Sentence Simplification using Syntactic Features and T5 model

SARA SARA MICHAEL IYASU, University of Twente, The Netherlands

Syntactic simplification seeks to alter a sentence's grammatical structure while ensuring its meaning is maintained, thereby making it more understandable for a wider audience. Despite being trained on extensive datasets, large language models mainly capture broad language patterns and often find it challenging to maintain nuanced meanings in complex sentences. This limitation can result in oversimplification or ambiguity. Therefore, it is crucial to strike a proper balance between simplification and correctness.

This research introduces an improvement to the attention mechanism by computing attention scores based on both hidden states and the grammatical relationships between words, rather than relying solely on hidden states as traditional attention mechanisms do. The paper also explains a second approach that incorporates a grammatical embedding layer to enhance the model's understanding of grammatical structure.

The paper investigates various methods for incorporating explicit grammatical information into the model. It adopts two strategies: one focuses on integrating grammatical information, while the other emphasizes fine-tuning the model for specific downstream tasks, which in this case is syntactic simplification. Fine-tuning the T5 model without explicit grammatical information yielded best results, with a rouge-1 score of 0.96, rouge-2 score of 0.9266, and R-L score of 0.9554 evaluated by ROUGE score.

Additional Key Words and Phrases: Syntactic simplification, attention scores, large language models, attention mechanism, T5, hidden states, embedding layer, BERT score

1 INTRODUCTION

Sentence Simplification is the process of reducing the complexity of a sentence to make it easier to understand. This process involves various types of simplification, including lexical simplification, paraphrasing, amount of compression and syntactical simplification [11]. In most cases, both lexical and syntactic simplification work hand in hand.

Lexical Simplification refers to replacing complex words with simpler alternatives. This is especially helpful for a wide range of audiences, such as people with language disabilities [3], cognitive impairments (e.g., aphasia) [2] or second-language learners, as it reduces the cognitive load required to understand the content. Sentences with intricate grammatical structures can be simplified by rephrasing or restructuring them, thereby reducing their complexity. However, this research will specifically focus on the implications of syntactic simplification. Sentence simplification using syntactic features emphasizes restructuring sentences rather than altering individual words, as is typically done in lexical simplification. The primary aim is to make sentences easier to understand while maintaining their original meaning. For example, Table 1 provides an illustration of syntactical simplification. Additionally, it can enhance the performances of many NLP tasks such as grammatical error correction, parsing, machine translation [4] [18], and so on. This research primarily focuses on simplifying complex sentences by breaking down long sentences with multiple clauses into shorter, standalone sentences, often by removing conjunctions and connectors.

Table 1. Complex and Simple Sentences

Complex: As soon as the game ended, they celebrated their victory.
Simple : The game ended. They celebrated their victory.

This research uses a large language model known as T5. LLM models employ attention mechanisms to identify relationships between different parts of a sequence. These mechanisms enable the model to concentrate on the most important parts of the input, allowing for a more adaptable and context-aware language generation and comprehension [14].

Large language models (LLMs) are not specifically designed to grasp grammar. Instead, they acquire patterns throughout the training process but lack the in-depth understanding of grammar that models trained directly with grammatical frameworks possess. During the pre-training process, LLMs may indirectly pick up grammatical structures while trying to identify patterns that reveal the relationships between words. Despite this, models like BERT, a large language model, have demonstrated that certain attention heads can accurately identify some grammatical relationships with an accuracy of 75% or higher [5]. Although large language models are not explicitly trained to understand grammar, different layers show varying levels of performance on specific grammatical tasks. This raises an important question: could incorporating explicit grammatical information further improve the performance of LLMs?

In this research, the model will be fine-tuned to perform syntactic simplification. A key part of this process is understanding how each word is grammatically connected to other words in a sentence. To achieve this, dependency parsing will be used, a natural language processing technique that studies the grammatical structure of sentences by identifying the relationships between words. The output of dependency parsing is usually a dependency tree or graph, which visually represents the syntactic connections in a sentence.

By incorporating additional features, such as grammatical information represented through a graph, can provide the model with enhanced contextual awareness. This approach enables the encoder to concentrate more effectively on grammatical attributes. This paper will show how grammatical bias will be integrated into the model.

2 PROBLEM STATEMENT

While large language models (LLMs) like BERT acquire a considerable understanding of grammar during their pre-training, they do not fully grasp it, as their understanding of grammar is largely an unintended consequence of the training process [5]. This study emphasizes that incorporating dependency trees structures that illustrate the grammatical relationships between words in a sentence can enhance the syntactic simplification task. By constructing

TScIT 42, January 31, 2025, Enschede, The Netherlands

a graph and using it as an additional feature in an encoder, this method can help identify which elements of a sentence should be removed, restructured, or kept.

2.1 Research Question

How can dependency trees be used to capture syntactic relationships between words and be integrated as additional input to an encoder-decoder model for generating syntactically simplified sentences?
Sub-Research Questions (RQ):

(1) What techniques can be employed to incorporate graph-based syntactic information into sequence-to-sequence models?
(2) How can semantic information derived from the model's attention mechanism be balanced with explicit syntactic biases provided by graph-based inputs to improve grammatical correctness and fluency?

3 RELATED WORK

This section will go through some of the related work in syntactical simplification using various techniques.

