Syntactic Knowledge-Infused Transformer and BERT models

Dhanasekar Sundararaman[1], Vivek Subramanian[2], Guoyin Wang[3], Shijing Si[4], Dinghan Shen[5], Dong Wang[6] and Lawrence Carin[7]

[1,4,6,7]*Duke University, USA*
[2,3]*Amazon Alexa AI*
[5]*Microsoft Dynamics 365 AI*

Abstract

Attention-based deep learning models have demonstrated significant improvement over traditional algorithms in several NLP tasks. The Transformer, for instance, is an illustrative example that generates abstract representations of tokens that are input to an encoder based on their relationships to all tokens in a sequence. While recent studies have shown that such models are capable of learning syntactic features purely by seeing examples, we hypothesize that explicitly feeding this information to deep learning models can significantly enhance their performance in many cases. Leveraging syntactic information like part of speech (POS) may be particularly beneficial in limited-training-data settings for complex models such as the Transformer. In this paper, we verify this hypothesis by infusing syntactic knowledge into the Transformer. We find that this *syntax-infused Transformer* achieves an improvement of 0.7 BLEU when trained on the full WMT '14 English to German translation dataset and a maximum improvement of 1.99 BLEU points when trained on a fraction of the dataset. In addition, we find that the incorporation of syntax into BERT fine-tuning outperforms $\text{BERT}_{\text{BASE}}$ on all downstream tasks from the GLUE benchmark, including an improvement of 0.8% on CoLA.

Keywords

Syntax, Semantics, Transformers

1. Introduction

Attention-based deep learning models for natural language processing (NLP) have shown promise for various machine translation and natural language understanding tasks. For word-level sequence-to-sequence tasks such as translation, paraphrasing, and text summarization, attention-based models allow a single token (in our case, subwords) to be represented as a combination of all tokens in the sequence [1]. The distributed context allows attention-based models to infer rich representations for tokens, leading to more robust performance. One such model is the Transformer, which features a multi-headed self- and cross-attention mechanism that allows many different representations to be learned for a given token in parallel [2]. The encoder and decoder arms contain several identical stacked subunits to learn embeddings for tokens in the source and target vocabularies.

Syntax is an essential feature of grammar from external knowledge that facilitates generation of coherent sentences. For instance, POS dictates how words relate to one another (*e.g.*, verbs represent the actions of nouns, adjectives describe nouns, *etc.*). Studies show that when trained for a sufficiently large number of steps, NLP models can potentially learn underlying patterns about text like syntax and semantics, but this knowledge is imperfect [3]. However, works such as [4], [5] show that NLP models that acquire even a weak understanding of syntactic structure through training demonstrate improved performance relative to their baseline counterparts. Hence, we hypothesize that explicit prior knowledge of syntactic information can benefit NLP models in various tasks.

The Transformer relies on a significant amount of data and extensive training to accurately pick up on syntactic and semantic relationships [3]. Previous studies have shown that an NLP model's performance improves with the ability to learn the underlying grammatical structure of a sentence [4, 5, 6]. It has been shown that simultaneously training models for machine translation, part of speech (POS) tagging, and named entity recognition provide a slight improvement over baseline on each task for small datasets [7]. Inspired by these previous efforts, we propose to utilize external knowledge, namely the syntactic features inherent in natural language sequences, to enhance the performance of the Transformer model.

We suggest a modification to the embeddings fed into the Transformer architecture that allows the tokens input to the encoder to attend not only to other tokens but also syntactic features including POS, case (upper or lower), and subword position [8]. Like POS, case is a categorical feature that can allow the model to distinguish common words from important ones. Subword tags can help bring cohesion among subwords of a complex word (say 'amal', 'ga', 'mation' of "amalgamation") so that their identity

ds448@duke.edu (D. Sundararaman)

CEUR Workshop Proceedings (CEUR-WS.org)

as a unit is not compromised by tokenization. These features are identified using a separate model (for POS) or are directly specified (for case and subword position) and are appended to the one-hot vector encoding for each token. For example, the feature-augmented subword tokens of 'amalgamation' would look like: 'amal,NOUN,0,S', 'ga,NOUN,0,M', 'mation,NOUN,0,E'. Here, 'NOUN' indicates the POS; 0 indicates lowercase; and 'S', 'M', and 'E' indicates that the subwords occur at the start, middle, and end of the word, respectively. Embeddings for the tokens and their features are learned jointly during the training. As the embeddings pass the Transformer layers, the representation for each token is synthesized using a combination of subword token and syntactic features.

