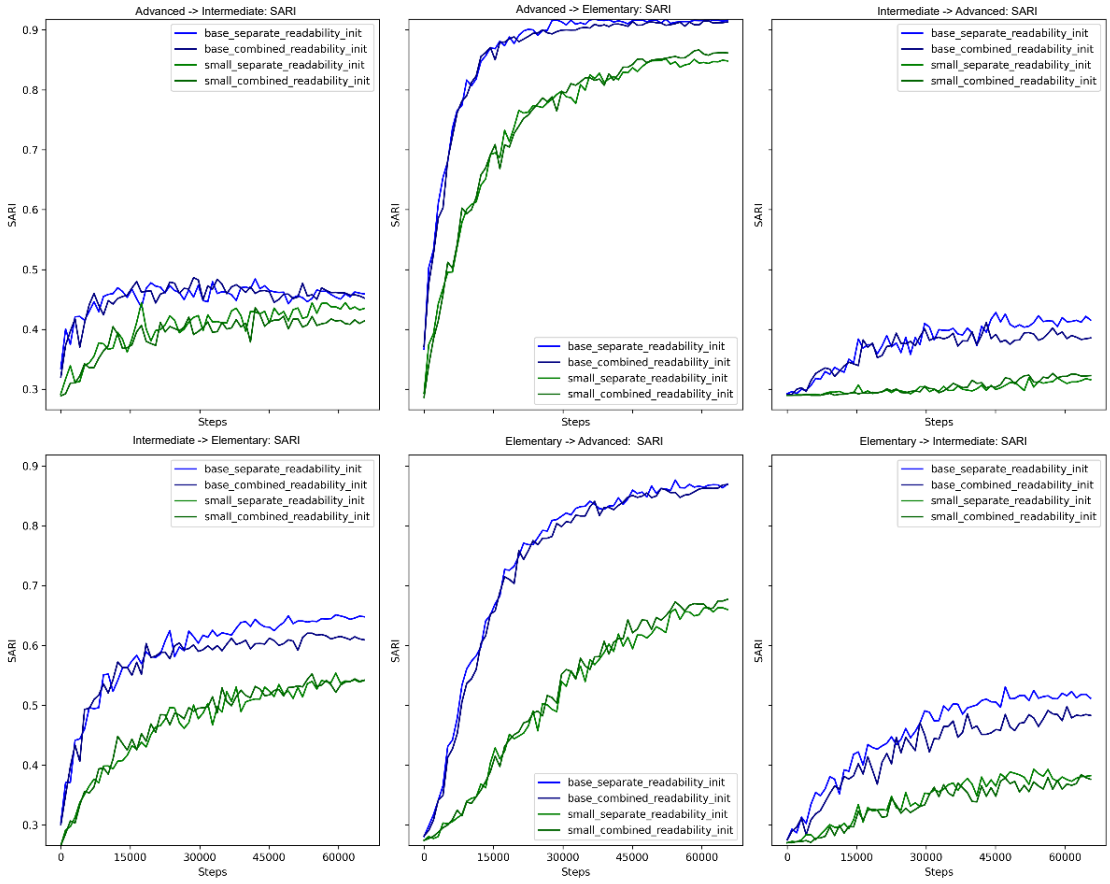

Figure 5: Learning curve of SARI for the Separate/Combined architecture

Frozen Task Embeddings

We further investigate whether it would be beneficial for the model to freeze the task embeddings, with their sole purpose to represent the readability level. The idea is that under these conditions, by freezing the embeddings we force the model to treat pairs with common readability levels as similar tasks. In the case of separate task embeddings, they would even be treated as identical tasks. That means when decoding the output of advanced-elementary and intermediate-elementary, the model will learn to produce the same characteristics for the two pairs hence possibly allowing greater positive transfer from one pair to another.

In the combined architecture, we compare the SARI performance between frozen and unfrozen runs in Figure 9. The results are largely similar, and the performance in terms of readability features, as illustrated by the GFI feature in Figure 10, is also comparable. However, the frozen embedding variant seems to show signs of more

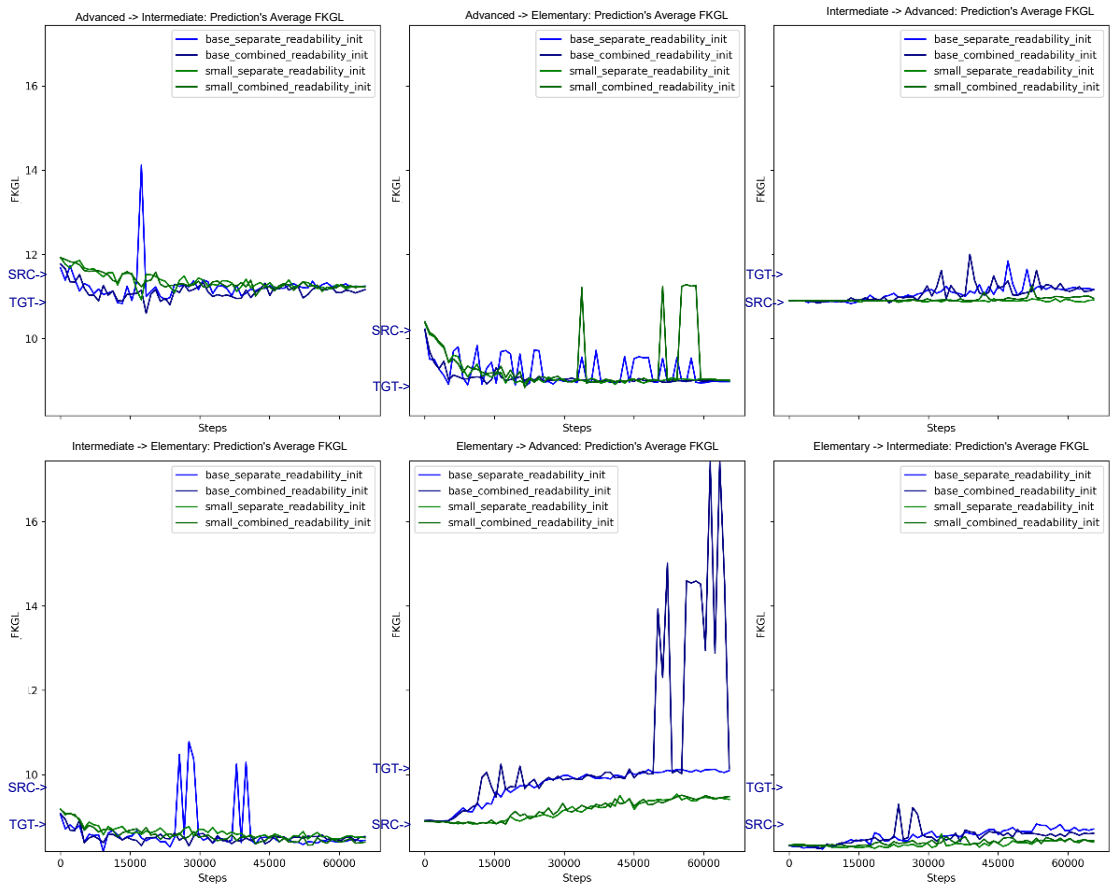

Figure 6: Learning curve of average predictions' FKGL in the Separate/Combined architecture

stable training. Therefore, we conclude that the automated metrics don't provide a definitive conclusion in this case.

In contrast, when considering the separate task embedding architecture, the results diverge significantly. Freezing the task embeddings consistently underperforms the non-frozen variant in four out of six pairs, as illustrated by SARI in Figure 11. The readability characteristics are consistent with this observation, and we illustrate this using GFI in Figure 12. Therefore, we conclude that with separate task embeddings for the encoder and decoder, keeping the task embedding representation trainable is beneficial for the performance of the model, which contradicts our initial hypothesis.

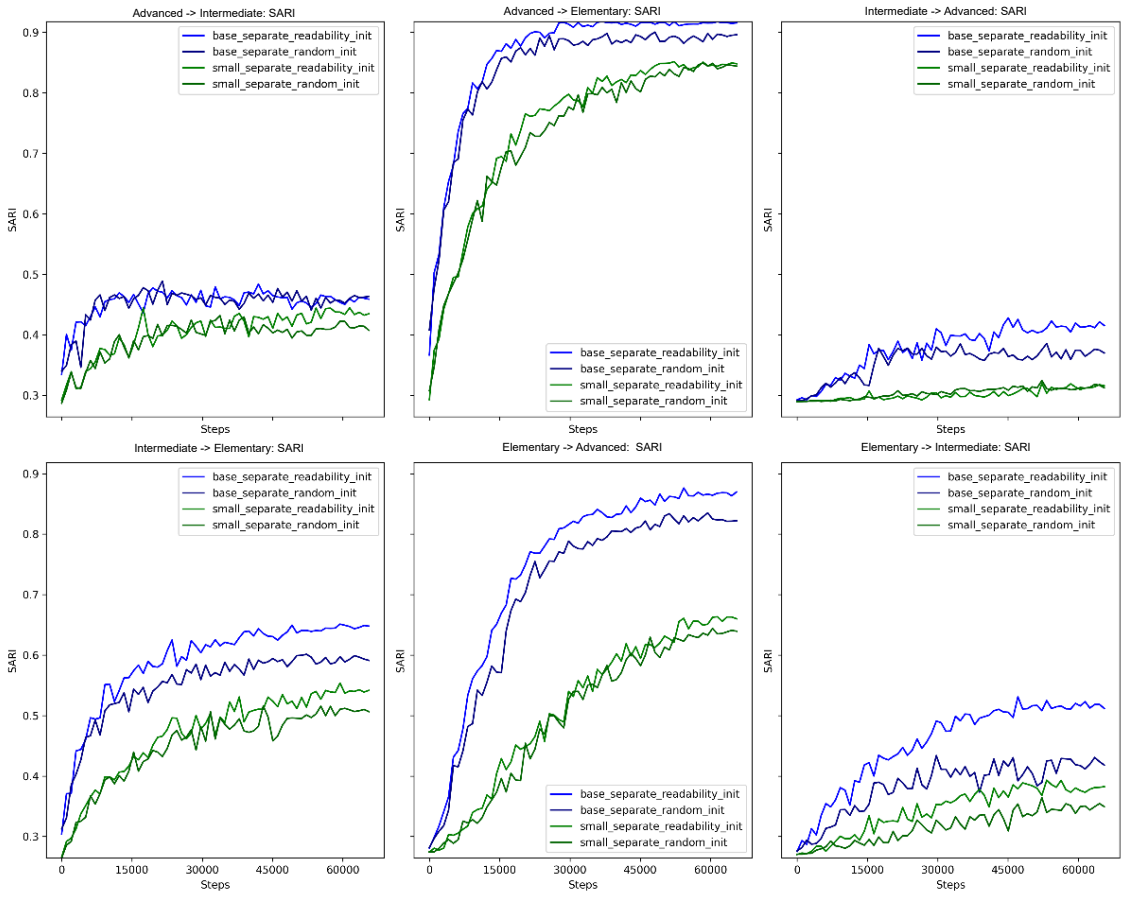

Figure 7: Learning curve of SARI in a Random/Readability Initialization in the Separate architecture

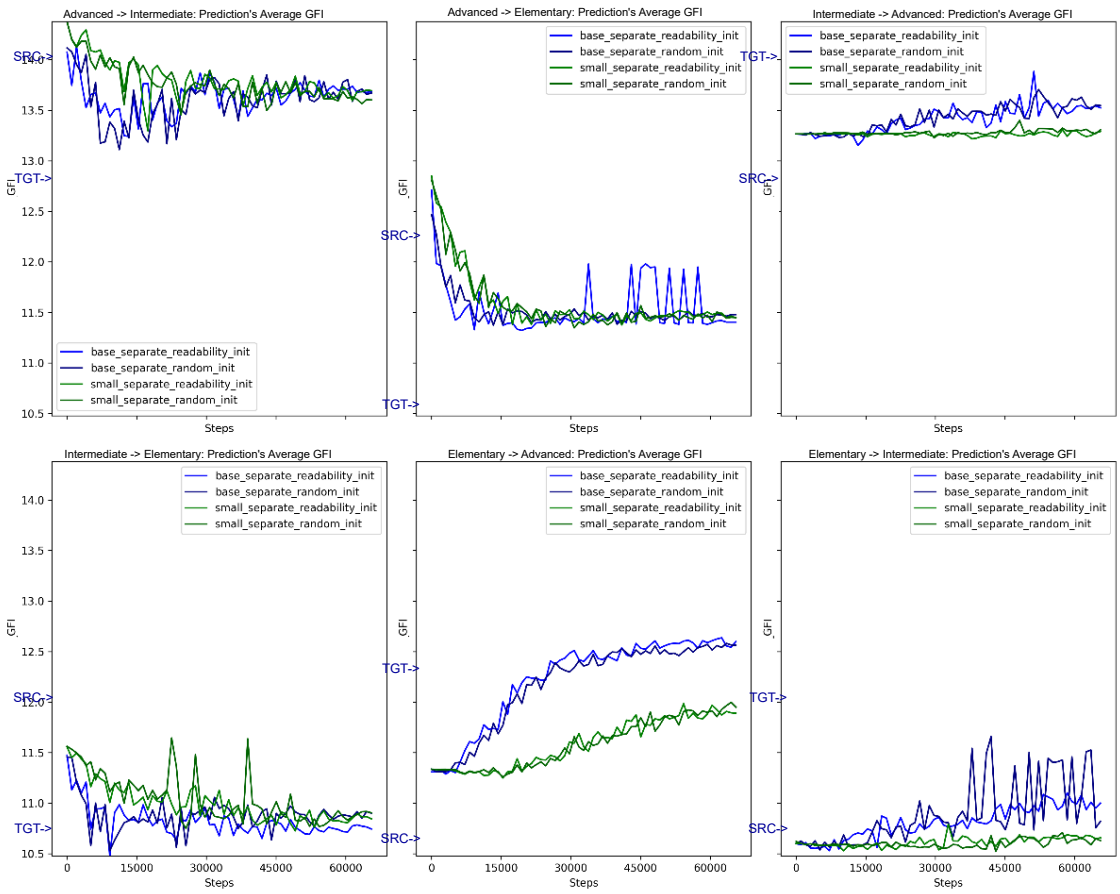

Figure 8: Learning curve of SARI in a Random/Readability Initialization in the Separate architecture