(1) In their 2020 paper, Martin et al. [11] have employed several methods to simplify sentences, focusing on four main attributes: the amount of word compression, paraphrasing, lexical simplification, and syntactical simplification. They have used control tokens as an additional input along with the sentences.
(2) In their work, Ma et al. [10] used matrices where each entry consists of relation vectors. These vectors represent either the grammatical relationship between two tokens or the number of steps each token takes to reach their ancestor in the dependency tree. Additionally, they incorporate a learnable gating mechanism to combine these relation vectors with the original attention scores that are computed.
(3) In their work, Bai et al. [1] different matrices were employed in this study, with each entry representing various types of grammatical distances between tokens. The use of distinct matrices allowed for the generation of varied attention outputs; these outputs were subsequently added together and transmitted to an additional attention layer for further processing. In this subsequent layer, the outputs were used as keys and values to compute a new attention output.

4 BACKGROUND INFORMATION

4.1 T5 model

The T5-small model is a transformer-based encoder-decoder architecture that looks like the Transformer proposed by Vaswani et al. [14]. It consists of blocks, each of which includes a self-attention layer, followed by an add-and-norm layer and a feed-forward layer [13]. T5-small comprises 12 blocks, with 6 blocks in the encoder part of T5-small. Within each block, there are eight attention heads. Each attention head operates independently to capture different aspects of the relationships or dependencies between elements in the input sequence. The outputs from these heads are then combined and further processed to enhance the model's understanding.

The decoder is similar to the encoder but includes an additional attention mechanism known as cross-attention after each self-attention layer, which focuses on the encoder's hidden state that later acts as query and key in the decoder part. Additionally, the decoder uses a unique form of attention called auto-regressive self-attention, which limits the model to attending only to tokens that come before the current position and everything that comes after this current token is masked [13]. In every attention layer, the input is transformed into query (q), key (k), and value (v) vectors through separate linear transformations. The attention score is then calculated as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{1}$$

Instead of using a single attention head to determine which parts of the input are important, the model uses multi-head attention. Each head focuses on different parts and aspects of the data. By projecting the data into different subspaces, the model captures different meanings within these contextual subspaces [14]. However, the exact role or function of each subspace or head is not explicitly understood.

4.2 How much Grammar do LLM models know?

Many studies have used techniques like probing [12], weight pruning [17], and other methods to investigate the roles of individual attention heads in transformer models. These studies aim to identify which specific linguistic features each head focuses on and how different heads contribute to various aspects of language processing. For instance, in their work, Voita et al. [17] used Layer-wise

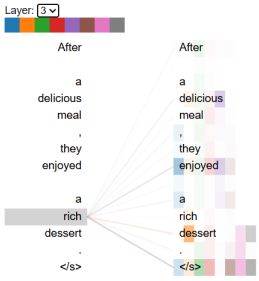

Fig. 1. head view in layer 3

Relevance Propagation (LRP) to measure how much each head contributes to the model's final decision. In a transformer model, each layer learns different types of relationships among words because each attention head has its own set of weights, and they receive different updates during the training process, leading to varying performance. Additionally, attention distances differ among attention heads, even within the same layer, due to different attention mechanisms. Some heads may focus on capturing relationships between words that are far apart, while others might concentrate on connections between nearby words. As shown in Fig. 1, is an example of attention being computed in layer 3 visualized using Bertviz [15], where thicker lines represent stronger relationship of a word with the rest of the words in the sentence. For instance, Jawahar et al. [8] indicated that higher layers, excel over lower layers when sentences include a significant distance between the subject and the verb. This implies that the higher layers are better at capturing complex syntactic relationships, especially those involving long-range dependencies. In contrast, the middle layers seem to focus on

simpler grammatical structures, emphasizing more straightforward syntactic patterns. In their paper Vig and Belinkov [16], they tried to understand what each attention head does by examining the mean attention distance, mean attention entropy, and attention variability of each head. For instance, if a head has high variability and high entropy how good is it at capturing grammar by distributing its attention across tokens.

The research presented in [5] has shown that different attention heads focus on different types of grammatical relationships, all with varying levels of accuracy. For instance, there is an 86.8% accuracy for the dobj relation between tokens in heads 8-10 in BERT, while noun modifiers attend to their nouns with a 94.3% accuracy at the det relation. All these results were obtained by learning patterns from large datasets during the pre-training process. Additionally, a study conducted by Htut et al. [6] has shown that there are some heads that focus on specific dependency types. However, there are no heads capable of comprehending all relationships, meaning none are specialized in comprehensive parsing or fully understanding the entire syntactic structure of a sentence.

Despite not being pre-trained on specific grammar tasks, large language models have shown good performance in this area.

4.3 Dependency Parsing

In this study, sentences will be pre-processed using dependency parsing, by using well-known Python library known as SpaCy to analyze grammatical relationships between words. This process produces a tree-like structure that represents the syntactic relationships in the sentence. The tree begins with a root node, typically the main verb of the sentence, which has no incoming arcs. All other words have exactly one incoming arc [9]. The tree expands by forming edges, which denote grammatical relationships between words. When two words are connected by an edge, it indicates a dependency, with one word being grammatically dependent on the other. An illustrative example of how a dependency tree is formed is shown in Fig. 4.

5 METHODOLOGY

This section outlines the approaches taken to address the research question.

Initially, the paper will evaluate the model without incorporating any grammatical features, fine-tuning it to understand its performance in this simplified state. Following this evaluation, the functionality of the model's attention heads will be analyzed by assessing both the average score and variance for each head. Based on these findings, the least important attention heads will be pruned.

