Table of Contents

Introduction to the Problem **2**

Background *2*

Problems *2*

Overall Directions to a Solution **4**

Dataset *4*

Study Methodologies *4*

How to Address Problems *6*

Evaluation Strategy: Two-Tier Evaluation *8*

Design of the Solution **10**

Overview of Content Recommendation System *10*

Evaluation Metric *11*

Advanced Evaluation Metric (Optional) *12*

Models in the Retrieval Stage *13*

Testing of the Solution **15**

Dataset *15*

Improving Models in the Retrieval Stage *15*

Results **18**

Overall Performance *18*

Performance Across Different Models *18*

Factors Affecting Model Performance *18*

Addressing the Problems *18*

Conclusions **19**

Appendixes **20**

Introduction to the Problem

Background

[To be completed]

Problems

The following are key challenges commonly encountered in content recommendation systems:

1. Data Sparsity (High Priority)

Definition: Data sparsity refers to the insufficient amount of user–item interaction data, which hampers the system's ability to generate accurate recommendations.

Impact: Limited interaction data adversely affects various recommendation algorithms. While collaborative filtering is particularly vulnerable, content-based and hybrid models also face challenges in extracting meaningful patterns. This deficiency can lead to less accurate predictions, reduced personalization, and, in some cases, overfitting of the available data.

2. Cold-Start Problem (High Priority)

Definition: The cold-start problem arises when new users or items do not have enough interaction history, making it difficult to provide personalized recommendations.

Impact: Consequently, new users often receive generic suggestions, and new items struggle to gain visibility. This ultimately reduces user engagement and satisfaction.

3. Long Tail Items (High Priority)

Definition: Long tail items are those that are less popular and receive fewer interactions, which frequently leads to their omission in recommendation lists.

Impact: Overlooking long tail items diminishes content diversity and user satisfaction while reinforcing the Pareto Principle, where a small subset of popular items dominates the recommendations.

4. Scalability (Medium Priority)

Definition: Scalability is the system's ability to manage an increasing number of users and items efficiently without suffering from significant performance degradation.

Impact: As the platform grows, rising computational demands can result in slower response times and a general decline in performance.

5. Diversity vs. Relevance Trade-off (Low Priority)

Definition: This challenge involves striking a balance between offering a diverse range of content and maintaining a high level of relevance to user preferences.

Impact: Overemphasizing relevance may produce overly narrow recommendations, whereas an excessive focus on diversity can compromise personalization.

6. Privacy Concerns (Low Priority)

Definition: Privacy concerns relate to the responsible handling of user data to ensure privacy while still providing personalized recommendations.

Impact: Ineffective data management can lead to privacy breaches, erosion of user trust, and non-compliance with regulatory standards.

7. Evaluation Challenges (Low Priority)

Definition: Evaluation challenges refer to the difficulties in accurately assessing recommendation algorithms, particularly when ground-truth labels or real-time user feedback are limited.

Impact: Without robust evaluation metrics, there is a risk of deploying suboptimal models, which can ultimately diminish user satisfaction.

Overall Directions to a Solution

Dataset

The Yelp dataset is employed in this research. Yelp is a platform that enables users to find, review, and recommend businesses across various categories, such as food, shopping, and entertainment, in eight metropolitan areas across the USA and Canada. This dataset was originally utilized for the Yelp Challenge and is available for academic research.
Please refer to Appendix 1 for detailed information regarding the dataset properties.

Study Methodologies

Our project aims to investigate content recommendation systems and address common challenges in the industrial domain. Consequently, a problem-solution approach serves as the primary framework for our research. The following methodologies are employed:

1. Comparative Evaluation

This method forms the backbone of our study by systematically comparing baseline and target models to assess their performance under various conditions.