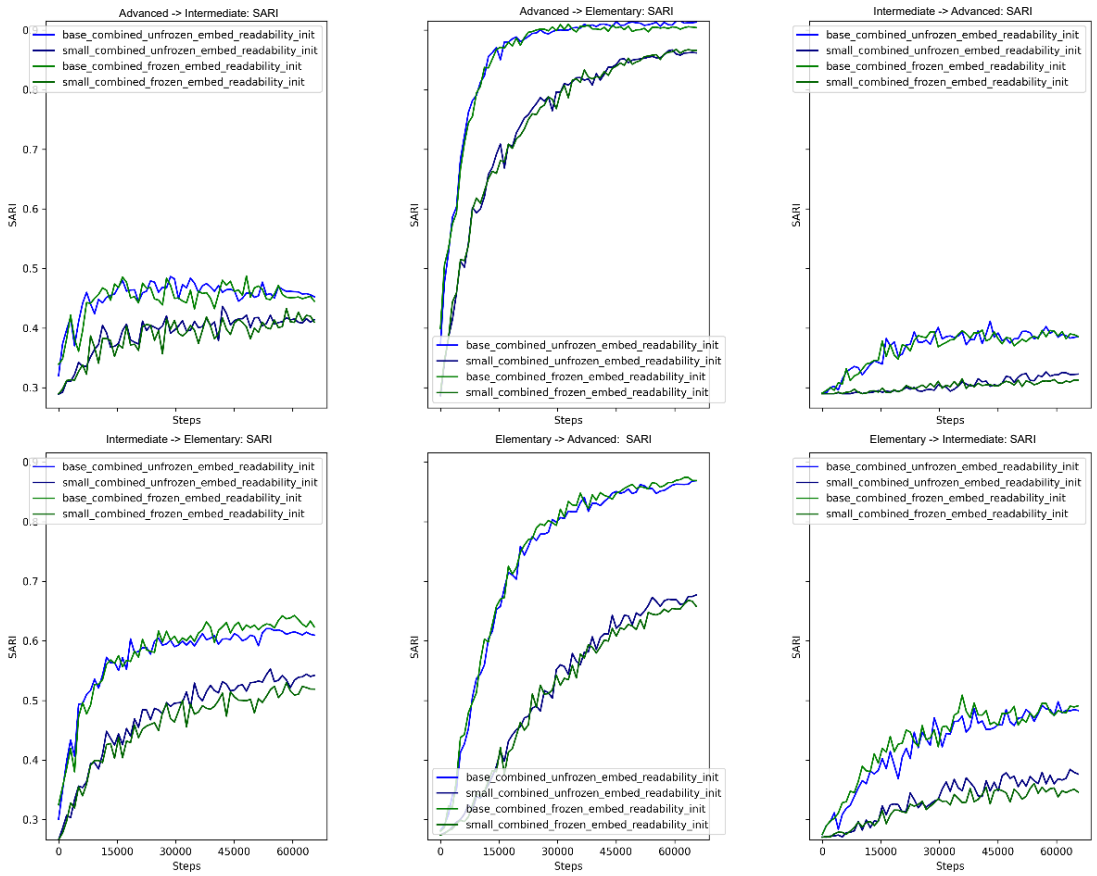

Figure 9: Learning Curve of SARI For Frozen/Unfrozen Embeddings in the combined architecture

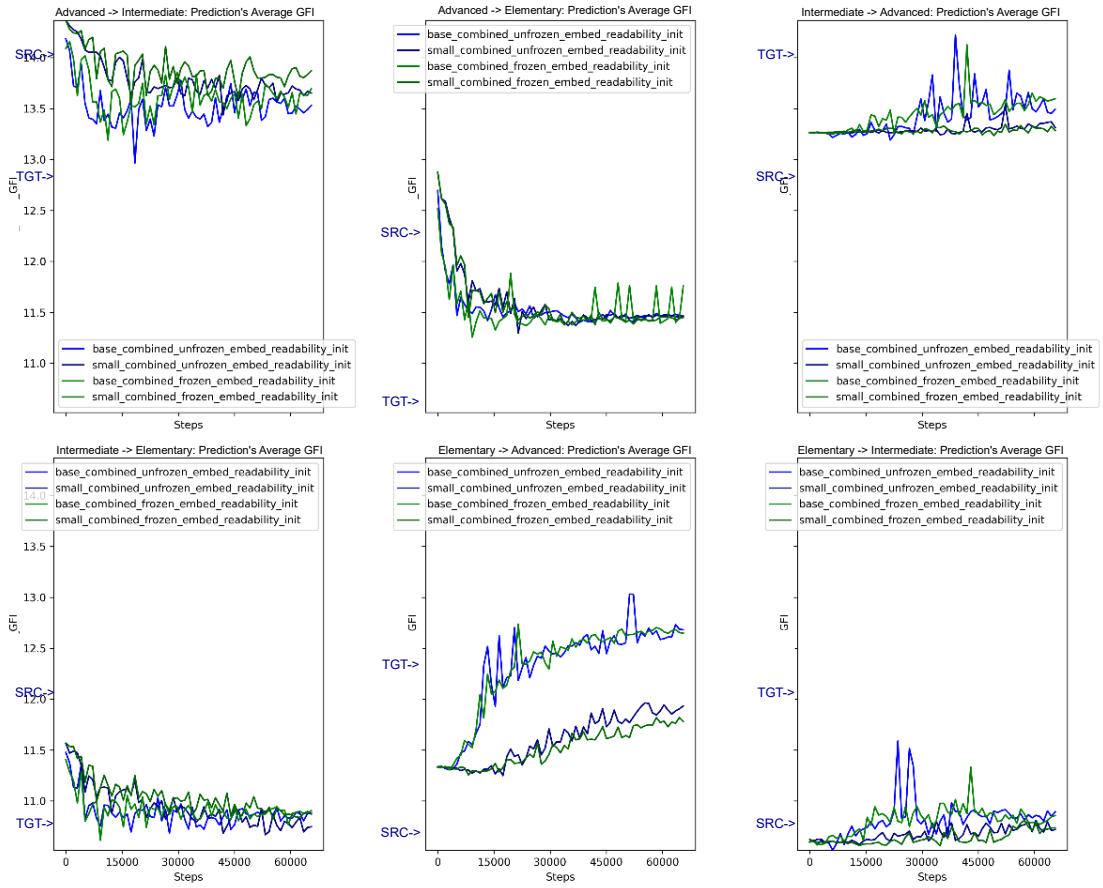

Figure 10: Learning Curve of average predictions' GFI For Frozen/Unfrozen embeddings in the combined architecture

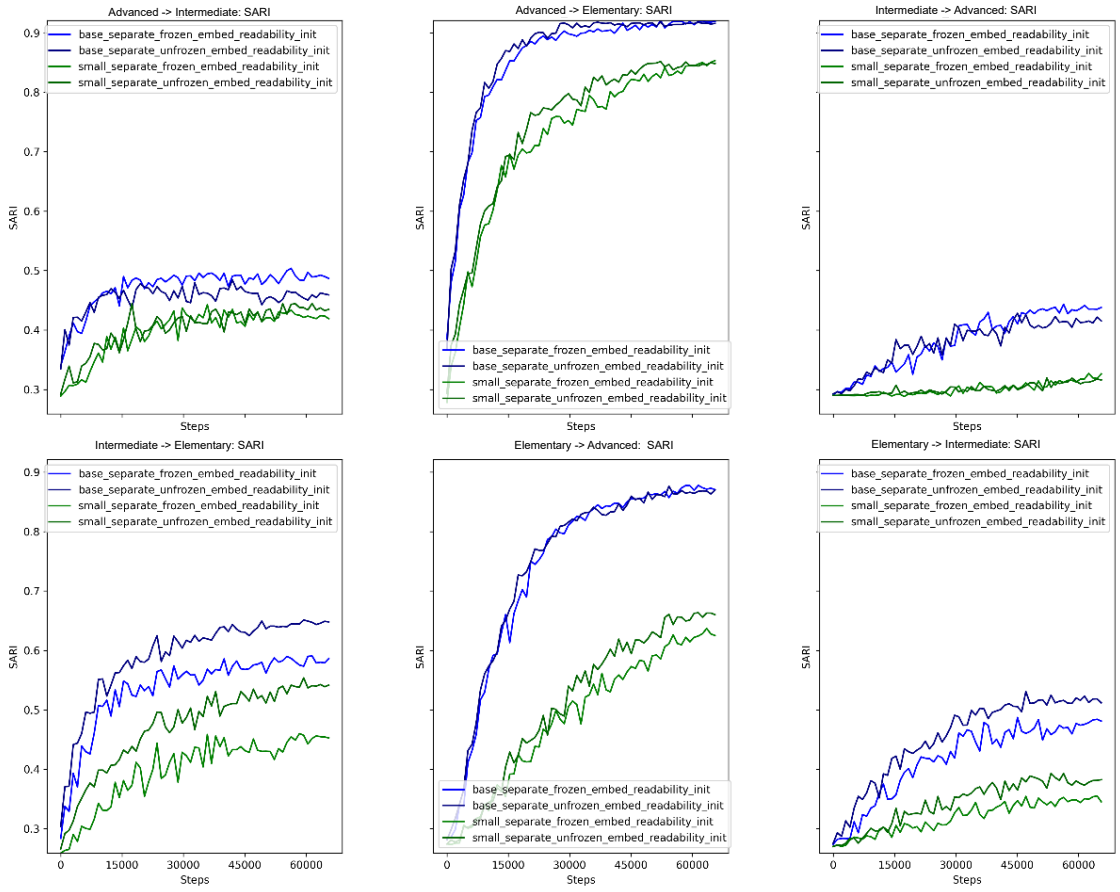

Figure 11: SARI of Frozen/Unfrozen embeddings in the separate architecture

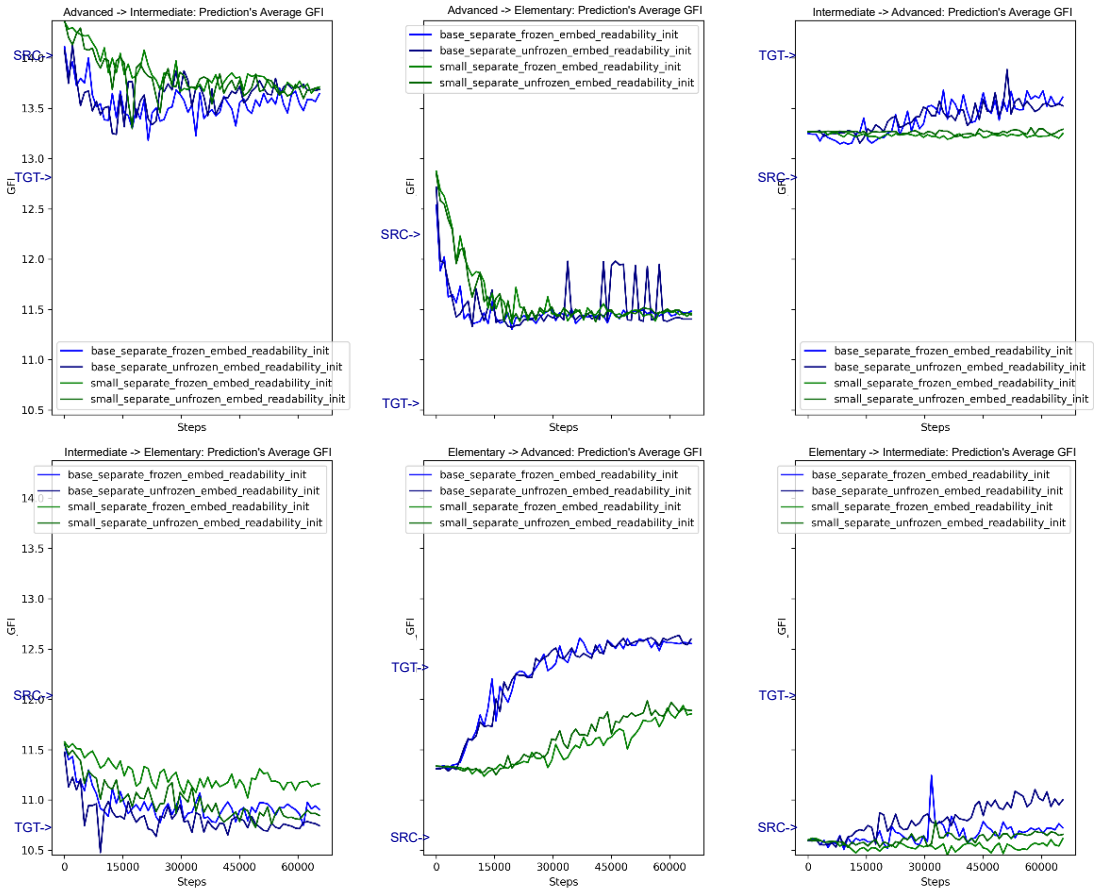

Figure 12: Learning curve for average predictions' GFI of in the separate architecture

5 Results

In this chapter, we begin by defining our experimental setups and identifying the baselines against which we will compare them. We then discuss, illustrate, and evaluate the results in two different settings.