Finally, grammar will be integrated into the model using two distinct methods: one will involve a matrix, while the other will utilize grammatical embeddings. By the end of the study, the performance of the model with explicit grammar inclusion to that of the model without grammar integration will be compared. The initial approach with grammar involved adjusting the attention layer to more effectively capture the dependency relationships between tokens. Before inputting the (seq_len, seq_len) matrix, where seq_len represents the sequence length, the relationships among words are first analyzed and then converted to grammatical distances and are filled into a matrix. If two words are directly connected (i.e., sharing an edge in the dependency tree), a grammatical distance of one is assigned to highlight the importance of their relationship. For words that are not directly connected, the total number of steps each token would need to take in order to reach the lowest common ancestor is computed. This is fed to the attention-layer which is later either added or multiplied with the score computed in the attention-layer before passing it to the softmax function. As shown in Fig. 2 are the changes made in the attention layer of T5.

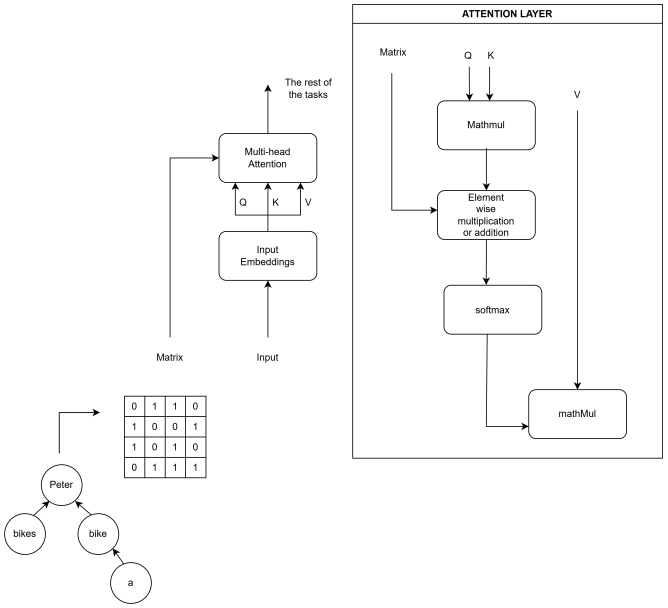

Fig. 2. matrix in the attention layer

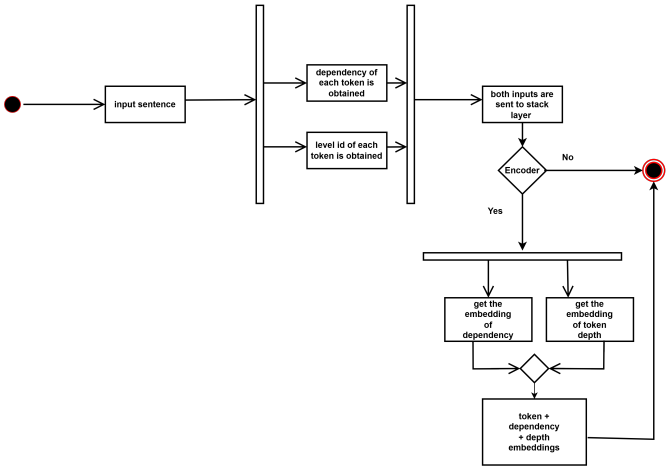

Fig. 3. Activity diagram of dependency and depth embeddings

For the second approach, two types of embeddings are created to be added to the token embeddings: dependency embeddings, which represent the embeddings of dependency relationships, and depth embeddings, which represent the embeddings of the depth of the tokens in the dependency tree. Each token's level in the tree and its relationship with its head are both sent to the stack layer. If it is

an encoder, both embeddings are obtained and added to the input embeddings to create new embeddings, which will then be sent to downstream tasks for further processing. Fig. 3 is an illustrative example of the process flow.

6 EXPERIMENT

6.1 Pruning Compared to Not Pruning

The model will initially be evaluated without considering pruning. The T5-small model consists of 6 encoder blocks, each containing 8 attention heads, resulting in a total of $6 \times 8 = 48$ attention heads in the encoder stack. Pruning will only be done when the model is being fine-tuned, with no grammar information taken into account in order to see if it has an effect.

To evaluate the attention score for each head in a layer. The following steps were taken:

- **Average Attention Score**: Compute the average attention score for each head for all the inputs
- **Variance of Attention Score**: Indicates how variant each attention score of a head is.

Referring to the methods used by Vig and Belinkov [16], they analyzed the roles of each head using the above approaches, along with other methods such as entropy. For instance, they utilized variability to determine which heads focused on position-based relationships rather than dependency-based ones. Therefore, high variability indicates a non-position-based relationship. They also employed entropy to assess whether the attention was more scattered or concentrated. This approach provided insights into which heads were most important within each layer. However, they used these methods to understand the function of each attention head. This paper will sum both values to determine which heads are unnecessary so that they can be pruned during inference time.

6.2 Dependency Matrix

The model's architecture has been extended to accept an additional input, namely a matrix, This matrix encodes grammatical information about the input sentence. The modified model now takes four main inputs:

- **Input**: The tokenized input sentence.
- **Attention**: It tells which parts of the input should be focused on and which parts are simply padding that can be ignored.
- **Labels**: The ground-truth data used by decoder during training.
- **Matrix**: A matrix of size (batch_size, seq_len, seq_length) that represents the grammatical relationships between tokens in the input sentence.