Steps:

1. **Setup Evaluation Metrics**: Define clear and measurable metrics to evaluate the models. Common metrics include:
   - Accuracy, Precision, Recall, F1-Score: Measure relevance and correctness of recommendations.
   - User Satisfaction: Measured through proxy metrics or qualitative assessments.
   - Intra-List Diversity (Optional): Evaluate the variety within recommendation lists.
   - Novelty (Optional): Determine how unexpected or fresh the recommendations are.
   - Coverage (Optional): Quantify the proportion of the dataset that is effectively recommended.
2. **Select Target Users/Items**: Choose the evaluation group based on the focus of the study. Options include:
   - The entire testing dataset for general evaluation.
   - Specific groups such as cold-start users, long-tail items, or niche categories for focused analysis.
3. **Select Baseline Model**: Define the existing or simple model for comparison, such as Collaborative Filtering or Content-Based Filtering. *Suggested baseline models (as proposed by Ruohan) include:*

- *Random Recommendation*
- *Best Uniform Recommendation (i.e., providing the same recommendation to all users).*

4. **Select Target Model**: Define the enhanced model under evaluation, such as DSSM or a hybrid recommender system.
5. **Statistical and Visual Analysis**:
   - Visualization: Employ bar charts, ROC curves, and trade-off plots.
   - Statistical Testing (Optional): Validate performance differences using paired t-tests or similar methods.

2. User Acceptance Test (UAT)

This method emphasizes direct involvement of users to evaluate the system's performance from a qualitative perspective, akin to a user journey mapping approach.

Steps:

1. **Define User Group**: Establish profiles for user groups through questionnaires and informed assumptions to reflect the diversity of the target audience.
2. **Design Scenarios**: Present users with predefined scenarios along with the recommendations generated by the system
3. **Collect Feedback**:
   - Use surveys or interviews to gather qualitative insights regarding relevance, usability, and satisfaction.
   - Ask users to rank or rate the recommendations and provide explanations for their preferences.
   - *As suggested by Ruohan, consider using GPT as an AI evaluator.*
4. **Analyze Results**:
   - Identify patterns in the feedback, such as common preferences or usability issues.
   - Use these insights to complement the quantitative findings from the Comparative Evaluation.

3. Case Studies and Scenario Simulation (Optional)

This method tests the system's behavior in specific, well-defined cases to illustrate its strengths and limitations. It is particularly useful for highlighting system behavior in edge cases or under unique conditions.

Steps:

1. **Define Use Cases**: Identify scenarios that emphasize specific challenges or features, such as:
   - Recommending niche items with limited data (addressing the long-tail problem).
   - Handling users with unique preferences.

- Adjusting recommendations to balance diversity and relevance.

2. **Qualitative Evaluation**: Examine the recommendations generated for these cases:
    - Highlighting successful examples where the system performed well.
    - Discussing limitations or unexpected outcomes to identify areas for improvement.

How to Address Problems

Below is the list of problems we identified and the step-by-step strategies to address them. Some problems are prioritized higher due to their significant impact on recommendation performance. For lower-priority issues, detailed strategies may not yet be developed.

1. Data Sparsity (High Priority)

Methodologies: *Comparative Evaluation*, *User Acceptance Test (UAT)*

Solution/Strategy Flow:

Problem	Solution
1. Demonstrate that the dataset exhibits sparsity, for example, by showing that many items have minimal interactions (such as reviews without associated clicks or likes). 2. Provide statistics (e.g., the number of users with fewer than five reviews). 3. (Optional) Illustrate the limitations of collaborative filtering using techniques such as Leave-One-Out Cross-Validation or metrics like Hit Rate and Top-K Evaluation.	1. Illustrate how models such as DSSM mitigate sparsity by leveraging additional data or embedding techniques. 2. Highlight performance improvements (e.g., increased recall or F1-score) after tuning parameters or using alternative loss functions. 3. Demonstrate overall system improvements despite the presence of sparse data.

2. Cold-Start Problem (High Priority)

Methodologies: *Comparative Evaluation*, *User Acceptance Test (UAT), Case Studies and Scenario Simulation (Optional)*

Solution/Strategy Flow:

Problem	Solution
1. (Optional) Relate the issue to data sparsity statistics, such as instances where users or items have zero interactions.	1. Detail how solutions, such as assigning initial labels or interests and leveraging demographic data,

2. Describe scenarios that illustrate the user journey for new users or the introduction of new items. 4.	can enhance recommendations for new users. 2. Showcase improvements in the user journey with tailored recommendations. 3. (Optional) Provide updated statistics demonstrating increased engagement for new users or items. 4.