In the first type of results discussion, referred to as the "Extended Evaluation", we perform a comprehensive analysis of the predictions and their characteristics of our approaches relative to all baselines. We then investigate the effects of expanding the hyperparameter generation network.

In the second section, referred to as the "Concise Evaluation Overview", we present a more concise, stand-alone evaluation that focuses solely on SARI. Within this section, we also delve into a discussion of the parameter efficiency of different approaches.

While there is some overlap between the "Extended Evaluation" and the "Concise Evaluation Overview", each section provides unique insights. The concise evaluation overview should be sufficient to grasp the main findings of the study, but the extended evaluation provides a more comprehensive understanding of the results and the value of our contributions.

5.1 Hyperformer++ Chosen Variation & Baselines

Hyperformer++ configuration

Ideally, we would evaluate each possible model over multiple training runs and compare their performance to draw the most reliable conclusions about our results. However, since we are unable to sufficiently replicate the experiments, we need an alternative fair evaluation comparison that is also accessible to the reader. As a result, we decided to include only the following two variations of our architecture in the final results:

- **H++ (Original)**: The original hyperformer++ architecture trained in a

multi-task setting under the randomly initialised task embedding as introduced explained in Mahabadi et al. [2021].

- **H++ (Combined)**: Our hyperformer++ variant trained in a multi-task setting under the combined task embedding readability initialization as introduced in Section 4.2.
- **H++ (Separate)**: Our hyperformer++ variant trained in a multi-task setting under the separate task embedding readability initialization as introduced in Section 4.2.

Baselines

Based on the related literature review, we select four approaches to act as baselines and represent the state of the art against our approaches. In the following list, we refer to these baselines as follows:

- **Pairwise Adapters (T5)**: **(Single Task)** Frozen-T5 with pairwise adapters to perform the transformation task. No prompt used.
- **Pairwise Full Model (T5)**: **(Single Task)** A fully parameterizable model for each pair of transformations. No prompt used.
- **Zero Shot (T5)**: Inference using off-the-shelf T5 with the prompt: "Simplify/Rewrite from {Source-Readability} to {Target-Readability} : {Input}"
- **String Prefix (T5)**[1].: **(Multi Task)** T5 fine-tuned in a multitask setting with the following prompt: "Simplify/Rewrite from {Source-Readability} to {Target-Readability} : {Input}"

5.2 Extended Evaluation

The results for these experiments are reported in the following tables: 9, 10, 11, and 12

- Table 9 shows the performance of T5-Small on the simplification transformation pairs.
- Table 10 shows the performance of T5-Small on the complexification trans-

[1] An underexplored multi-task setup method, which has a fair amount of success in past works [Johnson et al., 2017; Zhang et al., 2023; Mahabadi et al., 2021; Raffel et al., 2020]

formation pairs.

- Table 11 shows the performance of T5-Base on the simplification transformation pairs.
- Table 12 shows the performance of T5-Base on the complexification transformation pairs.

However, before looking into the performance results, we first want to highlight the two quality assurance columns labeled 'Fluency'[2] and 'True Paraphrases' (True P.), as introduced in Section 3.7.

Looking at these columns, we see that the vast majority of sentences (the lowest percentage is 94%) generated by each model are classified as fluent. The True Paraphrases column also suggests that all models generally preserve the original meaning. However, in both the small and basic variants, it is noteworthy that for most transformation pairs, the string prefix model is consistently classified as failing to preserve meaning more often (by 1%) than the hyperformer++ separate variant.

The performance observations in the extended evaluation results are consistent with those reported for the hyperformer in the GLUE benchmark [Mahabadi et al., 2021; Wang et al., 2018]. In their study, the authors found that multitasking – whether through the string prefix or, even greater, through the hyperformer++ architecture – improves overall performance.

Our results replicate this pattern, showing that multitasking models consistently outperform pairwise models across all tasks. This highlights the importance of sharing information between different transformation pairs. However, opposite to the original study, the string prefix baseline – even more so in the smaller models – consistently outperforms both separate and combined hyperformer++ architectures.

Our hypothesis for this discrepancy lies in the generative nature of our tasks, as opposed to those in the GLUE benchmark. This generative nature may require a larger number of trainable parameters. Consequently, each hyperformer++ architecture faces a parameter bottleneck due to its hypernet, which accounts for ($\approx$2%[3]) of the total model parameters. On the other hand, the string prefix baseline can leverage the entirety of the T5 model's parameters for these six transformation pairs.

[2]In the base variant, fitting all these models into a 16GB GPU memory proved challenging, so the fluency check was omitted in the base experiments. We assume (but don't report) that the results would be similar to the corresponding small variants.

[3]for the base model, $\approx$4% for the small model

Extra Evaluation: A Wider Hypernetwork

To test the hypothesis that the performance of the hyperformer++ architecture is bottlenecked by the size of the hypernetwork, we trained our variants of the T5 base hyperformer++ model using a hypernetwork 3.5 times wider (accounting for a total of 7% of the LM parameters). The variants that use a wider hypernetwork are denoted by the symbol †.

As can be seen from the results in Tables 13 (simplification) and 14 (complexification), using a wider hypernetwork does indeed enhances the SARI performance of our hyperformer++ variants. It not only achieves the highest performance within two pairs, but also makes our variants more competitive against the strong string prefix baseline across all transformation pairs. To clearly demonstrate the value of our contributions, we also trained an original variant of the hyperformer++ architecture, which consistently underperforms all of our variants.

5.3 Concise Evaluation Overview

In a concise summary of the main results of the study, we compare SARI across different architectures and transformation pairs for both sizes in Table 15. It is immediately apparent that pairwise models underperform compared to the multitask approach, likely due to the commonalities between different transformation tasks.

In particular, for the T5 small models, the string prefix setting significantly outperforms the hyperformer++ variants, including those with wider hypernetworks. However, for the T5 base models, this performance gap narrows significantly, occurring in only four of the six transformations, and decreases even further against the hyperformer++ variants with a wider hypernetwork. These observations suggest that, in the context of this study, we have successfully developed a task-specific variant of hyperformer++ which, despite using a minimal increase in additional parameters (2% or 7%), nearly matches the performance of a model that fine-tunes all of it's weights, demonstrating parameter efficient positive transfer between transformation pairs

As an advantage of conducting a bidirectional study, we can answer a final unique question. Which of the supertasks do the models generally perform better at, text simplification (three columns on the left) or text complexification (three columns on the right)? Although the differences are small, all models seem to perform better at text simplification. However, when comparing our hyperformer++ variant and

the string prefix approach, hyperformer++ seems to perform slightly better in the simplification task, while the string prefix variant seems to have a slight advantage in the text complexification task.

	0-1 Measures				Average Readability Features				
Simplification	SARI	Fluency	True P.	Copies	GFI	FRE	FKGL	ARI	ASL
Adv → Ele		0.985	0.976	0.000	11.33	68.04	08.89	11.00	20.57
Zero Shot	0.296	0.933	0.771	0.176	10.87	66.11	08.47	10.40	17.83
Pairwise Adapters	0.759	0.967	0.981	0.167	11.55	66.05	09.20	11.36	20.71
Pairwise Full Model	0.840	0.555	0.990	0.086	11.60	66.68	09.15	11.31	20.88
H++ (Combined)	0.867	0.957	0.976	0.057	**11.47**	67.16	09.02	**11.13**	**20.64**
H++ (Separate)	0.852	0.967	0.976	0.086	11.49	67.16	09.04	11.23	20.69
String Prefix	**0.938**	0.971	0.967	0.019	**11.47**	**67.55**	**09.00**	11.14	20.74
Adv → Int		0.991	0.932	0.000	13.27	59.15	10.91	13.47	23.71
Zero Shot	0.272	0.887	0.671	0.167	11.29	63.02	08.99	11.30	18.17
Pairwise Adapters	0.384	0.968	0.986	0.514	13.94	56.72	11.48	14.09	24.64
Pairwise Full Model	0.398	0.955	0.991	0.586	14.22	55.69	11.80	14.48	25.35
H++ (Combined)	0.436	0.955	0.959	0.338	13.44	58.39	11.01	13.53	**23.71**
H++ (Separate)	0.445	0.973	0.950	0.333	**13.30**	**59.00**	**10.92**	**13.46**	**23.70**
String Prefix	**0.467**	0.968	0.955	0.315	13.70	57.43	11.28	13.87	24.27
Int → Ele		0.976	0.880	0.000	10.60	69.91	08.32	10.30	19.37
Zero Shot	0.280	0.898	0.743	0.323	09.95	68.29	07.70	09.37	15.92
Pairwise Adapters	0.344	0.976	0.964	0.623	11.23	67.37	08.83	10.74	19.97
Pairwise Full Model	0.399	0.946	0.982	0.557	11.27	67.33	08.92	10.94	20.31
H++ (Combined)	0.553	0.964	0.928	0.311	**10.70**	**69.88**	**08.33**	**10.25**	19.39
H++ (Separate)	0.554	0.964	0.952	0.323	10.81	68.84	08.47	**10.36**	**19.36**
String Prefix	**0.636**	0.964	0.940	0.257	10.75	69.58	08.41	10.41	19.54

Table 9: **Extended Evaluation**: Target-level results for **T5-Small** on the onestop dataset in the **simplification** pairs. For SARI, where higher values indicate better performance, we emphasize the systems according to their scores. For the average readability features, we highlight the systems that perform closest to the level-specific references (which are provided in the intermediary rows).

	0-1 Measures				Average Readability Features				
Complexification	SARI	Fluency	True P.	Copies	GFI	FRE	FKGL	ARI	ASL
Ele → Adv		0.981	0.995	0.000	12.93	61.26	10.45	12.89	23.05
Zero Shot	0.297	0.923	0.766	0.205	09.91	70.68	07.63	09.46	16.98
Pairwise Adapters	0.504	0.952	1.000	0.486	11.50	67.16	09.11	11.16	20.97
Pairwise Full Model	0.795	0.976	0.990	0.205	12.55	63.82	09.98	12.34	22.59
H++ (Combined)	0.677	0.962	1.000	0.238	11.93	65.77	09.48	11.68	21.70
H++ (Separate)	0.663	0.967	0.986	0.281	11.92	65.68	09.45	11.67	21.51
String Prefix	**0.932**	0.976	0.995	0.010	**12.77**	**62.01**	**10.29**	**12.69**	**22.83**
Int → Adv		0.991	0.959	0.000	14.39	55.11	11.94	14.68	25.62
Zero Shot	0.282	0.910	0.775	0.238	11.08	64.36	08.83	10.94	18.27
Pairwise Adapters	0.304	0.986	1.000	0.946	13.27	59.09	10.91	13.46	23.70
Pairwise Full Model	0.327	0.982	0.995	0.779	13.36	58.78	11.01	13.63	23.93
H++ (Combined)	0.326	0.977	0.995	0.757	13.30	58.65	11.00	13.51	23.83
H++ (Separate)	0.319	0.977	0.995	0.793	13.28	58.96	10.94	13.49	23.75
String Prefix	**0.470**	0.982	0.977	0.432	**13.45**	**58.41**	**11.18**	**13.81**	**24.40**
Ele → Int		0.980	0.910	0.000	11.58	65.82	09.20	11.21	20.60
Zero Shot	0.288	0.928	0.748	0.299	09.33	73.24	07.00	08.85	15.88
Pairwise Adapters	0.299	0.964	0.994	0.850	10.62	69.84	08.35	10.32	19.43
Pairwise Full Model	0.308	0.976	0.988	0.856	10.64	70.03	08.34	10.33	19.49
H++ (Combined)	0.384	0.940	0.976	0.581	10.80	69.56	08.46	10.41	19.71
H++ (Separate)	0.393	0.958	0.988	0.557	10.65	69.33	08.43	10.32	19.49
String Prefix	**0.558**	0.976	0.970	0.251	**11.24**	**67.57**	**08.90**	**10.91**	**20.37**

Table 10: **Extended Evaluation**: Target-level results for **T5-Small** on the onestop dataset in the **complexification** pairs. For SARI, where higher values indicate better performance, we emphasize the systems according to their scores. For the average readability features, we highlight the systems that perform closest to the level-specific references (which are provided in the intermediary rows).

