Scaling Generative Pre-training for User Ad Activity Sequences

Sharad Chitlangia, Krishna Reddy Kesari & Rajat Agarwal

{chitshar, kkesari, agrajat}@amazon.com

amazon ads

AdKDD 2023

User Activity Sequence Models in Advertising

- **Why model user activity sequences?**
  - Information about temporal user behavioral patterns
  - Fine grained information as against feature aggregates
  - Lower reliance on feature engineering

- **Applications**
  - Customized ad response prediction
  - Personalization
  - Ad Fraud / Bot detection

- **Training Technique**
  - Deep sequence model as the backbone – LSTM, Transformer etc.
  - Ad response prediction uses supervised models
    - Labels are abundant
  - Fraud detection – large scale ground truth is unavailable
    - Unsupervised techniques / pre-training typically work better

Pre-training using Generative Models

Introduction

- Goal is to learn the input data distribution P(X)
- Pre-train a large neural network on large-scale unlabeled data using a proxy task
- Proxy tasks on sequential data:
  - Masked Token Prediction (BERT)
  - Next Token Prediction (GPT)
- Generates robust task-agnostic embeddings of the input
  - Improves performance on both supervised and unsupervised downstream tasks

Generative Pre-training

Formulation for User Ad Activity Sequences

- **Training Objective:** Predict the next event in users' activity sequence **(X_1… X_n)** using a neural network

$$L(S) = \sum_{u} \sum_{i} \log p(X_{i+1} | X_1, .., X_i; \theta)$$

- Each activity event **X_i** is defined using multiple **(k)** features **(F)**. Assuming conditional independence:

$$p(X_{i+1} | X_1, .., X_i) = \prod_{j=1}^{k} p(F_j(i+1) | X_1, .., X_i)$$

- For low cardinality features – compute the exact probability using softmax
- For high cardinality features – approximate the probability using negative sampling and contrastive loss
- Use any off-the-shelf deep sequence model for encoding the sequence
  - Transformers scale better

Scaling Properties

What do we mean by scaling of Generative Pre-training?

- How does model performance improve with increasing
  - Model size (Trainable non-embedding parameters)
  - Data size
  - Compute (GPU-hours or FLOPS)

- Performance is measured by test loss on the pre-training objective
- Also evaluated on downstream task performance

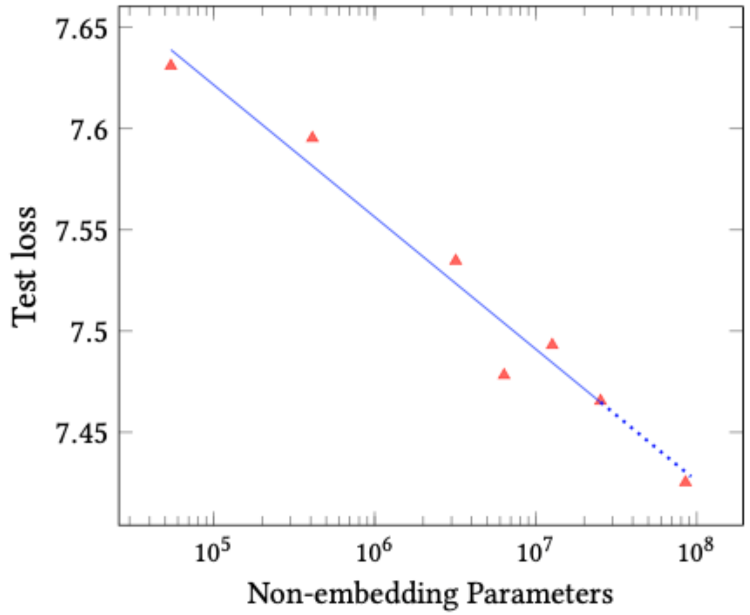
Scaling Properties – Model Size

How does increasing number of trainable parameters in the model impact test loss?

