Augmented Transformer for Machine Translation

Rajvi Kapadia
rnkapadia@umass.edu

Sahil Mishra
sahmishra@umass.edu

Shanu Vashishtha
svashishta@umass.edu

Zaid Farooq
zfarooq@umass.edu

Zhiyang Xu
zhiyangxu@umass.edu

1 Problem statement

Attention mechanisms have become an integral part of most natural language processing (NLP) tasks including machine translation systems. In this work, we use the attention based Transformer model (Vaswani et al., 2017) for the task of neural machine translation (NMT) and improve its performance through data and model augmentation (Tran et al., 2017; Ratner et al., 2017). To improve the performance of NMT using transformers we take three different approaches. First, we combine Part-of-Speech (POS) embeddings and predicate embeddings to the existing word embeddings for the purpose of incorporating linguistic information in our model. Second, we experiment with initializing the word embeddings with pre-trained contextualized embeddings (BERT) (Devlin et al., 2018). Finally, we experiment with Syntactically-informed self-attention (Strubell et al., 2018a). We believe conducting these experiments will provide us with information about the importance of using linguistic information while training Transformer-based models.

2 What you proposed vs. what you accomplished

In the project proposal and milestones we proposed to incorporate linguistic information in the attention based Transformer model for the task of neural machine translation (NMT). The lists of things we proposed to do were:

- Learn POS tag embeddings and add them to the word features
- Generate fix one hot vectors for POS tags and concatenate them to word features
- Learn embeddings for POS and concatenate them to word features
- Add LISA head (Strubell et al., 2018a) to the last layer of the encoder
- Use a separate encoder with one LISA head (Strubell et al., 2018a) for POS embeddings and add the outputs from POS encoder to the outputs of token encoder
- Train the model to predict both the POS and tokens for the target language
- Initialize BERT embeddings (Devlin et al., 2018) and fine-tune the embeddings on our dataset

We accomplished all the tasks we proposed. However, contrary to our hypothesis we were unable to surpass the performance of the baseline model. We discuss our results in detail in the following sections.

3 Related work

Many papers have been published that attempt to improve the performance of Neural Machine translation using transformers. While most of these make architectural changes to the transformer model (Vaswani et al., 2017) to improve its performance, we tried both architectural changes to the model as well as data pre-processing while keeping the model same. We discuss some of the existing work here.

In this work we chose to embed syntactic information into the Transformer based model because such techniques have shown promising results recently. (Akoury et al., 2019) propose the syntactically supervised Transformer, which first autoregressively predicts a chunked parse tree before non-autoregressively generating all of the target tokens in one shot conditioned on the predicted parse. Unlike the Latent Transformer model (Kaiser et al., 2018), this model does not suffer from the discretization bottleneck because the creation of the chunked sequence is supervised during training through an external syntactic parser. The experiments conducted show significant increase in decoding speeds without increasing the complexity of the model architecture.

Interestingly, sophisticated embeddings like ELMo (Peters et al., 2018) and BERT (Devlin et al., 2018) are found to have robust syntax embedded in them according to probes (Hewitt and Manning, 2019). Hence, more sophisticated word embedding architectures in the future may totally remove the need for adding explicit linguistic information to the model architecture,especially for high-resource target languages. In case of low-resource target languages, syntactic information may still play an important role. (Tsai et al., 2019) show that by fine-tuning a pre-trained BERT model, we can build a multilingual model that has comparable or better accuracy to state-of-the-art language specific models by using knowledge distillation (teacher-student learning) which is a compression technique in which a small model is trained to reproduce the behavior of a larger model. Sentences from Wikipedia are extracted ,tokenized and used to compute cross entropy loss for the logits of each wordpiece with the distilled Bert model against the original Bert model. This distillation loss is used to train the student (Distilled BERT) from scratch using the teacher (Original BERT (Devlin et al., 2018))'s logits on unlabeled data. Afterwards the student model is fine-tuned on the labeled data the teacher is trained on.

(Zhou and Xu, 2015) work on jointly predicting Semantic Role Labelling (SRL) spans and predicates in a deep bi-directional long short-term memory (DB-LSTM) model obtaining state of the art predicted predicate SRL.