Everything is pre-processed before sending the matrix to the encoder's self-attention layer. The matrix size should be equal to the sequence length of the input_id. The initial approach involves creating a matrix of seq_len by seq_len, but only filling the matrix with the grammatical relationships of words as the first step. This is necessary because SpaCy processes entire words, while a typical tokenizer breaks words into subwords when the word is not in its vocabulary. To address this issue, words are tokenized, and a dictionary is created to map to its corresponding subword components.

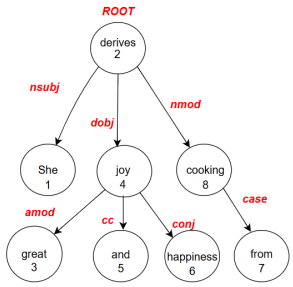

Fig. 4. dependency tree

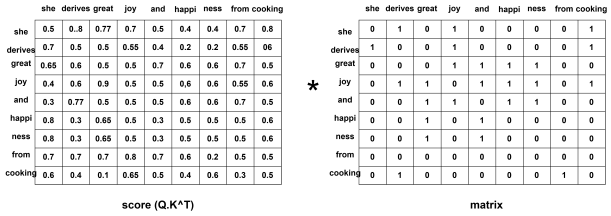

Fig. 5. matrix computed from dependency tree

For example, the word "Cinderella" is split into "Cin," "der," and "ella." Since the matrix already contains an entry for "Cinderella," this entry is copied for both the rows and columns, duplicating it $n - 1$ times, where n represents the total number of subwords. This approach is employed because the subwords are considered parts of the same word. As shown in Fig. 5 is an illustrative example of duplicating rows and column entries of subwords of a word.

In most studies, an adjacency matrix is used, typically focusing on the dependent word "attending" to the head, where [dependent, head] = 1 when there is a direct edge, and the reverse [head, dependent] = 0. According to Clark et al. [5], The attention scores computed by BERT, The dependent word tends to "attend" on the head word instead of the reverse, because each dependent is associated with only one head, whereas heads can have several dependents. However, in this study, matrix is expanded to explore whether reversing this relationship, so that both [dependent, head] = 1 and [head, dependent] = 1, could also have an impact. Introducing reverse dependencies is important because it may provide a more comprehensive understanding of word relationships. Both directions offer unique semantic insights. For example:

- nsubj : "The dog" is the subject of "eat
- opposite : "eat" is the predicate of the "the dog"

After constructing the matrix, it is refined further by applying a threshold. Any relationship that exceeds this threshold is considered grammatically distant and is disregarded. On the other hand, relationships below the threshold are considered grammatically close and are retained. The established threshold is a total sum of 4. This value was selected to ensure that the grammatical distances remain relatively close. If a token is a direct ancestor of another token, the step count is 1, as they are connected by a direct edge. However, if two tokens are not connected by a direct edge, their combined total

steps must equal 4 or less for them to be considered grammatically close.

So, instead of passing the score computed by $Q \cdot K^\top$ to the softmax function. The T5 model has been modified to accept a new parameter matrix, which is of size (batch_size, seq_len, seq_len). This is repeated 8 times so that it can be used by all the attention heads. The new shape of the matrix is (batch_size, num_heads , seq_len, seq_len) where num_heads represents the number of heads in each T5's attention layer. This method was only implemented in the encoder and not the decoder because the hidden state of the encoder will be passed to the cross-attention layer of the decoder.

Four different methods were used to handle this:

- **score * matrix :** Any value below the threshold is assigned a 1, while values above the threshold are set to 0. This creates a matrix of 1s and 0s, which is then multiplied element-wise with $Q \cdot K^\top$. This process is applied across all 8 heads in each attention layer. As a result, the attention computed is effectively modified by the matrix if it is not considered grammatically significant. In other words, it nullifies the attention scores for tokens that are not grammatically related

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top * \text{matrix}}{\sqrt{d_k}}\right)V \tag{2}$$

- **score + matrix :** Any value below the threshold is set to 0, while any value above it is set to negative infinity. This results in a matrix consisting of 0s and negative infinity values, which is then added element wise with $Q \cdot K^\top$.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^\top + \text{matrix}}{\sqrt{d_k}}\right)V \tag{3}$$

- **Gating mechanisms on scores :** Instead of fully discarding the score, this method provides the model with the flexibility to choose between score and bias_score. Here, bias_score is derived using Eq (2), and score is the original score computed by the attention layer. δ is initially set to 0.7

$$\text{adaptive\_score} = (1 - \delta) \cdot \text{bias\_score} + \delta \cdot \text{score} \tag{4}$$