3. Long Tail Items (High Priority)

Methodologies: *Comparative Evaluation*, *Case Studies and Scenario Simulation (Optional)*

Solution/Strategy Flow:

Problem	Solution
1. Explain the Pareto Principle and its effect on content recommendation. 2. Highlight challenges in training models with long-tail items, such as biased negative sampling in DSSM or issues encountered in self-supervised learning. 3. Provide statistics showing the low exposure of long-tail items.	1. Outline strategies to promote long-tail items, including tuning model parameters, employing balanced sampling methods, or incorporating filtering layers. 2. Present updated exposure statistics to demonstrate improved visibility for long-tail items.

4. Scalability (Medium Priority)

Methodologies: *Comparative Evaluation*

Strategy Flow:

- Evaluate system response times and performance metrics as the data volume increases.
- Incorporate scalability-focused techniques, such as caching, approximate nearest neighbor search, or distributed training.
- Compare system performance before and after scaling. Synthetic datasets can be used to simulate increased load and validate scalability.

5. Diversity vs. Relevance Trade-off (Low Priority)

Solution/Strategy Flow:

- Explore strategies to balance diversity and relevance, such as multi-objective optimization.
- Measure trade-offs using metrics like intra-list diversity and relevance scores.

Evaluation Strategy: Two-Tier Evaluation

The Two-Tier Evaluation approach assesses the recommendation model using two distinct settings: one that measures its scoring (or prediction) capability and another that simulates production-style retrieval. This dual strategy bridges the gap between offline performance and real-world system behavior.

Tier 1: Prediction/Scoring Evaluation

Objective: Measure how accurately the model predicts positive user–item interactions or ratings using a controlled test set with ground-truth labels.

Method:

Input	A set of user–item pairs sampled from historical interactions.
Process	1. Use the model to predict a rating or probability score for each pair. 2. Compare the predictions with actual labels (positive/negative).
Metrics	Accuracy, Precision, Recall, AUC (Area Under the ROC Curve), F1-score, etc.

Benefits:

1. Provides a clear assessment of the model's ability to differentiate between positive and negative interactions.
2. Helps diagnose issues related to model training, feature representation, and classification or regression performance.
3. Facilitates rapid iteration since the candidate set is smaller and the evaluation process is straightforward.

Tier 2: Retrieval Evaluation (Production Simulation)

Objective: Simulate a production environment by retrieving the top-k candidate items for a given user from a large catalog, thereby measuring how effectively the model's embedding or similarity function identifies items that the user actually likes.

Method:

Input	A user identifier and the entire (or a large subset of) item catalog.
Process	1. Retrieve the top-k nearest neighbor items based on the model's embedding or similarity score. 2. Determine whether the retrieved items include those with which the user has positively interacted (according to historical data).
Metrics	1. ***Recall@k***: *Fraction of truly positive items that appear within the top-k list.* 2. ***Precision@k***: *Fraction of items in the top-k list that are truly positive.* 3. ***NDCG*** *(Normalized Discounted Cumulative Gain): Measures the ranking quality of the retrieved list.*

Benefits:

1. Directly mimics the production scenario, revealing how the model performs when retrieval tasks are constrained by real-world factors such as runtime efficiency and large candidate sets.
2. Highlights potential challenges such as the “needle in a haystack” problem, where even a strong model might have low recall if only a small fraction of relevant items exists within a large catalog.
3. Helps verify whether the model’s training objectives align with the requirements of the retrieval task; a model optimized solely for prediction may underperform in ranking tasks.

Why Both Tiers Are Important

Complementary Insights:

- Prediction/Scoring Evaluation reveals whether the model can effectively differentiate between positive and negative interactions under controlled conditions.
- Retrieval Evaluation assesses whether these predictions translate into effective rankings in a real-world scenario, where only a few relevant items must be identified among many.

