Transfer Learning in Natural Language Processing

By

Sara Sultan

In

Deep Learning with Python

Introduction

Transfer learning is the process of pre-training a model on a large-scale data set and then using that pretrained model to conduct learning for another downstream task (i.e. target task). It is the idea of overcoming the isolated learning paradigm and utilizing knowledge acquired for one task to solve another related task. Transfer learning has been highly successful in the field of computer vision as most of the features learned in images are universal for all image processing applications. Natural Language Processing (NLP) can also benefit from transfer learning. In NLP, initial use of transfer learning was in terms of word embeddings like Word2Vec, Elmo, Skip gram and Glove. However these word embeddings are only a way to vectorize text, which is later used as input features to deep learning models. In the last year and a half, much progress has been made in introducing models that utilize full potential of transfer learning. These recent developments include GPT (Generative Pretrained Transformer), BERT(Bidirectional Encoder Representations from Transfomers) and XLNet.

In this project, use of transfer learning on BERT and XLNet is explored by fine-tuning them on Imdb movie reviews dataset. Fine-tuning of GPT-2 is more complicated and not pursued within the scope of this project. This report covers the theory behind BERT and XLNET, details some of the experiments performed and concludes with insights derived from the results.

Theory

BERT

BERT is the product of work from researchers at Google AI . It has caused a stir in the Machine Learning community by presenting state-of-the-art results in a wide variety of NLP tasks, including Question Answering (SQuAD v1.1), Natural Language Inference (MNLI), and other datasets. There are two steps in BERT framework: pre-training and fine-tuning. During pre-training, the model is trained on a huge corpus of unlabeled data. For finetuning, the BERT model is first initialized with the pre-trained weights, and these weights are then fine-tuned using labeled data from the target tasks. The distinctive feature of BERT is its unified architecture across different tasks.

How BERT works:

BERT makes use of Transformer, an attention mechanism that learns contextual relations between words (or sub-words) in a text. Transformer includes two separate mechanisms — an encoder that reads the text input and a decoder that produces a prediction for the task. Since BERT's goal is to generate a language model, only the encoder mechanism is necessary. As opposed to directional models, which read the text input sequentially (left-to-right or right-to-left), the Transformer encoder reads the entire sequence of words at once. Therefore it is considered bidirectional, though it would be more accurate to say that it's non-directional. This characteristic allows the model to learn the context of a word based on all of its surroundings (left and right of the word). The input is a sequence of tokens, which are first embedded into vectors and then processed in the neural network. The output is a sequence of vectors, in which each vector corresponds to an input token with the same index (Rani Horev Nov, 2018).

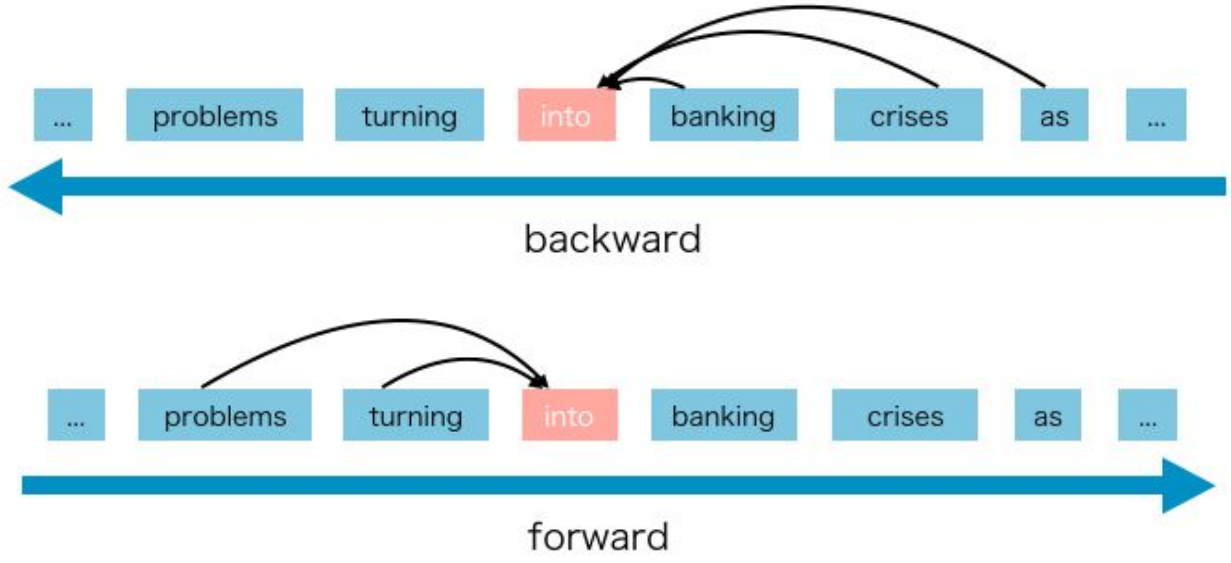
Defining a prediction goal is a challenge in training language models. Many models predict the next word in a sequence (e.g. “The child came home from ___”), a directional approach which inherently limits context learning. To overcome this challenge, BERT uses two training strategies:

Masked Leraning Model

Before feeding word sequences into BERT, 15% of the words in each sequence are replaced with a [MASK] token. The model then attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.