We apply our approach to English to German (EN-DE) translation on the WMT '14 dataset and demonstrate that the BLEU score of the feature-rich *syntax-infused Transformer* uniformly outperforms the baseline Transformer as a function of training data size. We also evaluate the proposed model on two low-resource languages – Korean and Sinhala – where the source translation language is always English. Finally, we experiment with this modification of embeddings on the $\text{BERT}_{\text{BASE}}$ model on several General Language Understanding Evaluation (GLUE) benchmarks and observe considerable improvement in performance on all the tasks. Examining the attention weights learned by the proposed model indicates that the attention is more dispersed than the baseline model.

To summarize, our contributions are as follows:

1. We propose a modification to the trainable embeddings of the Transformer model, incorporating external syntax information, including POS, case, and subword position.
2. We demonstrate superior performance on the WMT '14 EN-DE machine translation task and for low-resource machine translation.
3. We infuse pre-trained $\text{BERT}_{\text{BASE}}$ embeddings with syntax information and find that our model outperforms $\text{BERT}_{\text{BASE}}$ on all of the GLUE benchmark tasks.

2. Related Work

Previous works have sought to improve the self-attention module to aid NLP models. For instance, [9] introduced a Gaussian bias to model locality, to enhance model ability to capture local context while also maintaining the long-range dependency. Instead of absolute positional embeddings, [10] experimented with relative positional embeddings or distance between sequences and found that it led to a significant improvement in performance.

Adding linguistic structure to models like the Transformer can be thought of as a way of improving the attention mechanism. The POS and subword tags act as a form of relative positional embedding by enforcing the sentence structure. [11] encourages different attention heads to learn about different information like position and output representation by introducing a disagreement regularization. To model the local dependency between words more efficiently, [12] introduced distance between words and incorporated it into the self-attention.

Previous literature also has sought to incorporate syntax into deep learning NLP models. [13] used syntax dependency tree on a bidirectional RNN on translation systems by modeling the trees using Graph Convolutional Networks (GCNs) [14]. Incorporating syntax information by linearizing parse trees and adding a syntax-based distance constraint on the attention module helped significantly in Chinese-English and English-German translation [15].

Finally, [16] has incorporated source side syntax on Transformer encoders in a multi-task setup with parse trees. However, in this approach, the Transformer struggles to learn to parse and decode the syntactic sequences and only showed performance gains in low-resource languages. We append the syntactic information as sequences and decode only the target sequence, thus not complicating the Transformer's job. These works affirm that adding syntax information can help the NLP models translate better from one language to another and achieve better performance measures.

3. Background

3.1. Baseline Transformer

The Transformer consists of encoder and decoder modules, each containing several subunits that act sequentially to generate abstract representations for words in the source and target sequences [2]. For all $m \in \{1,\ 2,\ \ldots,\ M\}$, where M is the length of the source sequence, the encoder embedding layer first converts tokens $\mathbf{x}_m$ into embeddings $\mathbf{e}_m$: $\mathbf{e}_m = \mathbf{E}\mathbf{x}_m$ where $\mathbf{E} \in \mathbb{R}^{D \times N}$ is a trainable matrix with column m constituting the embedding for token m, N is the total number of tokens in the shared vocabulary, and $\mathbf{x}_m \in \{0,1\}^N : \sum_i x_{mi} = 1$ is a one-hot vector corresponding to token m. These embeddings are passed sequentially through six encoder subunits. Each of these subunits features a self-attention mechanism, that allows tokens in the input sequence to be represented as a combination of all tokens in the sequence. Attention is accomplished using three sets of weights: the key, query, and value matrices ($\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$, respectively). The key and query matrices interact to score each token in relation to other tokens, and the value matrix gives the weights to which the score is applied to generate output embedding of a given token.

Stated mathematically,

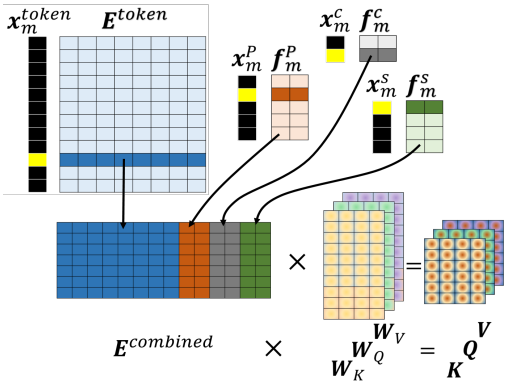
$$\begin{aligned}\mathbf{K} &= \mathbf{H}\mathbf{W}_K \\ \mathbf{Q} &= \mathbf{H}\mathbf{W}_Q \\ \mathbf{V} &= \mathbf{H}\mathbf{W}_V \\ \mathbf{A} &= \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{\rho}}\right)\mathbf{V}\end{aligned} \tag{1}$$

