CMPE 259 – Group 7: Project Report on ATLAS

Atlas working overview

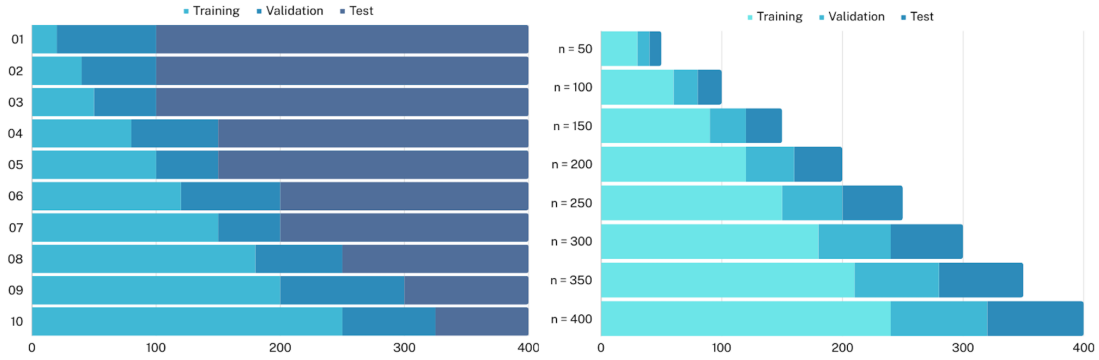
Atlas integrates a retrieval system into the language model, allowing it to pull in relevant information from external texts to answer questions. This approach helps Atlas retrieve relevant information from external texts to answer the questions. This is very useful in scenarios like our tasks(MCQA, FEVER, ODQA), where answers rely on external knowledge not included within the model's pre-trained data.

Project Description

For this project, we created a Multiple Question Answering dataset for Chapters 8, 9, 19 of the book" An introduction to statistical learning with applications in Python". We carefully created 100 MCQs from the chapters, and their respective passages into a JSONL format. The objective is to fine tune the Atlas model, to perform these experiments with varying training sizes using different training sizes to analyze how it performs. We have performed these experiments using the 400 sample passages and their respective MCQs, to see how each version of the model performed.

