LUMOS: Large User MOdel Series

Foundation models for User Behavior Prediction

Dhruv Nigam

August 21, 2025

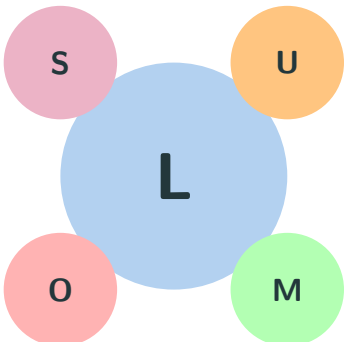
Annual Work Seminar

Presentation Outline

LUMOS: Large User MOdel Series

Origin:

- Spell in Harry Potter that creates light
- Greek for "light"
- Illuminating user behavior



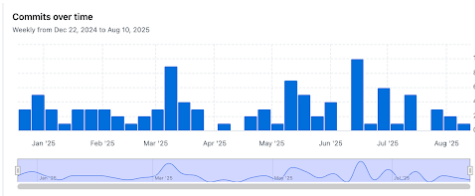
Why Names Matter

Using Deep Learning for forecasting of User attribute time series(UPMA)

Owned by dhruv.nigam • • •
Jun 20, 2023 • 1 min read • 19 people viewed

A Complete Platform

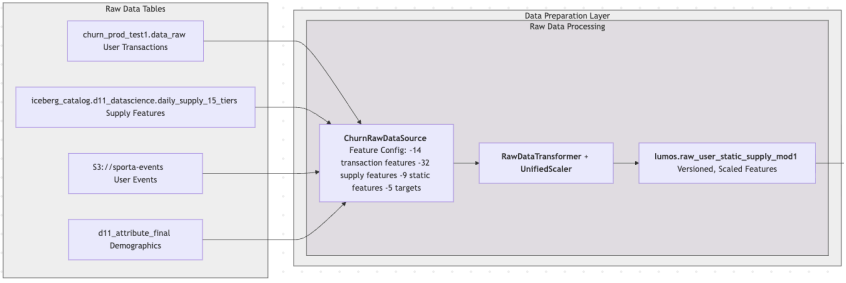
- **17,000+ lines of code** (8+ months development)
- Models: only **200 LOC (1%)**



Multiple Teams Actively Contributing:

- State-of-the-art models for churn and CEA prediction - less than 10 lines of code change to train either
- User and supply embeddings are valuable for communications teams.
- A deposit amount model is currently in development.

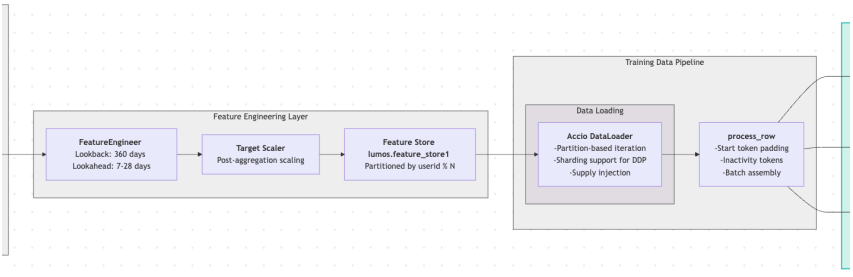
Platform Components: Raw Data Preparation



Software Stack

PySpark for large-scale data processing, **Delta Lake** for versioned data storage, and **MLflow** for tracking data scalers.

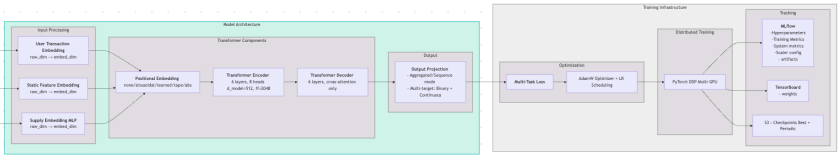
Platform Components: Training Preparation



Software Stack

PySpark for feature engineering and a custom **PyTorch** data loader (**Accio**) for efficient batch creation and DDP support. Currently migrating to **Ray Data**.

Platform Components: Model Training



Software Stack

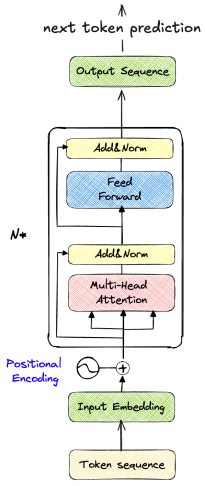
PyTorch DDP for distributed training, **MLflow** for experiment tracking, and **Databricks Jobs** for orchestration. Moving to **Ray Train** for better multi-instance parallelism.