where $\mathbf{H} = [\mathbf{h}_1\ \mathbf{h}_2\ \cdots\ \mathbf{h}_M]^\top \in \mathbb{R}^{M\times D}$ are the D-dimensional embeddings for a sequence of M tokens indexed by m; $\mathbf{W}_K$, $\mathbf{W}_Q$, and $\mathbf{W}_V$ all $\in \mathbb{R}^{D\times P}$ are the projection matrices for keys, queries, and values, respectively; ρ is a scaling constant (here, taken to be P) and $\mathbf{A} \in \mathbb{R}^{M\times P}$ is the attention-weighted representation of each token. Note that these are subunit-specific – a separate attention-weighted representation is generated by each subunit and passed on to the next. Moreover, for the first layer, $\mathbf{h}_m := \mathbf{e}_m$.

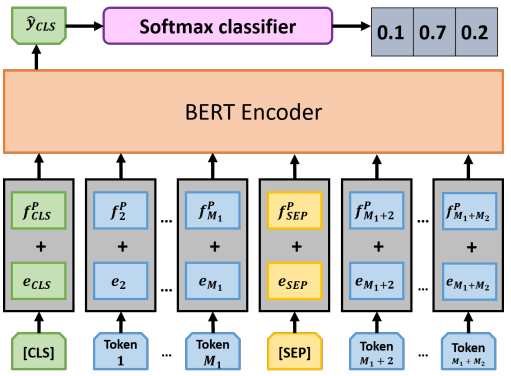
The final subunit then passes its information to the decoder, which also consists of six identical subunits that behave similarly to those of the encoder. One key difference between the encoder and decoder is that the decoder not only features self-attention but also cross-attention; thus, when generating new words, the decoder pays attention to the entire input sequence as well as to previously decoded words.

3.2. BERT

The token embeddings learned by the Transformer encoder can also be fine-tuned to perform a number of different downstream tasks. Bidirectional encoder representations of Transformers (BERT) [17] is an extension of the Transformer model that allows for such fine-tuning. The BERT model is essentially a Transformer encoder (with number of layers l, embedding dimension D, and number of attention heads α) which is pre-trained using two methods: masked language modeling (MLM) and next-sentence prediction (NSP). Subsequently, a softmax layer is added, allowing the model to perform various tasks such as classification, sequence labeling, question answering, and language inference.



(a) Syntax-infused Transformer



(b) Syntax-infused BERT

Figure 1: (a) Formation of attention matrices ($\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$) with syntactic information. The left column shows the token embedding matrix; the embedding matrices for the various features are shown on top. Embeddings for the features are either concatenated or summed (denoted by $\oplus$) and finally, concatenated to the token embeddings. Multiplication with learned weights results in $\mathbf{K}$, $\mathbf{Q}$, and $\mathbf{V}$. The attention matrices are double shaded to indicate the mix of token and syntax information. (b)The $\text{BERT}_{\text{BASE + POS}}$ model. Token embeddings are combined with trainable POS embeddings and fed into the BERT encoder. The final embedding of the [CLS] token is fed into a softmax for downstream classification tasks.

4. Knowledge-Infused Models

4.1. Syntax-infused Transformer

To aid the Transformer in acquiring and utilizing syntactic information for better translation, we (i) employ a pre-trained model[1] with high accuracy to tag words in the source sequence with their POS, (ii) identify the case of each word, and (iii) identify the position of each subword relative to other subwords that are part of the same word (subword tagging). We then append trainable syntax embedding vectors to the token embeddings, resulting in a combined representation of syntactic and semantic elements. Specifically, each word in the source sequence is first associated with its POS label according to syntactic structure. We then assign each subword token the POS label of the word from which it originated. For example, if the word `sunshine` is broken up into subwords `sun`, `sh`, and `ine`, each subword token would be assigned the POS NOUN. The POS embeddings $\mathbf{f}_m^P \in \mathbb{R}^d$ for each token (indexed by m) are then extracted from a

[1] https://spacy.io/

trainable embedding matrix using a look-up table.

