Transformers for Failure Prediction in High-Performance Computing System Logs

Halil Ozgur Demir, Allison Austin, Tejas Patil, Amey Gohil
ECS 289L

Motivation

- High-performance computing (HPC) systems are crucial for solving large-scale scientific problems
- High demand for effective monitoring and failure prediction

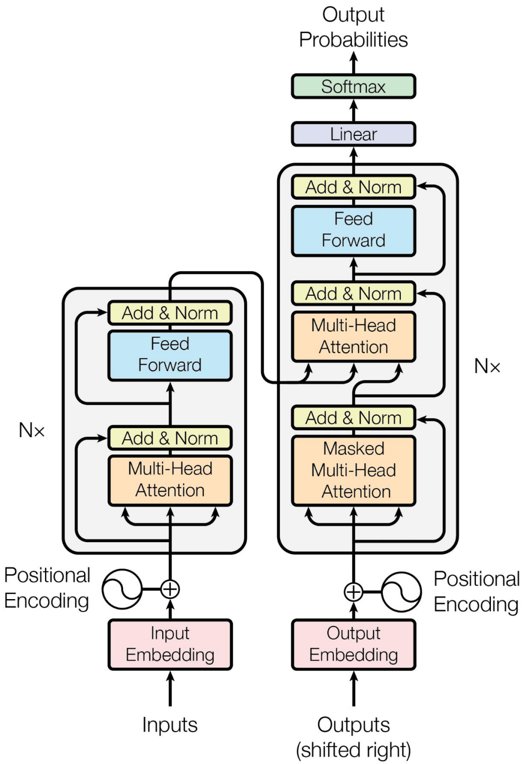
Existing Approaches:

- <u>Rule-Based:</u> Threshold monitoring, scalable but not generalizable
- <u>Time-Series:</u> Detects anomalies but doesn't scale well
- <u>Machine learning:</u> LSTMs and autoencoders, scalable and generalizable but need human intervention

Transformer models have been relatively unexplored as a method of log prediction

LLMs

- Designed to handle sequential data more effectively than RNNs and LSTMs
- Self-attention allows for handling long-term relations in data.
- Suitable for HPC log-based prediction
  - Analyzing log messages requires determining relationships between events, components, and jobs.
  - Failure interpretation
- LLMs used for this project
  - minGPT
  - logBERT