Training vs. Production Alignment:

- A model may perform well in prediction if optimized on a pairwise basis yet yield low recall when tasked with ranking an entire catalog.
- Running both evaluations helps identify if further tuning—or even a change in loss function (such as switching to a ranking loss)—is necessary to better align the model with production goals.

Feedback for Model Improvement:

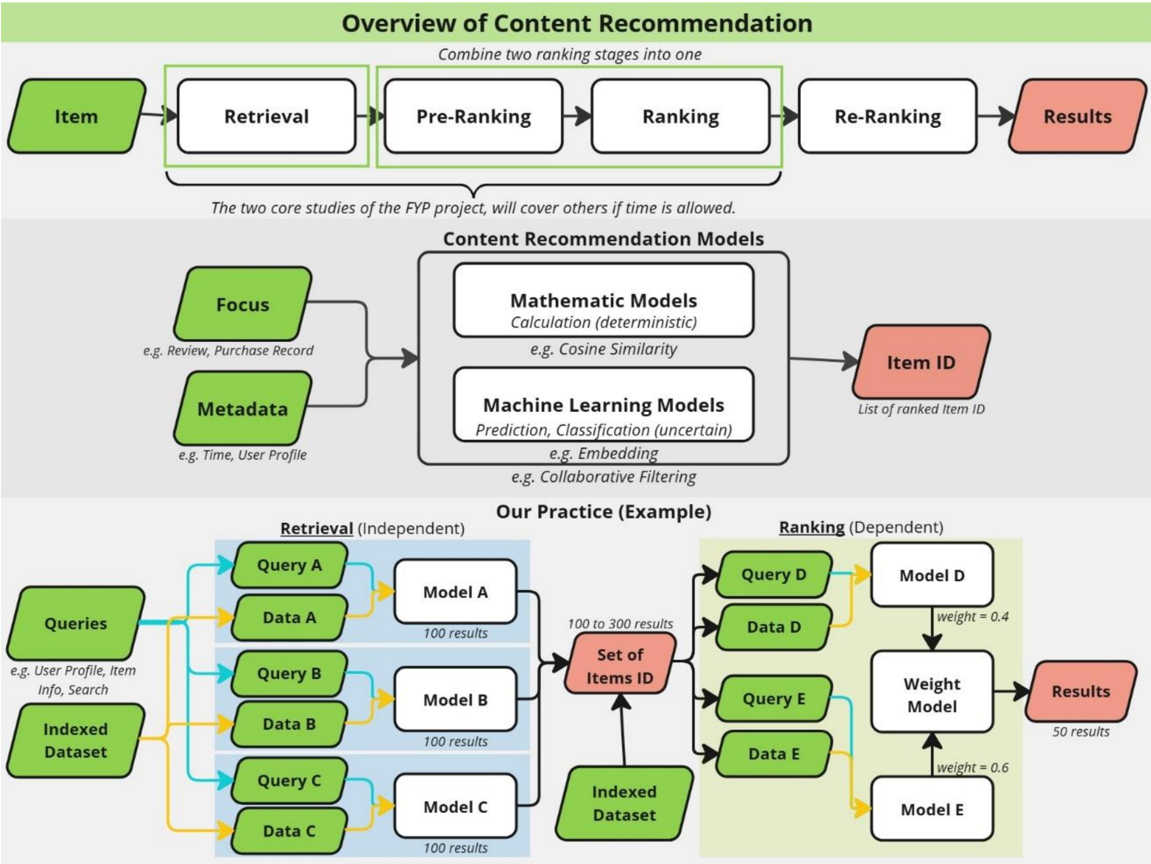
- If Tier 1 performance is strong but Tier 2 lags, it is essential to investigate factors such as the quality and structure of the embedding space, the efficiency of the nearest neighbor search, and the potential need for post-retrieval re-ranking steps.

Operational Considerations:

- Tier 2 evaluation ensures that runtime constraints and large-scale retrieval challenges are addressed, confirming that improvements made during development translate into tangible benefits when the system is deployed.