In a similar manner, we extract case and subword position features. For case, we use a binary element $z_m^c \in \{0, 1\}$ to look up a feature embedding $\mathbf{f}_m^c \in \mathbb{R}^d$ for each subword, depending on whether the original word is capitalized. For subword position, we use a categorical element $z_m^s \in \{S, M, E, O\}$ to identify a feature embedding $\mathbf{f}_m^s \in \mathbb{R}^d$ for each subword depending on whether the subword is at the start (S), middle (M), or end (E) of the word; if the subword comprises the full word, it is given a tag of O. These are then added onto the POS embedding and concatenated with the token embeddings $\mathbf{e}_m \in \mathbb{R}^{D-d}$ to create a combined embedding (see Figure 1a). Mathematically, in the input stage, $\mathbf{h}_m$ becomes:

$$[\mathbf{e}_m^\top \ \mathbf{f}_m^\top]^\top = \mathbf{h}'_m \in \mathbb{R}^D$$

where $\mathbf{f}_m = \mathbf{f}_m^P + \mathbf{f}_m^c + \mathbf{f}_m^s \in \mathbb{R}^d$ is the learned embedding for the syntactic features of subword m in the sequence of M subwords. The augmented embeddings are then passed through the Transformer's multi-headed attention, layer normalization, and feedforward modules, resulting in a syntax-infused hidden state of each encoder layer:

$$\begin{aligned}
\tilde{\mathbf{A}} &= [\mathbf{A}_0 \cdots \mathbf{A}_7]\mathbf{W}^O \\
\tilde{\mathbf{H}} &= [\mathbf{H}' \cdots \mathbf{H}'] \\
\tilde{\mathbf{Z}} &= \text{LayerNorm}(\tilde{\mathbf{A}} + \tilde{\mathbf{H}}) \\
\mathbf{Z}_l &= \text{LayerNorm}(\tilde{\mathbf{Z}} + \text{FeedForward}(\tilde{\mathbf{Z}}))
\end{aligned} \tag{2}$$

where $\tilde{\mathbf{H}}$ are the syntax-augmented embeddings, repeated eight times (once for each attention head); $\tilde{\mathbf{A}}$ is the concatenation of the attention-weighted representation of the output of the eight heads; and $\mathbf{Z}_l$ is the output of layer l. LayerNorm and FeedForward denote the layer normalization function and feedforward neural network layers, respectively, and are applied separately for each head. This is repeated L times in L layers, with the output of the previous layer treated as the input to the next. The final syntax-infused encoder output $\mathbf{Z}_L$ is attended to by the decoder, aiding the autoregressive text generation process in the final layer of the Transformer:

$$\begin{aligned}
\mathbf{z}'_{L,t} &= f(\mathbf{Z}_L, w_{1:t-1}) \\
\text{Scores}(t) &= \text{LayerNorm}(\mathbf{z'}_{L,t})\mathbf{W}_\mathcal{V} \\
P(w_t|w_{1:t-1}, \mathbf{z}'_{L,t}) &= \text{softmax}(\text{Scores}(t))
\end{aligned} \tag{3}$$

where f is the decoder, which undergoes the same process outlined in equation 2, and comprises cross-attention with the encoder output and a causal self-attention (parameters of the decoder omitted for conciseness). $t = 1, \ldots, T$ indexes decoded tokens w_t, and $\mathbf{W}_\mathcal{V}$ is a linear projection matrix mapping the decoder output to the vocabulary size. The scores for output tokens are now a function of $\mathbf{z}'_{L,t}$, which is aware of the source syntax.

4.2. Syntax-infused BERT

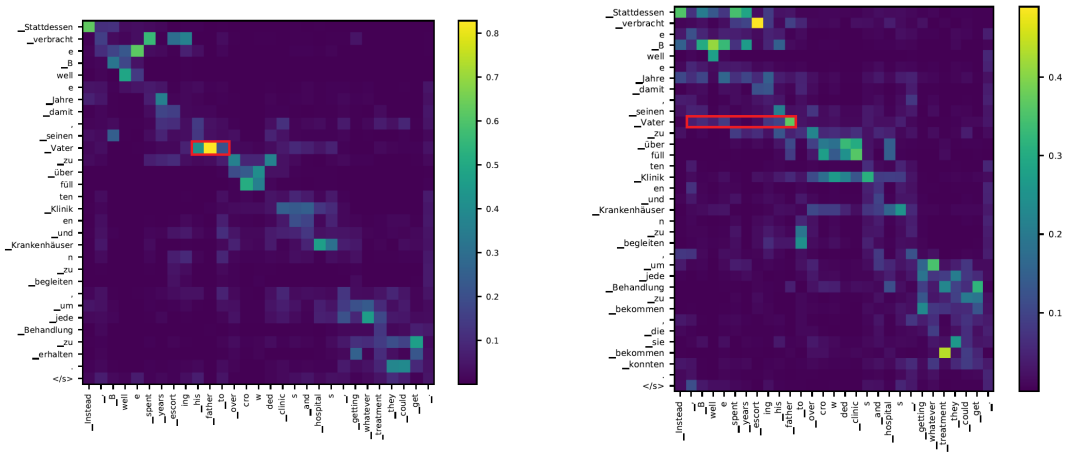
Adding syntactic features to the BERT model is a natural extension of the above modification to the Transformer. For a given token, its input representation is constructed by summing the corresponding BERT token embeddings with POS embeddings (see Figure 1b). Mathematically, the input tokens $\mathbf{h}'_m \in \mathbb{R}^D$ are given by $\mathbf{h}'_m = \mathbf{e}_m + \mathbf{f}_m^P$, where $\mathbf{e}_m$ is the BERT token embedding and $\mathbf{f}_m^P$ is the POS embedding for token m. For single sequence tasks, $m = 1, 2, \ldots, M$, where M is the number of tokens in the sequence; while for paired sequence tasks, $m = 1, 2, \ldots, M_1 + M_2$, where M_1 and M_2 are the number of tokens in each sequence. As is standard with BERT, for downstream classification tasks, the final embedded representation $\hat{\boldsymbol{y}}_{CLS}$ of the first token (denoted as [CLS]) is passed through a softmax classifier to generate a label. We modify the BERT$_{\text{BASE}}$ model using this approach and denote it as BERT $_{\text{BASE + POS}}$.