We also looked at several data augmentation techniques for improving the performance of the vanilla-Transformer. (Xie et al., 2019) present several data augmentation techniques for various NLP tasks that are very helpful in Semi Supervised learning (SSL). They find that instead of using baseline or naive data augmentation techniques if we use state of the art techniques for unlabeled data, SSL's performance can go up significantly. The two techniques they suggest for NLP are Back-Translation for question answering tasks and word replacing with TF-IDF for text classification. In the first technique an input in a language ”A” is translated into another language ”B” and then translated back to A to get another input with the same semantics. In the second technique, words with a lower TF-IDF score are replaced while those with a higher TF-IDF are kept which helps in determining the topic of the text.

(Diego Marcheggiani, 2017) use graph convolutional networks (GCNs), to model syntactic dependency graphs. GCNs over syntactic dependency trees are used as sentence encoders. They produce latent feature representations of words in a sentence. (Strubell et al., 2018b) present linguistically-informed self-attention which incorporates Parts Of Speech and predicates for the task of neural semantic role labeling. Unlike other self-attention mechanisms, this one has dedicated heads for Syntactic information. We experimented with using a LISA head in our transformer model to learn linguistic features.

(Glass et al., 2019) describe a new pre-training technique called Span selection which focuses the model on finding semantic connections between two sequences, and supports a style of cloze that can train deep semantic understanding without demanding memorization of world knowledge in the model. Their task at hand is question answering but a similar approach can be investigated for machine translation as well. (Yang et al., 2019b) introduce convolutional self-attention networks which enhance self-attention network models' ability to account for the dependencies amongst neighboring elements and to model the interaction between features extracted by multiple attention heads. The idea is to model locality for self-attention models and interactions between features learned by different attention heads in a unified framework. Authors use a 1-dimensional convolution to restrict the attention scope of the model to a fixed window of neighboring elements. They then extend the convolution to 2-dimensional area with the axis of attention head. This way model can interact local features across multiple attention heads.

(Chen et al., 2018) propose a hybrid model that combines the properties of vanilla RNNs, convolutional seq2seq models and transformer models. This seemingly outperforms all of the state of the art models for machine translation.

Context aware self-attention (Yang et al., 2019a) networks are another attempt at improving the performance of transformer models by contextualizing the transformations of the query and key layers. They use internal representations that embed both global and deep contexts.

Directional Self-attention Networks (Shen et al., 2017) are an improvement to the attention mechanisms in that the attention between elements from the input sequences is directional and multi-dimensional. They learn sentence embeddings based solely on the proposed attention model without any RNN/CNN. They are composed of directional self-attention with temporal order encoded, followed by a multi-dimensional attention that compresses the sequence into a vector representation.

Self-Attention with Relative Position Representations (Shaw et al., 2018) extend the self-attention mechanism to efficiently consider representations of the relative positions, or distances between sequence elements. They model the input as a labeled, directed, fully-connected graph.

(Dou et al., 2018) propose to do layer aggregation to simultaneously expose information learnt by the lower layers and higher layers in encoder decoder models by using multi-layer attention mechanism.

4 Your dataset

For our project, we are using the IWSLT'14 German to English dataset (Ott et al., 2019). For this project, we are focusing on the German → English translation task. Some of the examples from the training set are below -

S - *und was menschliche gesundheit ist, kann auch ziemlich kompliziert sein.*

T - *and it can be a very complicated thing, what human health is.*

S - *richard bransons leben auf 9000 metern*

T - *richard branson apos;s life at 30,000 feet*

The biggest challenge here is that sentences in themselves aren't complete rather they are fragments from a continuous talk and hence, difficult for a model to translate. The complete dataset statistics are present in Table 1.

set	sentences	En tokens	De tokens
train	172k	3.46M	3.24M
dev2010	887	20.1k	19.1k
tst2010	1,565	32.0k	30.3k
tst2011	1,433	26.9k	26.3k
tst2012	1,700	30.7k	29.2k

Table 1: IWSLT 2014 dataset statistics

4.1 Data preprocessing

In our pre-processing step, we tokenize the data using the Moses tokenizer, remove unwanted tokens, convert the remaining ones into lowercase, learn a byte pair encoding (subword NMT repository) using the train and valid split which is further used to transform the three splits. We also binarize the data for the training phase.

After the preprocessing step, the complete German dictionary has 8848 types while the English one has 6632 types. The training set has 160239 examples, the validation set has 7283 examples while the test set has 6750 examples. For the test set, we have a source sentence and a human verified translation which is used to calculate the BLEU score against the translated output for the task.

