Charlie Truong
CS-410: Text Information Systems
November 5, 2021

Deploying BERT Models in Applications

Introduction

Bidirectional Encoder Representations from Transformers (BERT) is a state-of-the-art pre-trained language model that can be easily fine-tuned for specific tasks such as text classification. This review of BERT will provide a high-level overview of BERT's design, how to build and deploy BERT models using Amazon Web Services (AWS) and Hugging Face libraries, and options for improving inference performance.

BERT Overview

BERT is a variation of a transformer encoder model [1]. In order to better encode an input word, transformer models use a technique called self-attention that leverages contextual embeddings from other inputs words. Unlike previous transformer models, BERT uses a masked language model (MLM) approach to pre-train its model. That is, a percentage of the input tokens is masked at random, and the model is trained to predict this hidden word. Masking these tokens allows BERT to use the surrounding context from both directions safely without leaking the output labels into the training inputs. In addition to MLM, BERT is also trained to better understand sentence relationships by predicting the next sentence given an input sentence.

BERT was pretrained on the BooksCorpus (~800M words) and Wikipedia (~2.5T words). Variations of the BERT model was trained with the base BERT model containing 110M parameters and the large BERT model consisting of 340M parameters. The largest transformer model at the time consisted of 235M parameters. Across various evaluation scores, BERT topped the then state-of-the-art models. For example, depending on the model architecture, BERT achieved a 4.5% to 7.0% accuracy improvement in the General Language Understanding Evaluation (GLUE) benchmark.

Deploying with Hugging Face and AWS

The Hugging Face transformers library is one of the easiest ways to use BERT and fine tune it. Hugging Face is a startup company specializing in enabling the development and deployment of natural language processing (NLP) models [2]. Its open-source transformers library provides useful methods to instantiate a pre-trained model as well as tokenizers to easily pass inputs into models [3]. Although the library supports a variety of models in addition to BERT, each of the models uses a common interface. So, it is simple to plug and play other models to evaluate how they perform differently.

The primary objects in the library are the following:

- Model classes – Wraps pre-trained PyTorch or Keras language models
- Configuration classes – Contains the parameters used to create the models
- Tokenizer classes – Has methods to encode strings to token embeddings as model inputs

AWS Sagemaker is an option for deploying Hugging Face models [4]. Sagemaker is a suite of machine learning tools provided within the AWS cloud platform [5]. At its core, it offers the ability to use Jupyter notebooks, run training jobs, and deploy model endpoints using a variety of server sizes and configurations. Sagemaker provides an SDK for working with common model implementations, which includes Hugging Face.

The primary object in Sagemaker's SDK is an Estimator class that abstracts the AWS API for the user [6]. When a user invokes an Estimator's "fit" method, then it is making AWS Sagemaker API calls to spin-up a server and execute the user-defined model training script. Sagemaker training jobs and model endpoints execute code in Docker containers, which are isolated virtualized environments that contain the code's required dependencies. So, the HuggingFace Estimator subclass ensures that the user's training script runs in an environment with the necessary Hugging Face libraries installed. The user can specify different versions of the transformers library if necessary.

Sagemaker offers deploying models via real time endpoints or batch inference jobs [7]. The chosen deployment solution should depend on the application's requirements. For instance, it would not be cost efficient to have a real time endpoint when a nightly job to update prediction values to a database or data file would be sufficient.

Speed-up Inference with Smaller Models

As previously mentioned, BERT is a large model with several hundred parameters. There may need to be optimization efforts to achieve one's cost and performance requirements. Before optimizing, it's important to establish an initial baseline for how the standard BERT model performs with typical hardware. Potentially, it will satisfy the project's goals with no further optimization. If improvements are necessary though, one may consider using GPU during inference as well evaluating smaller models that aim to achieve the same prediction performance as BERT.

