Project Report - Atlas RAG-based Multiple Choice QA

Describe in your own words how Atlas works.

Atlas is an advanced machine learning model which is based on retrieval-augmented language models. In essence, it combines the best features of generative AI models and retrieval-based models. The retrieval component of Atlas RAG is in charge of locating relevant data or evidence within a substantial data set or knowledge base. To do this, it employs a method known as "dense retrieval" to comb through the information. The query and document embedding representations in the dataset serve as a basis for this retrieval. Upon receiving a query, which can be either a question or a prompt, the retrieval model searches and identifies the most relevant passages or documents that could assist in generating a suitable answer. Once relevant documents are retrieved, the generative component of Atlas RAG comes into the picture which is a large language model called T5. The generative model also known as the reader takes the input query and the retrieved documents as inputs and generates a response. The presence of retrieved documents helps the model to produce more accurate, relevant, and informed responses. It's like the model has access to a curated set of information specifically for the given query. The best thing about the Atlas is that it has a lot of information already in its language model parameters because it is built on a pre-trained language model. Hence, Atlas can produce impressively good results with very few examples or even from instructions. Learning how to solve the Q&A challenge simply takes some fine-tuning on a few samples. Furthermore, Atlas generates more factual responses due to its retrieval component, in contrast to solely parametric approaches based on large language models. To put it briefly, Atlas is a combination of the best retrieval and generative models that generate answers that are more accurate and contextually relevant. It only needs some fine-tuning and a few-shot learning strategies to function well for our question-answering tasks, including natural questions, multiple-choice questions as well and fact-checking.

Describe the experiments you performed, your results, and how they compare to the results achieved in the original Atlas paper.

We ran experiments to fine-tune the Atlas model for the Multiple-Choice Question Answering task. The models were finetuned and evaluated using the MCQ datasets contributed by 4 teams. Accuracy and Debiased Accuracy were the metrics we tracked for performance evaluation of the different experiments. The Atlas paper has described the results of similar MCQ experiments on Massively Multitask Language Understanding (MMLU) datasets.

Base vs Large:

While the pretrained Atlas base model has 220M / 110M reader/retriever parameters, the large model has 770M / 110M parameters. In all our experiments, the large model performed better than the base model in terms of accuracy, almost consistently. This could be corroborated by the results in the Atlas paper where it performed increasingly better with increasing model size.

Without fine-tuning on our datasets, both models gave an accuracy of zero while the large model performed 50% better over the base model in terms of debiased accuracy. When fine-tuned on 10 examples, the difference between biased and debiased accuracy was more pronounced for the large model. When fine-tuned with 100 training examples, the large model outperformed the base model by just 6 points (See Figure 1). Both models showed an improvement in terms of accuracy as we increased the number of training epochs (See Figure 2).

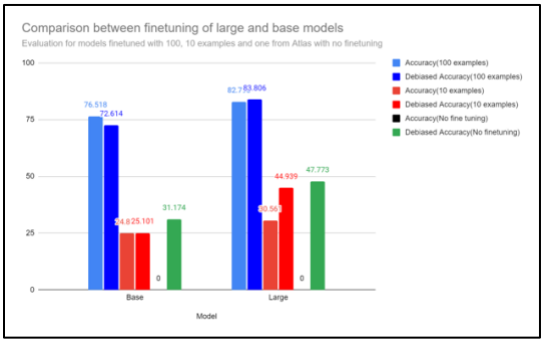

Figure 1: Training Examples vs Accuracy for Base & Large Models

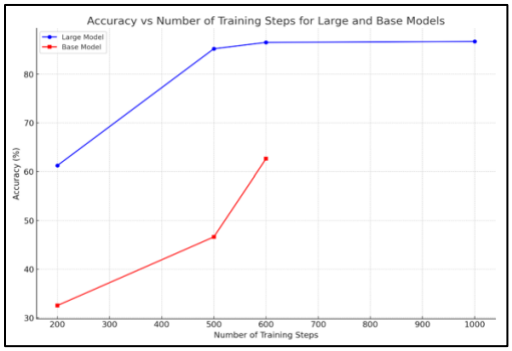

Figure 2: Epochs vs Accuracy for Base & Large

Number of examples used for fine-tuning:

Atlas is presented as a model capable of strong few-shot learning. In our experimentation, we fine-tuned it across varying numbers of training examples (0,5,10,20,30,50, 75, 100, 150, 225, 300) over 1000 epochs (See Figure 3 and Figure 4). The accuracy exhibited improvement with an increasing number of training examples before reaching a plateau. Interestingly, a subsequent increase in training examples led to a decline in accuracy, suggesting potential overfitting.