Our experiments involving BERT embeddings (Devlin et al., 2018) required the dataset to be preprocessed as well. BERT can take as input either one or two sentences and expects special tokens **[CLS]** and **[SEP]** to mark the beginning and end of each sentence respectively. All words were converted to lower-case before adding the special tokens. An example for En and De is shown below:

Before Tokenization: it can be a very complicated thing, the ocean.
After Tokenization: [CLS] it can be a very complicated thing, the ocean. [SEP]
Before Tokenization: das meer kann ziemlich kompliziert sein.
After Tokenization: [CLS] das meer kann ziemlich kompliziert sein. [SEP]

4.2 Data annotation

Since our dataset is a standard data set we didn't need to do data annotation of any form.

5 Baselines

Our baseline model is the original Scaled Dot Product Attention based Transformer model (Vaswani et al., 2017). The hyperparameters used are described below:

- Adam optimizer with an initial learning rate of $5e-4$

- Inverse square root scheduler
- Dropout = 0.3
- Weight Decay = $1e-4$

We trained the model for about 80 epochs. The loss value saturated around this time. We obtained a BLEU score of **34.72**, 68.9/43.1/29.0/19.9 (BP=0.960, ratio=0.961, syslen=126059, reflen=131161) on the test dataset when evaluating with a beam size of 5.

We couldn't find a pretrained model for reproducing the IWSLT2014 results in Pytorch. Neither did we come across an existing literature reporting the same. However, a benchmark BLEU score of 34.44 is reported for the IWSLT2015 dataset. Note that the test set for the two results are different. Since this was close to the value we obtained and independent trials by the team members resulted in similar final values, we decided to adopt this as our Baseline.

6 Your approach

After obtaining the POS tags, we will perform translation on the target language (English). Translation is a task in which the model understands a sentence $\mathbf{S} = \{s_1, s_2, ..., s_T\}$ in the source language to predict a sentence $\mathbf{t} = \{t_1, t_2, ..., t_T\}$ in the target language, where each output token t_i is one of $|V|$ possible tokens in the target language. In the vanilla transformer, the factorized conditional probability of target sentence is given by:

$$P(t_1, ..., t_T|S) = \prod_{i=1}^{t} P(t_i|t_1, ..., t_i, S)$$

The loss function we use is negative log likelihood:

$$\log P(\mathbf{t}|\mathbf{s}) = P(\mathbf{t}|\mathbf{s}) - \log \sum_{\mathbf{t}'} \exp(p(\mathbf{t}'|\mathbf{s}))$$

For this translation task, we come up with different approaches to incorporate linguistic information into our model:

Data Augmentation:

- The first approach (**POS Tags added to Embeddings**) is to use Pytorch built-in embedding layer to automatically generate embeddings for POS tags and add them to word embeddings. The embedding of POS tags will be updated during training.
- The second approach (**Concatenating one hot encoded POS tags**) is to create the POS embeddings as one hot vectors of length 24 and concatenate them with the existing word embeddings. The concatenated embeddings are passed into a linear layer and mapped back to the original size of word embeddings. The parameters in the linear layer are updated but the POS embeddings remain fixed.
- The third approach (**Concatenated learned POS tags**) is to use Pytorch built-in embedding layer to randomly initialize embeddings for POS tags of length 24 and concatenate them with word embeddings. Then the concatenated embeddings are passed into a linear layer and mapped back to the original space of word embeddings. Both the parameters in the linear layer and POS embeddings are updated during training.

Model Augmentation:

- The fourth approach (**One LISA head in the last encoder layer**) is to add one LISA head (Strubell et al., 2018a) on the last layer of the encoder. The LISA head has the bi-affine attention description (Hewitt and Manning, 2019) which is used for dependency parsing. We hypothesize this layer combined with POS information can help the model better understand the structure of the sentences.
- The fifth approach (**LISA layer trained with POS**) is to use a separate encoder with one LISA head (Strubell et al., 2018a) for POS tags. The output from POS encoder and token encoder will be added together and then sent to decoder. We expect the bi-affine attention (Hewitt and Manning, 2019) can provide structural information of the model. The performance of each model is shown in Table 2.

Contextualized Embeddings (BERT)

- Finally we initialized the vanilla pre-trained transformer model with **BERT Embeddings**. BERT embeddings (Devlin et al., 2018) extracted from the pre-trained HuggingFace transformer model have a dimensionality of 768. Training the vanilla-Transformer with 768 dimensional embeddings showed poor results. We believe that the model was overfitting to the training data due to a significant increase in the number of model parameters. Hence, we reduced the dimensionality of the BERT embeddings using Principal Component Analysis (PCA) before intitialization. We conducted three experiments with different dimensionality values - 512, 256 and 128.

