Syntax (and Attention) is All You Need

Xavier S. Mootoo
Electrical Engineering and Computer Science
York University
Toronto, ON M3J 1P3, Canada
`xmootoo@gmail.com`

Luca Vivona
Electrical Engineering and Computer Science
York University
Toronto, ON M3J 1P3, Canada
`lucavivona01@gmail.com`

Julian Forsyth
Electrical Engineering and Computer Science
York University
Toronto, ON M3J 1P3, Canada
`julianforsyth@hotmail.com`

Jayant Varma
Electrical Engineering and Computer Science
York University
Toronto, ON M3J 1P3, Canada
`Jayant@my.yorku.ca`

Abstract

In Natural Language Processing (NLP), pre-trained language models such as BERT (Bidirectional Encoder Representations from Transformers) have demonstrated remarkable capabilities in understanding contextual nuances within text. This study focuses on fine-tuning BERT for a specialized task: identifying and replacing words with their synonyms to enhance language diversity and fluency within our given contextual constraint. We have developed an innovative approach using BERT and an Attention Module to achieve this end. Our early results are mixed, but promising. Further work will require a more robust training set and some refinements to the model. Upon completion, our work will culminate in an application, *Syntax*, a unique and effective writing tool.

1 Introduction

1.1 The Application

Our goal is to produce an application, *Syntax*, which would function as a practical writing assistant for the everyday user. Given some text, which presumably the user has written themselves and wants to improve, Syntax will identify overused or imprecise word usages, and suggest superior alternatives. A micro-example to illustrate:

> *Input*: The house was extremely clean. It was very clean, with all the walls painted white.
>
> *Result*: The house was extremely clean. It was very **tidy**, with all the walls painted white.

Here, the word ”clean” is overused, leading to a poor flow and weak description. Syntax finds a preferable word, given the original word’s place in the greater context of the writing.

1.2 Assumptions

1. Syntax is designed for and trained on essay writing, academic writing, prose... i.e. formal and semi-formal writing. It is not compatible with poetic or highly expressive/informal writing styles.
2. Syntax is designed to be used with a maximum input size of 512 words, which amounts to several paragraphs. An essay, for example, would have to be divided into sections for individual processing.
3. In the application’s current state, English is assumed to be the language of input.

1.3 Importance

For many people, especially students young and old, writing and editing is a time consuming and difficult task. It's often difficult to know how to improve overall flow and stylistic fluency. This challenge becomes even more daunting if someone is writing in a second or third language. Our tool is of benefit in two key areas. Not only does it assist users in enhancing their writing by improving word choice and lexical variety, it also carries an educative component. With Syntax, language learners are able to see and learn new vocabulary possibilities in their own writing.

1.4 Similar Applications

Grammarly (https://www.grammarly.com/) is a similar product, which focuses on both grammatical and stylistic adjustments to enhance readability and impact.

Writer (https://writer.com/) is a Generative AI company that provides services of AI writer assistant mainly made for enterprises.

ProWritingAid (https://prowritingaid.com/) provides an AI software tool that enhances your writing by providing their end users with grammatical and stylistic improvements.

ChatGPT (https://chat.openai.com/), though a more general-purpose tool, it is capable of being prompted into indicating where there may be some grammatical errors in the user text.

Notion (https://www.notion.so/), though not specifically a writing assistant company, has an AI tool that helps the user in writing, and summarizing their notes.

Syntax is unique among these similar applications due to its specific functionality of targeting and replacing weak word choices, whether these words are overused or imprecise. This is not a feature which is available in any similar applications.

1.5 Adjustment to Part 1

Originally, we had the notion that we would be using Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) units for training our model. We had also considered using open-source LLMs such as GPT2 or Llama 2. Finally, we selected an architecture using Bidirectional Encoder Representations from Transformers (BERT) and an Attention Module. BERT was advantageous to other language models for its capability in evaluating specific word contexts.

2 Methodology

2.1 Design

We designed a sequence-to-sequence model using the BERT encoder (Devlin et al., 2018) along with an attention-based prediction head (Vaswani et al., 2017). This model aimed to transform the initial corpus of text with a maximum context size of $n = 512$ tokens, to a prediction corpus of text with the same number of tokens with the task of increasing the quality of the input text by substituting overused or poorly fitting words.