Data Preprocessing

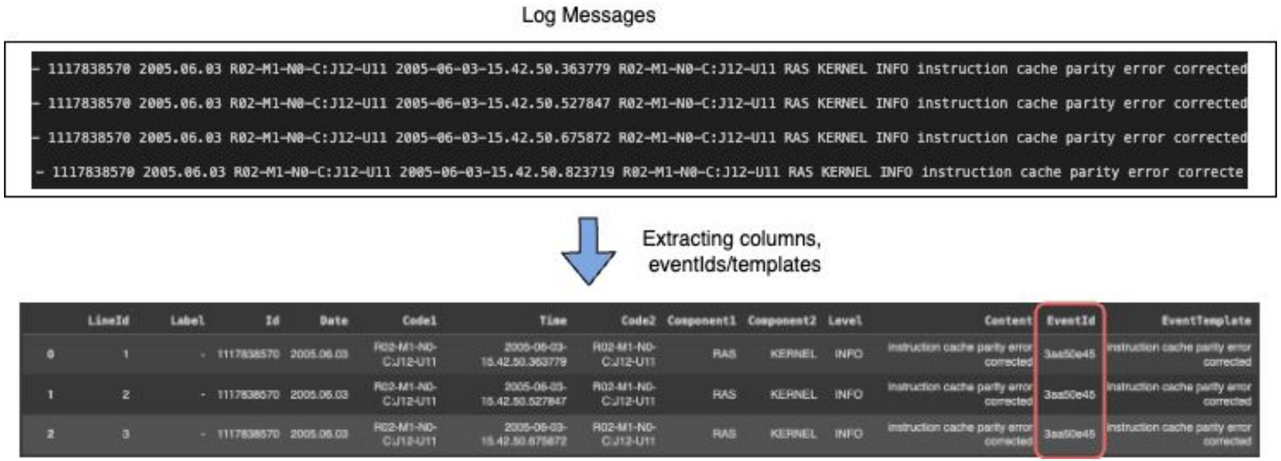
Datasets

System 20 HPC cluster

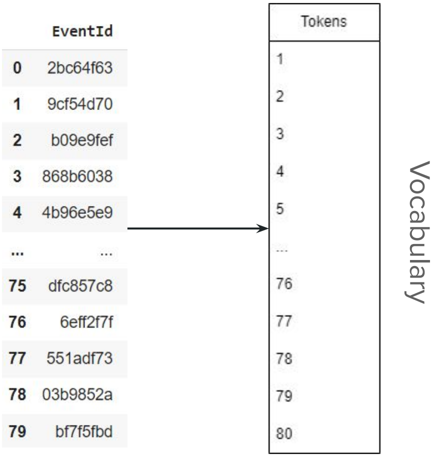
- 432,260 log messages, low frequency
- 80 unique log events, 21 failure events

Blue Gene/L (BGL) supercomputer

- 4,747,963 log messages, 348,460 anomalous, high frequency



Tokenizing Error IDs

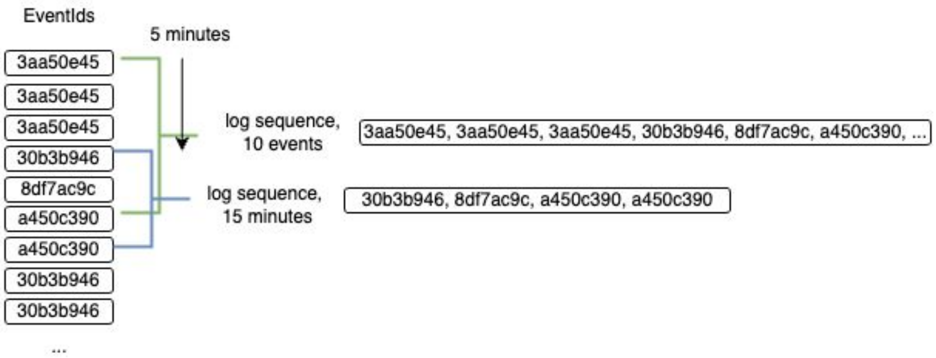


Data Preprocessing

Generating Log Sequences ➜

BGL

System 20



- *Input representation*
  - Sum of log key embedding and position embedding
  - Fed into the transformer encoder

$$\{x^{\mathrm{dist}}, x^1, x^2, x^3, \ldots, x^t, \ldots, x^{T-1}, x^T\}$$

$$x_j^t = e_j^t + t_j^t$$

LLMs

$$\mathrm{PE}(pos, 2i) = \sin\left(\frac{pos}{10000^{\frac{2i}{d\_model}}}\right)$$

$$\mathrm{PE}(pos, 2i+1) = \cos\left(\frac{pos}{10000^{\frac{2i}{d\_model}}}\right)$$

PE

minGPT

Simplified version of GPT-2 language model

Training for one token generation using fixed size sequence.

Autoregressive generation of sequences.

Unidirectional attention mechanism.

12 transformer and feedforward layers with embedding size 768.

LogBERT

Applying BERT model on HPC logs for anomaly detection

Captures contextual information for better results in prediction

Bidirectional self-attention mechanism.

Multi-head self-attention with 4 heads and a position-wise feed forward sub-layer of size 256.

Methods

Probability Inference

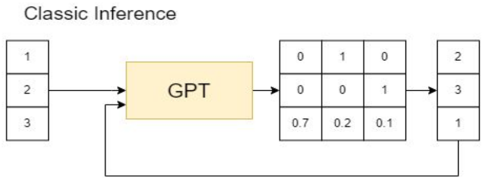
- Uses token probabilities generated by GPT model instead of directly using the token itself.