At the onset, we didn't expect our approaches to fail or perform worse than the baselines we established. We manged to complete all the implementations listed here. To accomplish this, we used the fairseq repository (Ott et al., 2019) and made changes to its codebase and the Huggingface repository (Wolf et al., 2019). The file transformer.py is the one where we modified most of our code. We also had to make changes to the way the data is read for which we changed the utils file as well. Some changes were also required in the fairseq_task file to make the information flow across different parts of the code. We were running our experiments mostly on an AWS cluster. Some of the POS codes were sufficient to run on our laptops.

7 Experiments and Results

Generate POS Tags: The first step for our experiments was to generate POS tags. We used two different taggers to get the POS tags for our dataset. They both are described below

- **Stanford log-linear POS tagger** - We used the Stanford log-linear POS tagger to get the POS tags for our German and English dataset. This tagger is discussed in detail here (Toutanova et al., 2003). The tagger is basically a java implementation with several models for POS tagging. We used the left3words model for English because it is faster than other models. Similarly we used a faster model for German POS tagging. Even though we successfully got the POS tags for our dataset, we are unable to use it as we discovered that the models after tagging produced output that had mismatched number of lines compared to the original input. The number of lines in the original dataset for both German and English were 178526 whereas after POS tagging they increase to 191973 for English and 196646 for German. We couldn't find out the reason behind it and decided to use the SpaCy tagger instead.

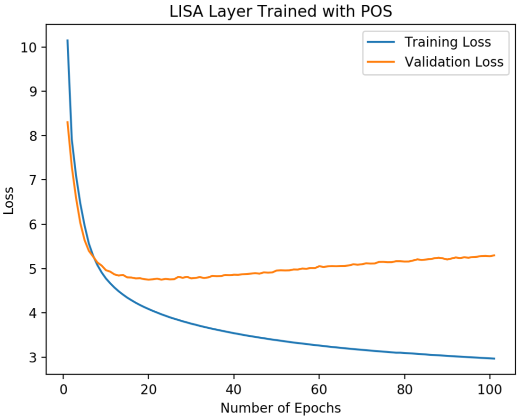

Figure 1: LISA Layer Trained with POS Tags.

- **SpaCy Tagger** - We used the SpaCy (Honnibal and Montani, 2017) library to tag our English and German datasets which is a well engineered library for many NLP tasks not limited by a specific domain and generalizes quite well to new datasets. We used en_core_web_sm model to label English sentences and de_core_news_sm for German. It takes 20 minutes to label each of the datasets. The output didn't have the lines mismatch problem we encountered for the previous tagger. One output for English is shown here: *And (cconj) it (pron) can (verb) be (aux) a (det) very (adv) complicated (adj) thing (noun) , (punct) what (det) human (adj) health (noun) is (aux) . (punct)*

Model	BLEU
Baseline transformer model	30.57
Adding POS tags	27.61
Concatenating one hot encoded POS tags	27.29
Concatenating learned POS tags	27.42
Prepending POS tags	15.97
One LISA head in the last encoder layer	27.80
LISA layer trained with POS	26.90

Table 2: Performance of Augmented models

Augmented models: After we obtained the POS tags, we had the dataset tokens in the form of *word_POS*. We split this data after reading, initialized embedding layers for each and performed one of the approaches listed above for our experiments. For training, we followed the same pre-processing steps except one difference. We didn't binarize the data this time instead used the raw implementation to conduct our experiments. This made us revise our baseline model. When we trained our model with this raw implementation, we obtained a baseline BLEU score of 30.57. Against this, we compared the models we trained later on.

Another change we did was add a POS dictionary for the model to read. Earlier in our pre-processing step, we created two dictionaries for the source and target languages. Since our dataset had a combined list of 24 POS tags, we created a new POS dictionary with these tags. We trained using the same set of hyper-parameters as mentioned before. Our max_tokens parameter was set to 4096. We used label smoothing CE loss with a label smoothing of 0.1 and a warmup updates of 4000.

