AI First Final Project: Course Assistance LLM

Romil Patel

Georgia Institute of Technology, rpatel778@gatech.edu

Abstract

Composed of two parts, using the GPT2 LLM, we first explore how hyperparameter choices affect model performance, then fine-tune models on course materials covering topics like PCA and Linear Regression all while using different tokenization strategies – No Tokenization, BPE, BBPE, and GPT-2. Models are tested against 50 tests about AI/ML topic and the ground truth using Levenshtein distance and ROUGE-L F1 score.

1 INTRODUCTION

As artificial intelligence continues to transform education, large language models (LLMs) offer support within the classroom such as personalization and automation of learning experiences. In this project, we explore the development of a Course Assistance LLM for Georgia Tech's new special topics course, AI First. The primary goal of this project is to experiment with different hyperparameters and tokenization strategies, then use the results to build a few models again using constant hyperparameters, but varying tokenization strategies.

The first part of this project, Model Analysis, investigates how architectural and training hyperparameters – such as learning rate, dropout, embedding size, number of layers and heads – effect model performance. Through a series of tests, we evaluate each configuration using quantitative metrics (Levenshtein distance and ROUGE-L F1 score). The choice of these metrics comes down to what our model is doing and what we want to achieve and how best to measure this. Levenshtein distance is a quantitative measure of simply measuring how close the approximated answer is to the ground truth, being the answer to the tests. A large distance means the model and ground truth are significantly different, whereas a small distance means the model and ground truth are closely related. ROUGE-L F1 score measures the longest common subsequence between a generated response and the ground truth. The score takes precision (how much of the generated output overlaps with the ground truth) and recall (how much of the reference is captured in the generated output) and uses both scores to generate a balanced metric that properly assesses how our model is performing. Its ranges differ from 0 being extremely different from the ground truth, and a score closer to 1 would mean the model's content matches very well with the ground truth.

The second part of this project, we tailor our model using parameters we find suitable during the first part. We use data from class lectures, labs, recitations and practice problems as .txt files for the model to use to train (the default data is a Shakespeare play (Shakespeare.txt)). This customization allows a user to learn more about a topic such as PCA or linear regression and as well illustrates the flexibility of LLMs for a specific fine-tuning.

Ultimately, this project is a culmination of everything taught during the AI First curriculum and use to develop a model using careful experimentation and thoughtful design in designing a topic-aligned LLM.

1.1 Part I: Parameter Selection

Our first step for Part I is to identify our situation and parameters we wish to experiment with. In this situation, our problem becomes one of classification as the model needs to classify what the needs of the student are and to respond accordingly with accurate information. With that in mind, our hyperparameters for experimentation are as follows: learning rate, optimizers, batch size, number of heads and layers, dropout, block size, embedding size, and for BPE, BBPE and GPT-2 the vocab size.

The parameters in particular affect either the architecture of the model or model learning in general. Learning rate effects the weight updating process, using the loss function, it tells how much to update the weights based on the loss function. Optimizers control how the model's weights are updated during training, so a good optimizer such as Adam improves the way weights are updated as poor tuning can lead to underfitting or overfitting. Batch size is the number of training samples processed before weights are updated, so a larger batch size gives more accurate gradients (used for weight updates), results in faster computation and a smaller batch size adds more noise when updating which can help prevent overfitting and helps with generalization.

The number of layers affects the architecture of the model where it adds extra depth to the model allowing for higher capacity and abstraction, but this also is slower to train and results in overfitting, but a model with too few layers results in a model unable to learn complex patterns and underfit. Similarly, more heads allow for more processes to run in parallel, so it can better capture relationships between tokens, but also can over-complicate a process and result in overfitting. Dropout disables neurons during training so a model doesn't overfit, fixing a high variance problem, but if you disable too many you can prevent learning and get a model that is extremely underfit. The last few parameters, block size is the number of tokens a model can see at a time, this can allow a model to learn more, but results in memory and computation costs, embedding size is the dimensions of the token embeddings which allows for a more richer representation of the token and improves performance, but if its too large, you can have overfitting and slower training. The last parameter is only applicable to models that utilize tokenization strategies such as BPE, BBPE and GPT-2, vocab size is the number of distinct tokens the model can use. A smaller vocab size uses shorter sequences, but has better generalization, helps with overfitting, or a larger vocab can include more tokens, but also increases overfitting risk.