Debiasing is employed to mitigate potential biases that a model may develop toward specific answer letters, important in few-shot learning settings. As the number of training examples was increased, the difference/gap between accuracy and debiased accuracy decreased and at fine tuning with 100 examples, the difference was insignificant. This observation aligns with the trends illustrated in

the Atlas paper wherein as the amount of training data increases, the incremental benefits of debiasing become less significant. We also noticed that the number of training epochs also interplay with these observations and finding the balance between the number of training examples and the number of epochs is crucial in these few-shot settings which is being described in the next section.

Figure 3: Number examples vs Accuracy I

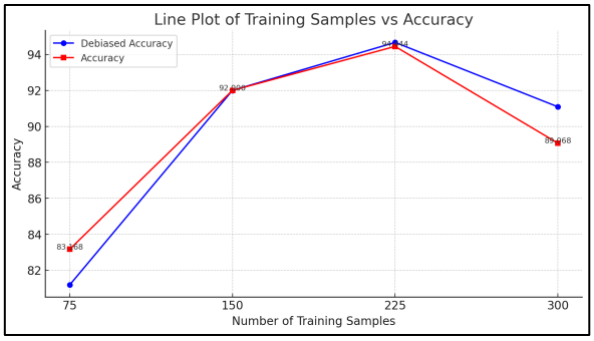

Figure 4: Number examples vs Accuracy II

Number of training epochs:

Training epochs refer to the number of times a machine learning model processes and learns from the entire training dataset. During each epoch, the model goes through the entire dataset once, updating its weights and biases based on the training examples. With a higher number of train steps, the model seemed to overfit the training data, especially in few-shot settings with less number of training examples.

Indeed, the large model demonstrated better results with the smallest number of train steps set at 128, outperforming scenarios with 256, 512, 1024, and 2056 steps for 10 training examples. This implies that a shorter training duration proved to be more effective when dealing with a smaller number of examples. This becomes more evident when plotting the accuracy on varying numbers of train steps at two different sample sizes. The accuracy decreased as we increased the number of epochs for training on 10 examples. Whereas it improved for the model trained on 100 examples (See Figure 5). When selecting hyperparameters for the MMLU experiments, the Atlas paper opted for a smaller number of train steps in smaller few-shot settings which aligns with our observations.

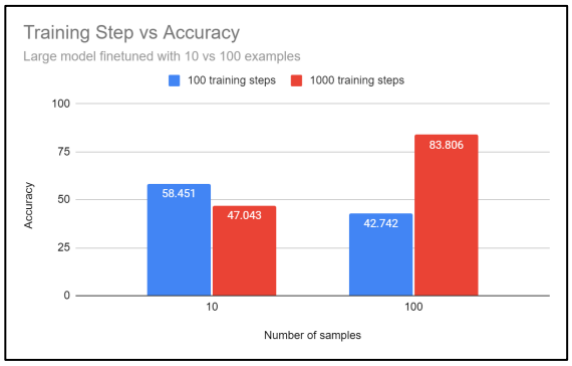

Figure 5: Number of epochs vs Accuracy

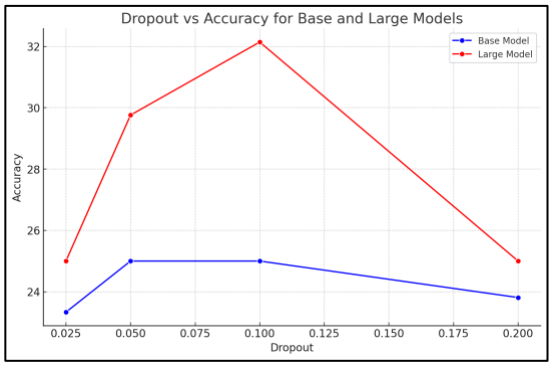

Figure 6: Dropout vs Accuracy

Dropout:

Dropout is a regularization technique used in neural networks to prevent overfitting which forces the network to learn more robust features that perform well with a variety of random subsets of the other neurons by randomly dropping neurons. We tried with other dropout rates, such as 0.025, 0.05, 0.1, and 0.2, and discovered that 0.1 was the most optimal value for the large model (See figure 6). We conducted this experiment while adjusting several parameters, including the number of training samples, epochs, and model type (large or base), and we discovered that the outcomes were consistent.