- **Gating mechanisms on attention outputs :** In the final approach, instead of employing a single matrix and using only one attention output, two matrices are used. The concept of using multiple matrices for relationships was inspired by the work of Bai et al. [1]. However, this approach has been modified by introducing a gating mechanism to combine outputs from the different attention mechanisms based on the chosen two matrices. Instead of adding another attention layer, as done in their work, this paper stops the process here. The first matrix has binary entries for tokens that are directly connected, while the second matrix includes all tokens that are not directly connected; for these tokens, the entries represent the sum of the steps required to reach their common ancestors. Additionally, a threshold of 4 was chosen. The approach has been modified by introducing a gating mechanism to combine the outputs from the different attention mechanisms computed based on the two matrices. A learnable parameter α was introduced to enable the model to dynamically adjust its preference between two attention outputs. Since different layers of the model focus on varying levels of grammatical distance, this parameter provides greater flexibility, allowing the model to adapt and ensure grammatical correctness across layers. α is set to 0.5 in all layers in the beginning with a learning rate of 1e-4. Overtime, the values change in different layers reflecting what the layer focuses on. Consequently, the final output is generated as follow.:

$$\text{attention\_output1} = \text{softmax}\left(\frac{QK^\top * \text{matrix1}}{\sqrt{d_k}}\right)V \tag{5}$$

$$\text{attention\_output2} = \text{softmax}\left(\frac{QK^\top * \text{matrix2}}{\sqrt{d_k}}\right)V \tag{6}$$

$$\text{attention} = \alpha \cdot \text{attention\_output1} + (1 - \alpha) \cdot \text{attention\_output2} \tag{7}$$

6.3 Dependency Embedding

- **Input**: The tokenized input sentence.
- **Attention**: It tells which parts of the input should be focused on and which parts are paddings that can be ignored.
- **Labels**: The ground-truth data used by the decoder during training.
- **Dependency Relationship**: Represents the relationship between a token and its direct ancestor, with a size of (batch_size, seq_len).
- **Level IDs**: Represents the depth of the token in the dependency tree(batch_size, seq_len).

By examining the dependency relationships in SpaCy, 45 distinct grammatical relationships were used. To incorporate these into the model, a separate embedding layer was created for these relationships, keeping it separate from the word embedding layer. This approach ensures that the grammatical relationships are placed in their own embedding space, allowing similar relationships to cluster in the same subspace during training. For each token, both its dependency (word → head) and its position within the syntactic tree are taken into account. Using depth embedding was inspired by the work of Ma et al. [10]. However, this approach has been modified in this paper by incorporating dependency embedding as well. As a result, both the token's depth in the tree and its dependency relationship are represented in a higher-dimensional space, specifically with a dimension of 512. The new embedding layers for both level_ids and dependency relationships are initialized with dimensions that match the hidden size of T5-small. Both of their weights are randomly initialized using He initialization further processing

$$\text{embedding} = \text{token\_embedding} + \text{depend\_embedding} + \text{level\_id} \tag{8}$$

7 EVALUATION

7.1 Evaluation Metrics

The accuracy was evaluated using two evaluation metrics which are ROUGE and Bert Score

(1) **ROUGE-1**: This metric evaluates the overlap of individual words between the predicted sentence and the ground truth. It

measures how many uni-grams from the ground truth appear in the predicted sentence.

(2) **ROUGE-2**: This metric evaluates the overlap of pairs of consecutive words between the predicted sentence and the ground truth. It calculates how many bi-grams from the ground truth also appear in the predicted sentence.

(3) **ROUGE-L**: This metric measures the Longest Common Subsequence (LCS) between the predicted sentence and the ground truth. The LCS is the longest sequence of words that appear in both the predicted sentence and the ground truth while preserving order.

(4) **BERT Score**: This metric assesses the similarity of each token in the predicted sentence compared to those in the ground truth. Similarity between tokens is evaluated using contextual embeddings. Additionally, the method employs greedy matching to enhance the score by aligning each token in the predicted sentence with the most similar one in the reference based on embeddings. [19]

(5) **Event overlap Score**: This metric evaluates the similarity of each token in the predicted sentence to the corresponding token in the ground truth, calculated on a per-event basis. For example, if the ground truth consists of three sentences and the prediction also contains three sentences, the token similarity is computed individually for each pair of sentences. The number of matching words is obtained and divided by the total number of words in the ground truth for each sentence. Once this is computed, the final score is obtained by summing each score and dividing it by the total number of events in the ground truth. For instance, (0.63 + 0.73 + 0.33) / 3

$$\text{Event Overlap Score} = \frac{\sum(\text{overlap\_per\_event})}{\text{len(events\_in\_ground truth)}} \tag{9}$$

(6) **Event split Score**: This metric assesses the total number of events identified in the generated text and compares it to the number of events in the reference sentence. Here, p represents the number of events in the predicted sentences, while r refers to the number of events in the reference sentence.

$$\text{Event Split Score} = \frac{\min(p, r)}{\max(p, r)} \tag{10}$$

7.2 Experimental Setup

For this experiment, a dataset of 1,000 of sentences was used The sentences were divided into training, validation, and test sets with 80%, 10%, and 10%, respectively. The dataset consists of pairs of complex and simple sentences. The complex sentences use connectors like "Before," "After," and "When," while the simple sentences remove these connectors to present a sequential order. In other words, the simple sentences consist of distinct events or actions.