5. Datasets and Experimental Details

We train our syntax-infused model on the WMT '14 EN-DE (German) dataset, which consists of 4.5M training sentence pairs. Validation is performed on newstest2013 (3000 sentence pairs), and testing is on the newstest2014 dataset (2737 sentence pairs, [18]). For low-resource machine translation (1M sentence pairs or less), we employ the OpenSubtitles EN-KO (Korean) and EN-SI (Sinhala) datasets [19]. The latter was found by [20] to be a low-resource language with morphology and syntax significantly different from that of English. We subsample these datasets from 1.3M and 601K sentence pairs, respectively, to 500K sentence pairs, and train on 90%, validate on 9%, and test on 1%. One of the key reasons to incorporate POS features to EN sequences is that parsers that infer syntax from EN sentences are typically trained on a greater number and variety of sentences and are therefore more robust than parsers for other languages. Unlike [16], in our experiments, we always translate sequences from EN to other languages as the objective is to improve the translation performance and not to decode the POS sequences.

5.1. Machine translation

For EN-DE, we train both the baseline and syntax-infused Transformer for 100,000 steps. All hyperparameter settings of the baseline Transformer, including embedding dimensions of the encoder and decoder, match those of [2]. We train the syntax-infused Transformer model using 512-dimensional embedding vectors. In the encoder, $D = 492$ dimensions are allocated for subword token em-



(a) Baseline (EN-DE)

(b) Syntax-infused (EN-DE)

Figure 2: Comparison of attention for example sentences translated by baseline and POS Transformer models (obtained from the last layer). Rows depict the attention score for a given target subword to each of the subwords in the source sequence. In syntax-infused models for EN-DE translation, we find that attention is more widely distributed across subwords. For instance, the subword "Vater" (the German word for "father") attends mostly to the nearby subwords "his" and "father" in the base model while "Vater" also attends to the more distant words "Bwelle" (a person) and "escorting" in the syntax-infused model. This suggests that the syntax-infused model is able to better connect disparate parts of a sentence to aid translation. Note that the number of rows in the baseline and syntax-infused Transformer are different because each produces different predictions.

Data Fraction	Number of Sentences	Baseline Transformer	Syntax-infused Transformer
1%	45k	1.10	**1.67**
5%	225k	8.51	**10.50**
10%	450k	16.28	**17.28**
25%	1.1M	22.72	**23.24**
50%	2.25M	25.41	**25.74**
100%	4.5M	28.94	**29.64**

Table 1
BLEU scores for different proportions of the data for baseline Transformer vs syntax-infused Transformer for the EN-DE task on newstest2014.

	Transformer		RNN	
Lang.	Baseline	Syntax	Baseline	Syntax
EN-SI	11.12	**11.73**	8.24	7.95
EN-KO	3.87	**4.19**	2.86	1.80

Table 2
BLEU scores improvements on low resource languages including Sinhala and Korean for the baseline and syntax-infused models.

beddings while $d = 20$ for feature embeddings (chosen by hyperparameter tuning). In the decoder, all 512 dimensions are used for subword token embeddings (since we are decoding words, not word-POS pairs).