Next Sentence Prediction (NSP)

In the BERT training process, the model receives pairs of sentences as input and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. During training, 50% of the inputs are a pair in which the second sentence is the subsequent sentence in the original document, while in the other 50%, a random sentence from the corpus is chosen as the second sentence. The assumption is that the random sentence will be disconnected from the first sentence.



XLNet

XLNet is the most recent addition to the NLP arsenal, beating BERT at about 20 NLP benchmarks. It is a generalized autoregressive pretraining method. Autoregressive (AR) language model is a kind of model that uses the context word to predict the next word, but here the context word is constrained to two directions, either forward or backward.

How XLNet works

The AR language model only can use the context either forward or backward, so how to let it learn from bi-directional context? The language model consists of two phases, the pre-train phase, and fine-tune phase. XLNet focuses on pre-train phase. In the pre-train phase, it proposes a new objective called Permutation Language Modeling as shown below (Xu Liang et al., June, 2019):

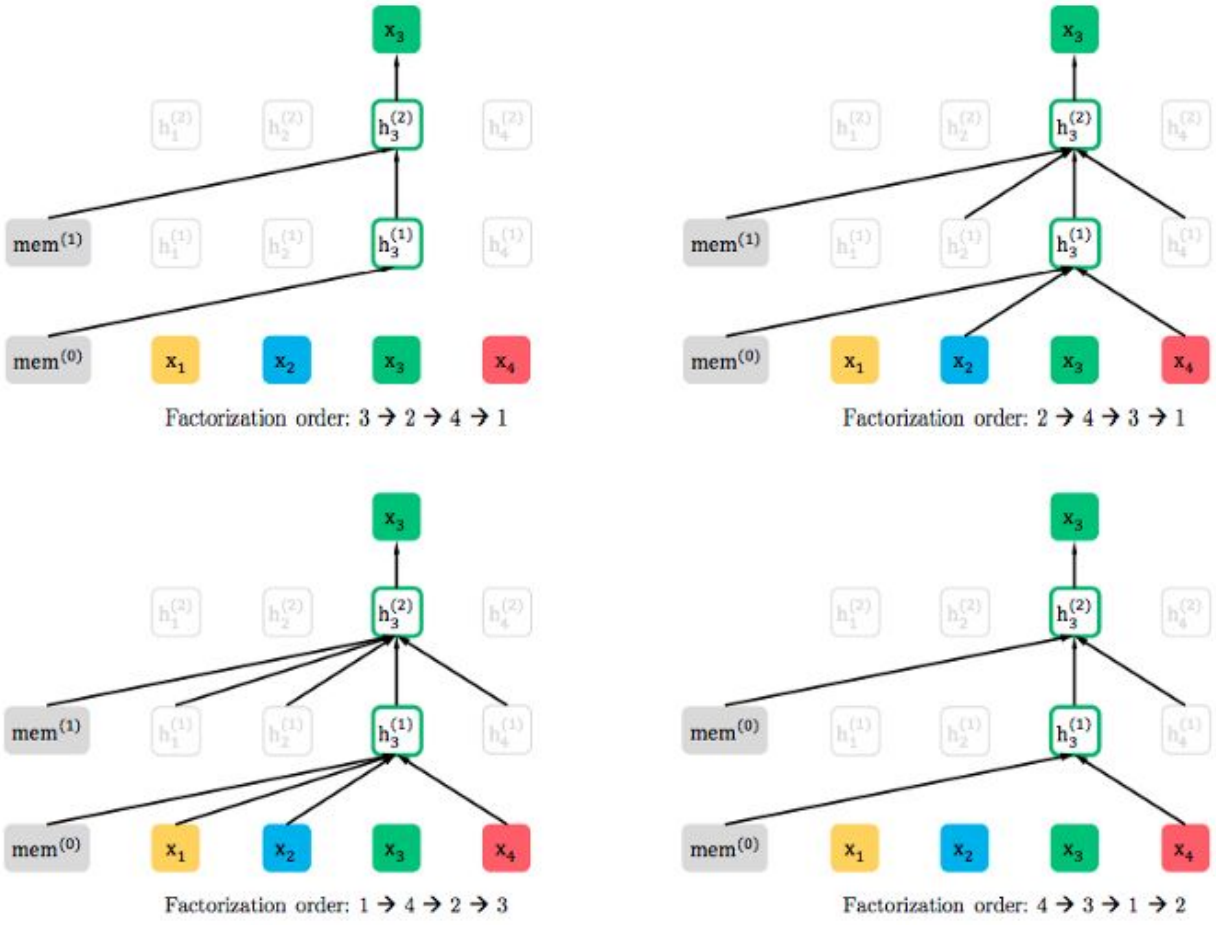


Figure 1: Illustration of the permutation language modeling objective for predicting x_3 given the same input sequence $\mathbf{x}$ but with different factorization orders.

Experiments

Performance of BERT and XLNET is tested on Imdb Movies Review dataset. The dataset consists of 12500 training samples and 12500 test sample. The text need to be classified into two classes: positive review and negative review. Hyperparameters identified for the experiments inlcude:

- Sentence Length (Sequence Length)
- Learning Rate

- Batch Size

First the training and test data is cleaned and tokenized using the vocabulary set provided alongside pretrained models. Then the pretrained models are loaded and modified for the Imdb tasks, while preserving the weights. Categorical Cross-entropy is used as the loss metric. Adam optimizer is used along with learning rate warm-up so as to preserve the learning from pre-training while fine-tuning the model.

Using BERT

To prepare BERT for fine-tuning, its pretrained model is loaded, masked layer used for pretraining is removed and fully-connected layer is added with softmax activation. During the experiments, it was seen that model performance is very sensitive to learning rate, batch size and sentence length. Having too high a learning rate resulted in model starting to diverge. A higher learning rate could only be used with a higher batch size. Batch size, in turn, was limited by gpu memory. Sentence length is also an important hyperparameter. It was tested with 128, 256, 512 and fine-tuning with 256+prediction with 512.