	0-1 Measures				Average Readability Features				
Simplification	SARI	Fluency	True P.	Copies	GFI	FRE	FKGL	ARI	ASL
Adv → Ele		0.985	0.976	0.000	11.33	68.04	08.89	11.00	20.57
Zero Shot	0.271	0.842	0.621	0.214	08.83	70.84	06.64	08.43	13.13
Pairwise Adapters	0.840		0.986	0.105	11.66	65.30	09.39	11.54	21.09
Pairwise Full Model	0.837		0.990	0.105	11.61	66.42	09.20	11.34	20.94
H++ (Combined)	0.915		0.971	0.029	11.47	67.36	09.01	11.14	20.69
H++ (Separate)	0.918		0.971	0.029	**11.40**	67.45	**08.97**	**11.12**	**20.59**
String Prefix	**0.939**		0.962	0.019	11.49	**67.53**	09.00	11.15	20.76
Adv → Int		0.991	0.932	0.000	13.27	59.15	10.91	13.47	23.71
Zero Shot	0.260	0.901	0.594	0.171	08.59	69.79	06.82	08.80	13.26
Pairwise Adapters	0.424		0.986	0.459	14.00	56.70	11.60	14.27	25.14
Pairwise Full Model	0.410		0.991	0.482	14.20	55.86	11.78	14.47	25.38
H++ (Combined)	**0.484**		0.955	0.194	**13.48**	**58.54**	**11.05**	**13.58**	**23.94**
H++ (Separate)	**0.484**		0.977	0.203	13.60	58.22	11.13	13.65	24.10
String Prefix	0.474		0.964	0.342	13.73	57.10	11.34	13.94	24.31
Int → Ele		0.976	0.880	0.000	10.60	69.91	08.32	10.30	19.37
Zero Shot	0.286	0.911	0.670	0.221	08.84	74.19	06.28	07.97	13.55
Pairwise Adapters	0.416		0.976	0.461	11.03	68.41	08.63	10.53	19.74
Pairwise Full Model	0.406		0.988	0.467	11.15	68.12	08.80	10.77	20.28
H++ (Combined)	0.620		0.952	0.293	10.90	68.99	08.53	10.49	19.67
H++ (Separate)	**0.651**		0.958	0.210	**10.72**	**69.61**	**08.40**	**10.39**	**19.51**
String Prefix	0.647		0.940	0.257	10.86	68.94	08.52	10.53	19.62

Table 11: **Extended Evaluation**: Target-level results for **T5-Base** on the onestop dataset in the **simplification** pairs. For SARI, where higher values indicate better performance, we emphasize the systems according to their scores. For the average readability features, we highlight the systems that perform closest to the level-specific references (which are provided in the intermediary rows).

	0-1 Measures				Average Readability Features				
Complexification	SARI	Fluency	True P.	Copies	GFI	FRE	FKGL	ARI	ASL
Ele → Adv		0.981	0.995	0.000	12.93	61.26	10.45	12.89	23.05
Zero Shot	0.287	0.904	0.710	0.219	08.25	74.20	06.17	07.48	13.08
Pairwise Adapters	0.800		1.000	0.195	12.55	63.54	10.03	12.43	22.66
Pairwise Full Model	0.796		1.000	0.205	12.61	63.49	10.05	12.40	22.72
H++ (Combined)	0.869		0.981	0.062	12.68	62.74	10.14	12.47	22.66
H++ (Separate)	0.876		0.990	0.071	12.60	62.79	10.12	12.46	22.61
String Prefix	**0.930**		0.990	0.033	**12.75**	**61.93**	**10.30**	**12.69**	**22.82**
Int → Adv		0.991	0.959	0.000	14.39	55.11	11.94	14.68	25.62
Zero Shot	0.251	0.887	0.617	0.189	08.95	68.05	07.12	08.66	13.47
Pairwise Adapters	0.345		0.982	0.662	13.37	59.09	11.02	13.63	24.14
Pairwise Full Model	0.341		0.995	0.748	13.35	58.75	10.97	13.52	23.76
H++ (Combined)	0.411		0.977	0.342	13.41	58.62	11.11	13.66	24.23
H++ (Separate)	0.428		0.955	0.392	**13.59**	**57.90**	**11.25**	**13.88**	24.41
String Prefix	**0.480**		0.959	0.392	13.54	58.32	11.19	13.76	**24.38**
Ele → Int		0.982	0.910	0.000	11.58	65.82	09.20	11.21	20.60
Zero Shot	0.278	0.874	0.598	0.191	07.40	76.72	05.43	06.90	11.55
Pairwise Adapters	0.330		0.982	0.647	10.83	68.62	08.59	10.58	19.73
Pairwise Full Model	0.322		0.994	0.725	10.62	69.87	08.35	10.31	19.45
H++ (Combined)	0.498		0.958	0.311	10.86	68.10	08.64	10.53	19.60
H++ (Separate)	0.531		0.952	0.263	10.94	67.80	08.69	10.58	19.64
String Prefix	**0.571**		0.970	0.275	**11.30**	**67.11**	**08.97**	**11.03**	**20.38**

Table 12: **Extended Evaluation**: Target-level results for **T5-Base** on the onestop dataset in the **complexification** pairs. For SARI, where higher values indicate better performance, we emphasize the systems according to their scores. For the average readability features, we highlight the systems that perform closest to the level-specific references (which are provided in the intermediary rows).

	0-1 Measures				Average Readability Features				
Simplification	SARI	Fluency	True P.	Copies	GFI	FRE	FKGL	ARI	ASL
Adv → Ele		0.985	0.976	0.000	11.33	68.04	08.89	11.00	20.57
H++(Original)	0.902		0.976	0.038	11.52	67.32	09.05	11.20	20.84
H++ (Combined)	0.915		0.971	0.029	11.47	67.36	09.01	11.14	20.69
H++ (Combined)†	0.917		0.962	0.033	**11.39**	**67.60**	**08.97**	**11.10**	20.65
H++ (Separate)	0.918		0.971	0.029	11.40	67.45	**08.97**	11.12	**20.59**
H++ (Separate)†	0.918		0.967	0.024	11.47	67.26	09.02	11.15	20.66
String Prefix	**0.939**		0.962	0.019	11.49	67.53	09.00	11.15	20.76
Adv → Int		0.991	0.932	0.000	13.27	59.15	10.91	13.47	23.71
H++ (Original)	0.475		0.977	0.230	13.74	57.53	11.30	13.87	24.41
H++ (Combined)	0.484		0.955	0.194	**13.48**	58.54	11.05	13.58	23.94
H++ (Combined)†	0.491		0.964	0.212	13.56	58.29	11.12	**13.60**	24.11
H++ (Separate)	0.484		0.977	0.203	13.60	58.22	11.13	13.65	24.10
H++ (Separate)†	**0.504**		0.968	0.140	13.53	**58.58**	**11.06**	**13.61**	**24.01**
String Prefix	0.474		0.964	0.342	13.73	57.10	11.34	13.94	24.31
Int → Ele		0.976	0.880	0.000	10.60	69.91	08.32	10.30	19.37
H++ (Original)	0.619		0.940	0.222	10.88	68.89	08.53	10.47	19.62
H++ (Combined)	0.620		0.952	0.293	10.90	68.99	08.53	10.49	19.67
H++ (Combined)†	0.653		0.952	0.234	10.81	69.24	08.52	10.49	19.77
H++ (Separate)	0.651		0.958	0.210	**10.72**	**69.61**	**08.40**	**10.39**	**19.51**
H++ (Separate)†	**0.656**		0.952	0.251	10.76	69.58	08.44	**10.39**	19.64
String Prefix	0.647		0.940	0.257	10.86	68.94	08.52	10.53	19.62

Table 13: **Extra Results**: Target-level results for **T5-Base** on the onestop dataset in the **simplification** pairs. For SARI, where higher values indicate better performance, we emphasize the systems according to their scores. For the average readability features, we highlight the systems that perform closest to the level-specific references (which are provided in the intermediary rows). †shows a model using a wider hypernetwork (x3.5 params)

	0-1 Measures				Average Readability Features				
Complexification	SARI	Fluency	True P.	Copies	GFI	FRE	FKGL	ARI	ASL
Ele → Adv		0.981	0.995	0.000	12.93	61.26	10.45	12.89	23.05
H++ (Original)	0.837		1.000	0.100	12.57	63.65	10.00	12.38	22.60
H++ (Combined)	0.869		0.981	0.062	12.68	62.74	10.14	12.47	22.66
H++ (Combined)†	0.889		1.000	0.062	12.63	62.78	10.15	12.55	22.70
H++ (Separate)	0.876		0.990	0.071	12.60	62.79	10.12	12.46	22.61
H++ (Separate)†	0.901		0.990	0.038	12.61	63.02	10.10	12.52	22.66
String Prefix	**0.930**		0.990	0.033	**12.75**	**61.93**	**10.30**	**12.69**	**22.82**
Int → Adv		0.991	0.959	0.000	14.39	55.11	11.94	14.68	25.62
H++ (Original)	0.383		0.973	0.410	13.63	57.84	11.27	13.87	24.43
H++ (Combined)	0.411		0.977	0.342	13.41	58.62	11.11	13.66	24.23
H++ (Combined)†	0.406		0.977	0.351	13.53	58.50	11.20	13.80	**24.54**
H++ (Separate)	0.428		0.955	0.392	**13.59**	**57.90**	**11.25**	**13.88**	24.41
H++ (Separate)†	0.414		0.977	0.338	13.56	57.93	11.16	13.68	24.05
String Prefix	**0.480**		0.959	0.392	13.54	58.32	11.19	13.76	24.38
Ele → Int		0.982	0.910	0.000	11.58	65.82	09.20	11.21	20.60
H++ (Original)	0.438		0.982	0.407	10.82	68.28	08.59	10.53	19.52
H++ (Combined)	0.498		0.958	0.311	10.86	68.10	08.64	10.53	19.60
H++ (Combined)†	0.516		0.940	0.275	10.90	67.91	08.68	10.60	19.69
H++ (Separate)	0.531		0.952	0.263	10.94	67.80	08.69	10.58	19.64
H++ (Separate)†	0.541		0.964	0.222	11.11	67.72	08.74	10.62	19.82
String Prefix	**0.571**		0.970	0.275	**11.30**	**67.11**	**08.97**	**11.03**	**20.38**

Table 14: **Extra Results**: Target-level results for **T5-Base** on the onestop dataset in the **complexification** pairs. For SARI, where higher values indicate better performance, we emphasize the systems according to their scores. For the average readability features, we highlight the systems that perform closest to the level-specific references (which are provided in the intermediary rows). †shows a model using a wider hypernetwork (x3.5 params)

Model	LM Params	Per Task	Adv → Ele	Adv → Int	Int → Ele	Ele → Adv	Int → Adv	Ele → Int	**AVG**
Single-Task Training									
T5$_{small}$	$6.0\times$	100%	0.840	0.398	0.399	0.795	0.327	0.308	0.511
T5-Adapters$_{small}$	$1 + 6 \times 0.01$	0.74%	0.759	0.384	0.344	0.504	0.304	0.299	0.432
T5$_{base}$	$6.0\times$	100%	0.837	0.410	0.406	0.796	0.341	0.322	0.519
T5-Adapters$_{base}$	$1 + 6 \times 0.01$	0.87%	0.840	0.424	0.416	0.800	0.345	0.330	0.526
Multi-Task Training									
H++$_{small}$(Original)	$1.04\times$	0.67%	0.827	0.415	0.494	0.652	0.324	0.345	0.510
H++$_{small}$(Separate)	$1.04\times$	0.67%	0.852	0.445	0.554	0.663	0.319	0.393	0.538
H++$_{small}$(Separate)†	$1.15\times$	2.50%	0.869	0.445	0.555	0.716	0.322	0.379	0.548
T5 String Prefix$_{small}$	$1.0\times$	16.67%	**0.938**	**0.467**	**0.636**	**0.932**	**0.470**	**0.558**	**0.667**
H++$_{base}$(Original)	$1.02\times$	0.40%	0.902	0.475	0.619	0.837	0.383	0.438	0.609
H++$_{base}$(Separate)	$1.02\times$	0.40%	0.918	0.484	0.651	0.876	0.428	0.531	0.648
H++$_{base}$(Separate)†	$1.07\times$	1.15%	0.918	**0.504**	**0.656**	0.901	0.414	0.541	0.656
T5 String Prefix$_{base}$	$1.0\times$	16.67%	**0.939**	0.474	0.647	**0.930**	**0.480**	**0.571**	**0.673**

Table 15: **Concise Evaluation**: SARI Performance of the T5 model across all different tasks. †shows a model using a wider hypernetwork (3.5 times the params). The LM Params column indicates the proportion of parameters required for the approach (relative to the total LM params of the model). The Per Pair column indicates the proportion of trained parameters (relative to the total LM Params of the model) to perform each task.