The model architecture consists of six encoder and six decoder layers, with eight heads for multi-headed attention. Parameters are initialized from a Glorot uniform distribution [21]. We use a dropout rate of 0.1 and batch size of 4096. We utilize the Adam optimizer to train the model with $\beta_1 = 0.9$ and $\beta_2 = 0.998$; gradients are accumulated for two batches before updating parameters. A label-smoothing factor of 0.1 is employed.

The training settings are exactly the same for low-resource translation except that we train the Transformer models for 50,000 steps owing to the reduced data complexity. The dimension d of features $\mathbf{f}_m$ is chosen to be 20 by doing a grid search over the range of 8 to 64.

5.2. Natural language understanding

The General Language Understanding Evaluation (GLUE) benchmark [22] is a collection of different natural language understanding tasks evaluated on eight datasets: Multi-Genre Natural Language Inference (MNLI), Quora Question Pairs (QQP), Question Natural Language Inference (QNLI), Stanford Sentiment Treebank (SST-2), the Corpus of Linguistic Acceptability (CoLA), the Semantic Textual Similarity Benchmark (STS-B), Microsoft Research Paraphrase Corpus (MRPC), and Recognizing Textual Entailment (RTE). For a summary of these datasets, see [17]. We use POS as the syntactic feature for BERT for these tasks. Aside from the learning rate, we use identical hyperparameter settings to fine-tune both the $\text{BERT}_{\text{BASE}}$

Reference	Baseline Transformer	Syntax-infused Transformer
Parken in Frankfurt könnte bald empfindlich teurer werden .	Das Personal war sehr freundlich und hilfsbereit .	Parken in Frankfurt könnte bald spürbar teurer sein .
Die zurückgerufenen Modelle wurden zwischen dem 1. August und 10. September hergestellt .	Zwischen August 1 und September 10.	Die zurückgerufenen Modelle wurden zwischen dem 1. August und 10. September gebaut
Stattdessen verbrachte Bwelle Jahre damit , seinen Vater in überfüllte Kliniken und Hospitäler zu begleiten , um dort die Behandlung zu bekommen , die sie zu bieten hatten .	Stattdessen verbrachte Bwelle Jahre damit , seinen Vater mit über füllten Kliniken und Krankenhqusern zu beherbergen .	Stattdessen verbrachte Bwelle Jahre damit , seinen Vater zu überfüllten Kliniken und Krankenhäusern zu begleiten , um jede Behandlung zu bekommen , die sie bekommen konnten .
Patek kann gegen sein Urteil noch Berufung ein legen .	Patek kann noch seinen Satz an rufen .	Patek mag sein Urteil noch Berufung ein legen .

Table 3
Translation examples of baseline Transformer vs. syntax-infused Transformer on the EN-DE dataset. The text highlighted in blue represents words correctly predicted by the syntax-infused model but not by the baseline Transformer.

System	MNLI 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	Average -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{BASE + POS}}$	84.4/**83.6**	**71.4**	**90.8**	**93.9**	**52.9**	**86.0**	**89.0**	**66.9**	**79.9**

Table 4
GLUE test results scored using the GLUE evaluation server. The number below each task denotes the number of training examples. The scores in bold denote the tasks for which $\text{BERT}_{\text{BASE + POS}}$ outperforms $\text{BERT}_{\text{BASE}}$.

and $\text{BERT}_{\text{BASE + POS}}$ models for each task. This includes a batch size of 32 and 3 epochs of training for all tasks. For each model, we also choose a task-specific learning rate among the values $\{5, 4, 3, 2\} \times 10^{-5}$, which is standard for $\text{BERT}_{\text{BASE}}$.

6. Experimental Results

6.1. Machine translation

We evaluate the impact of infusing syntax into the baseline Transformer on the WMT '14 EN-DE translation task as well as on the low resource EN-SI and EN-KO language pairs. There are multiple ways to incorporate feature embeddings into the subword token embeddings, such as direct summation, vector projection, or concatenation. For a fair comparison to the baseline Transformer, we fix a total of 512 dimensions for representing both the token embeddings and feature embeddings. One important tradeoff is that as the dimensionality of the syntax information increases, the dimensionality for token embeddings decreases. Since POS, case, and subword tags have only a limited number of values they can take, dedicating a high dimensionality for each feature proves detrimental. We find that the total feature dimension for which the gain in BLEU score is maximized is 20 (found through grid search). This means that (1) each feature embedding dimension can be allocated to 20 and summed together or (2) the feature embeddings can be concatenated to each other such that their total dimension is 20. Therefore, in order to efficiently learn the feature embeddings while also not sacrificing the token embedding dimensionality, we find that summing the embeddings for all three different features of $d = 20$ and concatenating the sum to the subword token embeddings of $D = 492$ gives the maximum performance on translation.