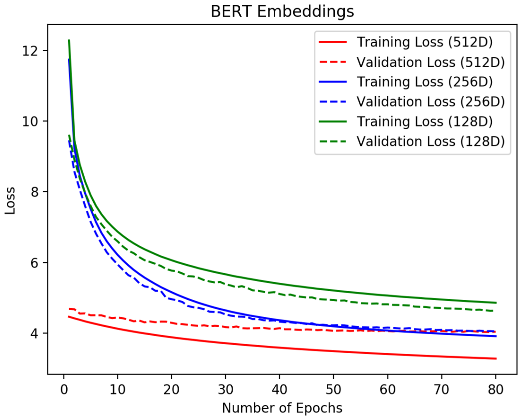

Figure 2: Performance of BERT Embeddings of various sizes.

Models initialized with BERT Embeddings: After adding [CLS] and [SEP] token to each sentence in the dataset, word embeddings for the IWSLT'14 dataset were extracted using the HuggingFace (Wolf et al., 2019) pre-trained models trained on the De and En datasets. We trained the vanilla transformer with these pre-trained BERT embeddings. The results are shown in Table 3.

8 Error analysis

We performed error analysis by looking at around 100 sample outputs for each model. We analysed them based on semantic and syntactic mistakes in translation. We compared the errors made by the model and have summarized our findings below along with a few examples. Some of the examples on which our baseline model performs poorly are shown below :

Model	BLEU
Baseline transformer model	34.50
BERT 512-d embeddings	31.37
BERT 256-d embeddings	31.68
BERT 128-d embeddings	25.53

Table 3: Performance of BERT models

Title	Model
Pred. 1	POS tags added to embeddings
Pred. 2	Concatenating one hot encoded POS tags
Pred. 3	Concatenated learned POS tags
Pred. 4	One LISA head in the last encoder layer
Pred. 5	LISA layer trained with POS
Pred. 6	Prepending POS tags
Pred. 7	BERT Embeddings

Table 4: index and the corresponding model

1. **Source**: die vereinigten staaten haben heute die höchste inhaftierungsrate der welt .
   Target: the united states now has the highest rate of incarceration in the world .
   Baseline: the united states today has the highest incarceration rate in the world .
   Prediction-1: the united states today has the highest fertility rate of the world
   Prediction-2: the united states today has the highest incarceration rate in the world .
   Prediction-3: the united states today has the highest fertility rate in the world .
   Prediction-4: the united states today has the highest mortality rate in the world .
   Prediction-5: the united states today has the highest average rate of the world .
   Prediction-6: the united states is the largest killer of the world.
   Prediction-7:
   - 512D: the united states today has the highest incarceration rate in the world .
   - 256D: the united states today has the highest incarceration rate in the world .
   - 128D: the united states today have the highest rate of the world .

2. **Source**: ihre eltern wurden in den 1840ern in virginia als sklaven geboren .
   Target: her parents were born in slavery in virginia in the 1840
   Baseline: her parents were born as slaves in the 1840s in virginia .
   Prediction-1: their parents were born in the design of virginia as slaves
   Prediction-2: her parents were born as a slave in virginia as a slave .
   Prediction-3: her parents were born at the mahabharata in virginia when slaves was born .
   Prediction-4: their parents were born as a slave in the 1920s in virginia .
   Prediction-5: their parents were born in the 1920s in virginia as slaves .
   Prediction-6: her parents were born in the middle of the city , and her parents were born in vietnam
   Predicition-7:
   - 512D: their parents were born in virginia as slaves in the 1840s .
   - 256D: their parents were born in virginia in the 1840s as slaves .
   - 128D: their parents were born in the 1840s in virginia as slaves .

3. **Source**: heute möchte ich ihnen von kreisen und offenbarungen erzählen
   Target: today i want to tell you about circles and epiphanies.
   Baseline: today i want to tell you about circles and revelations .
   Prediction-1: today i ¡unk¿ ¡unk¿ like to tell you about circles and revelations.
   Prediction-2: today , i want to tell you about circles and revelations .
   Prediction-3: today i want to tell you about circles and revelations .
   Prediction-4: today i want to tell you about circles and revelations.
   Prediction-5: today i want to tell you about circles and revelations .
   Prediction-6: going to tell you about the importance of ladies and gentlemen .
   Prediction-7:
   - 512D: today i want to tell you about circles and openings .
   - 256D: today i want to tell you about circles and revelations .
   - 128D: today i want to tell you about circles and openness .