LLM Architecture Foundation

Standard LLM Architecture: Decoder-Only

Major Components:

- Tokenization
- Token Embeddings
- Positional Embeddings
- Self-Attention



LLM Components: 1. Tokenization

- **Process**: Converts raw text into a sequence of discrete tokens.
- **Method**: Typically uses a vocabulary-based mapping with subword units like Byte-Pair Encoding (BPE) or WordPiece.
- **Output**: A sequence of integer IDs representing the tokens.

Mathematical Representation

$$\text{Text} \rightarrow \text{sequence of tokens } T = \{t_1, t_2, \ldots, t_n\}$$

LLM Components: 2. Token Embeddings

- **Purpose**: Transforms one-hot encoded tokens into dense, continuous vector representations.
- **Mechanism**: A learnable embedding table (matrix) maps each token ID to a high-dimensional vector.
- **Benefit**: Captures semantic relationships between tokens (e.g., "king" is to "queen" as "man" is to "woman").

Mathematical Representation

$$E = W_e \cdot t_i$$

where t_i is the one-hot vector for a token and W_e is the embedding matrix.

LLM Components: 3. Positional Embeddings

- **Problem**: The self-attention mechanism is permutation-invariant and lacks a sense of sequence order.
- **Solution**: Inject information about the position of each token in the sequence.
- **Common Method**: Sinusoidal functions of different frequencies are added to the token embeddings.

Mathematical Representation

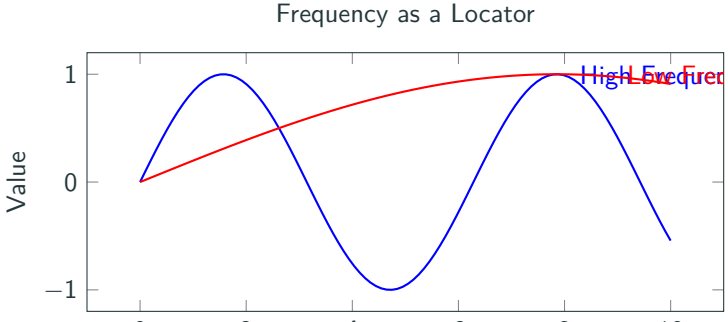
$$PE_{(pos,2i)} = \sin(pos/10000^{2i/d_{\text{model}}})$$

$$PE_{(pos,2i+1)} = \cos(pos/10000^{2i/d_{\text{model}}})$$

Positional Embeddings: The Intuition

- **Relative Positioning**: The dot product between positional embeddings for two positions, pos and $pos + k$, depends only on the relative distance k, not on the absolute position pos. This is due to the properties of sine and cosine addition, allowing the model to learn relative positions.

$$PE_{pos}^{T} \cdot PE_{pos+k} = \sum_{j=0}^{d/2-1} \cos(k/10000^{2j/d})$$



LLM Components: 4. Self-Attention

- **Function**: Allows each token to attend to all other tokens in the sequence to build contextual representations.
- **Core Idea**: Computes a weighted sum of all token values, where the weights are determined by the similarity between the current token (query) and other tokens (keys).
- **Benefit**: Enables long-range dependency modeling and parallel processing.

Scaled Dot-Product Attention

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

Visualizing Self-Attention

Attention Score Calculation:

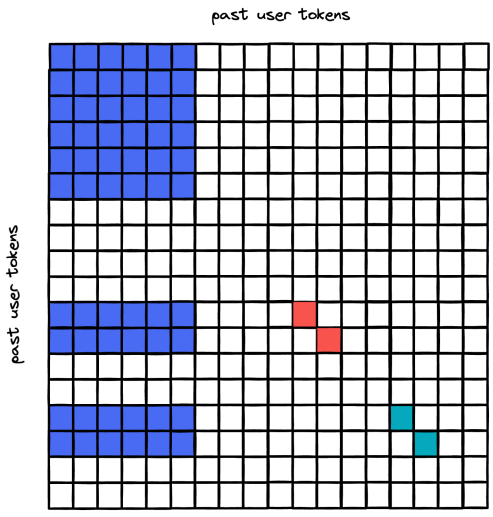
$$\text{score}(q_i, k_j) = q_i \cdot k_j^T$$