1.2 Part I: Hyperparameter and Tokenization Separation and Results

After determining which hyperparameters to experiment with, the next step was to identify the best organization method, so thus it seemed efficient to repeat each hyperparameter experiment under different tokenization strategies. The first was to run all experiments under no tokenization strategy and to see what the hyperparameter does to a raw encode-decode methodology of parsing inputs to basic tokens. The next tokenization strategy, BPE, works with individual characters as tokens and merges together frequent tokens with new words/sub words. The next tokenization strategy, like BPE is Byte-Level BPE, which instead of using each character as a token, uses raw bytes of characters, which helps improve recognition of new characters such as emojis or special symbols making this strategy extremely robust. The last strategy is based off BBPE, which is what GPT-2 uses where raw bytes of characters are also used as tokens but has a fixed vocabulary of 50,257 tokens to use also making it very robust.

After conducting our experiments, we can see various trends and make conclusions based on Rouge Scores. Of the many experiments, the few that showed interesting results are learning rate, number of heads and layers. As we can see in Figure 1, already having a tokenization strategy vastly improves the ROUGE score vs not having a tokenization strategy. In addition, we can see that using a GPT-2 tokenizer with a learning rate of .0001 produces the greatest ROUGE scores. This is probable because using 20 iterations, the model properly adjusts weights according to the loss function. Another trend that is visible is that when you increase the learning rate, the ROUGE score decreases and can be contributed to the learning rate overstepping gradients and not updating properly, so it oscillates.

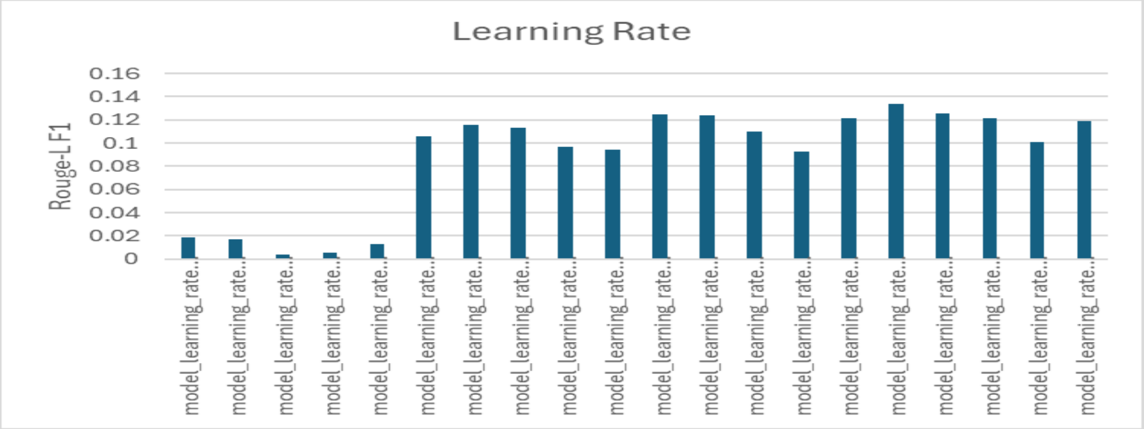


Figure 1. Learning rates of all models of different tokenization strategies, every 5 models is a different tokenization strategy, going from no token, BPE, BBPE, GPT-2.

The next hyperparameter that showed interesting results is the number of layers. A clear we can see is that when you increase the number of layers, it allowed for better understanding of the training data and seemed to improve results. Thus, we can conclude that as we increase the number of layers we can expect to see better results until a certain number of layers, or else we get high amount of overfitting, and the model becomes less generalized. However, using a BBPE strategy and 4 layers also had great scores, which can be concluded that 4 layers may be the most effective setting for the number of layers for the strategy.