This prediction was trained to match the target corpus of text, which was the same number of tokens as the input, but with certain words substituted for higher quality of writing. Denote n as the number of tokens and d as the size of the dictionary. The input and target texts are represented by:

$$x = \begin{bmatrix} x^{\langle 1\rangle} & x^{\langle 2\rangle} & \cdots & x^{\langle n\rangle} \end{bmatrix} = \begin{bmatrix} x^{\langle 1,1\rangle} & x^{\langle 1,2\rangle} & \dots & x^{\langle 1,n\rangle} \\ x^{\langle 2,1\rangle} & x^{\langle 2,2\rangle} & \dots & x^{\langle 2,n\rangle} \\ \vdots & \ddots & \dots & \vdots \\ x^{\langle d,1\rangle} & x^{\langle d,2\rangle} & \dots & x^{\langle d,n\rangle} \end{bmatrix} \in \{0,1\}^{d\times n}, \tag{1}$$

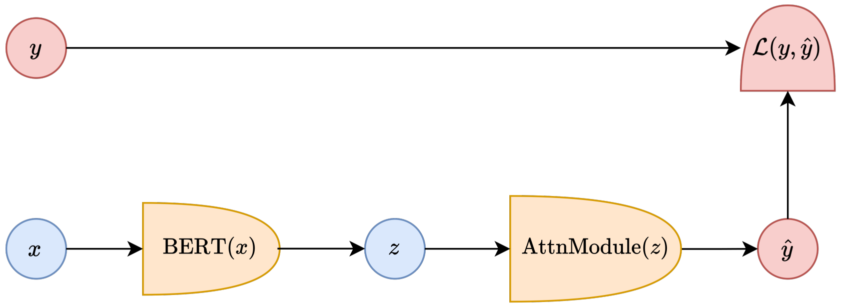

Figure 1: The Syntax model architecture. An input x is fed into the BERT module to create an encoding z, before an attention module is applied to form the predictions $\hat{y} = (y_1, y_2)$ to match the target $y = (y_1, y_2)$ as closely as possible with respect to the loss $\mathcal{L}$.

$$y_2 = \begin{bmatrix} y_2^{\langle 1 \rangle} & y_2^{\langle 2 \rangle} & \cdots & y_2^{\langle n \rangle} \end{bmatrix} = \begin{bmatrix} y_2^{\langle 1,1 \rangle} & y_2^{\langle 1,2 \rangle} & \cdots & y_2^{\langle 1,n \rangle} \\ y_2^{\langle 2,1 \rangle} & y_2^{\langle 2,2 \rangle} & \cdots & y_2^{\langle 2,n \rangle} \\ \vdots & \ddots & \cdots & \vdots \\ y_2^{\langle d,1 \rangle} & y_2^{\langle d,2 \rangle} & \cdots & y_2^{\langle d,n \rangle} \end{bmatrix} \in \{0,1\}^{d \times n}. \tag{2}$$

where each column $x^{\langle i \rangle} \in \{0,1\}^d$ and $y_2^{\langle i \rangle} \in \{0,1\}^d$ are the one-hot encodings of the i^{th} token in the input and target respectively. We refer to y_2 as the **synonym label**. In addition to y_2 we used a **replacement label**:

$$y_1 = \begin{bmatrix} y_1^{\langle 1 \rangle} & y_1^{\langle 2 \rangle} & \cdots & y_1^{\langle n \rangle} \end{bmatrix} \in \{0,1\}^n, \tag{3}$$

which indicated at each token i whether it had been replaced or not in the target, relative to the input, by one-hot encoding:

$$\hat{y}_1^{\langle i \rangle} = \begin{cases} 1 & \text{if token } i \text{ has been replaced} \\ 0 & \text{otherwise} \end{cases}. \tag{4}$$