Attention Weights (Softmax):

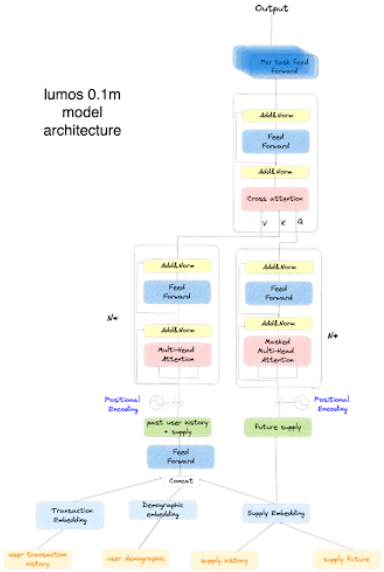
$$\alpha_{ij} = \frac{\exp(\text{score}(q_i, k_j))}{\sum_{l=1}^{N} \exp(\text{score}(q_i, k_l))}$$

Final Output Vector:

$$z_i = \sum_{j=1}^{N} \alpha_{ij} v_j$$



LUMOS High-Level Architecture



LUMOS Architecture Deep Dive

LUMOS Tokenization: Daily-Level Tokens

Key Innovation: Multi-Modal Token Design

User Behavior

13 Transaction Features

- Joins, deposits
- Withdrawals, wins
- Activity counts
- Daily aggregation

Supply Data

32 Supply Features

- Sports calendar
- Count of events
- Time of events(morning vs night)

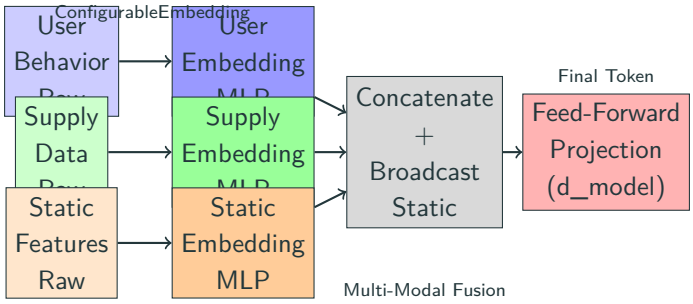
Static Features

9 User Attributes

- Demographics
- Device info
- Registration data
- Age at registration

Token Assembly: Embed $\rightarrow$ Concatenate $\rightarrow$ Feed-Forward $\rightarrow$ Rich Daily Representation

Token Construction Process

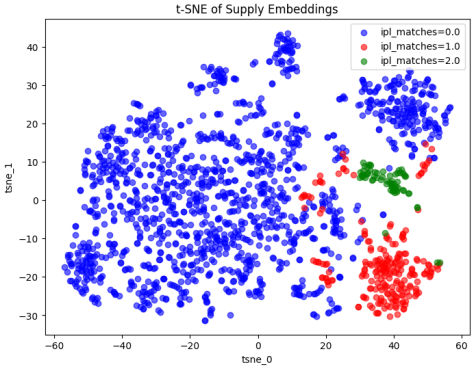


vs. LLMs

LLMs: Discrete text tokens from fixed vocabulary

LUMOS: Continuous numerical vectors from multi-modal daily aggregations

The Power of Specialized Embeddings



Isolating Embedding Pathways Unlocks New Applications:

1. **Static User Embeddings**
   - **Use Case**: New user personalization, acquisition assessment.
   - Stable, demographic-based representation.

2. **Supply Embeddings**
   - **Use Case**: Supply similarity analysis
   - Allows for "what-if" scenarios by embedding hypothetical future supply.

Positional Embeddings: Ablation Study

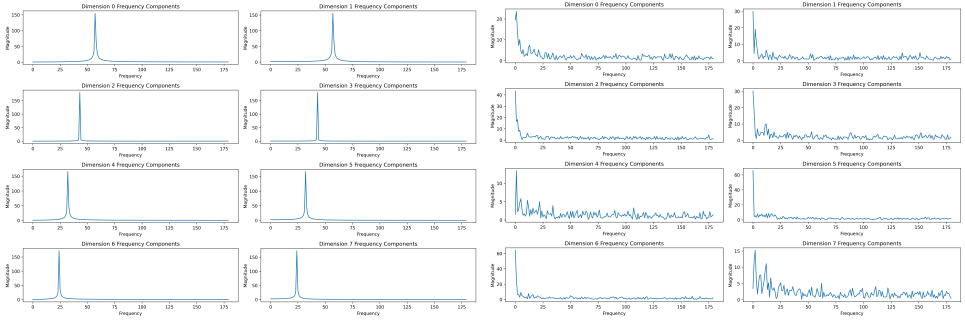
Comparing different positional encoding strategies