6 Limitations & Future Work

While our models consistently generate sentences that are similar to the reference sentence across multiple metrics, this does not guarantee that these sentences will always be the user's preferred choice, or even found useful in real-world scenarios. The importance of evaluating the quality of generated text from a human perspective cannot be overstated. This is why modern chatbots like BlenderBot3 [Shuster et al., 2022] and ChatGPT [OpenAI, 2023] continuously incorporate human feedback during and after training. Regrettably, due to the significant technical overhead of this project, we have supplemented human evaluation with fluency and meaning preservation checks via Language Models. While these offer a helpful indication, they are by no means equivalent, and this choice remains a limitation of this study.

While this work has presented some interesting results, we believe that there is still a plethora of insights about the properties and capabilities of our architectures that can be gained through further experimentation. One such interesting property that could be explored is whether our models have acquired the ability, through multitask training, to perform in a zero or few-shot scenarios on transformation pairs that they haven't encountered before. This could be achieved by omitting a transformation pair during training and then evaluating performance on the omitted pair.

Another aspect that we have not explored at all in this study is the effect of different ways of aligning sentences as training data for our models. We trained our models on Onestop parallel English sentence aligned using a cosine similarity method. We expect that the model might show different behavior if it was trained on sentences aligned with a different method, such as alignment using a language model.

While the trend in natural language processing research is to be predominantly focused on the English language, a trend that our study unfortunately follows, we found it difficult during our preliminary search to find suitable resources for conducting this specific study in another language. However, we were able to identify the German simplification/summarization datasets [Ebling et al., 2022] and [Anschütz et al., 2023] which, with additional effort, could potentially be transformed into valuable resources for the task we have explored. Given access to parallel sen-

tences in multiple readability levels and languages, it would be possible to replicate this study in a multilingual setting by replacing T5 with a multilingual model such as mT5 [Xue et al., 2021]. However, a critical aspect of any potential multilingual follow-up study would be to modify the multitask scheduler to accommodate multilingual learning.

7 Conclusion

In this thesis, we have made several contributions to the study of the task of re-generating sentences at different readability levels. We started by examining the available corpora and analyzing their readability features. We then established an evaluation pipeline that evaluates the performance of the models using several traditional readability features as well as reference-based metrics such as SARI. As part of our evaluation methodology, we used externally trained language models to ensure that our models generate paraphrases that maintain grammatical correctness.

Within a logical sequence of experiments, we evaluate our proposed architectural variations to the hyperformer++ architecture. This leads us to our main contribution, a unique variant of hyperformer++, specifically designed to tackle the sentence re-generation to target specific readability levels task within a multitask framework. Finally, we compare our approach with other common pairwise or multitask solutions. Our results suggest that pairwise approaches often struggle to extract sufficient signals from the training data, which might also be due to the nature of the data itself.

Within this comparison, we show that our proposed method (using only 2% trained parameters) performs competitively against strong multitask baselines that fine-tune all model parameters. Specifically, our two key architectural additions, initialization by an ordinal readability representation and separation of the task representation between the encoder and decoder, both contribute significantly to bridging the performance gap of the original hyperformer++ to the strong string prefix baseline.

References

A. Aghajanyan, A. Gupta, A. Shrivastava, X. Chen, L. Zettlemoyer, and S. Gupta. Muppet: Massive multi-task representations with pre-finetuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 5799–5811, 2021.

M. Anschütz, J. Oehms, T. Wimmer, B. Jezierski, and G. Groh. Language models for German text simplification: Overcoming parallel data scarcity through style-specific pre-training. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1147–1158, Toronto, Canada, July 2023. Association for Computational Linguistics. URL `https://aclanthology.org/2023.findings-acl.74`.

A. Ansell, E. M. Ponti, J. Pfeiffer, S. Ruder, G. Glavaš, I. Vulić, and A. Korhonen. MAD-G: Multilingual adapter generation for efficient cross-lingual transfer. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4762–4781, Punta Cana, Dominican Republic, Nov. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.findings-emnlp.410. URL `https://aclanthology.org/2021.findings-emnlp.410`.

T. Deutsch, M. Jasbi, and S. M. Shieber. Linguistic features for readability assessment. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 1–17, 2020.

S. Ebling, A. Battisti, M. Kostrzewa, D. Pfütze, A. Rios, A. Säuberli, and N. Spring. Automatic text simplification for German. *Frontiers in Communication*, 7:706–718, 2022.

L. Ermakova, E. Sanjuan, J. Kamps, S. Huet, I. Ovchinnikova, D. Nurbakova, S. Araújo, R. Hannachi, E. Mathurin, and P. Bellot. Overview of the CLEF 2022 SimpleText Lab: Automatic simplification of scientific texts. In *International Conference of the Cross-Language Evaluation Forum for European Languages*, pages 470–494. Springer, 2022.

L. Ermakova, E. SanJuan, S. Huet, O. Augereau, H. Azarbonyad, and J. Kamps. CLEF 2023 SimpleText Track: What happens if general users search scientific texts? In *European Conference on Information Retrieval*, pages 536–545. Springer, 2023.

R. Gunning. *The Technique of Clear Writing*. McGraw-Hill, 1952. ISBN 9787000014190. URL `https://books.google.ch/books?id=Zl5bAAAAMAAJ`.

D. Ha, A. M. Dai, and Q. V. Le. Hypernetworks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017. URL `https://openreview.net/forum?id=rkpACe1lx`.

N. Houlsby, A. Giurgiu, S. Jastrzebski, B. Morrone, Q. De Laroussilhe, A. Gesmundo, M. Attariyan, and S. Gelly. Parameter-efficient transfer learning for NLP. In *International Conference on Machine Learning*, pages 2790–2799. PMLR, 2019.

C. Jiang, M. Maddela, W. Lan, Y. Zhong, and W. Xu. Neural CRF model for sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7943–7960, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.709. URL `https://aclanthology.org/2020.acl-main.709`.

M. Johnson, M. Schuster, Q. V. Le, M. Krikun, Y. Wu, Z. Chen, N. Thorat, F. Viégas, M. Wattenberg, G. Corrado, et al. Google's multilingual neural machine translation system: Enabling zero-shot translation. *Transactions of the Association for Computational Linguistics*, 5:339–351, 2017.

J. D. M.-W. C. Kenton and L. K. Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of NAACL-HLT*, pages 4171–4186, 2019.

T. Kew and S. Ebling. Target-level sentence simplification as controlled paraphrasing. In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, pages 28–42, 2022.

J. Kincaid. *Derivation of New Readability Formulas: (automated Readability Index, Fog Count and Flesch Reading Ease Formula) for Navy Enlisted Personnel*. Research Branch report. Chief of Naval Technical Training, Naval Air Station Memphis, 1975. URL `https://books.google.ch/books?id=4tjroQEACAAJ`.

K. Krishna, J. Wieting, and M. Iyyer. Reformulating unsupervised style transfer as paraphrase generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 737–762, 2020.

B. W. Lee and J. Lee. Prompt-based learning for text readability assessment. In *Findings of the Association for Computational Linguistics: EACL 2023*, pages 1774–1779, 2023.

B. W. Lee, Y. S. Jang, and J. Lee. Pushing on text readability assessment: A transformer meets handcrafted linguistic features. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 10669–10686, 2021.

X. L. Li and P. Liang. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4582–4597, 2021.

Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*, 2019.

R. K. Mahabadi, S. Ruder, M. Dehghani, and J. Henderson. Parameter-efficient multi-task fine-tuning for transformers via shared hypernetworks. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 565–576, 2021.

M. Martinc, S. Pollak, and M. Robnik-Šikonja. Supervised and unsupervised neural approaches to text readability. *Computational Linguistics*, 47(1):141–179, 2021.

A. Nighojkar and J. Licato. Improving paraphrase detection with the adversarial paraphrasing task. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 7106–7116, Online, Aug. 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.552. URL `https://aclanthology.org/2021.acl-long.552`.

OpenAI. GPT-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

J. Pfeiffer, I. Vulić, I. Gurevych, and S. Ruder. MAD-X: An adapter-based framework for multi-task cross-lingual transfer. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7654–7673, 2020.

J. Philip, A. Berard, M. Gallé, and L. Besacier. Monolingual adapters for zero-shot neural machine translation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4465–4470, 2020.

E. A. Platanios, M. Sachan, G. Neubig, and T. Mitchell. Contextual parameter generation for universal neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 425–435, Brussels, Belgium, Oct.-Nov. 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1039. URL `https://aclanthology.org/D18-1039`.

A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, et al. Language models are unsupervised multitask learners, 2019. URL `https://d4mucfpksywv.cloudfront.net/better-language-models/language-models.pdf`.

C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *The Journal of Machine Learning Research*, 21(1):5485–5551, 2020.

P. P. Ray. Chatgpt: A comprehensive review on background, applications, key challenges, bias, ethics, limitations and future scope. *Internet of Things and Cyber-Physical Systems*, 2023.

K. Shuster, J. Xu, M. Komeili, D. Ju, E. M. Smith, S. Roller, M. Ung, M. Chen, K. Arora, J. Lane, et al. Blenderbot 3: a deployed conversational agent that continually learns to responsibly engage. *arXiv preprint arXiv:2208.03188*, 2022.

E. A. Smith and R. Senter. Automated readability index. *AMRL-TR. Aerospace Medical Research Laboratories*, pages 1–14, 1967. URL `https://api.semanticscholar.org/CorpusID:38558516`.

T. Tanprasert and D. Kauchak. Flesch-kincaid is not a text simplification evaluation metric. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 1–14, 2021.

S. Vajjala. Trends, limitations and open challenges in automatic readability assessment research. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 5366–5377, 2022.

S. Vajjala and I. Lučić. OneStopEnglish corpus: A new corpus for automatic readability assessment and text simplification. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 297–304, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/W18-0535. URL `https://aclanthology.org/W18-0535`.

S. Vajjala and D. Meurers. On improving the accuracy of readability classification using insights from second language acquisition. In *Proceedings of the seventh workshop on building educational applications using NLP*, pages 163–173, 2012.

A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. Bowman. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, 2018.

A. Warstadt, A. Singh, and S. R. Bowman. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7: 625–641, 2019.

W. Xu, C. Callison-Burch, and C. Napoles. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297, 2015.

L. Xue, N. Constant, A. Roberts, M. Kale, R. Al-Rfou, A. Siddhant, A. Barua, and C. Raffel. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, 2021.

K. Yang and D. Klein. Fudge: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3511–3535, 2021.

B. Zhang, B. Haddow, and A. Birch. Prompting large language model for machine translation: A case study. *arXiv preprint arXiv:2301.07069*, 2023.

T. Zhang, V. Kishore, F. Wu, K. Q. Weinberger, and Y. Artzi. BERTScore: Evaluating text generation with BERT. *arXiv preprint arXiv:1904.09675*, 2019.

A Architecture Variations Supplementary Learning Curves

Hyperparameter Setup

1. **Initial Learning Rate**
   - Default: 0.0003
   - Short Description: The model initial learning rate.
2. **Training Steps**
   - Default: 65000
   - Short Description: The finetuning steps (batches).
3. **LR Scheduler**
   - Default: `linear`
   - Short Description: Which lr scheduler to use.
4. **Temperature**
   - Default: 1
   - Short Description: Defines the temperature value for sampling across the multiple datasets.
5. **Top** K
   - Default: 50
   - Top K words from softmax to sample for generation.
6. **Beam Size**
   - Default: 4

- The sequences considered simulatenously for generation. Beam size = 1 is greedy decoding.