Error Analysis: We compare the translations of the different experiments based on syntax and semantics, and we find minor syntactic errors in all the models which still allow for a good semantic translation. We find the model with concatenated one hot encoded tags (Prediction 2) as the one with the best observed syntax and semantic translations failing on very few ambiguous sentences. For example, in the first sentence, only the prediction 2 correctly translates "incarceration" which is a rare word in the corpus. This model has better performance because it can put more attention on the syntactically important word in the sentence.
POS tags added to embeddings (Prediction 1) perform semantically well on most of the sentences but fail to grasp syntactic and grammatical structures of English sentences.

The model with one LISA head (Prediction 4) in the last encoder layer) performs well semantically after the concatenated one hot embeddings although there are some noticeable semantic errors that could have possibly been reduced by some more syntactic supervision. But as is clear from the loss curve, the model has overfit to our data.
Prepending POS tags (Prediction 6) does not show great semantic or syntactic understanding of the English language showing that the model has slightly overfit. We can also see that in all of the three examples, concatenating one hot encoded POS tags gives better translation than just adding the POS tags to embeddings. This is strange, given that in the BLEU score the behavior is opposite. This is most probably because we got unlucky in terms of picking the examples from the dataset. Surely, in the rest of the dataset, adding POS tags to embeddings does better than concatenating one hot encoded POS tags.

It is also strange to see that concatenating POS tags and then finetuning the embeddings (Prediction 3) actually ends up learning a very deviant subject words in the sentence. For example, in the second sentence, the model somehow learns to predict "mahabharata" which is completely irrelevant to the context of the sentence. This is because the model when finetuning the emebeddings also finetunes POS embeddings since they are concatenated with the word embeddings and this causes the difference in prediction for the main subject words of the sentence.

The experiments involving the BERT embeddings (Devlin et al., 2018) show some promising results but are still unable to beat the baseline score.

Firstly, it is clear to see that 128-Dimensional embeddings are unable to capture all the important features in the source language. As seen in the learning curves in Figure 2 the validation loss for this model is always lower than the training loss which means this model is under-fitting. The training loss went to plateaus around 5 as the learning slows down. Since the training curve plateaus, we can say that the training has converged.

The models trained with 512-dimensional and 256-dimensional embeddings have very similar performances but have a slightly low BLEU score than the baseline. Looking at the loss curves for the the model trained 512-D embeddings it is clear that the model overfit on the training data as the validation loss does not seem to reduce at all. In case of the model with 256-D embeddings, the model seems to train well and may have better performances after some possible hyperparameter tuning.

9 Contributions of group members

- Zhiyang: use SpaCy to generate POS tags for German / preprocess datasets / modify Fairseq transformer model to add/concatenate POS embeddings to token embeddings / modify Fairseq transformer model to incorporate LISA head in the encoder / write reports

- Zaid: Wrote the script to get the BERT embeddings for the English and German dataset / Used the Stanford POS tagger to tag the German and English dataset. This was later found to be changing the number of lines in the dataset so we had to switch to SpaCY / Contributed to the error analysis section of the report / Read 5 extra papers for citing in the related word section of the final report. This is on top of the 4 core papers that the whole team read.

- Shanu: Used SpaCy to generate POS tags for English / Pre-process datasets / Modify Fairseq transformer model to add and concatenate POS embeddings to token embeddings / modify Fairseq transformer model to incorporate LISA head in the encoder / Train models on AWS / Wrote reports.

- Sahil Mishra: Preprocess datasets to incorporate BERT embeddings / performed Principal Component Analysis (PCA) to reduce the dimensionality of 768-dimensional BERT embeddings to 512, 256 and 128 respectively / trained Fairseq vanilla-Transformer model with BERT embeddings initializations / write reports

- Rajvi : Read papers / Looked into the existing LISA implementation / Looked into the existing BERT embeddings code / Debugging elmo/bert embeddings,looked into error analysis

10 Conclusion

In this work, we conducted multiple experiments to test the significance of augmenting Transformer-based models with external linguistic information. Our initial intuition was that the models themselves weren't incorporating POS information and hence, we explored that line of thought. During these experiments we learned that while there are exceptional open-source tools like HuggingFace (Wolf et al., 2019) and Fairseq (Ott et al., 2019) it can be significantly difficult to use different datasets with these existing models. It is not trivial to hack around their codebase and accomplish what you set out to do. The way that the datasets are read also takes a while to understand and we had to spend time figuring out how to change them. Some models can also be extremely brittle and can perform poorly even with minor changes like increasing the dimension size of the embeddings.