We report results on the WMT '14 EN-DE translation task in Table 1. We vary the proportion of data used for training and observe the performance of both the baseline and syntax-infused Transformer. The syntax-infused model markedly outperforms the baseline model, offering an improvement of 0.57, 1.99, 1, 0.52, 0.33, and 0.7 points, respectively, when trained on 1, 5, 10, 25, 50, and 100% of the full training data. It is notable that the syntax-infused model translates the best relative to the baseline when only a fraction of the dataset is used for training. Specifically, the maximum improvement is 1.99 BLEU points when only 10% of the training data is used. This shows that explicit syntax information is most helpful under limited training data conditions. As shown in Figure 2(a)-(b), the syntax-infused model is better able to capture connections between tokens that are far apart

Sentence 1	Sentence 2	True label
The Qin (from which the name China is derived) established the approximate boundaries and basic administrative system that all subsequent dynasties were to follow .	Qin Shi Huang was the first Chinese Emperor .	Not entailment
Steve Jobs was attacked by Sculley and other Apple executives for not delivering enough hot new products and resigned from the company a few weeks later.	Steve Jobs worked for Apple.	Entailment

Table 5
Examples of randomly chosen sentences from the RTE dataset (for evaluation of entailment between pairs of sentences) that were classified by $\text{BERT}_{\text{BASE + POS}}$ but not by $\text{BERT}_{\text{BASE}}$.

yet semantically related, resulting in improved translation performance. In addition, Table 3 shows a set of sample German predictions made by the baseline and syntax-infused Transformer.

In Table 2, we compare the performance of baseline and syntax-infused Transformers on low resource languages, namely Korean and Sinhala. These languages are often under represented in the NLP literature and fall under categories of languages with limited training data. Results confirm the effectiveness of our methodology of adding syntax to source sequences to improve efficiency, irrespective of the training data size. In addition, while we find there are improvements in BLEU scores with the syntax-infused Transformer, Table 2 also shows the effect of addition of syntax to RNN-based networks for the same translation tasks leads to a drop in performance. For this experiment, we employed an 2-layer LSTM encoder and 2-layer LSTM decoder with 500 hidden states each and augmented encoder tokens with syntax features in the same way as for the Transformer. [5] confirms our observation that recurrent structures struggle to learn the subject-verb agreement and that better architectures may be required to retain the complex syntactic structures present in the languages. On the other hand, Transformers have been known to learn syntax with sufficient training and an increased amount of data. Studies [23] have also shown that explicit syntax supervision in Transformers yield superior results.

6.2. Natural language understanding

Results obtained for the $\text{BERT}_{\text{BASE + POS}}$ model on the GLUE benchmark test set are presented in Table 4. $\text{BERT}_{\text{BASE + POS}}$ outperforms $\text{BERT}_{\text{BASE}}$ on all tasks, with the exception of matched MNLI. The improvements range from marginal to significant, with a maximum improvement of 0.8 points of the POS model over $\text{BERT}_{\text{BASE}}$ on CoLA. Fittingly, CoLA is a task which assesses the linguistic structure of a sentence, which is explicitly informed by POS embeddings. Moreover, $\text{BERT}_{\text{BASE + POS}}$ outperforms $\text{BERT}_{\text{BASE}}$ on tasks that are concerned with evaluating semantic relatedness. For examples of predictions where $\text{BERT}_{\text{BASE + POS}}$ performs better than $\text{BERT}_{\text{BASE}}$ made on the RTE dataset, see Table 5.

7. Conclusions

Infusing external syntax into NLP models has proven to be an effective method to improve performance on generation as well as classification tasks. In this work, we have infused the Transformer and BERT models with syntactic features including POS, word case, and subword position using a simple concatenation of token embeddings and trainable feature embeddings from external knowledge. We conducted experiments on the large-scale WMT '14 EN-DE translation task and low-resource EN-KO and EN-SI OpenSubtitles datasets. We found significant improvements in EN-DE translation, especially as the size of the training dataset is reduced. These results were confirmed in the low-resource setting, suggesting that our method performs well regardless of the training data size, a feature that makes it less rely on the training data. We then modified BERT by adding trainable syntactic embeddings directly to the input and found that our $\text{BERT}_{\text{BASE + POS}}$ model performs better than baseline on a number of GLUE downstream tasks. Thus, we have proven both the text generation and classification applications benefits from knowledge injection.