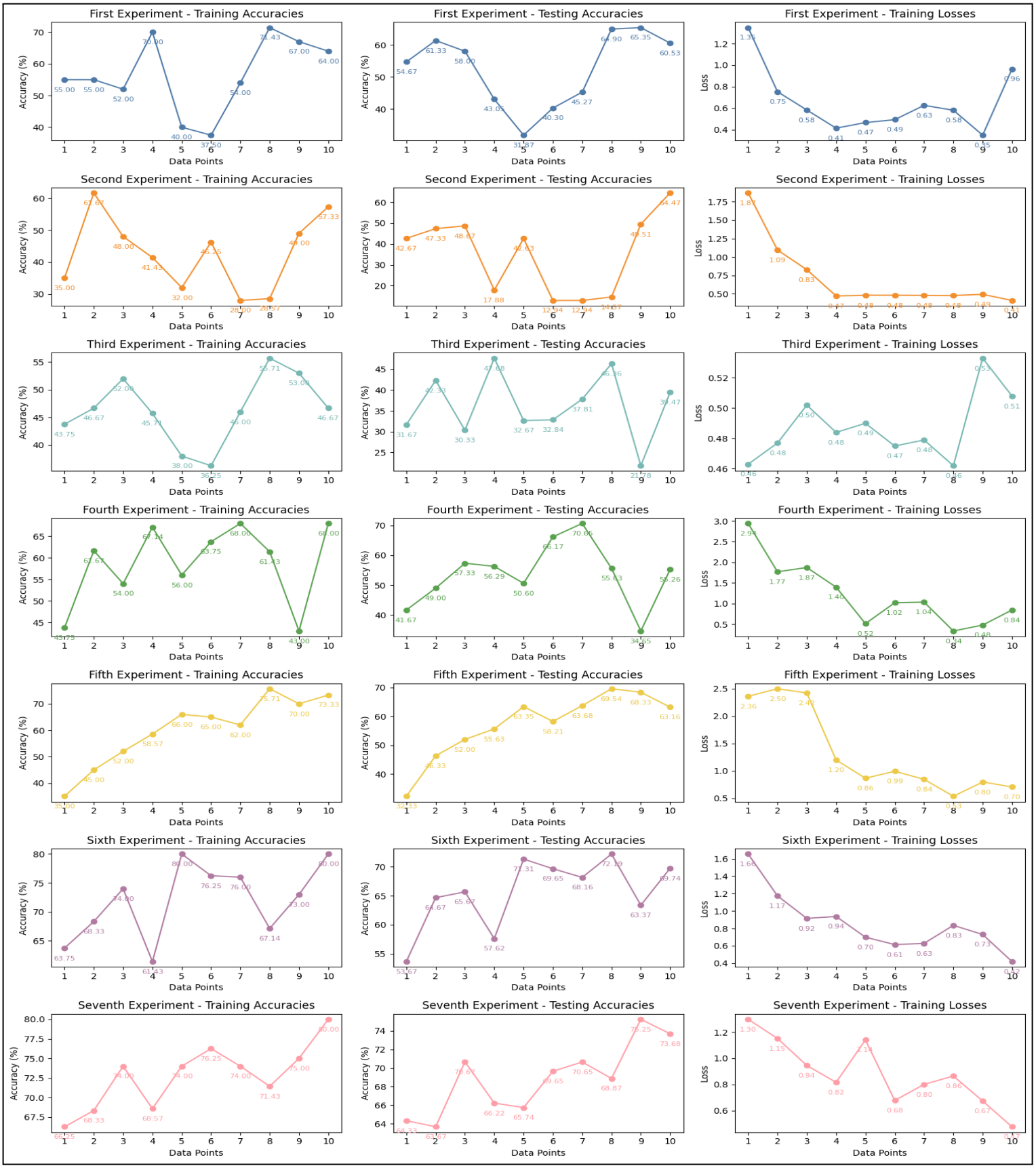
Experiments Performed

To perform the experiments we took two different splits for each individual.
(i) One where the entire set of 400 samples was used in different proportions; And
(ii) one where the samples used were used in the traditional 60:20:20 ratio.



For the first set of experiments, we experimented with 10 different training sizes and performed 7 sets of experiments on them, for a total of 70 experiments. Prior to that, we performed multiple test runs to determine the appropriate "text_max_length", "STEP_SIZE", and the most accurate "precision" parameter; to not change them for the rest of the experiments. The default text_max_length is 128, and precision is fp32 for all our experiments. We determined that the text length>128, and STEP_SIZE > 1000 had no much impact, except that they significantly slowed down the training.

Experiment Parameters	Best Training Accuracy	Best Testing Accuracy	Lowest Losses(train,test)
500 steps, n_context = 40, lr = 5e-5, lr_retriever = 1e-5, lr scheduler = linear (Model : Base)	67.143	65.347	(0.349, 0.315)
500 steps, Lr = 0.0002, lr_retriever = 1e-5, n_context = 10, n_context_ret = 20, lr scheduler = linear (Model: Base)	56.667	64.474	(0.410, 0.365)
500 steps, lr = 0.0002, lr_retriever = 1e-5, n_context = 10, retriever_context = 20, lr scheduler = cosine (Model: Base)	33.000	46.358	(0.462, 0.426)
1000 steps, lr = 5e-5, lr_retriever = 5e-5, n_context = 20, retriever_context = 30, lr scheduler = cosine (Model: Base)	66.333	70.647	(0.337, 0.346)
1000 steps, lr = 5e-5, lr_retriever= 5e-5, n_context = 20, retriever_context = 30, lr scheduler = linear (Model: Base)	74.286	69.536	(0.534, 0.751)
1000 steps, lr = 1e-4, lr_retriever = 1e-5, context = 20, retriever_context = 30, lr scheduler = cosine (Model: Large)	80.500	72.185	(0.419, 0.609)
1000 steps,lr = 1e-4, lr_retriever = 1e-5, context = 20, retriever_context = 30, lr scheduler = linear (Model: Large)	79.000	74.247	(0.475, 0.560)



The first half of this project was trained in datasets with varied training/dev/test splits. The second half is trained in datasets with varied sample sizes, divided into training, development, and testing sets, with a 60%, 20%, and 20% split, respectively. Below we present a detailed analysis of data collected from thirteen experiments conducted to evaluate the performance of an ATLAS Multiple Choice QA model across different parameters. The experiments vary in learning rate (lr), number of contexts (n_context), per GPU batch size, dropout rates, weight decay, warmup steps, total steps, evaluation frequency (eval_freq), and model size. The model's performance was assessed based on debiased accuracy, accuracy, and evaluation loss (eval_loss).

