**Overview:** The document provides a comprehensive guide on designing a machine learning system for LinkedIn feed ranking with a focus on problem statement, metrics design, and requirements. It details the different aspects necessary for building a personalized feed ranking system to enhance long-term user engagement.

**Key Sections:**

1. **Problem Statement:**
   * **Objective:** Design a personalized LinkedIn feed to maximize long-term user engagement.
   * **Engagement Metrics:** Primary metric used is Click Through Rate (CTR), though user frequency can also be considered.
   * **Activity Types on LinkedIn Feed:**
     + Connections (e.g., A connects with B)
     + Informational (e.g., article shares)
     + Profile (e.g., profile updates)
     + Opinion (e.g., likes/comments on posts)
     + Site-specific (e.g., endorsements)
   * **CTR Variation:** Different activity types have varying CTRs, influencing model design and training data generation.
2. **Metrics Design and Requirements:**
   * **Metrics:**
     + **Offline Metrics:** Use normalized cross-entropy (NCE) and Area Under Curve (AUC) to evaluate model performance.
     + **Online Metrics:** Focus on user engagement levels post-deployment, measured by conversion rates (CTR).
   * **CTR Calculation:** CTR=number of clicksnumber of times feed is shownCTR = \frac{\text{number of clicks}}{\text{number of times feed is shown}}CTR=number of times feed is shownnumber of clicks​
   * **NCE Calculation:** NCE=−(plog⁡(p)+(1−p)log⁡(1−p))NCE = -\left( p \log(p) + (1-p) \log(1-p) \right)NCE=−(plog(p)+(1−p)log(1−p))
3. **Requirements:**
   * **Training:**
     + Handle large volumes of data, ideally in distributed settings.
     + Address data distribution shifts by retraining models multiple times per day.
     + Ensure high levels of personalization.
     + Maintain data freshness to avoid repetitive feeds.
   * **Inference:**
     + **Scalability:** Handle large user activity volumes (300 million users).
     + **Latency:** Ensure feed ranking processes within 50ms to fit into the overall 200ms response time.
     + **Data Freshness:** Ensure awareness of previously seen activities to avoid redundancy.
4. **Summary:**
   * **Metrics:** Aim for reasonable normalized cross-entropy.
   * **Training:** High throughput and multiple daily retrains.
   * **Personalization:** Support high levels of personalization.
   * **Inference:** Maintain latency between 100ms to 200ms and high data freshness to avoid repetitive feeds.

**Additional Insights:**

* The document underscores the complexity of creating a real-time feed ranking system that caters to the personalized needs of a vast user base while ensuring swift performance and up-to-date content delivery. It stresses the importance of continuous model retraining and the handling of non-stationary data to maintain relevance and user engagement.

**Conclusion:** The outlined approach provides a robust framework for designing a feed ranking system with a clear emphasis on metrics and requirements critical for achieving high user engagement and satisfaction on LinkedIn.

### **Detailed Summary of the Document "Feed Ranking Model - Machine Learning System Design"**

**Overview:** The document provides an in-depth guide on building a feed ranking model for machine learning systems, particularly focusing on feature engineering, training data collection, model selection, and evaluation. The goal is to enhance the personalization and relevance of LinkedIn feed content to maximize user engagement.