Using XLNet

Since XLNet was recently released, there are not many keras implementations of XLNet available that are suitable for fine-tuning. This resulted in digging into original tensorflow implementation by the authors and figuring out the less obvious hyperparameters. The same needed to be done for data preprocessing where XLNet uses special characters to identify separators and sentence end. Significant improvement was achieved by these correct parameters. One of the parameters that can be further explored concerns the use of two fully-connected (FC) layers with a dropout layer in-between. In our experiment, we used the first FC layers 256 units and tanh activation and second FC layer with 2 units and softmax activation. Other variants for fine-tuning XLNet found across different forums included not using the first FC layer at all or using 1024 units first FC layer.

Results

Results of most successful runs are shown in the table below.

Table 1

Dataset	Model	Sentence Length	Batch Size	Learning Rate	Accuracy
Imdb	BERT	128	4	10^{-5}	88%
Imdb	BERT	256	4	10^{-5}	91.3%
Imdb	BERT	512	1	$5x10^{-6}$	93.3%
Imdb	BERT	512	2	$5x10^{-6}$	93.9%
Imdb	XLNet	256	4	$5x10^{-5}$	92.36%
Imdb	XLNet	512	1	$5x10^{-6}$	92.7%
Tweet Sentiment	BERT	32	64	10^{-6}	38%

It can be seen that best results on Imdb dataset are achieved with BERT running with 512 sentence length and 2 batch size. The batch size was limited by gpu memory. It can be seen that increasing the batch size from 1 to 2 resulted in a 0.6% performance gain. This also explain that XLNet while performing better than BERT at 256 sentence length does not perform as well on 512 sentence length. With higher batch sizes, XLNet is expected to outperform BERT as shown by public benchmarks.

A small experiment was also performed on 13 class tweet sentiment data (used in first class assignment) with 36.5% test accuracy achieved. This also almost the similar result as achieved by running a much smaller model. However, the training accuracy was also at 38% which meant that there is a lot more room for the model to learn. Doing a hyperparameter space search and spelling corrections can help in improving the results.

Conclusion

Transfer learning using BERT and XLNet improves the model performance significantly and it is very easy to setup fine-tuning of these models. However the models are very prone to hyperparameters and need multiple runs to find the best hyperparameters. With parameters size ranging hundereds of millions, training these models is compute intensive. TPUs are preferred over GPUs as they provide higher memory, thus making it possible to use higher batch size. All in all, feasibility of transfer learning has ushered in an exciting era for the applications of NLP.