Method	Description	Validation Loss	ROC-AUC
Learned	Trainable embeddings that adapt to the data's temporal patterns.	**0.3347**	**0.9296**
Sinusoidal	Fixed, non-learnable embeddings based on sine and cosine functions.	0.3372	0.9284
TAPE	A variant of sinusoidal embeddings with adjusted scaling.	0.3394	0.9275
Absolute	A simple linear scaling of positions.	0.3556	0.9207

Key Finding

Learned embeddings perform the best, suggesting that for user behavior time series, adaptable embeddings are superior at capturing dataset-specific temporal patterns.

Sinusoidal vs. Learned Embeddings



Analysis

Learned embeddings are less effective at absolute and relative positional encoding. They primarily capture lower frequency components, suggesting that precise, high-frequency positional information may be less critical for this task. The model appears to benefit more from broader, lower-frequency temporal signals.

Cross-Attention: The Key Innovation

Conditioning Predictions on Future Known Events

Self-Attention (Encoder):

- Historical context processing
- Information diffusion within sequence
- Rich token representations

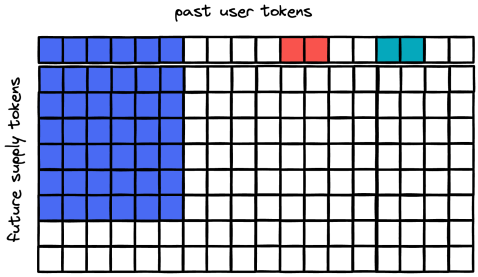
Cross-Attention (Decoder):

- **Queries**: Future supply data
- **Keys/Values**: Historical user context
- **Innovation**: Condition on known future events

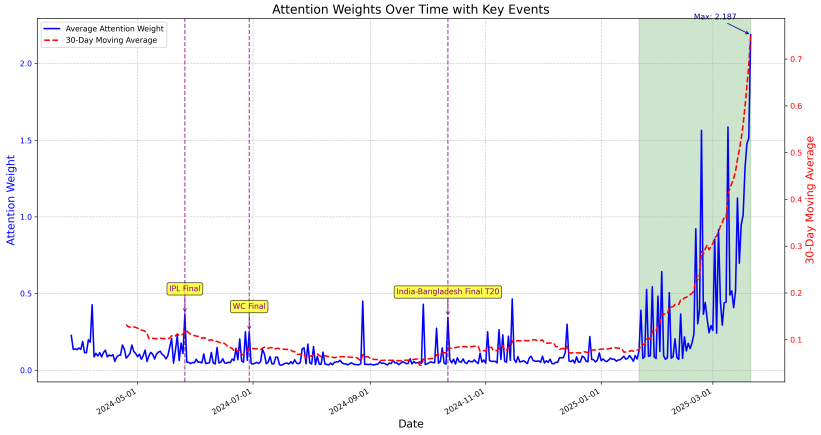
Visualizing Cross-Attention

Attention matrix is not square: Future Supply (Queries) x Historical Context (Keys)

- The cross-attention matrix has dimensions (Future Window x Historical Window), for example, 28x360.
- Each row corresponds to a future day's supply, which acts as a "query."
- Each column represents a day in the user's past behavior.
- The cell (i, j) shows how much the model focuses on the user's behavior on past day j when considering the supply on future day i.
- This allows the model to answer questions like: "Given the big match next Sunday



Attention Pattern Analysis



Ablation Study: Role of Supply Data

Quantifying the Impact of Supply Data on Churn Prediction

Run Name	Validation Loss	ROC-AUC
Full Model	**0.3331**	**0.9300**
No Past Supply	0.3349	0.9297
No Future Supply	0.3350	0.9294
No Supply Data	0.3345	0.9298

Conclusion

The results clearly indicate that both past and future supply data are crucial for the model's performance. The highest performance is achieved with the full model, and removing either past or future supply degrades the results, confirming their importance in providing necessary context for user behavior prediction.