While many existing works either suggest that Transformers are capable of learning syntax implicitly or attempt to train Transformers to predict syntax in a multi-task setting, our results suggest that, when available, it is preferable to directly incorporate syntactic features as inputs to the NLP model to improve performance. As a next step, we plan to study the alignment of tokens between source and target sequences to determine the effect of adding syntax into the model. In addition, we plan to investigate syntax infusion into other NLP tasks such as co-reference resolution. In this context, syntactic information could be helpful in matching tokens to their antecedents by virtue of parts of speech, for instance.

References

[1] M.-T. Luong, H. Pham, C. D. Manning, Effective approaches to attention-based neural machine translation, arXiv preprint arXiv:1508.04025 (2015).

[2] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in neural information processing systems, 2017, pp. 5998–6008.

[3] G. Jawahar, B. Sagot, D. Seddah, S. Unicomb, G. Iñiguez, M. Karsai, Y. Léo, M. Karsai, C. Sarraute, É. Fleury, et al., What does bert learn about the structure of language?, in: 57th Annual Meeting of the Association for Computational Linguistics (ACL), Florence, Italy, 2019.

[4] A. Kuncoro, C. Dyer, J. Hale, D. Yogatama, S. Clark, P. Blunsom, Lstms can learn syntax-sensitive dependencies well, but modeling structure makes them better, in: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), 2018, pp. 1426–1436.

[5] T. Linzen, E. Dupoux, Y. Goldberg, Assessing the ability of lstms to learn syntax-sensitive dependencies, Transactions of the Association for Computational Linguistics 4 (2016) 521–535.

[6] H. Chen, S. Huang, D. Chiang, J. Chen, Improved neural machine translation with a syntax-aware encoder and decoder, arXiv preprint arXiv:1707.05436 (2017).

[7] J. Niehues, E. Cho, Exploiting linguistic resources for neural machine translation using multi-task learning, arXiv preprint arXiv:1708.00993 (2017).

[8] R. Sennrich, B. Haddow, Linguistic input features improve neural machine translation, arXiv preprint arXiv:1606.02892 (2016).

[9] B. Yang, Z. Tu, D. F. Wong, F. Meng, L. S. Chao, T. Zhang, Modeling localness for self-attention networks, arXiv preprint arXiv:1810.10182 (2018).

[10] P. Shaw, J. Uszkoreit, A. Vaswani, Self-attention with relative position representations, arXiv preprint arXiv:1803.02155 (2018).

[11] J. Li, Z. Tu, B. Yang, M. R. Lyu, T. Zhang, Multi-head attention with disagreement regularization, arXiv preprint arXiv:1810.10183 (2018).

[12] J. Im, S. Cho, Distance-based self-attention network for natural language inference, arXiv preprint arXiv:1712.02047 (2017).

[13] J. Bastings, I. Titov, W. Aziz, D. Marcheggiani, K. Sima'an, Graph convolutional encoders for syntax-aware neural machine translation, arXiv preprint arXiv:1704.04675 (2017).

[14] T. N. Kipf, M. Welling, Semi-supervised classification with graph convolutional networks, arXiv preprint arXiv:1609.02907 (2016).

[15] J. Li, D. Xiong, Z. Tu, M. Zhu, M. Zhang, G. Zhou, Modeling source syntax for neural machine translation, arXiv preprint arXiv:1705.01020 (2017).

[16] A. Currey, K. Heafield, Incorporating source syntax into transformer-based neural machine translation, in: Proceedings of the Fourth Conference on Machine Translation (Volume 1: Research Papers), 2019, pp. 24–33.

[17] J. Devlin, M.-W. Chang, K. Lee, K. Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, arXiv preprint arXiv:1810.04805 (2018).

[18] B. Zhang, I. Titov, R. Sennrich, Improving deep transformer with depth-scaled initialization and merged attention, arXiv preprint arXiv:1908.11365 (2019).

[19] P. Lison, J. Tiedemann, Opensubtitles2016: Extracting large parallel corpora from movie and tv subtitles (2016).

[20] F. Guzmán, P.-J. Chen, M. Ott, J. Pino, G. Lample, P. Koehn, V. Chaudhary, M. Ranzato, The flores evaluation datasets for low-resource machine translation: Nepali-english and sinhala-english, arXiv preprint arXiv:1902.01382 (2019).

[21] X. Glorot, Y. Bengio, Understanding the difficulty of training deep feedforward neural networks, in: Proceedings of the thirteenth international conference on artificial intelligence and statistics, 2010, pp. 249–256.

[22] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, S. R. Bowman, Glue: A multi-task benchmark and analysis platform for natural language understanding, arXiv preprint arXiv:1804.07461 (2018).

[23] C. Wang, S. Wu, S. Liu, Source dependency-aware transformer with supervised self-attention, arXiv preprint arXiv:1909.02273 (2019).