Readability Initialisation

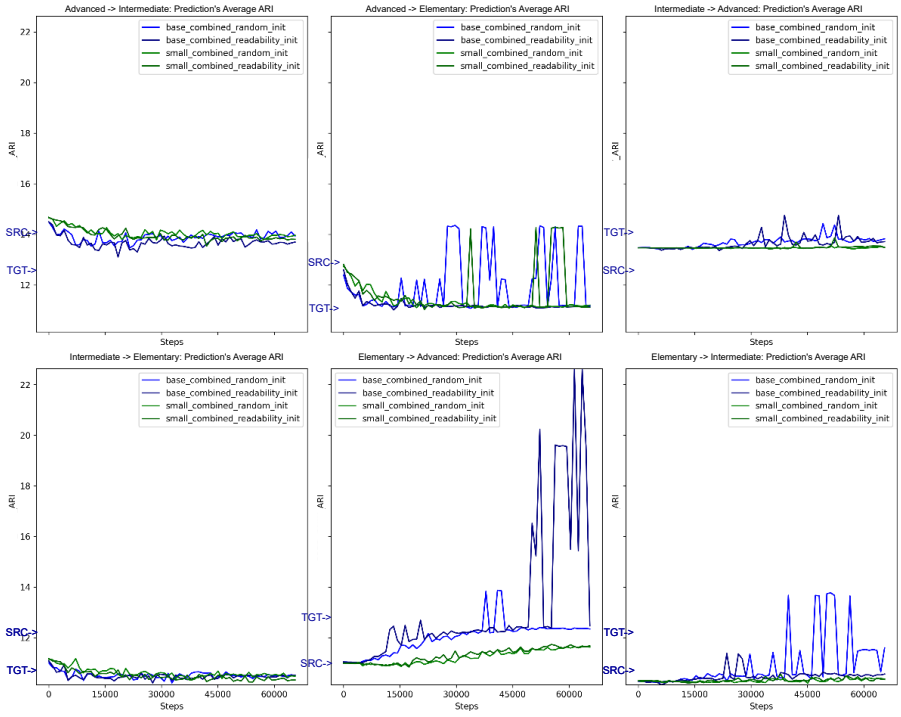

Figure 13: Average Predictions' ARI Random/Readability Initialisation Combined

Separate Task Embeddings Instance for Encoder and Decoder

Frozen Task Embeddings

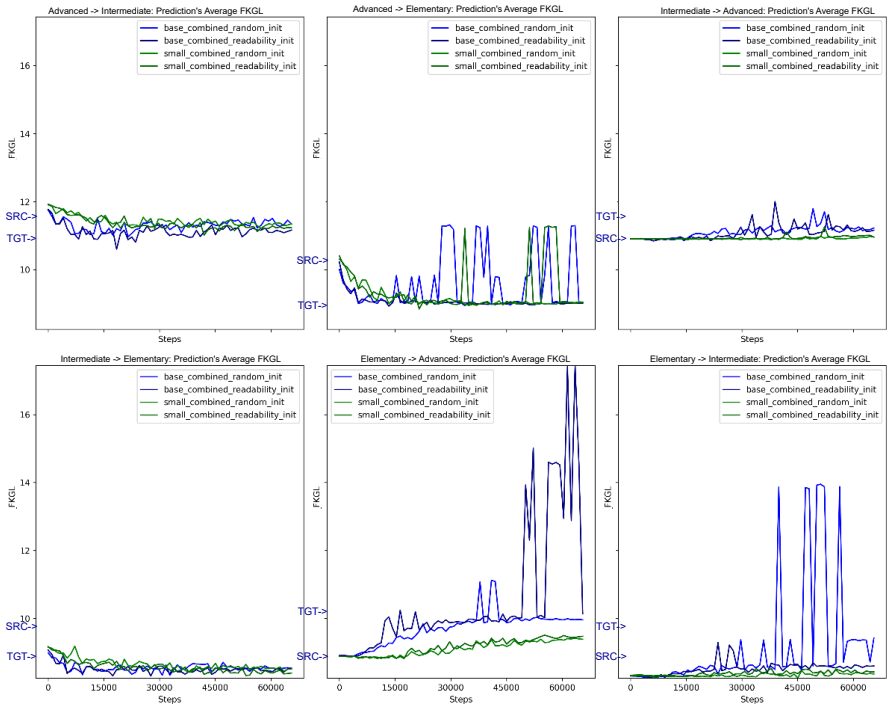

Figure 14: Average Predictions' FKGL Random/Readability Initialisation Combined

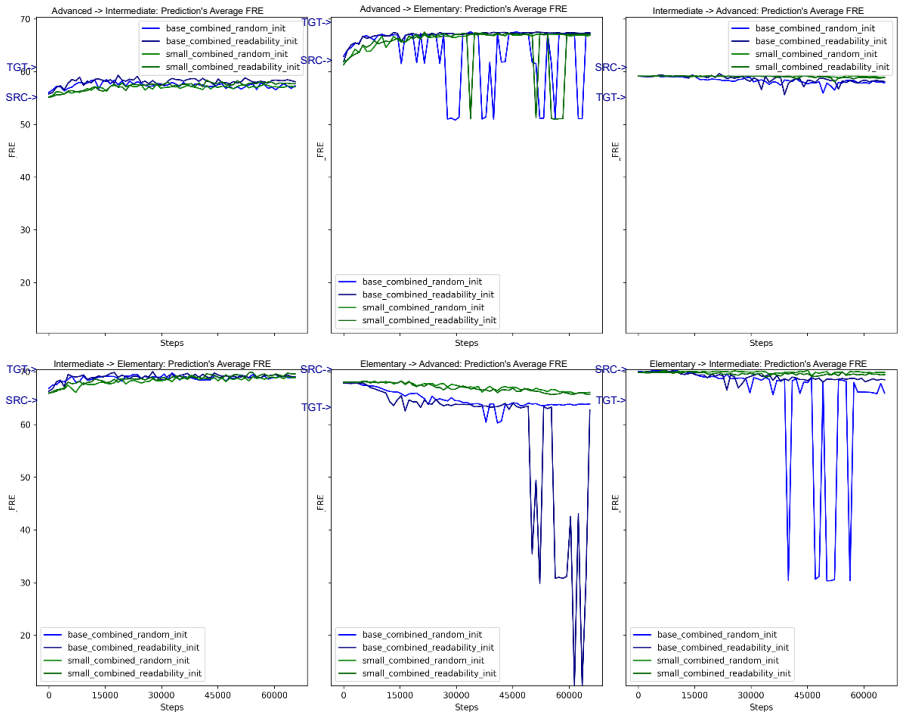

Figure 15: Average Predictions' FRE Random/Readability Initialisation Combined

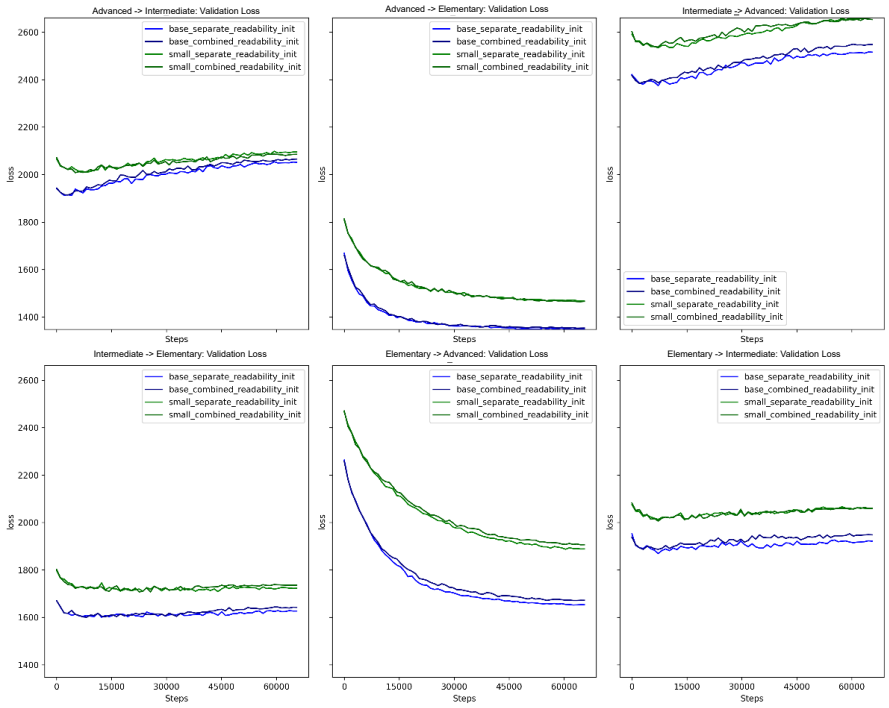

Figure 16: Validation loss of Separate/Combined Embeddings

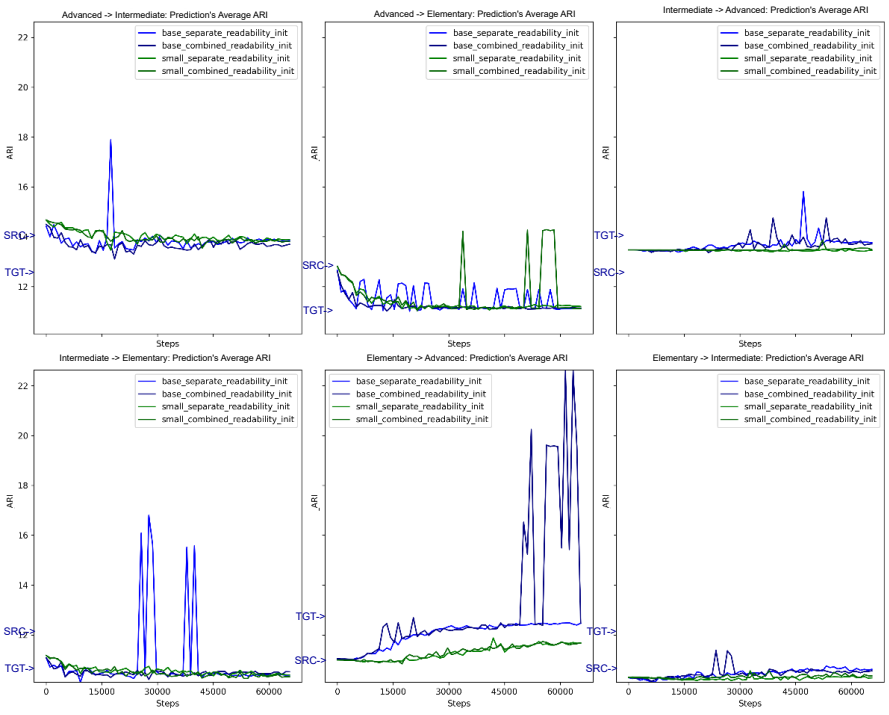

Figure 17: Average Predictions' ARI Separate/Combined Embeddings

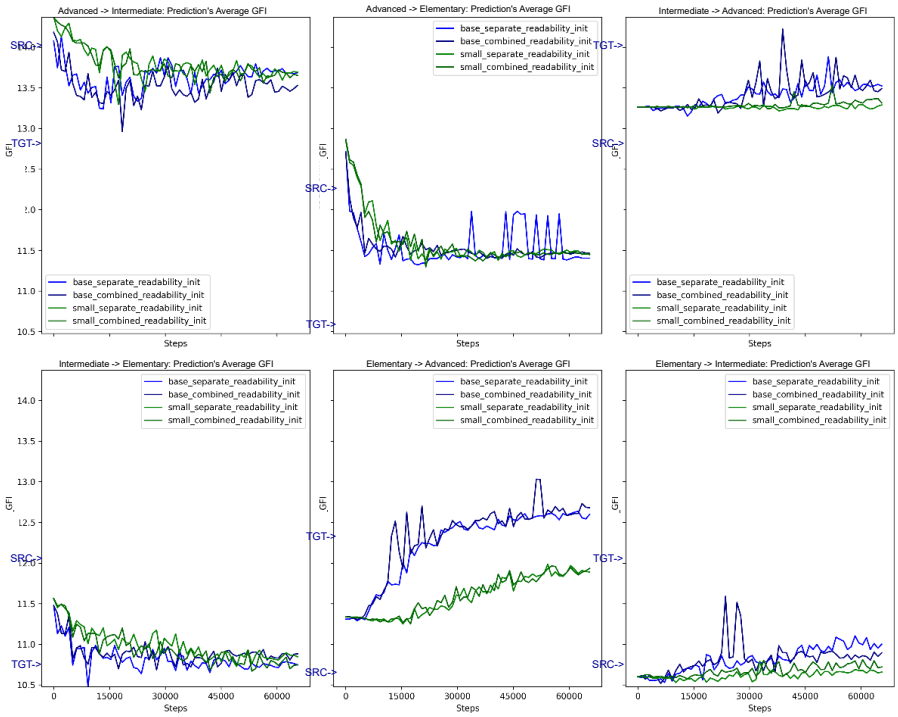

Figure 18: Average Predictions' GFI Separate/Combined Embeddings

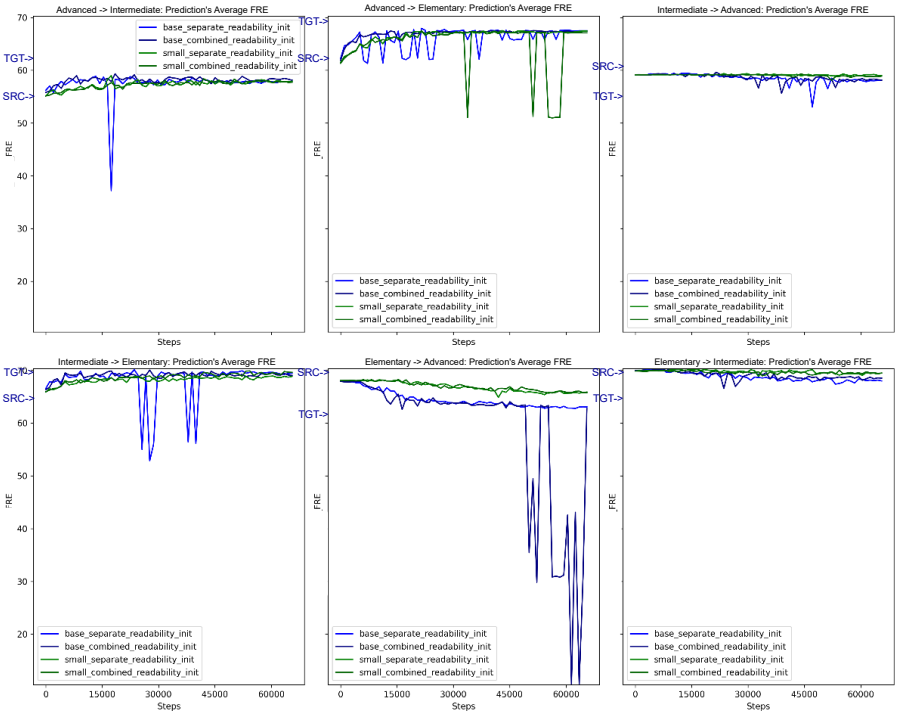