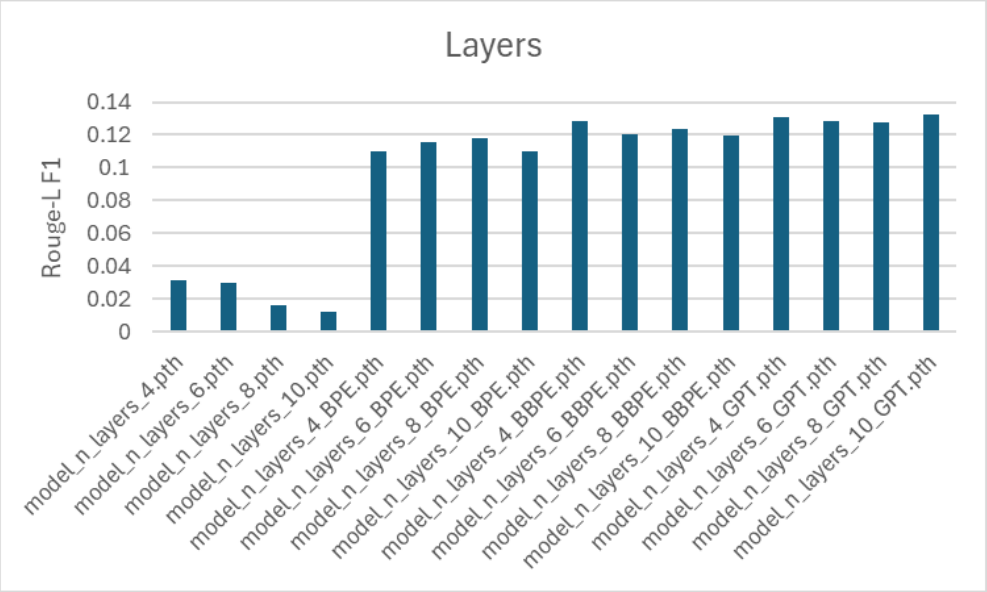


FIGURE 2. Different models of different tokenization strategies and different number of layers.

The last hyperparameter that showed interesting results is the number of heads. Overall, when increasing heads, we can see that it improved performance, and the most effective strategy for a high number of heads would be to use GPT-2, but another thing to note is that we need to have number of embedding be able to be divided by the number of heads, otherwise errors occur, so for 10 heads, embedding size was set to 390, but that didn't have a significant effect on the output as it was a minute detail that didn't effect results.

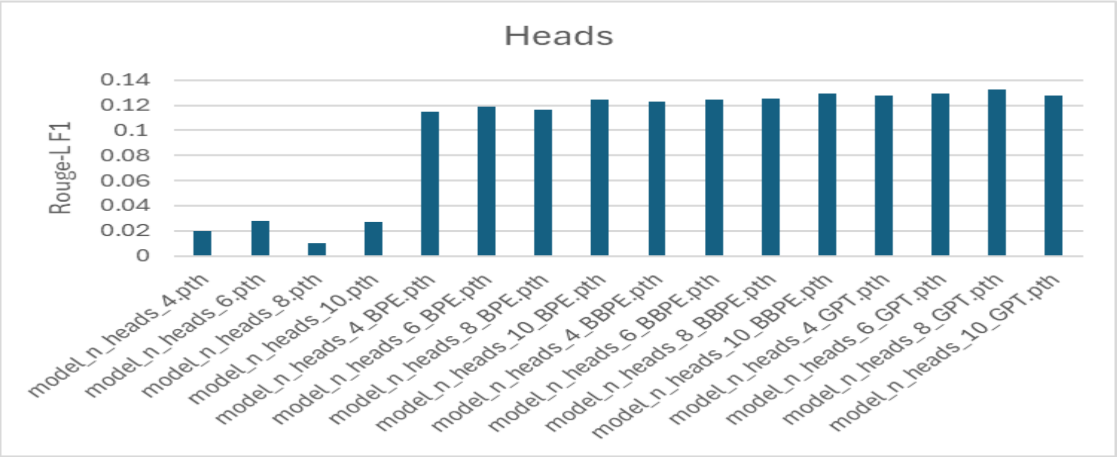


Figure 3. Number of heads for all strategies and performance scores for all.

2 PART II: STUDENT CUSTOMIZATION

2.1 Part II: Hyperparameter selection

After testing the hyperparameters in the first part of the project, we can see that the most efficient parameters will be to have learning parameter at 0.001 as we saw the best score for that particular learning rate, and due to hardware limitations and time constraints, we will use the next best number for layers and heads, being 8 and we will hold embedding size, dropout and the other parameters as defaulted as the average text file length will be around 2000-4000 characters, so these parameters should hold. The only thing that needed to be clarified is that for BBPE, the token size will be held at 8000 as that is what it was trained on and should be sufficient to train using class data. The model will be trained using all class data such as lectures, recitations, labs and practice problems to provide better results during testing.