Given an input $x \in \{0,1\}^{d \times n}$ an encoding:

$$\text{BERT}(x) = z \in \mathbb{R}^{k \times n} \tag{5}$$

is produced, which contains the contextual encodings for each token from $\{0,1\}^d$ to $\mathbb{R}^k$. Given the encoding z, we form our prediction $\hat{y} = (\hat{y}_1, \hat{y}_2)$ through the attention module described below. With z, we first generate the query, value, and key for two attention mechanisms:

$$Q_i = W_{qi} z, K_i = W_{ki} z, V_i = W_{vi} z,$$

for $i = 1, 2$, where $W_{qi}, W_{ki}, W_{vi} \in \mathbb{R}^{\ell \times k}$ are learnable weight matrices. Given the above we compute two attention mechanisms:

$$A_i = \text{Attention}(Q_i, K_i, V_i) = V_i \, \text{softmax}\left(\frac{Q_i^T K_i}{\sqrt{\ell}} \right) \in \mathbb{R}^{\ell \times n}, \tag{6}$$

for $i = 1, 2$. Attention(Q_i, K_i, V_i) is the attention operation from the Transformer, where softmax is computed column-wise on $\frac{Q_i^T K_i}{\sqrt{\ell}}$ (Vaswani et al., 2017). We use A_1 and A_2 to compute the final prediction of the model. The first component of the prediction is the **replacement probabilities**, given by:

$$\hat{y}_1 = \sigma(W_1(A_1 + A_2)) = \begin{bmatrix} \hat{y}_1^{\langle 1 \rangle} & \hat{y}_1^{\langle 2 \rangle} & \cdots & \hat{y}_1^{\langle n \rangle]} \end{bmatrix} \in [0,1]^n, \tag{7}$$

where $W_1 \in \mathbb{R}^{1 \times \ell}$ is a learnable weight matrix, σ is the sigmoid function, and:

$$\hat{y}_1^{\langle i \rangle} = p(y_1^{\langle i \rangle} = 1 | x), \tag{8}$$

models the conditional probability that $y_1^{\langle i \rangle} = 1$ given the input x for each $1 \leq i \leq n$, i.e. the i^{th} token in the target text is a replacement for the i^{th} token in the input. The second output of the model is the **synonym probabilities** defined by:

$$\hat{y}_2 = \text{softmax}(W_2(A_2)) = \begin{bmatrix} \hat{y}_2^{\langle 1 \rangle} & \hat{y}_2^{\langle 2 \rangle} & \cdots & \hat{y}_2^{\langle n \rangle} \end{bmatrix} = \begin{bmatrix} \hat{y}_2^{\langle 1,1 \rangle} & \hat{y}_2^{\langle 1,2 \rangle} & \cdots & \hat{y}_2^{\langle 1,n \rangle} \\ \hat{y}_2^{\langle 2,1 \rangle} & \hat{y}_2^{\langle 2,2 \rangle} & \cdots & \hat{y}_2^{\langle 2,n \rangle} \\ \vdots & \ddots & \cdots & \vdots \\ \hat{y}_2^{\langle d,1 \rangle} & \hat{y}_2^{\langle d,2 \rangle} & \cdots & \hat{y}_2^{\langle d,n \rangle} \end{bmatrix} \in \mathbb{R}^{d \times n}, \tag{9}$$

where $W_2 \in \mathbb{R}^{d \times \ell}$ is a learnable weight matrix and the softmax function is applied column-wise to produce a discrete probability distribution $\hat{y}_2^{\langle j \rangle} \in \mathbb{R}^d$ for every token j. For our experiments we used $d = 30522$ unique tokens to represent our dictionary. To be specific, for each $1 \leq i \leq d$ and $1 \leq j \leq n$ we modeled

$$\hat{y}_2^{\langle i,j \rangle} = p(y_2^{\langle i,j \rangle} = 1 | x), \tag{10}$$

as the probability that the target text at position i used token j. Together, $\hat{y}_1$ and $\hat{y}_2$ formed our full prediction for modeling the probabilities of which words needed to be replaced and what words they should be replaced with.