Figure 19: Average Predictions' FRE Separate/Combined Embeddings

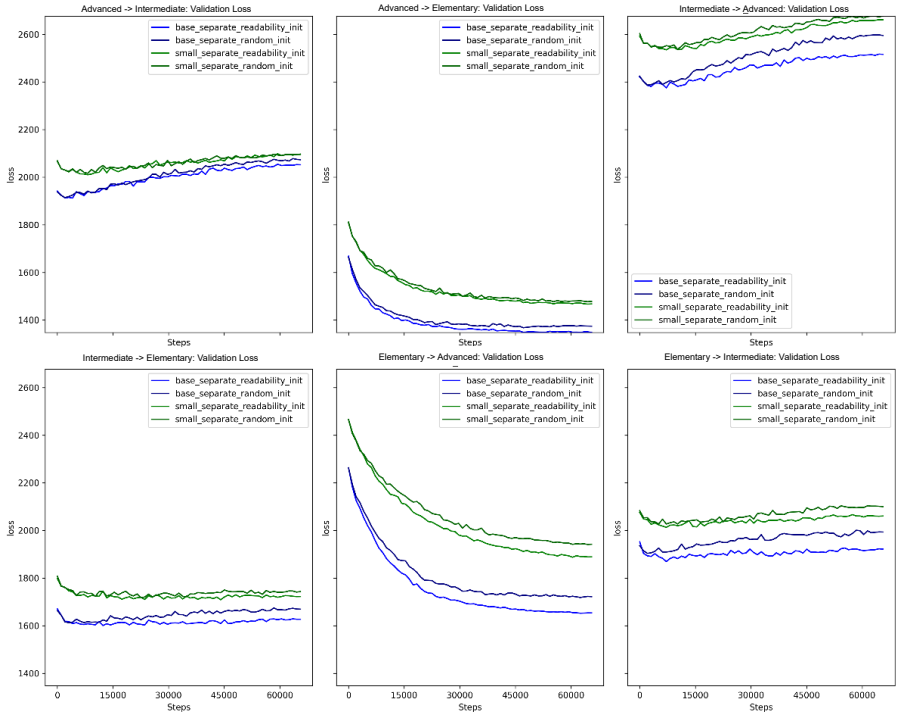

Figure 20: Validation Loss of Random/Readability Initialisation Separate

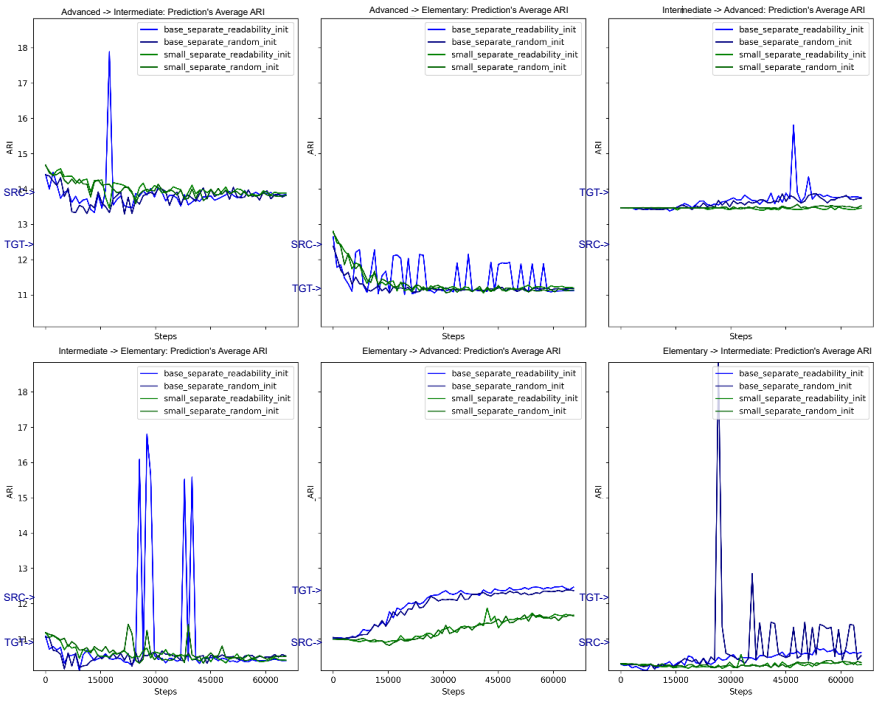

Figure 21: Average Predictions' ARI Random/Readability Initialisation Separate

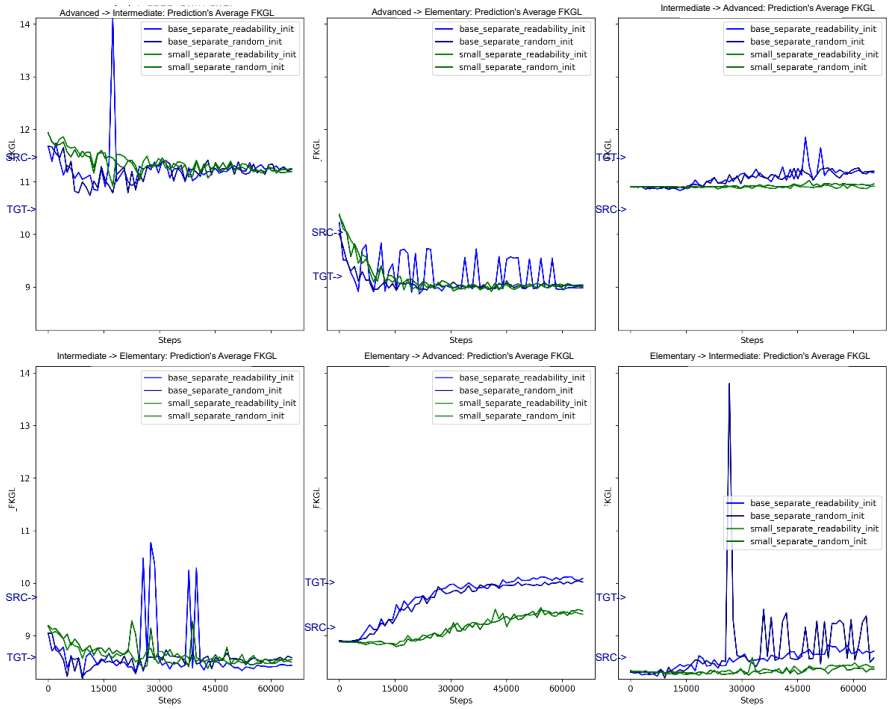

Figure 22: Average Predictions' FKGL Random/Readability Initialisation Separate

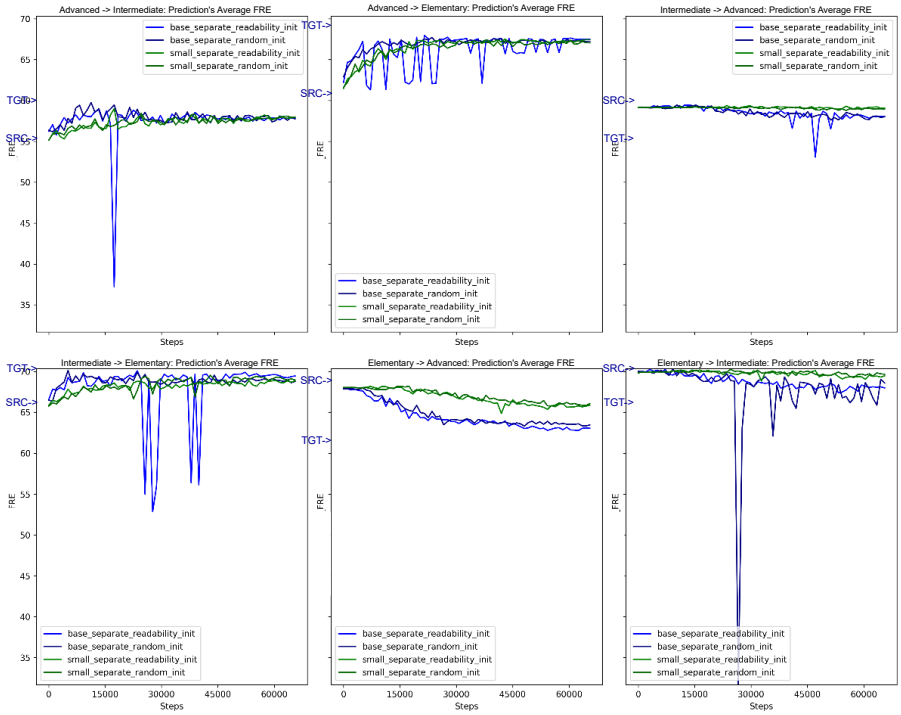

Figure 23: Average Predictions' FRE Random/Readability Initialisation Separate

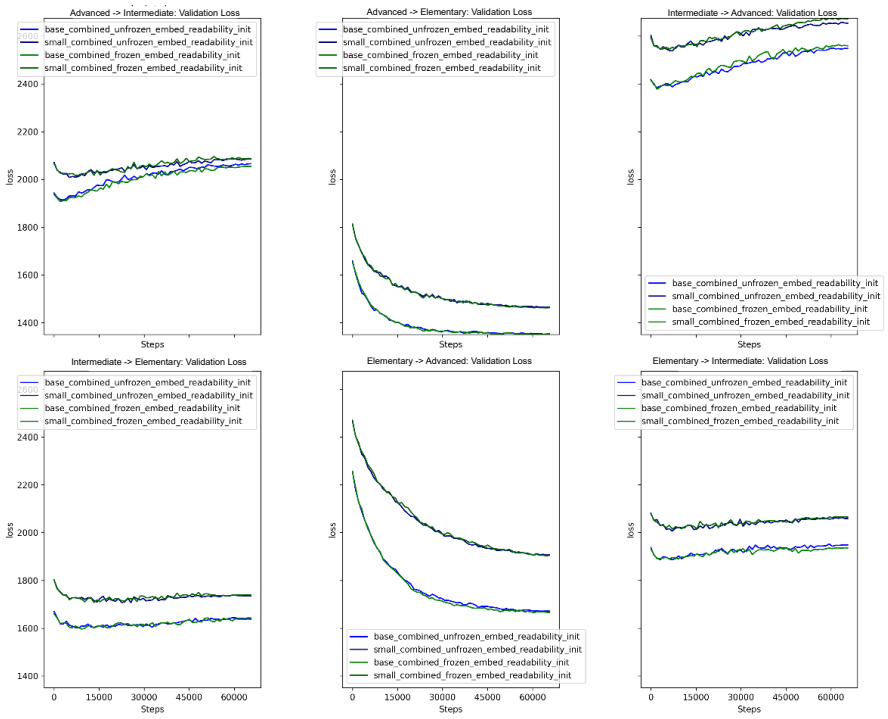

Figure 24: Validation Loss of Frozen/Unfrozen Combined Readability Initialisation

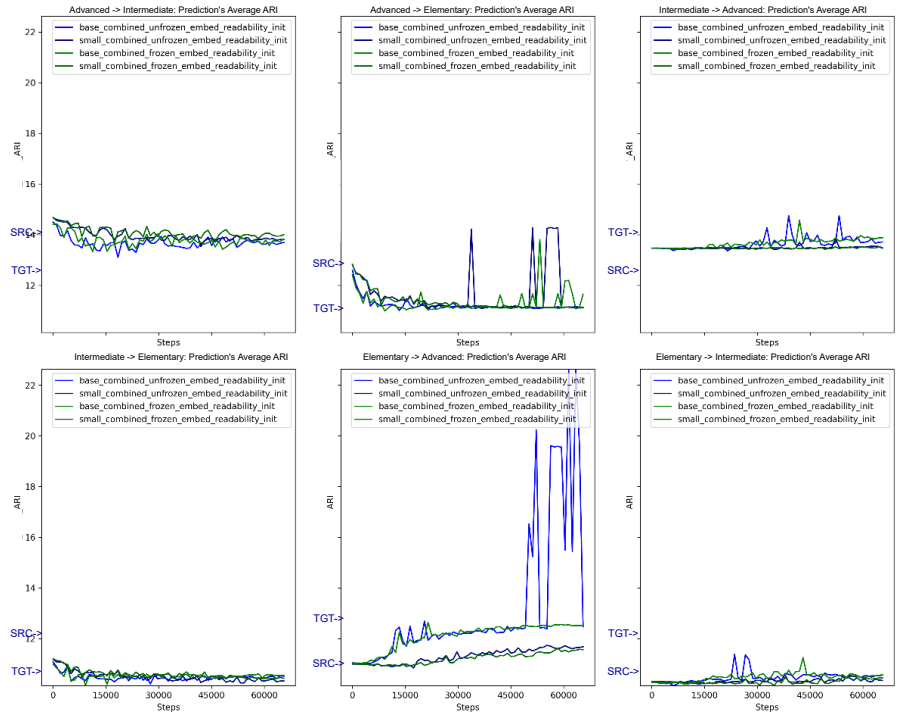

Figure 25: Average Predictions' ARI of Frozen/Unfrozen Combined Readability Initialisation