2.2 Part II: Results and Errors

Overall, results showed promise. As seen in Figure 4, models trained with no tokenizer and with the GPT-2 tokenizer showed progress, but BPE and BBPE had errors that either were extremely far from the ground truth or were unable to be tested due to either corrupt file or some parameters were altered during training and during testing, those parameters were not being referenced correctly. For the model using BPE, the score was at a flat 0, which can be due to the decoding algorithm not being written properly and due to some bugs, the decoded characters were being represented as characters that would not be in the alphabet or very specific characters.

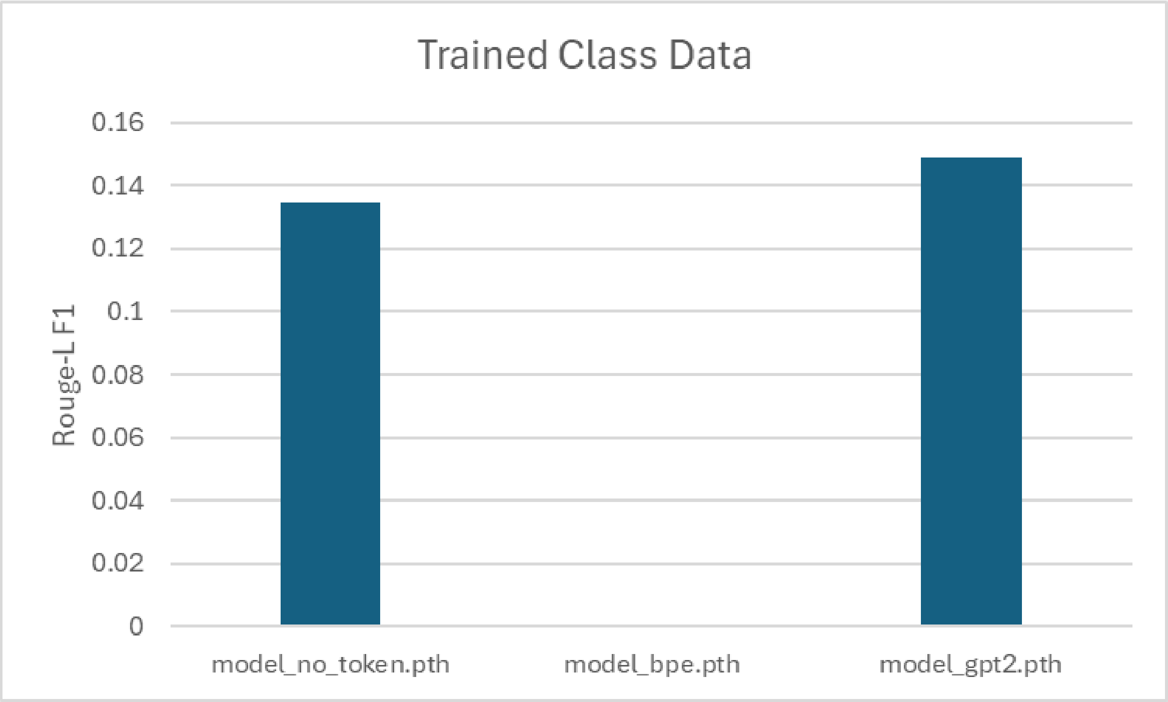


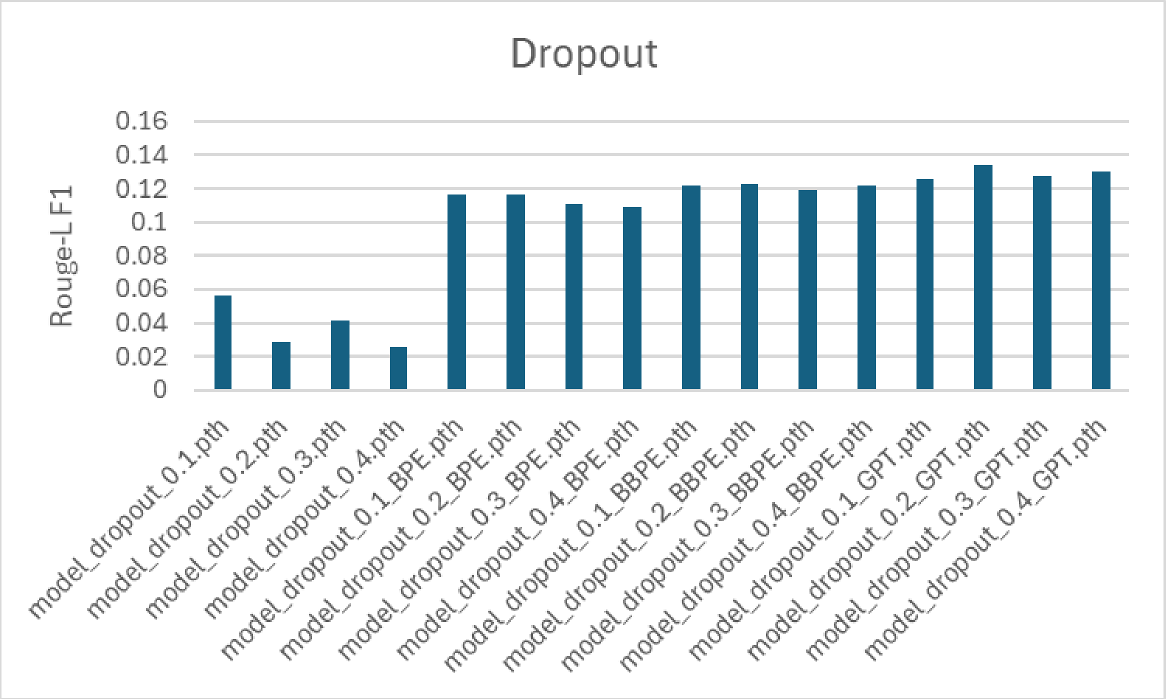
Figure 4. Models trained off class data and different tokenization strategies, there is error which it addressed.