**Key Sections:**

1. **Feature Engineering:**
   * **User Profile:**
     + Features include job title, industry, and demographics.
     + For low cardinality features, use one-hot encoding.
     + For higher cardinality features, use embeddings.
   * **Connection Strength:**
     + Represented by the similarity between users.
     + Embeddings can measure the distance vector between users.
   * **Age of Activity:**
     + Can be considered as a continuous feature or binned value depending on the sensitivity.
   * **Activity Features:**
     + Include type of activity, hashtag, media, etc.
     + Use activity embeddings to measure similarity between activity and user.
   * **Cross Features:**
     + Combine multiple features to capture interactions between them.
     + Example provided in the Machine Learning System Design Primer.
2. **Training Data Collection:**
   * **Chronological Order Ranking:**
     + Rank posts in chronological order to collect click/not-click data.
     + Issues include serving bias and data sparsity.
   * **Random Serving:**
     + Rank posts in random order.
     + May lead to poor user experience and does not help with data sparsity.
   * **Feed Ranking Algorithm:**
     + Rank top feeds and permute them randomly for data collection.
     + Provides some randomness and helps models learn diverse activities.
   * **Data Selection:**
     + Select data for a specific period (e.g., last month, last 6 months).
     + Balance between training time and model accuracy.
     + Downsample negative data to handle imbalanced datasets.
3. **Model Selection:**
   * **Probabilistic Sparse Linear Classifier (Logistic Regression):**
     + Popular for computation efficiency and working well with sparse features.
     + Use distributed training (e.g., Logistic Regression in Spark or Alternating Direction Method of Multipliers).
   * **Deep Learning:**
     + Use fully connected layers with Sigmoid activation for the final layer.
     + Resample training data to address data imbalance.
     + Keep validation and test sets intact for accurate performance estimation.
4. **Model Evaluation:**
   * **Data Split:**
     + Split data into training and validation sets.
   * **Replay Evaluation:**
     + Use data up to time ttt for training.
     + Use test data from time t+1t + 1t+1 and reorder ranking based on the model during inference.
     + Record a match if click prediction is accurate at the correct position.
   * **Evaluation Considerations:**
     + Determine the optimal size of the training data set.
     + Decide the frequency of model retraining.
     + Adjust various hyperparameters for optimal performance.

**Additional Insights:**

* The document emphasizes the importance of feature engineering to capture relevant user and activity characteristics.
* It highlights the need for balanced and comprehensive training data to build robust models.
* The selection of appropriate models and evaluation methods is critical for achieving high performance in feed ranking systems.
* Ensuring scalability and efficiency in both training and inference phases is crucial for handling large volumes of user data and maintaining a responsive system.

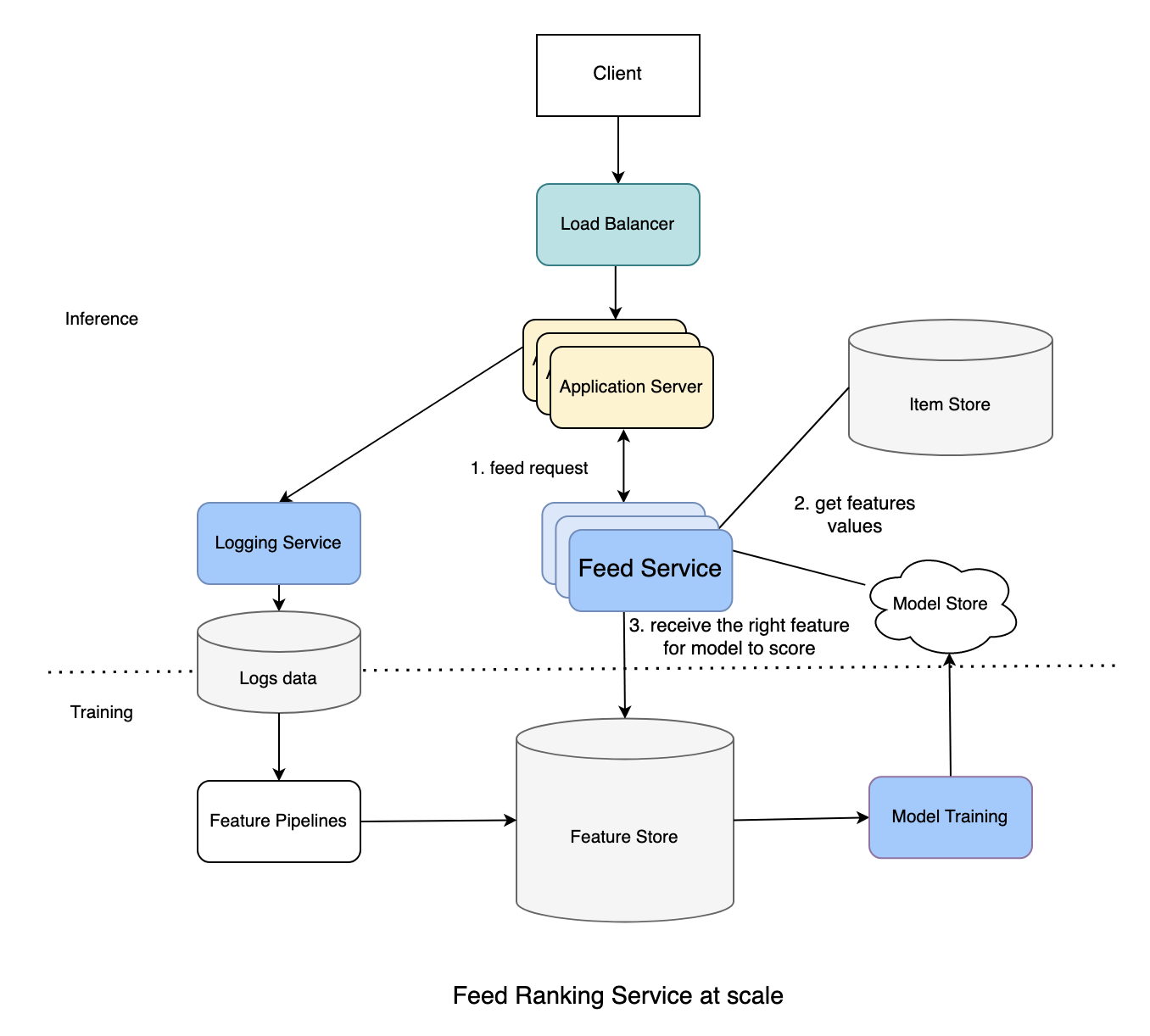
**Conclusion:** The outlined approach provides a structured methodology for developing a sophisticated feed ranking model that leverages advanced feature engineering, comprehensive training data, efficient model selection, and thorough evaluation to enhance user engagement on LinkedIn.