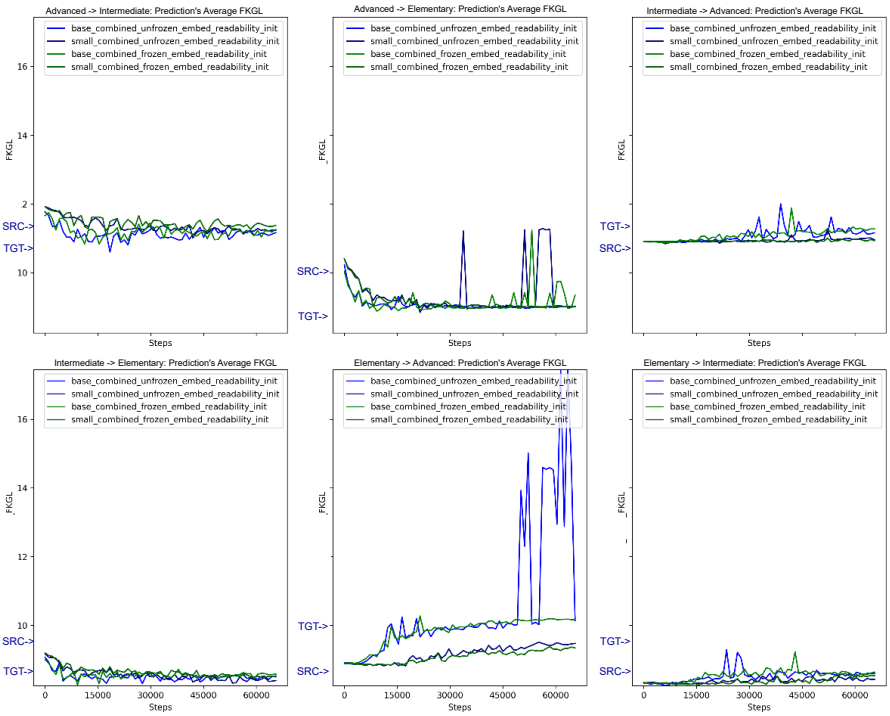

Figure 26: Average Predictions' FKGL of Frozen/Unfrozen Combined Readability Initialisation

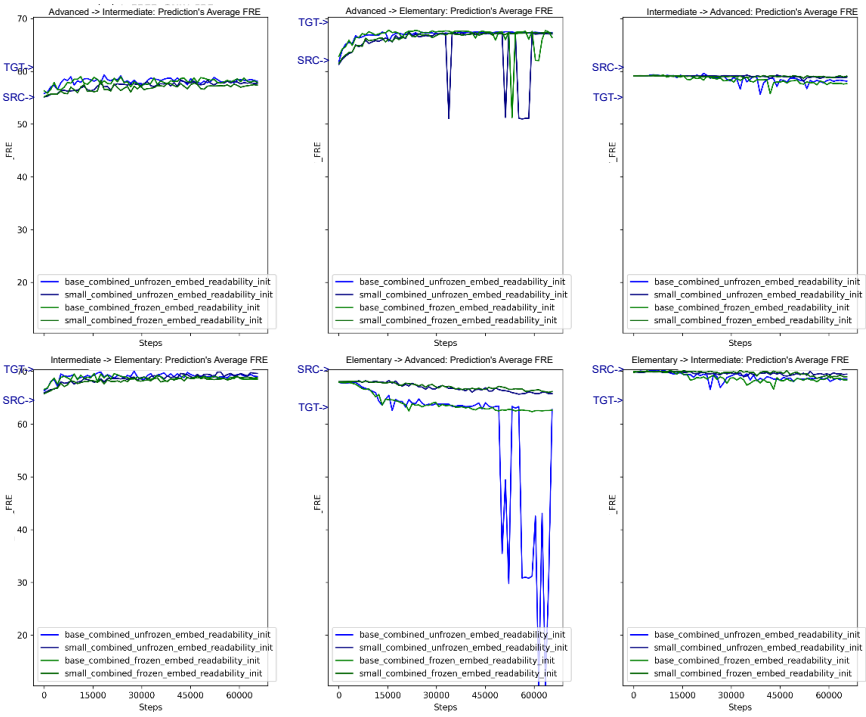

Figure 27: Average Predictions' FRE of Frozen/Unfrozen Combined Readability Initialisation

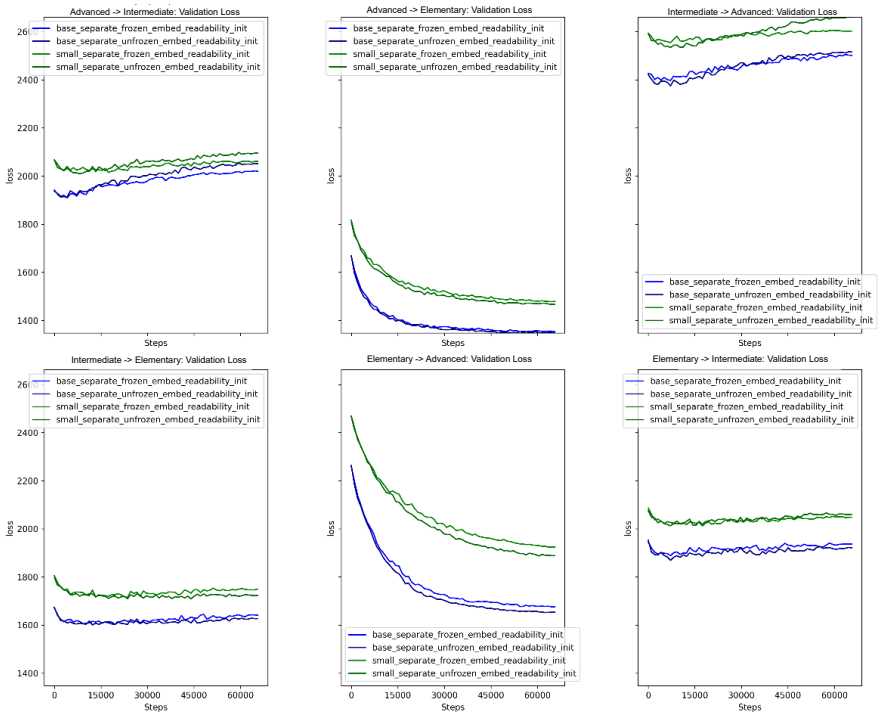

Figure 28: Validation Loss of Frozen/Unfrozen Separate Readability Initilisation

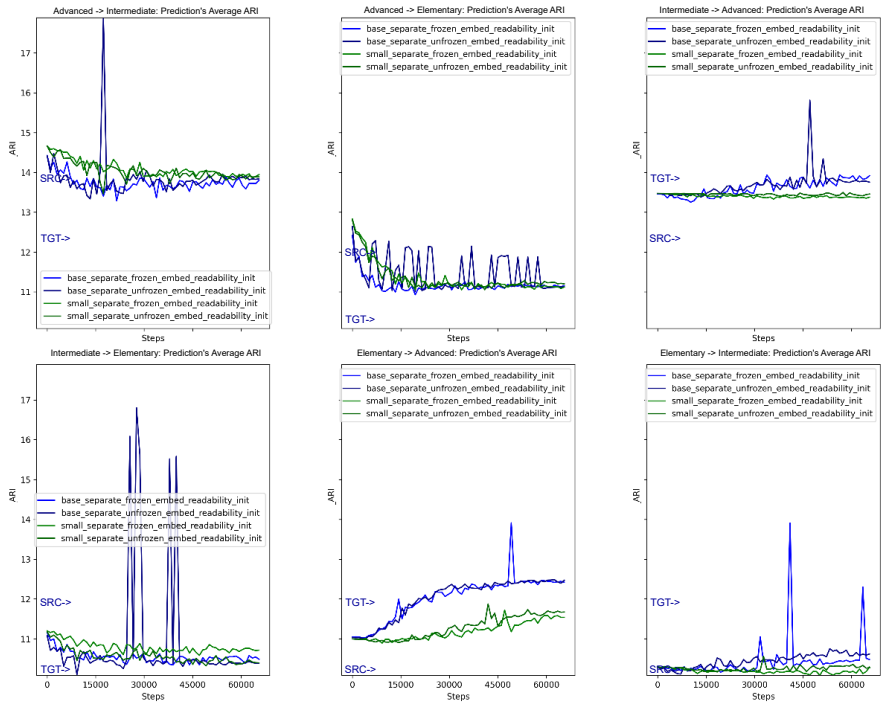

Figure 29: Average Predictions' ARI of Frozen/Unfrozen Separate Readability Initialisation

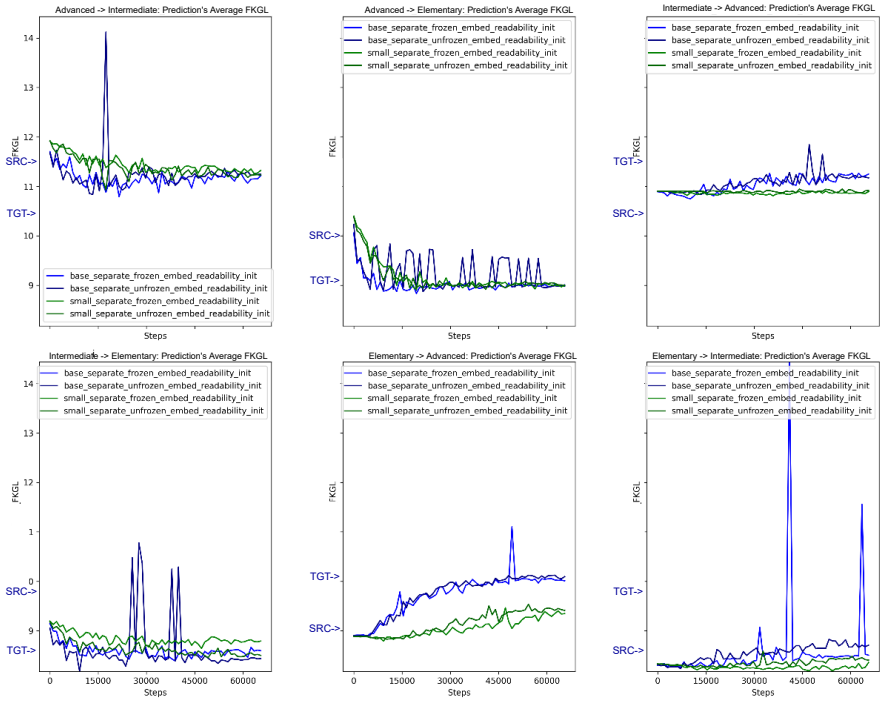

Figure 30: Average Predictions' FKGL of Frozen/Unfrozen Separate Readability Initialisation

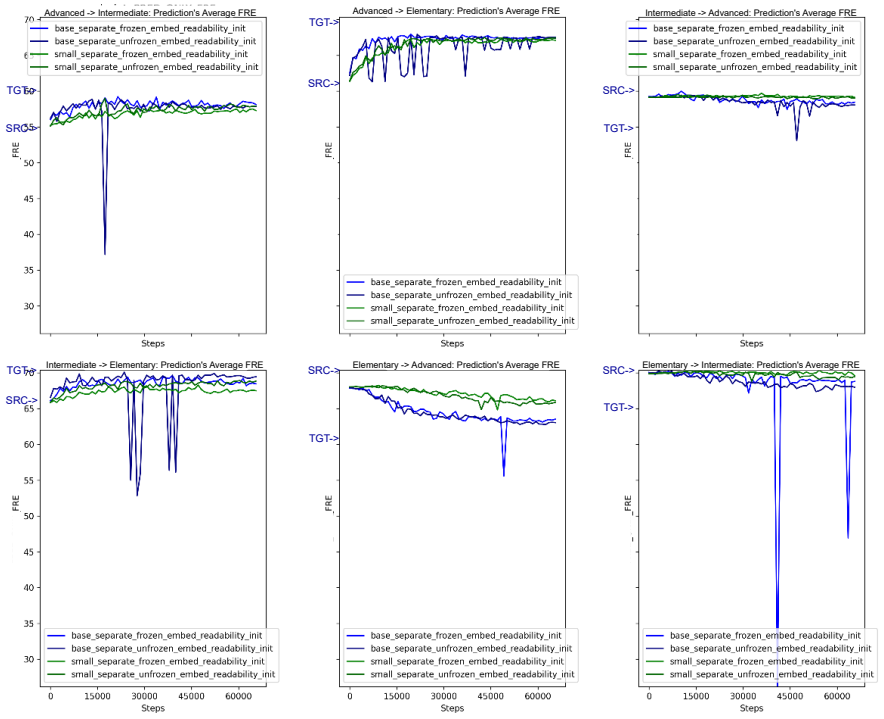

Figure 31: Average Predictions' FRE of Frozen/Unfrozen Separate Readability Initialisation