- Considers only non-embedding parameters
- Trains on all available data (number of users in the ad program)

- Test loss follows a power law relationship with parameter count
- Can extrapolate to predict the performance of a larger model

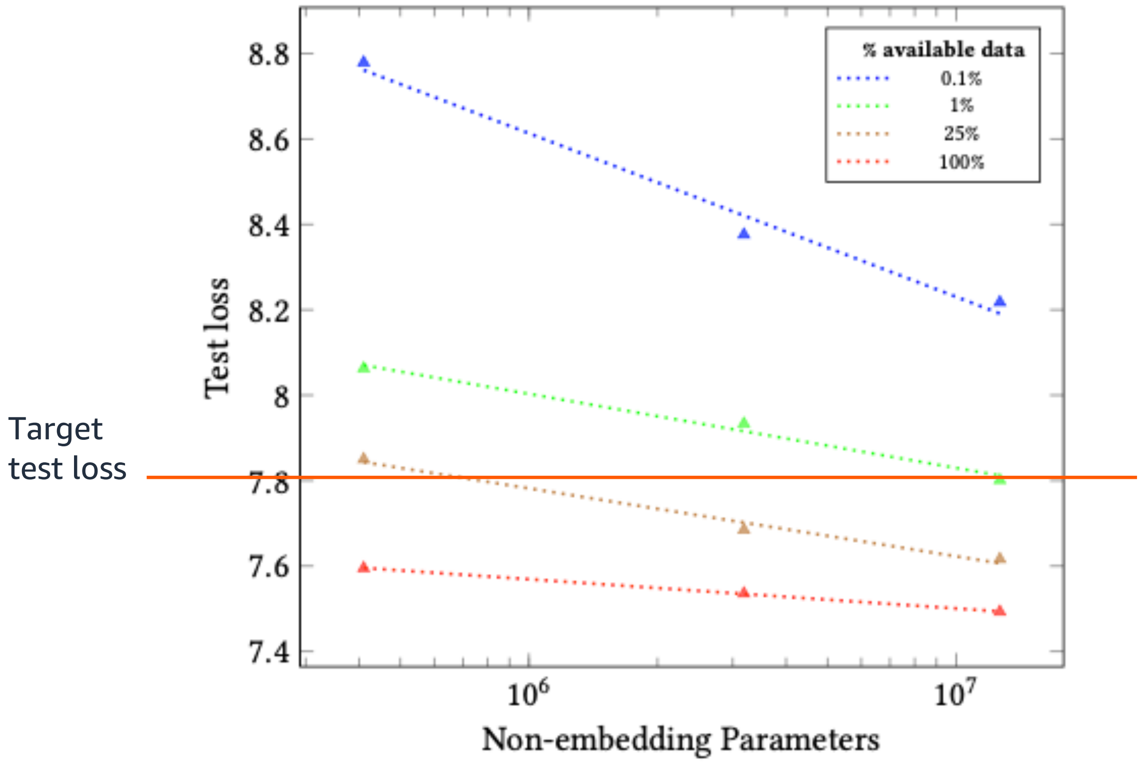
- Will eventually saturate at some point due to bounded dataset size



Scaling Properties – Data Size

How does dataset size impact test loss for different model sizes?

- Larger models are more data efficient:
  - Need lesser data to achieve a target test loss

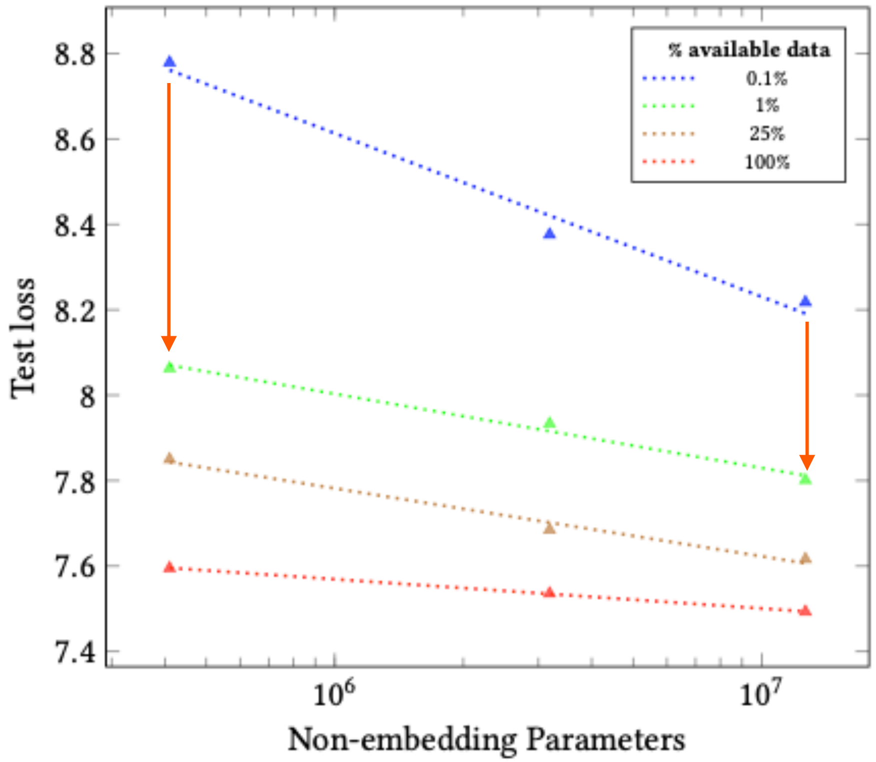


Scaling Properties – Data Size

How does dataset size impact test loss for different model sizes?