Low-Rank Adaptation and PEFT: The T5-small model comprises of about 60 million parameters [13]. Fine-tuning this pre-trained model can pose challenges due to hardware limitations. LoRa a type of PEFT (parameter-efficient fine-tuning) provides a solution by enabling the fine-tuning of only a limited number of additional parameters, while the majority of parameters remain unchanged. This approach significantly reduces computational costs. Training just a few parameters can achieve performance levels similar to those of a model that has undergone full parameter training. [7] The following parameters were used:

- `r = 8`
- `lora_alpha = 64`
- `lora_dropout = 0.01`

Approach 1: Dependency Matrix: The AdamW optimizer was selected for this study due to its superior performance compared to the Adam optimizer. AdamW effectively decouples weight decay from gradient-based updates, which leads to better convergence. A learning rate of 1e-4 was chosen, aligning with common practices for training T5 models, which typically use learning rates of either 1e-4 or 3e-4. Additionally, ReduceLROnPlateau was used as a learning rate scheduler with a patience of 2 epochs. If the validation loss continues to increase in two consecutive epochs, the learning rate will be reduced by a factor of 0.1. The batch size was set to 4 to allow for frequent weight updates, given that the dataset is relatively small. The experiment was run for 10 epochs. **Approach 2: Dependency Embeddings and Depth embeddings** The learning rate, number of epochs, and learning rate scheduler are consistent with those used in the dependency matrix approach. However, there are two key differences in this case: two new embeddings have been initialized, specifically the depth_embeddings and dependency_embeddings, respectively. Furthermore, both embeddings are initialized with He initialization, and they share the same learning rate as the other parameters, which is set at 1e-4.

7.3 Results

Baseline: The BERT score evaluates precision, which measures how many words in the prediction closely match the words in the reference based on contextual embeddings, and recall, which measures how many words from the reference are included in the prediction. The F1 score, which balances both precision and recall, is very high, indicating good results and suggesting that the contextual meaning of the sentences is maintained. However, the event overlap score is the lowest, measuring how well the predicted events align with the reference events. This low score could also be due to the default settings used in this paper. The split score is high, showing that the model effectively splits sentences into events. However, when evaluating split scores, ROUGE and event overlap should also be considered, as the split score does not account for the order of events, which may be incorrect. When analyzing all metrics, the model demonstrates good performance in splitting events while maintaining the correct order, all without explicit grammar information.

R1	R2	RL	Precision	Recall	F1	EO	Split
0.9649	0.9266	0.9554	0.9767	0.9734	0.9749	0.8808	0.9763

Table 2. Evaluation Metrics for Baseline

Baseline with pruning: Pruning certain heads in the T5-small model did not yield positive results, as it significantly reduced most of the computed scores, with many falling below 0.90. This outcome can be attributed to the fact that T5-small already has a limited number of heads per layer, making further pruning unlikely to produce significant improvements.

R1	R2	RL	Precision	Recall	F1	EO	Split
0.8703	0.7985	0.8261	0.9406	0.9173	0.8984	0.6518	0.8863

Table 3. Evaluation Metrics for pruning

Gating mechanism on score: In Fig 6, it is shown that different layers of the encoder exhibit varying delta rates. Initially, delta is set to 0.7, where the formula is delta * bias_score + (1 - delta) * scores. This experiment was conducted with a learning rate of 1e-4. Across all layers, there was a clear preference for using the original score, as none of the delta values dropped below 0.69 throughout all epochs, despite the high learning rate. This explains why the scores evaluated remained almost identical to the baseline, with only minor differences.

```
tensor(0.7091, requires_grad=True)
tensor(0.7045, requires_grad=True)
tensor(0.7070, requires_grad=True)
tensor(0.7020, requires_grad=True)
tensor(0.7218, requires_grad=True)
tensor(0.7243, requires_grad=True)
```

Fig. 6. The values of delta at different blocks in T5 encoder

R1	R2	RL	Precision	Recall	F1	EO	Split
0.9603	0.9294	0.9590	0.9799	0.9765	0.9781	0.8802	0.9747

Table 4. Evaluation Metrics for gating mechanism on scores

score * matrix: When comparing to the baseline, the BERT Score did decrease, but not significantly, indicating that the contextual meaning of the predicted sentence matches that of the reference. It only went down by 1%. However, the ROUGE score has decreased significantly, which indicates that there is less overlap occurring between the predicted sentence and the reference. For example, ROUGE-2 is the most significantly impacted, possibly because it struggles to account for grammatically distant words that form meaningful bigrams. This issue could arise if two consecutive words in the reference sentences are placed together, but the distance to their lowest common ancestor in the dependency graph exceeds the threshold used in this paper; they receive a value of zero in the matrix. As a result, this affects how they attend to each other in the attention score computed by the attention layer. The number of splits also decreased by 3%, which indicates that some of the sentences are not split into the correct number.

R1	R2	RL	Precision	F1	Recall	EO	Split
0.9360	0.8873	0.9255	0.9688	0.9643	0.9603	0.8278	0.9488

Table 5. Evaluation Metrics for score * matrix

score + matrix: As shown in the findings, a decline in all the scores indicates that the use of negative infinity penalizes distant relationships. Unlike the score * matrix, event overlap and ROUGE scores have dropped significantly; however, the split score has not decreased in comparison to the score * matrix. This means that even though the events are split into the correct number, the order may not be maintained. There are fewer words that overlap with each other, and the BERT score has decreased significantly, indicating that there are either fewer words generated in comparison to the reference sentence or that the words generated do not match contextually.