Accuracy Scores												
Hyperparameters	exp2	exp3	exp4	exp5	exp6	exp7	exp8	exp9	exp10	exp11	exp12	exp13
lr	0.00005	0.00005	0.00005	0.00005	0.00005	0.00005	0.00005	0.00005	0.00005	0.00005	0.00004	0.00005
n_context	10	10	10	10	12	14	8	8	8	8	8	8
per_gpu_batch_size	2	4	3	4	4	4	4	4	4	4	4	4
dropout	0.1	0.1	0.1	0.15	0.1	0.1	0.1	0.1	0.1	0.1	0.1	0.1
weight_decay	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
warmup_steps	50	50	50	50	50	50	50	50	50	70	50	50
total_steps	500	500	500	500	500	500	500	550	450	500	500	500
eval_freq	20	30	30	30	30	30	30	30	30	30	30	30
size	base	base	base	base	base	base	base	base	base	base	large	large
n=50	50.0	70.0	80.0				50.0				60.0	50.0
n=100	20.0	65.0	45.0				70.0				80.0	85.0
n=150	30.0	53.3	73.3				66.7				83.3	73.3
n=200	45.0	55.0	42.5				57.5				67.5	77.5
n=250	62.0	68.0	56.0				66.0				70.0	66.0
n=300	65.0	70.0	36.7				71.7	63.3	60.0	60.0	81.7	75.0
n=350	48.6	45.7	62.9	35.7	61.4	58.6	64.3				80.0	71.4
n=400	56.8	44.4	63.0				50.6				67.9	69.1

In assessing the models performance, larger datasets notably improved accuracy, especially in Experiments 03, 04, 08, 12, and 13, with Experiment 02 being an exception. Larger per GPU batch sizes (Experiment 03 vs 02) enhanced performance, indicating better generalization. Varying learning rates and contexts showed inconsistent results; increasing context numbers didn't always lead to better outcomes. A notable finding was in Experiment 05, where a higher dropout rate of 0.15 resulted in reduced performance compared to experiments with a dropout rate of 0.1, indicating a possible optimal range for dropout rates. A higher dropout rate negatively impacted learning. The parameters of weight decay and warm-up steps were kept constant throughout the experiments, making it difficult to assess their individual impacts. Similarly, variations in total steps and evaluation frequency seemed to have a minor influence on overall results. Experiment 12 excelled, particularly with larger datasets. Its combination of a slightly lower learning rate, large model size, and specific settings (lr=4e-5, n_context=8, batch size=4, dropout=0.1, weight decay=0.01, warmup=50 steps, total 500 steps, evaluation every 30 steps) led to the highest debiased and overall accuracy, suggesting the importance of model size and learning rate for complex pattern recognition and effective training convergence.

Comparison with ATLAS

ATLAS did not explicitly provide detailed results for the MCQA. Our highest performing model was having a training accuracy of ~80% and testing accuracy of ~75%. Atlas highlighted its efficiency in tasks like NaturalQuestions, TriviaQA, and FEVER and to match or exceed performance in tasks like MMLU. There's an instance where Atlas achieved a 42% accuracy on Natural Questions using only 64 training samples. However, Atlas has been significantly updated since the paper came out, and is now expected to have much better performance. As the peak performance mentioned in Atlas was around 75-85%, as shown with some tasks in the paper, our fine-tuned version performed on par with the Atlas paper.

Insights

For similar tasks, it is recommended to use a large model size coupled with a learning rate around 4e-5 to 5e-5. While increasing the dataset size generally improves performance, special attention should be paid to the batch size and dropout rate to optimize learning. A per GPU batch size of 4 and a dropout rate around 0.1 seem to be effective settings. This analysis underscores the importance of carefully tuning machine learning parameters to optimize performance. Experiment 12 offers a robust model configuration for tasks similar in nature to those tested. However, it is crucial to consider the specificities of the task at hand, as different datasets and objectives might require adjustments to these parameters.

Challenges Faced During the Experiments

During our experiments, we faced common challenges such as Out of Memory Error, resolved by reducing batch size to fit the model within GPU memory limits. The CUDA Devices Busy or Unavailable Error was tackled by ensuring the proper shutdown of previous sessions and running experiments during less busy periods to free up GPU resources. Learning rate estimation, a trial-and-error process, was streamlined through a range test, selecting the rate that most effectively reduced loss. Addressing these challenges systematically improved the efficiency and effectiveness of model training and deployment.