Learning Rate:

Learning rate is a hyperparameter that determines the step size at each iteration while moving toward a minimum of a loss function. When fine-tuning a model such as Atlas RAG, the learning rate plays a crucial role in determining how the model adjusts its understanding of the relationship between the generative and retrieval components. An optimal learning rate has the potential to improve the model's accuracy, stability, and early convergence. After testing out a number of different learning rates, including 1e-5, 2e-5, 2e-5, 4e-5, 5e-5, and 4e-4, we found that the 4e-5 learning rate was the most effective for both base and large models (See Figure 7). It so happens that the Atlas training script's default settings use the same value.

Weight Decay:

Weight decay is a form of regularization that adds a penalty term to the loss function to discourage the learning of a more complex or flexible model. In the context of the Atlas RAG model, weight decay works by shrinking the weights of the neural network parameters towards zero by a small amount each time the weights are updated during training. Appropriate value of weight decay can help in controlling overfitting, simplifying models as well as improving the generalization. We experimented with the various values of the weight decay such as 0.01, 0.05, 0.1 and 0.2 and found out the value of weight decay 0.05 to be the most optimal one for the large model (See Figure 8).

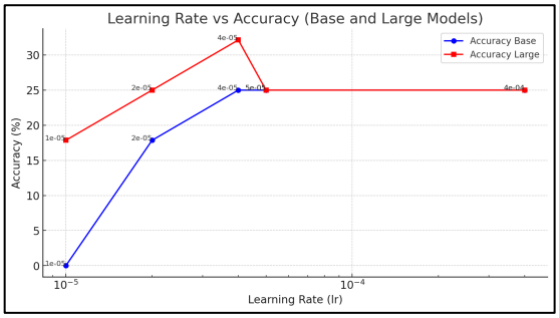

Figure 7: Learning Rate vs Accuracy

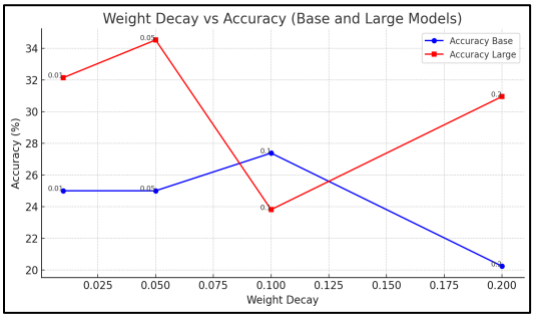

Figure 8: Weight Decay vs Accuracy

Comparison with Atlas Paper

The highest accuracy marked in Atlas paper for a full/transfer learning task is 56.3%. For MMLU tasks, Atlas has achieved up to 74.4% accuracy with a model with 11B parameters. With the help of various fine-tuning techniques, we were able to achieve an accuracy of 92% and an evaluation loss of 0.195 with our most optimal model which was trained on 150 training samples.

Insights

What Works Well:

In training Atlas models, setting the learning rate to 4e-5, a standard in the Atlas program, has proven highly effective. This learning rate, coupled with a weight decay of 0.05, strikes an optimal balance by preventing overcomplexity in the model while enhancing its capacity to generalize to new data. A dropout rate of 0.1 has been found to be ideal, aiding the model in developing a robust and flexible learning approach without over-reliance on specific features. Notably, the larger model variant demonstrated superior accuracy compared to the base model, underscoring the potential of larger models in handling complex tasks more effectively. Furthermore, increasing the training sample size significantly improved accuracy for both large and base models. The strategy of over-retrieval with reranking, enabled by the "retrieve_with_rerank" flag, also yielded positive results. Importantly, selecting 'all' for answer order permutations in the "multiple_choice_train_permutations" parameter enhanced the model's proficiency in answering multiple-choice questions. This approach effectively eliminated any bias towards particular answer orders, thereby contributing to an overall improvement in accuracy.

What doesn't work well:

In evaluating the model's performance, several approaches revealed mixed outcomes. Utilizing various gold score modes such as EMDR, PPMEAN/PDIST, and loop, did not significantly improve the accuracy rate for both base and large models. Additionally, the strategy of over-retrieval with reranking, activated by the "retrieve_with_rerank" flag, resulted in increased computational demands and longer processing times. A slight enhancement in accuracy was noted when the temperature gold score was adjusted from 0.001 to 0.1. However, when the 'single' option was selected in the "multiple_choice_train_permutations" parameter, the model's effectiveness in handling varying answer orders in multiple-choice questions did not improve. This lack of flexibility led to a higher incidence of errors, as the model developed a bias towards a specific order of answers.

Challenges

In our project, we faced a few big challenges. First, we found errors in the multiple-choice questions (MCQ) dataset we started with. Another big problem was that we didn't have access to high-performance computers (HPC) at all times, which made our work much harder, especially for complex tasks. One of the toughest parts was training the larger models, as it required a lot of time, adding more complexity to the project.