Chapter 1

Resources

1.1 Models and Tokenizers

The models used in this project come from two different sources. The GPT-2 XL model is based on the OpenAI's GPT2 model[6]. The GPT-2 XL model has 48 attention modules as compared to the 12 found in the base GPT2 model. This gives the GPT-2 XL model more capability in terms of language modelling tasks but also requires significantly more computing resources to train and work with[5]. This translates to 117 million parameters in the GPT2 model vs 1.6 billion parameters in the GPT-2 XL model. Given that the COMET-DISTIL models are based on the GPT-2 XL model, we opted to utilize the naive GPT-2 XL model as our baseline for comparison. We downloaded the GPT-2 XL model by using the Hugging Face model library[2]. As our tokenizer, we used the pre-trained tokenzier also available from the Hugging Face model library[2].

The second model we used was the COMET-DISTIL model[9] that was trained on the ATOMIC-10X knowledge graph. The COMET-DISTIL model was downloaded from the author's provided website that can be found on their GithHub page[4]. The tokenizer used with this model was also downloaded from the same source. The tokenizers for both models were adjusted to add EOS tokens to ensure that they could work with the datasets that were used for evaluating the models.

In order to train and test the models, both models had different language modelling heads (depending on the task) connected to the models. The heads were then trained on the training set of the datasets and then evaluated on the evaluation set of the datasets.

1.2 Datasets

In order to test the performances of the respective models, several standard datasets are available. This included the widely recognized GLUE dataset[8].

The benchmark was chosen based on its wide variety of tasks and the difficulty associated with each of these tasks.

1.3 Compute Resources

To train and evaluate the performance of the respective models, Lambda Compute Instances at the University of Alberta was utilized. The machines included two Nvidia N100 Gpus with 24GB of memory. Due to the large size of the GPT-2 XL Model (5.2 Billion Parameters), the DeepSpeed[7] library from Microsoft was also utilized to help fit the models within the GPU memory.

1.4 Development Environment

1. Python 3.8.13
2. Conda 4.11.0 (Environment Manager)
3. jupyterlab 3.3.2 (IDE / Notebook)
4. PyTorch 1.11.0 (Deep Learning Framework)
5. transformers 4.27.0.dev0 (Hugging Face Library)
6. DeepSpeed 0.8.1 (Model Training Utility)

1.5 Python, JupyterLab, and Conda

We utilized a combination of tools and environments that are foundational in the modern data science and machine learning workflows: Python, JupyterLab, and Conda to conduct these experiments.

1. **Python**: Python has become the backbone of the modern Data Science and Machine Learning world due to its ease of use and the widely supported libraries built on top such as TensorFlow, Pandas and PyTorch. In addition, it is also highly performant as it is able to utilize bindings with other low level languages such as C/C++ for intensive computing tasks. This is essentially how numpy is able to do math so quickly!
2. **JupyterLab**: An interactive development environment that provides a platform for writing and executing code, visualizing data. It's an evolution of the original Jupyter notebooks.
3. **Conda**: An open-source package management system and also an environment management system. Conda makes it easy to install multiple versions of software packages and their dependencies in isolated environments. This ensures consistent and reproducible results across different stages of a project and across different team members' setups.

1.6 PyTorch

PyTorch[3] is one of the most well-known Deep Learning frameworks. Developed originally by the Facebook's Research Lab (FAIR), it is currently an integral part of the Hugging Face library which was also utilized for this project. It is used both by researchers and developers. It offers several advantages which allow us to train our models effectively even when facing low memory constraints. This is especially important when dealing with large language models such as GPT-2 XL.

Some of the offerings that come with PyTorch are it's Dynamic Computation Graph which allow us to change the model architecture on the fly. Another advantage is the rich ecosystem of libraries that either support or utilize PyTorch such as Hugging Face and DeepSpeed.

1.7 Hugging Face

Another key piece of software that was utilized to perform these experiments was the Hugging Face[10] library. It is an open source framework that provides many state-of-the-art implementations of NLP models, particularly transformer-based models such as BERT, GTP-2 and GPT-2-xl.

1. **Pre-trained Models**: The library boasts a broad selection of pre-trained models. This feature significantly eases the process of fine-tuning models for specific tasks, removing the necessity to train extensive models from the ground up. We capitalized on this by utilizing the GPT-2-xl models available from the model hub.

2. **Flexibility**: While the Transformers library seamlessly integrates with popular deep learning frameworks like PyTorch and TensorFlow, users have the freedom to select their preferred backend. In addition, Hugging Face offers native integrations with Zero and DeepSpeed libraries making model training possible even with limited hardware resources.

3. **User-friendly API**: The design of the API is intuitive, ensuring that tasks such as tokenization, training, and inference are both streamlined and accessible with minimal coding. This user-centric design approach caters to both novices and experts in the field of NLP. The library also comes with a very hand run_glue.py script that we were able to modify to run our experiments for the glue dataset.

1.8 Deepspee and ZeRO

DeepSpeed[7] is an open-source deep learning optimization library that offers several techniques for distributed training and model parallelism. One of its most renowned components is ZeRO (Zero Redundancy Optimizer), a memory optimization technology.

1. **Scalability**: DeepSpeed drastically improves the scale and speed of training. It can support model sizes of over 100 billion parameters on current generation hardware.

2. **ZeRO (Zero Redundancy Optimizer)**: A key innovation of DeepSpeed, ZeRO optimizes the memory utilization across distributed model training. By reducing memory redundancy across the data parallelism stages, ZeRO enables the training of models that are up to 10x larger without any significant increase in computational resources.

3. **Memory Efficiency**: Through its optimization techniques, such as activation checkpointing, DeepSpeed allows for more significant model sizes and batch sizes without the need for proportional increases in memory.

4. **Integration**: DeepSpeed is designed to be easily integrated into existing PyTorch models, making it a versatile choice for many deep learning applications.

1.9 Weights & Biases

We also utilized Weights & Biases (W&B)[1], a platform designed to help teams track their machine learning experiments, visualize metrics, and share findings.

1. **Experiment Tracking**: W&B provides comprehensive logging of hyperparameters, outputs, code, and metrics. By having a centralized record, researchers and developers can easily compare different runs and iterations, making it simpler to iterate over model architectures and parameters.

2. **Visualization**: Through interactive charts and graphs, the platform facilitates a deeper understanding of model behavior. This aids in identifying patterns, anomalies, or areas of improvement in model training and validation.

3. **Reproducibility**: The platform ensures that every detail of the experiment, from code to configurations, is logged. This is crucial for both verifying results and building upon previous work without redundancy.

4. **Collaboration**: W&B promotes collaboration by allowing teams to share findings, make comments, and build on each other's work seamlessly. This feature is especially beneficial for distributed teams or projects with multiple stakeholders. This allowed our team to collaborate and share findings on this project as we were all located remotely.

For the experiments detailed in this report, Weights & Biases was employed to monitor the training process, analyze results, and ensure reproducibility across different runs.