First, GPUs can be used during inference to improve performance. AWS Sagemaker provides an Elastic Inference feature to attach additional GPU acceleration to non-GPU machines [8]. This would be less expensive than standing up a traditional GPU machine normally used for training. There are also less expensive GPU machines such as the G4 instance types that are not ideal for training but can be a cost-effective solution for inference [9].

Additionally, there are other models that aim to achieve similar performance to BERT but are smaller. Smaller models will be faster and more efficient when serving predictions and likely will be good enough in real-world applications.

Distilbert is one example developed by Hugging Face [10]. It uses knowledge distillation where a smaller model trains to approximate the behavior of a larger model such as BERT. Distilbert achieves 97% of the original BERT model's GLUE benchmark performances but with 40% fewer parameters.

Roblox, a gaming company, was able to make use of Distilbert plus additional improvements to handle a load of one billion inferences a day under 20ms [11]. Just using Distilbert over the base BERT model achieved nearly two times improvement in latency, from 330ms to 171ms. They made further performance improvements by not zero padding their inputs when the batch size is just one during inference. Lastly, they used quantization, a technique to make model weight representations smaller, to make the model more efficient at a slight cost to its prediction performance. Ultimately, they were able to achieve this strong performance using just CPUs, which is more cost efficient than serving with GPUs.

Again, premature optimization should be avoided. As with any project, one should understand the requirements first, where potentially initial performance results are sufficient for your goals.

Conclusion

BERT is a breakthrough for natural language processing due to its high performance as well as its ability to be easily fine-tuned for different domains and applications. The Hugging Face transformers library provides easy access to work with BERT and similar models, and cloud providers like AWS offer simple methods to deploying BERT models into production. If necessary, there are options for improving a deployed BERT model's inference speed such as using GPU machines or using a lighter-weight BERT-inspired model like Distilbert.

References

1. J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", arxiv.org, October 11, 2018. [Online]. Available: https://arxiv.org/pdf/1810.04805.pdf [Accessed October 23, 2021]
2. Hugging Face Homepage. [Online]. Available: https://huggingface.co [Accessed October 23, 2021]
3. Hugging Face, "Transformers". [Online]. Available: https://huggingface.co/transformers/index.html [Accessed October 23, 2021]
4. "Deploy Models to Amazon SageMaker". [Online]. Available: https://huggingface.co/docs/sagemaker/inference [Accessed October 31, 2021]
5. AWS Sagemaker Product Page. [Online]. Available: https://aws.amazon.com/pm/sagemaker/ [Accessed October 31, 2021]
6. AWS Sagemaker Estimator Documentation. [Online]. Available: https://sagemaker.readthedocs.io/en/stable/api/training/estimators.html [Accessed October 31, 2021]
7. AWS Sagemaker Batch Transform Documentation. [Online]. Available: https://docs.aws.amazon.com/sagemaker/latest/dg/batch-transform.html [Accessed October 31, 2021]
8. Q. Li, D. Ping, and L. Yu, "Fine-tuning a PyTorch BERT model and deploying it with Amazon Elastic Inference on Amazon SageMaker", July 15, 2020. [Online]. Avaiable: https://aws.amazon.com/blogs/machine-learning/fine-tuning-a-pytorch-bert-model-and-deploying-it-with-amazon-elastic-inference-on-amazon-sagemaker/ [Accessed October 31, 2021]
9. AWS G4 EC2 Instance Types Product Page. [Online]. https://aws.amazon.com/ec2/instance-types/g4/ [Accessed October 31, 2021]
10. V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "DistilBERT, a distilled version of BERT: smaller, faster, cheaper, and lighter", arxiv.org, October 2, 2019. [Online]. Available: https://arxiv.org/pdf/1910.01108.pdf [Accessed November 3, 2021]
11. "How We Scaled Bert to Serve 1+ Billion Daily Requests on CPUs", May 27, 2020. [Online]. Available: https://blog.roblox.com/2020/05/scaled-bert-serve-1-billion-daily-requests-cpus/ [Accessed November 3, 2021]