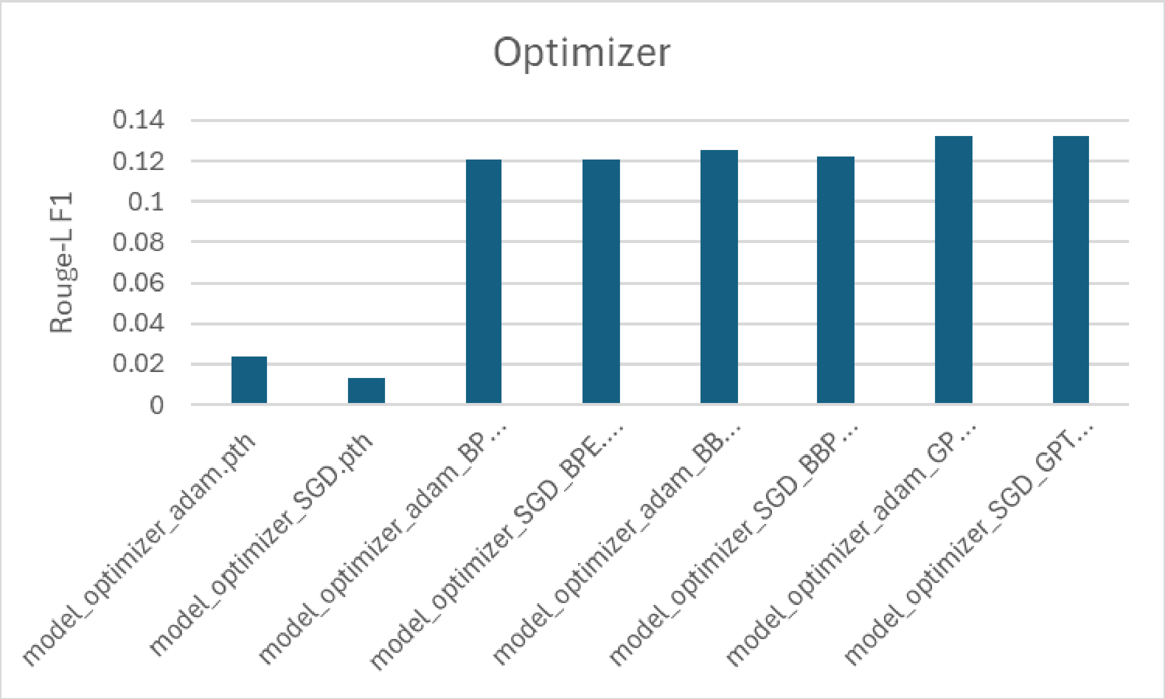
3 CONCLUSION

This project demonstrates the impact of hyperparameter tuning and tokenization strategies when training a large language model (LLM) using the GPT-2 architecture. Through systemic experimentation with different model configurations, we observed their effects on performance such as ROUGE-L F1 and Levenshtein Distance. We found that models using the GPT-2 tokenization strategy performed better than other strategies, while improper decoding and errors with BPE and BBPE led to admissible results. Ultimately, with proper finetuning and proper decoding and training of certain models, we think that results may improve significantly and shows how to effectively train a model using different parameters and content tailored for a particular situation.

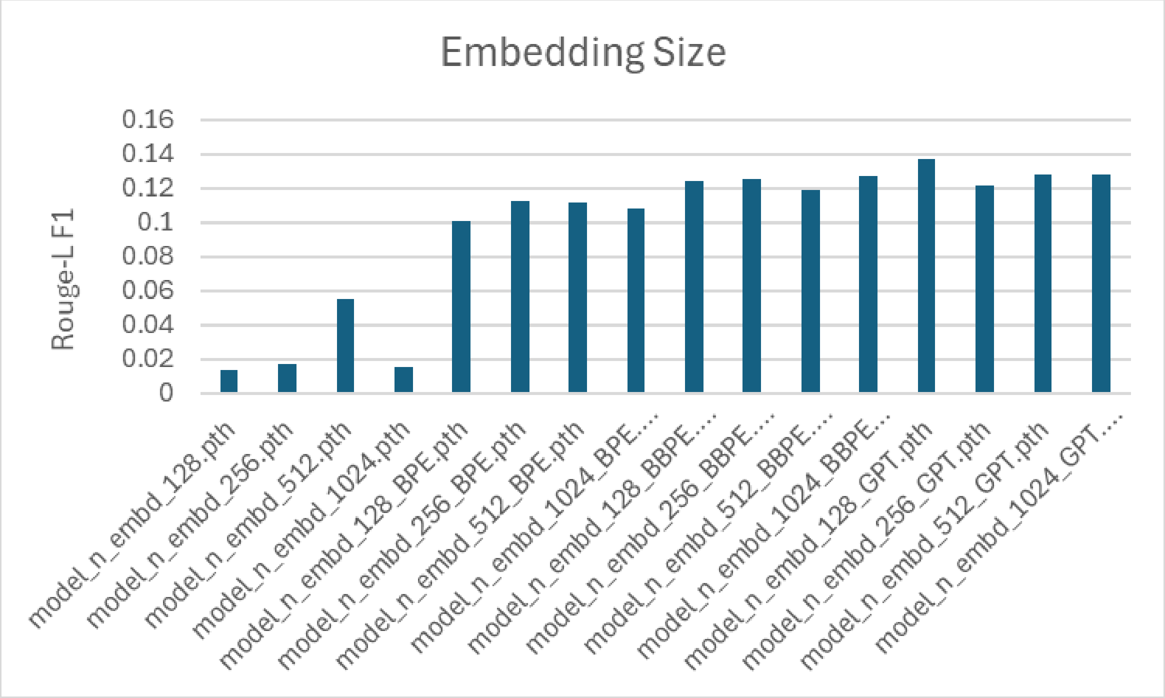
A APPENDICES

The rest of the results from the different tests.