- Larger models are more data efficient:
  - Need lesser data to achieve a target test loss

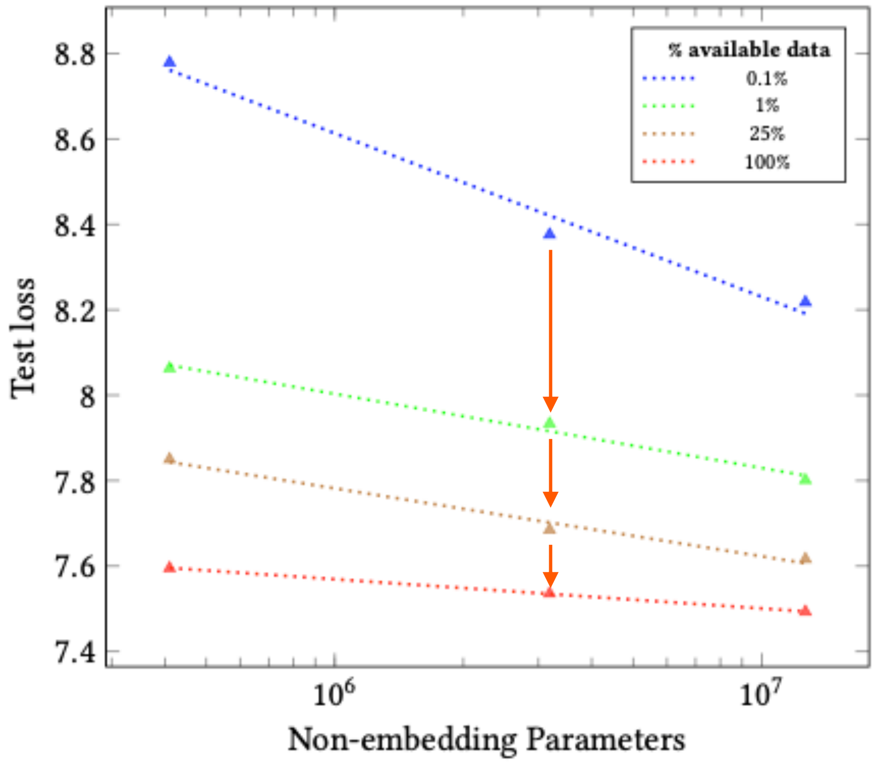
- Smaller models benefit more from increasing dataset than larger models



Scaling Properties – Data Size

How does dataset size impact test loss for different model sizes?

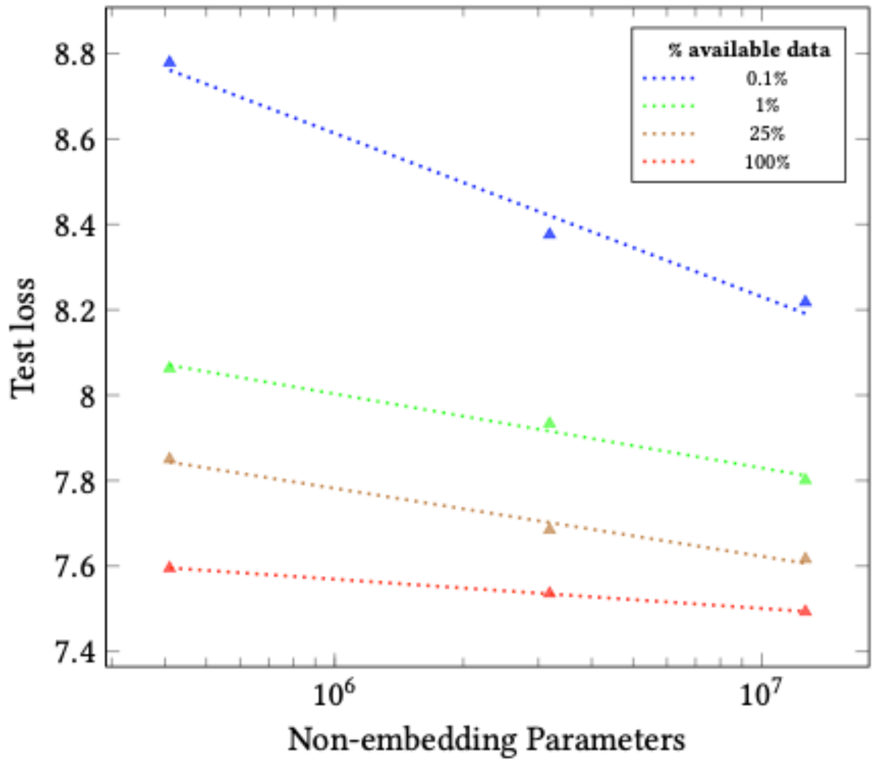
- Larger models are more data efficient:
  - Need lesser data to achieve a target test loss
- Smaller models benefit more from increasing dataset than larger models
- For a fixed model size – increasing data size gives diminishing returns