R1	R2	RL	Precision	Recall	F1	EO	Split
0.9042	0.8394	0.8623	0.9402	0.9342	0.9368	0.726	0.9491

Table 6. Evaluation Metrics for score + matrix

Gating mechanisms on attention outputs: In Fig 7 it is shown that different layers of the encoder exhibit varying rates of alpha. In some areas of the encoder, there is a preference for using direct edges, represented by matrix 1, which performs better than matrix 2, which includes distances between distinct tokens. Over the course of the epoch, the values began to change. The lower layers and upper layers tend to have slightly lower values than the middle layers. This experiment was conducted with a learning rate of 1e-4. This behavior is consistent with the understanding that different layers in LLM models focus on various aspects even without using extra grammatical information. Different layers have different attention heads; hence, they focus on different tokens at varying distances.

```
tensor(0.4735, requires_grad=True)
tensor(0.4727, requires_grad=True)
tensor(0.5011, requires_grad=True)
tensor(0.4912, requires_grad=True)
tensor(0.4716, requires_grad=True)
tensor(0.4720, requires_grad=True)
```

Fig. 7. The values of alpha at different blocks in T5 encoder

R1	R2	RL	Precision	F1	Recall	EO	Split
0.9076	0.8302	0.8724	0.9401	0.9368	0.9342	0.7532	0.9566

Table 7. Evaluation Metrics for gating attention outputs

The split score is slightly higher than the other two approaches; however, the event overlap and the ROUGE scores are lower. This suggests that either the order of the sentences is not maintained, or if it is maintained, some of the generated words do not match the reference. This observation is also supported by the drop in the F1 score of the BERT score, as well as the lack of overlap among the words.

Dependency Matrix and Level ID: Based on the results, the embeddings did not possess the appropriate weights to influence the token representations effectively. As a result, adding additional information to the token embeddings before further downstream tasks did not cause any meaningful changes. It was almost as if nothing had been added to the token embeddings, and their contribution did not produce any noticeable effect. When observing the outputs, they were nearly identical to the baseline results. This is why, unlike other approaches, the scores computed using both BERT and ROUGE metrics did not drop significantly. The BERT Score was almost the same as the baseline because the outputs in this case were not corrupted. This stability in attention scores ensured that there were no negative effects on the model's performance.

R1	R2	RL	Precision	F1	Recall	EO	Split
0.9589	0.9251	0.9527	0.9778	0.9760	0.9744	0.8552	0.9727

Table 8. Evaluation Metrics for dependency matrix and level ID

7.4 Discussion

Dependency_embeddings and level_ids: This approach was not better than the baseline. Both metrics, the ROUGE and BERT score were almost similar compared to those computed for the baseline. The reason behind this is that both dependency embeddings and depth embeddings are not pre-trained. Despite setting their learning rate to 1e-4 to facilitate faster learning, the data used was too small. A large dataset is required for these embeddings to attain the correct weights, and they should also be pre-trained on downstream tasks first. The right approach would have been to train them separately using different datasets so that they can learn and attain the right weights; in other words, grammatical relationships that tend to closely relate to one another would fall into the same subspace. Once that is done, these pre-trained weight embeddings could be used to fine-tune the T5 model for syntactic simplification, or both processes could be done simultaneously.

Matrix: First of all, the baseline model performed well, demonstrating that the T5 model effectively simplified sentences based on events. This is evidenced by the results, where the F1 score was the highest among all approaches, indicating that the generated sentences matched contextually well with the reference. Moreover, when considering both the split score and event overlap, as well as the ROUGE score, T5 correctly sequenced the events, highlighting its strong performance even without explicit grammatical information.

Pruning was applied only to the model that excluded grammatical information. A close examination of the attention heads in each layer provided some insights; however, due to the smaller size of the T5- small model, further pruning led to incorrect outputs upon closer inspection.

When evaluating the other approaches, it is clear that they all underperformed. This is evident in the decreasing number of correctly ordered splits when considering event overlap, ROUGE score, and split scores. Additionally, a drop in the BERT score indicates that the number of tokens that contextually align with references has reduced.The paper did not investigate the attention heads when applying grammatical information, which could have assessed the impact of changes by computing metrics such as entropy and variability. For instance, if the matrix were applied at certain layers, variability could have been closely examined to determine whether it shifted from being mostly position-based (low variability) to more dependency-related (high variability). Similarly, it would have been valuable to evaluate whether entropy changed from a more focused to a dispersed state. So, these approaches would help with picking the right threshold and deciding where to apply the matrix. Among all the approaches, the application of a gating mechanism on scores performed the best. This is because the attention was not completely modified by the matrix, allowing the model to prioritize its preferences through the use of learnable parameters. It was observed that the model predominantly relied on the attention score computed by the attention layer.

8 CONCLUSION

In conclusion, this paper has explored various methods for incorporating grammatical information into the model, enabling its use in the encoder's attention layer. The goal was to modify the attention scores to place greater emphasis on grammatical relationships, thereby facilitating the breakdown of syntactically complex sentences into simpler ones. Specifically, the objective was to ensure that words lacking grammatical connections do not attend to each other and that those with grammatical relationships are highlighted.

Despite the presence of grammatical information, the model demonstrated limited learning from the proposed matrices. This paper primarily concentrated on implementing this grammatical information across all layers. Future research will require a more granular analysis to understand the specific functions of each attention head and how they can be adjusted in consideration of the additional grammatical data. Simply adding the matrix did not help the model; other techniques such selecting heads that focus on grammatical features and applying the matrix only to those heads may be more effective. Overall, large language models (LLMs) perform exceptionally well after fine-tuning, as evidenced by the baseline results. T5, for instance, has been pre-trained on over 100 billion tokens. Various studies have indicated that, even without explicit training in grammatical structures, attention heads often incorporate grammatical considerations, at least to some extent, across all layers. No single attention head or layer is solely responsible; rather, the workload is distributed among all. Consequently, the aim of this research was to achieve a slight increase in accuracy because the accuracy of the baseline was already high. If improvement is to be pursued, only slight enhancements can be anticipated, and these will require a deeper understanding of what each head is actually doing, rather than simply applying a matrix in each layer and a lot more data.