Beyond Next Token Prediction

Masked Prediction Strategy:

- **Not predicting entire tokens**
- Focus on user transaction features only
- Selective prediction: **5 of 13** raw attributes
- Target-specific outputs

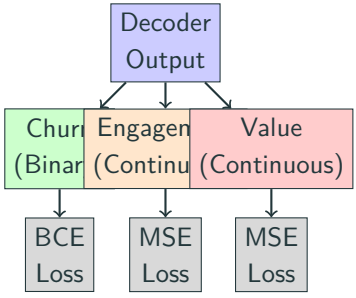
Multi-Task Learning:

- **Churn prediction** (binary)
- **CEA prediction** (continuous)
- **Dep amount** prediction (continuous)
- Shared representations across tasks

Loss Function Strategies:

- MultiTaskLoss: Uncertainty weighting

Prediction Targets



Key Difference

LLMs: Next token in sequence
LUMOS: Specific behavioral outcomes

User Embeddings

Extracting User Embeddings: Key Considerations

1. **Extraction Point:**
   - **Pre-Encoder**: Raw, uncontextualized inputs.
   - **Post-Encoder**: Rich, contextualized representations capturing historical sequence patterns.

2. **Aggregation Strategy:**
   - How to distill a sequence of embeddings into a single user-state vector.
   - Options: Mean, Max, or Exponentially Weighted Average.

3. **Recency Weighting (λ):**
   - Controls the emphasis on recent vs. long-term behavior in aggregation.
   - **Low** λ: Uniform weighting (long-term patterns).
   - **High** λ: Strong weight on recent events (short-term dynamics).

4. **Training Objective:**
   - **Multi-Task**: Embeddings benefit from shared representations across diverse tasks.
   - **Single-Task**: Embeddings are highly specialized for one objective.

User Embedding Performance

Benchmarking User Embeddings on Downstream Tasks

Model	Method (Extraction, λ)	Churn AUC	CEA Spear.	Pub. CEA Spear.	Net Win AUC	Withdraw AUC
Multi-Task	('post_encoder', 0)	0.9086	**0.3198**	0.4414	0.7121	0.9054
Multi-Task	('post_encoder', 0.1)	0.9087	0.3172	**0.4429**	0.7128	0.9059
Multi-Task	('post_encoder', 1)	0.9090	0.3047	0.4422	0.7141	0.9768
Multi-Task	**('post_encoder', 10)**	**0.9802**	0.1518	0.4403	**0.7219**	**0.9077**
Single-Task	('post_encoder', 0)	0.9063	0.3820	0.4107	0.6544	0.9735
Single-Task	('post_encoder', 0.1)	0.9063	0.3737	0.4125	0.6543	0.9736
Single-Task	('post_encoder', 1)	0.9067	0.3764	0.4087	0.6574	0.9740
Single-Task	('post_encoder', 10)	0.9077	0.3333	0.3931	0.6708	0.9048

Do you need a transformer?

Transformer vs. Deep Neural Networks

Model Architecture	Parameters	Validation Loss
Transformer (LUMOS)	**2.49M**	**0.3347**
Large Deep NN	11.57M	0.3566
Small Deep NN	2.30M	0.3575

Philosophy & Future Directions

LUMOS Philosophy: End Feature Engineering

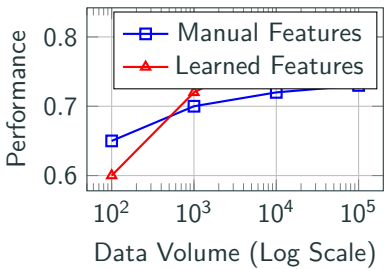
Traditional Approach:

- Manual feature engineering
- Domain expert knowledge required
- Brittle, hard to maintain
- Limited scalability

LUMOS Approach:

- **Learned representations**
- End-to-end optimization
- Automatic pattern discovery
- Scalable with data volume

Scalability Comparison



Core Insight

Learned representations scale with data while manual features plateau.

Future Directions & Next Steps

Immediate Enhancements:

- **SwiGLU activation** integration
- Scaling law experiments
- Flash Attention optimization
- Model compression techniques

Architecture Innovations:

- MoE
- Advanced positional encodings
- Alternative attention mechanisms

Platform Developments:

- Automated hyperparameter tuning
- Online learning capabilities
- Ray

Research Opportunities:

- Multi-objective optimization
- Treatment effect modeling
- Uncertainty quantification
- Explainable AI features

Vision: Enable teams to build sophisticated user behavior models with deep ML expertise, focusing on architecture and business logic while LUMOS handles the engineering complexity.