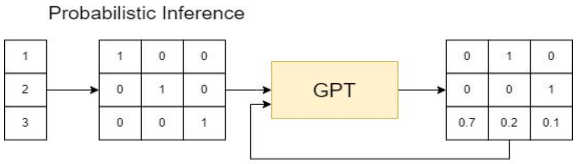
Additive Attention Mechanism

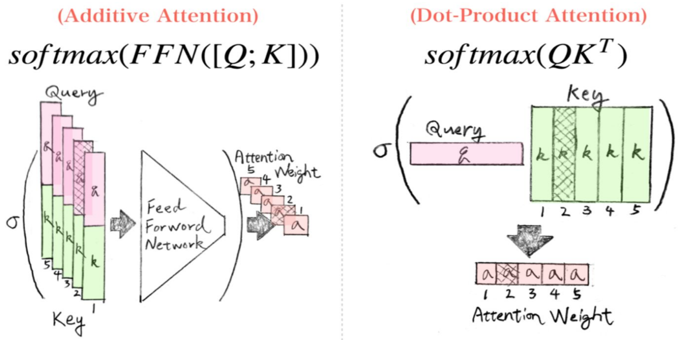
- Uses single-layer feedforward neural network
- Hyperbolic tangent for nonlinearity, scaling factor v

Hierarchical Attention Mechanism

- Processes sequences at event-level and sequence-level







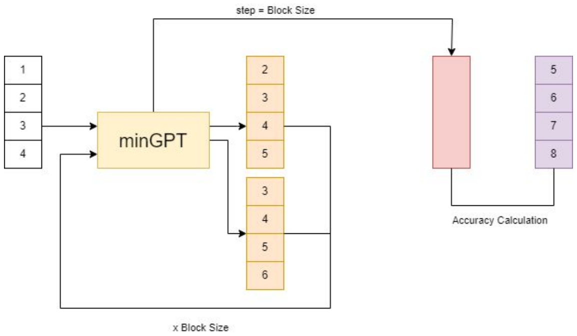
$$f_{\text{att}}(\mathbf{h}_i, \mathbf{s}_j) = \mathbf{v}_a{}^{\top} \tanh(\mathbf{W}_1 \mathbf{h}_i + \mathbf{W}_2 \mathbf{s}_j)$$

Results

BERT was best-performing model across both datasets

Changing attention mechanism only improved models trained on System 20 dataset

- Baseline BERT model performed best on BGL dataset
- BERT + HA model performed best on System 20 dataset



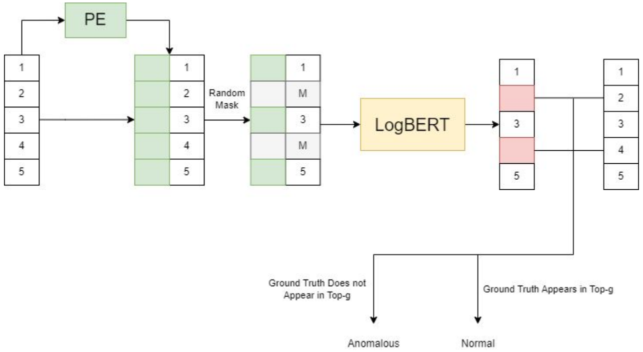


Table 3: Performance of each model on anomalous log sequence prediction task.

	Sys20				BGL			
	Accuracy	Precision	Recall	F-1 score	Accuracy	Precision	Recall	F-1 score
GPT	45.62	98.28	9.86	17.92	83.65	80.61	36.41	50.16
GPT + PI	48.65	95.28	19.90	32.93	83.75	75.28	33.33	46.21
GPT + AA	45.62	100	9.84	17.92	-	-	-	-
BERT	80.12	68.97	1.54	3.02	**95.61**	**94.61**	**83.57**	**88.75**
BERT + AA	78.14	43.81	**31.89**	**36.91**	88.28	72.34	70.27	71.29
BERT + HA	**81.61**	**82.53**	10.47	18.59	93.17	86.30	79.69	82.86

Conclusions

- Significant difference in model performance for GPT vs BERT
  - Techniques that improved performance of GPT resulted in worse performance for BERT
- Probabilistic Inference
  - Increased performance in Sys20, decreased performance in BGL
  - Increase in number of log keys influence probabilistic inference negatively
  - Different techniques on determining probabilities
- When applying techniques to one model, we see different performance changes across datasets
- Log parsing and dataset characteristics has large impact on LLM performance
  - Parameters being unchanged between LLMs and datasets, no parameter tuning