9 FUTURE WORK

Future research could concentrate on gathering more data, particularly if Approach 2 is used with dependency embedding and depth embedding. This would enhance the embeddings' ability to cluster related words or relationships. Whereas, for Approach 1, it would be beneficial to explore the complexities of how attention scores function by seeking to understand the role of each attention head through the analysis and decide whether to apply matrices to specific heads, possibly employing different thresholds for each head instead of applying matrices to all heads in each layer.

REFERENCES

[1] Jiangang Bai, Yujing Wang, Yiren Chen, Yaming Yang, Jing Bai, Jing Yu, and Yunhai Tong. 2021. Syntax-BERT: Improving Pre-trained Transformers with Syntax Trees. arXiv:2103.04350 [cs.CL] https://arxiv.org/abs/2103.04350

[2] J. Carroll, G. Minnen, Y. Canning, S. Devlin, and J. Tait. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology.*

[3] John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying Text for Language-Impaired Readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Henry S. Thompson and Alex Lascarides (Eds.). Association for Computational Linguistics, Bergen, Norway, 269–270. https://aclanthology.org/E99-1042/

[4] R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and Methods for Text Simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics.* https://aclanthology.org/C96-2183/

[5] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D. Manning. 2019. What Does BERT Look at? An Analysis of BERT's Attention. In *Proceedings of the 2019 ACL Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, Tal Linzen, Grzegorz Chrupała, Yonatan Belinkov, and Dieuwke Hupkes

(Eds.). Association for Computational Linguistics, Florence, Italy, 276–286. https://doi.org/10.18653/v1/W19-4828

[6] Phu Mon Htut, Jason Phang, Shikha Bordia, and Samuel R Bowman. 2019. Do Attention Heads in BERT Track Syntactic Dependencies? *arXiv preprint arXiv:1911.12246* (2019).

[7] Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. LoRA: Low-Rank Adaptation of Large Language Models. (2021). arXiv:2106.09685 [cs.CL] https://arxiv.org/abs/2106.09685

[8] Ganesh Jawahar, Benoît Sagot, and Djamé Seddah. 2019. What Does BERT Learn about the Structure of Language?. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Anna Korhonen, David Traum, and Lluís Màrquez (Eds.). Association for Computational Linguistics, Florence, Italy, 3651–3657. https://doi.org/10.18653/v1/P19-1356

[9] Daniel Jurafsky and James H. Martin. 2025. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models* (3rd ed.). https://web.stanford.edu/~jurafsky/slp3/ Online manuscript released January 12, 2025.

[10] Junhua Ma, Jiajun Li, Yuxuan Liu, Shangbo Zhou, and Xue Li. 2022. Integrating Dependency Tree Into Self-attention for Sentence Representation. *arXiv preprint arXiv:2203.05918* (2022). ICASSP 2022, https://doi.org/10.48550/arXiv.2203.05918.

[11] Louis Martin, Benoît Sagot, Éric de la Clergerie, and Antoine Bordes. 2020. Controllable Sentence Simplification. arXiv:1910.02677 [cs.CL] https://arxiv.org/abs/1910.02677

[12] Onkar Pandit and Yufang Hou. 2021. Probing for Bridging Inference in Transformer Language Models. *arXiv preprint arXiv:2104.09400* (2021). https://arxiv.org/abs/2104.09400 Submitted on 19 Apr 2021.

[13] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2019. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *arXiv preprint arXiv:1910.10683* (2019). https://arxiv.org/abs/1910.10683 Version 4, last revised 19 Sep 2023.

[14] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. *arXiv preprint arXiv:1706.03762* (2017). https://arxiv.org/abs/1706.03762 Version 7, last revised 2 Aug 2023.

[15] Jesse Vig. 2019. Visualizing Attention in Transformer-Based Language Representation Models. arXiv:1904.02679 [cs.HC] https://arxiv.org/abs/1904.02679

[16] Jesse Vig and Yonatan Belinkov. 2019. Analyzing the Structure of Attention in a Transformer Language Model. *arXiv preprint arXiv:1906.04284v2* (2019). https://doi.org/10.48550/arXiv.1906.04284

[17] Elena Voita, David Talbot, Fedor Moiseev, Rico Sennrich, and Ivan Titov. 2019. Analyzing multi-head self-attention: Specialized heads do the heavy lifting, the rest can be pruned. In *Proceedings of ACL 2019*. https://doi.org/10.48550/arXiv.1905.09418

[18] Lulu Wang, Aishan Wumaier, Tuergen Yibulayin, and Maihemuti Maimaiti. 2024. Syntax-guided controllable sentence simplification. *Neurocomputing* 587 (2024), 127675. https://doi.org/10.1016/j.neucom.2024.127675

[19] Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. (2020). arXiv:1904.09675 [cs.CL] https://arxiv.org/abs/1904.09675

A ADDITIONAL DETAILS

During the preparation of this work, the author(s) used Grammarly to check grammar and spelling mistakes. Used ChatGPT to be able to create tables in Latex. After using these tools/service, the author(s) reviewed and edited the content as needed and take(s) full responsibility for the content of the work.