To measure how well our predictions performed, we computed a loss on a per token basis and then averaged the results. To compute the total loss, for each token $1 \leq i \leq n$, we first computed:

$$\text{BCE}(y_1^{\langle i \rangle}, \hat{y}_1^{\langle i \rangle}) = -\left[y_1^{\langle i \rangle} \log \hat{y}_1^{\langle i \rangle} + (1 - y_1^{\langle i \rangle}) \log(1 - \hat{y}_1^{\langle i \rangle}) \right], \tag{11}$$

$$\text{CE}(y_2^{\langle i \rangle}, \hat{y}_2^{\langle i \rangle}) = -\sum_{j=1}^{d} y_2^{\langle j,i \rangle} \log \hat{y}_2^{\langle j,i \rangle}. \tag{12}$$

The above equations are the Binary Cross-Entropy and Cross-Entropy equations respectively, which are the canonical criteria for modeling Equations (8) and (10) within binary classification and multi-class classification settings. Given this, we defined the **replacement loss** as the average of Equation (11) and the **synonym loss** as the average of Equation (12) across all n tokens, given by the following:

$$\mathcal{L}_1(y_1, \hat{y}_1) = \frac{1}{n} \sum_{i=1}^{n} \text{BCE}(y_1^{\langle i \rangle}, \hat{y}_1^{\langle i \rangle}), \tag{13}$$

$$\mathcal{L}_2(y_2, \hat{y}_2) = \frac{1}{n} \sum_{i=1}^{n} \text{CE}(y_2^{\langle i \rangle}, \hat{y}_2^{\langle i \rangle}). \tag{14}$$

The total loss was given by a weighted sum of both loss functions:

$$\mathcal{L}(y, \hat{y}) = \lambda \, \mathcal{L}_1(y_1, \hat{y}_1) + \gamma \, \mathcal{L}_2(y_2, \hat{y}_2), \tag{15}$$

for hyperparameters $\lambda, \gamma > 0$, where $y = (y_1, y_2)$ and $\hat{y} = (\hat{y}_1, \hat{y}_2)$. In our experiments we selected $\lambda = \gamma = 1$.

2.2 Dataset

The dataset comprised of pairs (x, y) in the form of text; x representing the input text, and y representing the output text with certain words replaced for synonyms. This was encoded in a `.csv` format whereby only x was included in the first column with a ˜_˜ flag at any word that was replaced in the output y, along with the replaced synonyms in order in the second column. This was then one-hot encoded using the BERT tokenizer, converting it back to x and y as described in (2.1). Since little data fit our narrowly defined format, we required our data to be generated, which was done by hand by all members of the group. We extracted writing samples from various sources, such as Wikipedia, news articles, and essays. From here, we often ”reverse-engineered” the data, replacing certain words with inferior or repetitive choices. By this process we produced (x, y) pairs.

2.3 Model Training

Preprocessing: Bert requires an encoding of words or tokens using a technique called WordPiece Tokenization. We tokenized our data accordingly. We made use of BERT's pre-trained weights, and these remain unadjusted, as they're optimized in contextualizing language within the last hidden layer.

Training: We conducted our training loop using mini-batches of our dataset. We selected Adam Optimization as the mechanism for updating our parameters, as it has been shown to be one of the most effective in training NLP models. The Loss Function, as described in previous section, is updated in each forward pass.

Training Loop

```
for epoch in range(num_epochs):
    for batch, labels in batch_data:

        pred_syn, pred_rep = model(**batch)
        loss = criterion(pred_syn, pred_rep, labels)

        # Backward pass and optimization
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
```

Figure 2: The training loop is no different from other training loops denoted within the parameters of the function, we forward through the network giving predictions which then are fed to our loss function where we zero out the gradients in the optimizer, compute the weights within in the forward pass, and update our weights

2.4 Prediction

To form a prediction, given the outputs $\hat{y} = (\hat{y}_1, \hat{y}_2)$, we defined a threshold value for the replacement probabilities $\hat{y}_1$ given by $\tau \in [0, 1]$. That is,

$$\text{Replace}(\hat{y}_1^{\langle i \rangle}) = \begin{cases} \text{replace word} & \text{if } \hat{y}_1^{\langle i \rangle} \geq \tau \\ \text{don't replace} & \text{otherwise} \end{cases} \tag{16}$$