Scaling Properties – Data Size

How does dataset size impact test loss for different model sizes?

- Larger models are more data efficient:
  - Need lesser data to achieve a target test loss
- Smaller models benefit more from increasing dataset than larger models
- For a fixed model size – increasing data size gives diminishing returns
- **Takeaway**: Increase model size and data size in tandem to maximize performance
- **In practice:** Dataset is bounded. Train the largest model given your compute budget.



Scaling Properties – Compute

How to achieve the lowest test loss given fixed compute budget (GPU-hours)

For fixed model size, higher gradient steps, lower batch size

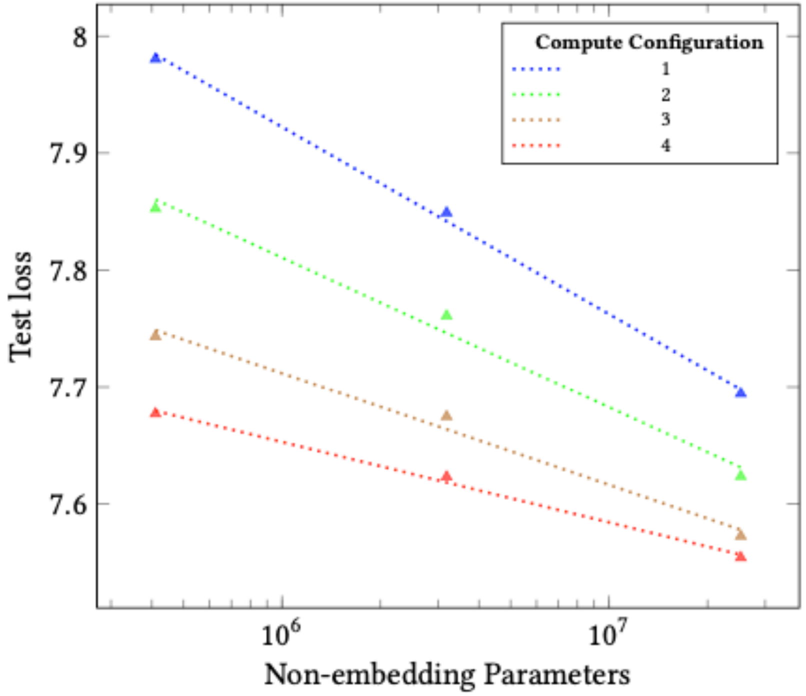
Configuration	GPUs	Time (minutes)	Learning rate
1	64	15	0.0008
2	32	30	0.0004
3	16	60	0.0002
4	8	120	0.0001

- Fix GPU hours (monetary cost):
  - Tradeoff number of serial gradient update steps vs model size vs batch size
  - GPU utilization is maximized for every model by changing batch size

- Larger models are more sample efficient in achieving a fixed test loss
  - For each compute configuration (fixed number of GPUs and wall clock time) larger models process lesser data

- For a fixed model size, increasing gradient steps is more efficient than increasing batch size