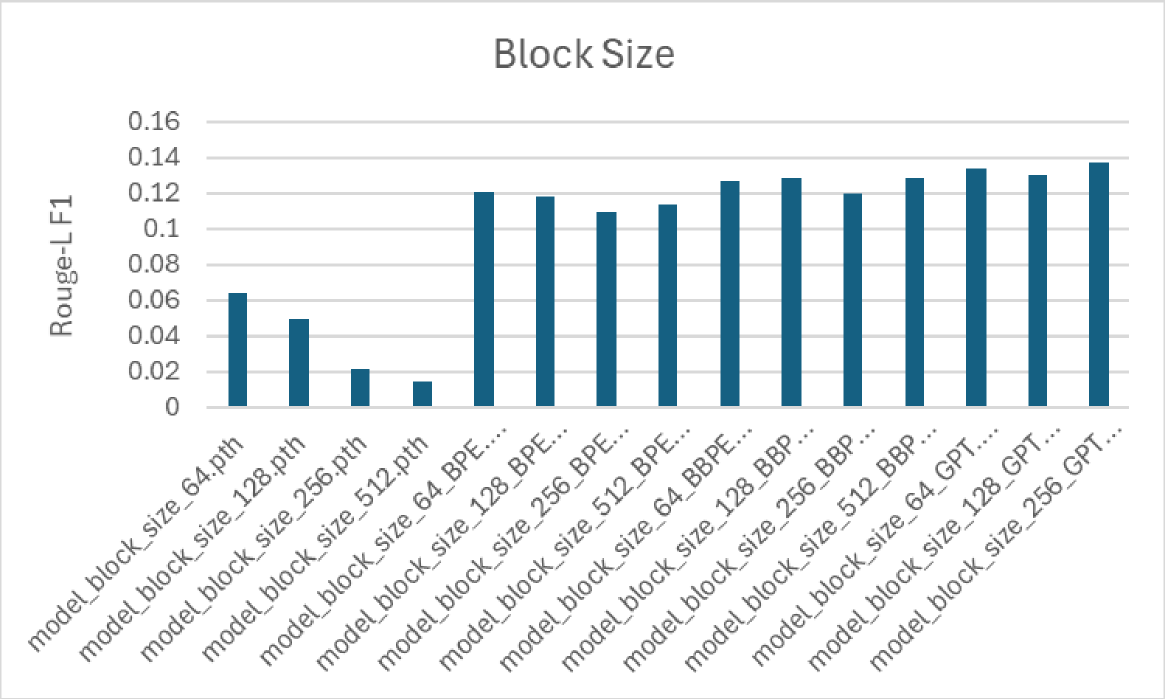


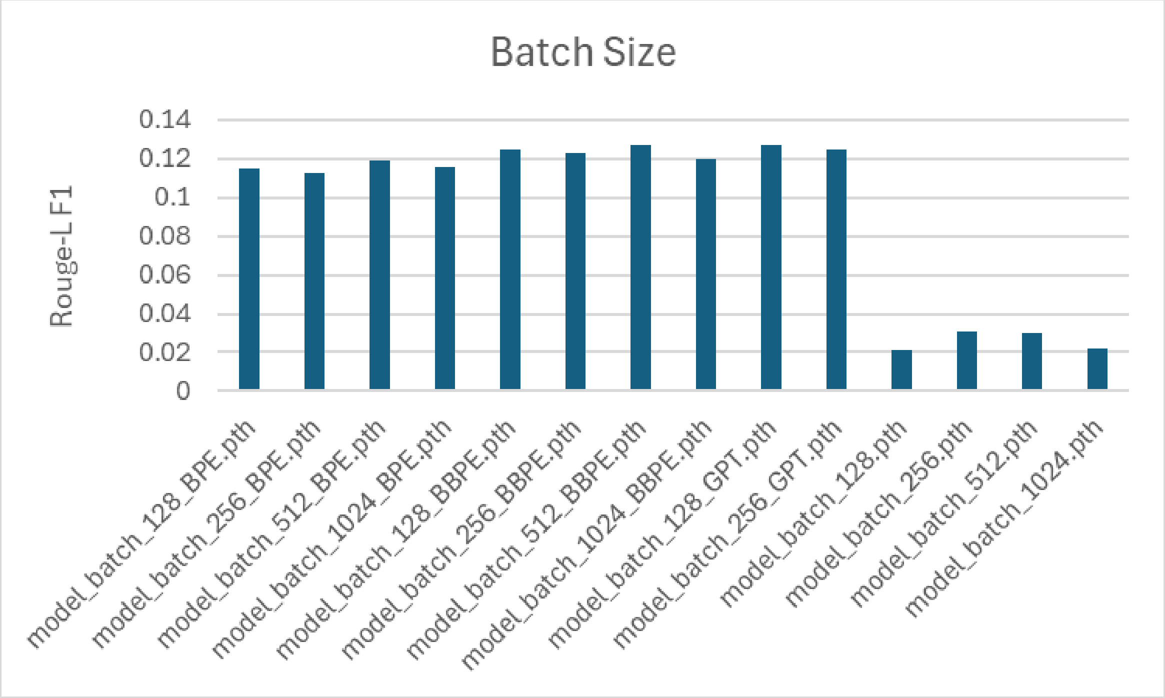


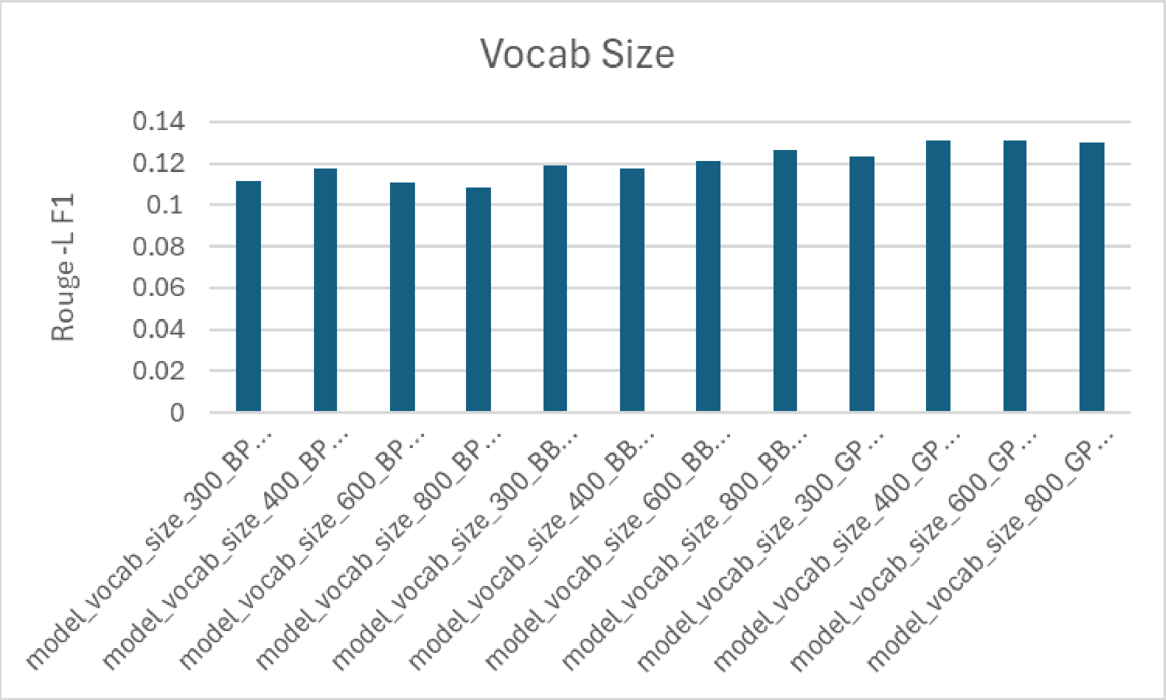
Optimizer



Embedding Size


Block Size
Rouge-L F1
0.16
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
model_block_size_64.pth
model_block_size_128.pth
model_block_size_256.pth
model_block_size_512.pth
model_block_size_64_BPE....
model_block_size_128_BPE...
model_block_size_256_BPE...
model_block_size_512_BPE...
model_block_size_64_BBPE...
model_block_size_128_BBP...
model_block_size_256_BBP...
model_block_size_512_BBP...
model_block_size_64_GPT....
model_block_size_128_GPT...
model_block_size_256_GPT...


Batch Size
Rouge-L F1
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
model_batch_128_BPE.pth
model_batch_256_BPE.pth
model_batch_512_BPE.pth
model_batch_1024_BPE.pth
model_batch_128_BBPE.pth
model_batch_256_BBPE.pth
model_batch_512_BBPE.pth
model_batch_1024_BBPE.pth
model_batch_128_GPT.pth
model_batch_256_GPT.pth
model_batch_128.pth
model_batch_256.pth
model_batch_512.pth
model_batch_1024.pth


Vocab Size
Rouge -L F1
0.14
0.12
0.1
0.08
0.06
0.04
0.02
0
model_vocab_size_300_BP…
model_vocab_size_400_BP…
model_vocab_size_600_BP…
model_vocab_size_800_BP…
model_vocab_size_300_BB…
model_vocab_size_400_BB…
model_vocab_size_600_BB…
model_vocab_size_800_BB…
model_vocab_size_300_GP…
model_vocab_size_400_GP…
model_vocab_size_600_GP…
model_vocab_size_800_GP…