for each token $1 \leq i \leq n$. For our experiments we chose $\tau = 0.6$, note that this hyperparameter is independent of model training, and thus can be tuned after the model is trained. It can act as a "temperature" parameter indicating how much you would like the model to replace words, if τ is high it will be very selective in choosing words to replace, whereas if τ is low it will replace many words. If a token i was chosen to be replaced by the Equation (16), the appropriate token for replacement corresponded to the maximum probability across the dictionary in $\hat{y}_2$:

$$\text{Synonym}(\hat{y}_2^{\langle i \rangle}) = \underset{1 \leq j \leq d}{\arg\max}\, \hat{y}_2^{\langle j,i \rangle} \tag{17}$$

for each token $1 \leq i \leq n$, where it corresponded to the index in the dictionary. Converting all the above into one-hot encodings, and then into words, we can obtain the desired prediction.

3 Results

3.1 Evaluation

Our current evaluation is done by hand, assessing by human judgement the quality of a small sample of prediction results. A more quantitative analysis approach should be necessary to assess output consistency and performance on large scales. A larger collection of data will be required to perform this analysis.

3.2 Results

Our results are mixed, with the model often producing errant results. A word replacement may be a closely related word, ex. "*hunting* dog" $\rightarrow$ "*gun* dog" , which is insufficiently precise to be synonymous and thus results in absurdity.

4 Discussions

4.1 Implications

The outcomes learned thus far in our initial application falls short of our set expectation. While we maintain a high level of confidence in the architecture and implementation of our model, the flaws lie within the lack of diversity and insufficient data volume available for the model to be trained on. We aspire to address this limitation further down the line as we are able to generate a larger corpus of data-sets to enhance the robustness and efficacy of our model.

4.2 Strengths

The strength of our application is lies in its specialization and flexibility. Whereas other applications such as chatGPT or grammarly can be useful writing aids and are more general purpose, they often fail to identify overused words or words that do not fit within the context, even if the text is grammatically correct. Often, when prompted to improve the writing quality of the text, the choice of word substitution does not fit well, and often uses words with low frequency leading to unnatural looking writing. By specifically training a model specialized on this one task, our model has the opportunity to use many of the features of LLMs, such as a large transformed-based encoder, along with a specialized task fine-tuned this area.

4.3 Limitations

1. Data generation, and data augmentation. Data generation proved to be a difficult task, as we need to implement a variety of synonym mappings from the input to the output so that the model could generalize to many different types of text. Due to the limited ability for four people to implement this data generation process by hand, which in our particularly context was not automatable, it greatly limited the size of the training set and diversity of examples. We attempted to use data augmentation techniques such as paraphrasing, whereby we would take one input and target text and then rephrase it, but keep the synonym mapping the same, although this was also very time consuming and had to be done by hand. The challenge of this was to generate semantically equivalent and diverse sentences while preserving context, maintaining grammatical correctness, and addressing domain-specific vocabulary.

2. Fixed number of tokens, and so it may struggle with capturing long-range dependencies in text. This is, however, a limitation driven by our current access to computational resources.

3. BERT model and their larger variance require a high amount of memory on both GPU and CPU. This can be limiting in trying to deploy these models locally or in production with limited memory.

4. BERT Base is a relatively large model with approximately 100 Million parameters. Retraining it, either from scratch or from its pretrained state, is outside of the question with our modest compute power. So we are limited to the fixed representation of BERT.

4.4 Future Directions

1. Experimenting with how the attention outputs A_1 and A_2 are used, for the output. For example, what about $\hat{y}_2 = \text{softmax}(W_2(A_1 + A_2))$ instead of $\hat{y}_2 = \text{softmax}(W_2(A_2))$,

and similarly for $\hat{y}_1$.

2. One major improvement we can make is for **multiple word subtitution**, as the model is restricted to single word substitution and must return a sequence length equal to the input. Thus, this will allow for more flexibility as certain synonyms like “knowledgeable“ and “well-informed” can be used, leading to diversity of synonym pairings and greater generalization.
   log
3. Incorporating **reinforcement learning** techniques that leverage human feedback. By integrating human insights into the training process, we can fine-tune the model with a more nuanced understanding of contextual syntax and improve its overall capabilities of providing better replacements, and unique synonyms. This iterative feedback loop helps the model adapt and refine its predictions, leading to enhanced performance and more accurate outcomes.

REFERENCES

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.