Our main takeaway is that these models somehow have linguistic information encoded in them. We were initially surprised by our results because we expected a significant improvement over the baseline because our models had more linguistic information. We later realized that our model may be overfitting to the training data due to the increase in parameters. We tried different ways but we couldn't beat the baseline we established.

Future work could include working with a larger dataset like the WMT so that we have a more concrete evaluation of our hypothesis. We would also like to conduct multiple experiments for tuning the hyperparameters more to fit to the dataset better.

References

Akoury, N., Krishna, K., and Iyyer, M. (2019). Syntactically supervised transformers for faster neural machine translation. *arXiv preprint arXiv:1906.02780*.

Chen, M. X., Firat, O., Bapna, A., Johnson, M., Macherey, W., Foster, G., Jones, L., Parmar, N., Schuster, M., Chen, Z.-F., Wu, Y., and Hughes, M. (2018). The best of both worlds: Combining recent advances in neural machine translation. In *ACL*.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Diego Marcheggiani, I. T. (2017). Encoding sentences with graph convolutional networks for semantic role labeling. *arXiv:1703.04826*.

Dou, Z.-Y., Tu, Z., Wang, X., Shi, S., and Zhang, T. (2018). Exploiting deep representations for neural machine translation. In *EMNLP*.

Glass, M., Gliozzo, A., Chakravarti, R., Ferritto, A., Pan, L., Bhargav, G. P. S., Garg, D., and Sil, A. (2019). Span selection pre-training for question answering.

Hewitt, J. and Manning, C. D. (2019). A structural probe for finding syntax in word representations. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4129–4138.

Honnibal, M. and Montani, I. (2017). spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear.

Kaiser, Ł., Roy, A., Vaswani, A., Parmar, N., Bengio, S., Uszkoreit, J., and Shazeer, N. (2018). Fast decoding in sequence models using discrete latent variables. *arXiv preprint arXiv:1803.03382*.

Ott, M., Edunov, S., Baevski, A., Fan, A., Gross, S., Ng, N., Grangier, D., and Auli, M. (2019). fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of NAACL-HLT 2019: Demonstrations*.

Peters, M. E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., and Zettlemoyer, L. (2018). Deep contextualized word representations. *CoRR*, abs/1802.05365.

Ratner, A. J., Ehrenberg, H., Hussain, Z., Dunnmon, J., and Ré, C. (2017). Learning to compose domain-specific transformations for data augmentation. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Shaw, P., Uszkoreit, J., and Vaswani, A. (2018). Self-attention with relative position representations. In *NAACL-HLT*.

Shen, T., Zhou, T., Long, G., Jiang, J., Pan, S., and Zhang, C. (2017). Disan: Directional self-attention network for rnn/cnn-free language understanding. In *AAAI*.

Strubell, E., Verga, P., Andor, D., Weiss, D., and McCallum, A. (2018a). Linguistically-informed self-attention for semantic role labeling. *CoRR*, abs/1804.08199.

Strubell, E., Verga, P., Andor, D., Weiss, D., and McCallum, A. (2018b). Linguistically-informed self-attention for semantic role labeling. *CoRR*, abs/1804.08199.

Toutanova, K., Klein, D., Manning, C. D., and Singer, Y. (2003). Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 173–180, Stroudsburg, PA, USA. Association for Computational Linguistics.

Tran, T., Pham, T., Carneiro, G., Palmer, L., and Reid, I. (2017). A bayesian data augmentation approach for learning deep models. In *Advances in Neural Information Processing Systems (NeurIPS)*.

Tsai, H., Riesa, J., Johnson, M., Arivazhagan, N., Li, X., and Archer, A. (2019). Small and practical bert models for sequence labeling. *arXiv preprint arXiv:1909.00100*.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., and Brew, J. (2019). Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Xie, Q., Dai, Z., Hovy, E. H., Luong, M.-T., and Le, Q. V. (2019). Unsupervised data augmentation for consistency training.

Yang, B., Li, J., Wong, D. F., Chao, L. S., Wang, X., and Tu, Z. (2019a). Context-aware self-attention networks. *ArXiv*, abs/1902.05766.

Yang, B., Wang, L., Wong, D., Chao, L., and Tu, Z. (2019b). Convolutional self-attention networks.

Zhou, J. and Xu, W. (2015). End-to-end learning of semantic role labeling using recurrent neural networks.