### **Detailed Summary of the Document "Feed Ranking System Design - Machine Learning System Design"**

**Overview:** The document provides a detailed guide on designing a feed ranking system for LinkedIn, covering assumptions, calculations, high-level design, scaling, and a summary. The aim is to create a robust and scalable system to handle feed ranking for a large user base efficiently.

**Key Sections:**

1. **Calculation & Estimation:**
   * **Assumptions:**
     + 300 million monthly active users.
     + Each user sees 40 activities per visit and visits 10 times per month.
     + This results in 120 billion observations/samples per month.
   * **Data Size:**
     + Click Through Rate (CTR) is estimated at 1% for 1 month, resulting in 1 billion positive labels and 110 billion negative labels.
     + Each data point collects hundreds of features, with each row taking 500 bytes.
     + For 120 billion rows, the total data size is approximately 60 Terabytes.
     + To manage costs, store the last 6 months to 1 year of data in a data lake and archive older data in cold storage.
2. **High-level Design:**
   * **Feature Store:**
     + Stores feature values with low latency (<10ms) access for scoring during inference.
     + Examples include MySQL Cluster, Redis, and DynamoDB.
   * **Item Store:**
     + Stores all user-generated activities and the models corresponding to users.
     + Ensures consistent user experience by using the same feed ranking method for each user.
   * **System Flow:**
     + Client sends a feed request to the Application Server.
     + Application Server requests feeds from the Feed Service.
     + Feed Service retrieves the latest model from the Model Repository, fetches features from the Feature Store, and gets feeds from the Item Store.
     + The model returns recommended feeds sorted by the likelihood of click-through rate.
3. **Scaling the Design:**
   * **Scaling Feed Service:**
     + Represent both Retrieval Service and Ranking Service for better visualization.
   * **Scaling Application Server:**
     + Scale out the Application Server and use a Load Balancer to distribute the load.



1. **Summary:**
   * **Model Building:**
     + Developed binary classification models with custom loss functions to reduce sensitivity to background click-through rates.
   * **Training Data Generation:**
     + Created processes to generate training data for machine learning models.
   * **Training and Inference Scalability:**
     + Implemented scaling solutions for both Application Server and Feed Services to manage large user bases and ensure efficient performance.

**Additional Insights:**

* The document emphasizes the importance of low-latency feature access and the need for efficient data storage and retrieval systems to handle the vast amounts of data generated by users.
* It highlights the critical role of scaling both the application and feed services to maintain performance and user experience as the number of users grows.
* The use of feature stores and item stores ensures that the system can deliver personalized and relevant content to users in real-time.

**Conclusion:** The document provides a comprehensive blueprint for designing and scaling a feed ranking system, focusing on efficient data handling, robust model deployment, and scalability to support a large user base. It combines advanced machine learning techniques with practical system design principles to create a high-performance feed ranking solution for LinkedIn.