- **Theme:** Increase model size, decrease batch size, increase serial steps

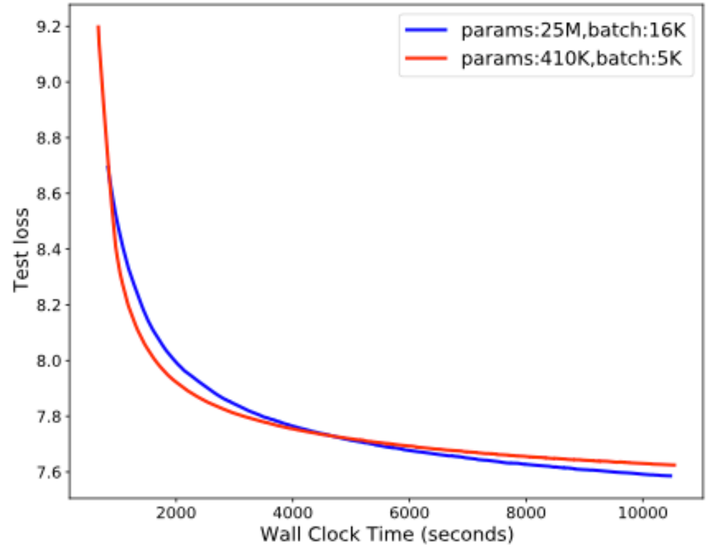


Scaling Properties – Compute

How much can we decrease batch size and increase model size, given fixed compute budget? (GPU-hours)

Parameters	GPUs	Time (minutes)	Batch size (per GPU)	Test loss
201,649,152	8	120	48	7.732
85,136,640	8	120	128	7.601
25,273,856	8	120	256	7.554
410,240	8	120	960	7.677

- Lower batch size leads to noisier gradients
  - Larger model with **batch size < B_{min}** will underperform smaller model
- Scale model size keeping with batch size B_{min}
- But increasing model size decreases serial steps
  - Need minimum wall clock time (W_{min}) before larger model outperforms all smaller models
- **Takeaway:** Increase model size till the extent both B_{min} and W_{min} fit your compute budget



Parameters	GPUs	Time (minutes)	Batch size (global)	Test loss
25,273,856	32	30	8192	7.623
410,240	8	120	7680	7.677

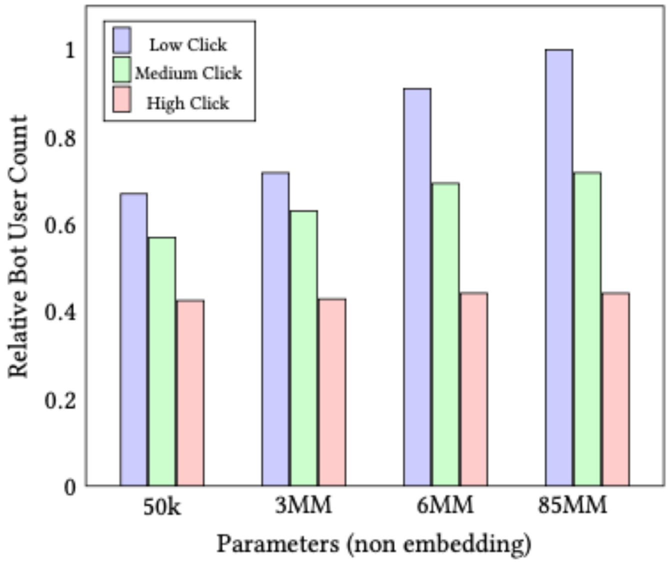
Downstream Task Evaluation

Does improvement in pre-training test loss translate to better downstream task performance?

- Evaluated on two downstream tasks in advertising
  - **Supervised –** Conversion Prediction
  - **Unsupervised -** Bot detection
- Lower test loss (larger model) leads to improved downstream performance on both tasks
- Larger models were significantly better in detecting bots with shorter sequence lengths
  - Points to better representation learning on limited data with increasing model size

Table 5: Lift over downstream task performance relative to 54K model

Params	IVR @ fixed FPR	pConversion AUC
3,186,432	+1.63 %	+0.02 %
6,359,552	+3.44 %	+2.51 %
85,136,640	+4.09 %	+3.57 %



Learnings

Putting it all together

- Pre-training test loss follows power law scaling with increasing model size
- Larger models are more data and compute efficient
- Downstream task performance doesn't follow power law with model size
  - Cannot exactly predict final downstream performance but it will likely improve
- Dataset size is bounded in ads, but it's not a key constraint at current model sizes
  - Even with 1% data, the test loss did not saturate
- Compute efficient training requires scaling model size at a fixed minimum batch size
  - But give it a minimum wall clock time to see the benefits
  - Efficient training need not converge – aim for the lowest test loss given your budget

Thank you!

Questions?

agrajat@amazon.com

amazon ads