Design of the Solution

Overview of Content Recommendation System



Content recommendation systems typically operate through four sequential stages before presenting the final recommendations to users:

1. **Retrieval**: The initial stage where a broad set of candidate items is identified.
2. **Pre-Ranking**: A preliminary filtering of the candidates based on lightweight scoring.
3. **Ranking**: A more detailed ranking of the filtered items using complex models.
4. **Re-Ranking**: A final adjustment to the ranked list to optimize user satisfaction.

By leveraging recommendation models with varying levels of complexity at each stage, the system can efficiently process millions of items, narrow them down to thousands of candidates, and ultimately present a curated top-10 list.

In this research, our focus is on two primary stages:

1. **Retrieval Stage**: Where we implement three distinct models.
2. **Combined Pre-Ranking and Ranking Stage**: Where we implement two models to refine the candidate list further.

Evaluation Metric

1. Accuracy

Measures the overall correctness of the recommendations.

2. Precision / Precision@*K*

Evaluates the relevance of the retrieved items, with *Precision@K* specifically assessing the top-*K* recommendations.

3. Recall / Recall@*K*

Often considered the most crucial metric in content recommendation systems, recall measures the proportion of relevant items that are successfully retrieved. *Recall@K* focuses on the top-*K* items.

4. F1 Score

Provides a balanced view by combining both precision and recall into a single measure.

5. Weighted F-Beta Score

A generalized form of the F1 score where the parameter β adjusts the relative importance of recall versus precision. It is defined as:

$$F_\beta = (1+\beta^2)\frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

- When recall is more critical, β is set greater than 1 (e.g., β = 2).
- When precision is more critical, β is set less than 1 (e.g., β = 0.5).

In our research, the weighted Fβ-score is particularly relevant because recall is key for retrieval models. For example, ratings of 4 stars or higher are considered as indicating user interest, whereas ratings of 3 stars or lower are not; however, marginal ratings may still be acceptable. Ensuring that the retrieval model returns a sufficient number of relevant items makes recall especially important.

6. Mean Reciprocal Rank (MRR)

MRR evaluates the rank of the first relevant item in the recommendation list. The Reciprocal Rank (RR) for an individual query is defined as:

$$RR = \frac{1}{rank\ of\ the\ first\ relevant\ item}$$

The Mean Reciprocal Rank is then computed as the average RR over all users:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank\ of\ the\ first\ relevant\ item\ for\ user\ i}$$

Example:

- User A: Relevant item at rank 3 → Reciprocal Rank = $\frac{1}{3}$
- User B: Relevant item at rank 1 → Reciprocal Rank = $\frac{1}{1}$
- User C: No relevant item → Reciprocal Rank = 0

Thus, if there are three users, the MRR would be: $MRR = \frac{1}{3}\left(\frac{1}{3} + 1 + 0\right) = 0.44$

Advanced Evaluation Metric (Optional)

1. Coverage

Measures the proportion of the catalog (either items or users) that the system is capable of recommending. For example:

$$Item\ Coverage = \frac{Number\ of\ unique\ items\ recommended}{Total\ number\ of\ items\ in\ the\ catalog}$$

High coverage indicates that the system recommends a diverse range of items rather than solely focusing on popular ones.

2. Diversity

Quantifies the dissimilarity among items within a single user's recommendation list. Diversity is typically calculated using a similarity metric (e.g., cosine similarity) to compute pairwise distances between items:

$$Diversity = 1 - \frac{\sum_{i \neq j} Similarity(i, j)}{Number\ of\ item\ pairs}$$

This metric helps balance personalization with exploration.

3. Novelty

Assesses how unexpected or unique the recommendations are, often rewarding the recommendation of less popular items. One example metric is the Inverse Popularity Score:

$$Novelty = \frac{1}{N} \sum_{i=1}^{N} -\log_2(popularity\ of\ item\ i)$$

4. Serendipity

Evaluates the extent to which recommendations are both surprising and delightful, going beyond mere relevance. This can be measured by comparing the expected recommendations (e.g., as predicted by a baseline collaborative filtering model) to the actual recommendations generated.

5. User Effort

Measures the effort required by users to locate relevant items, such as the average position of relevant items in the recommendation list:

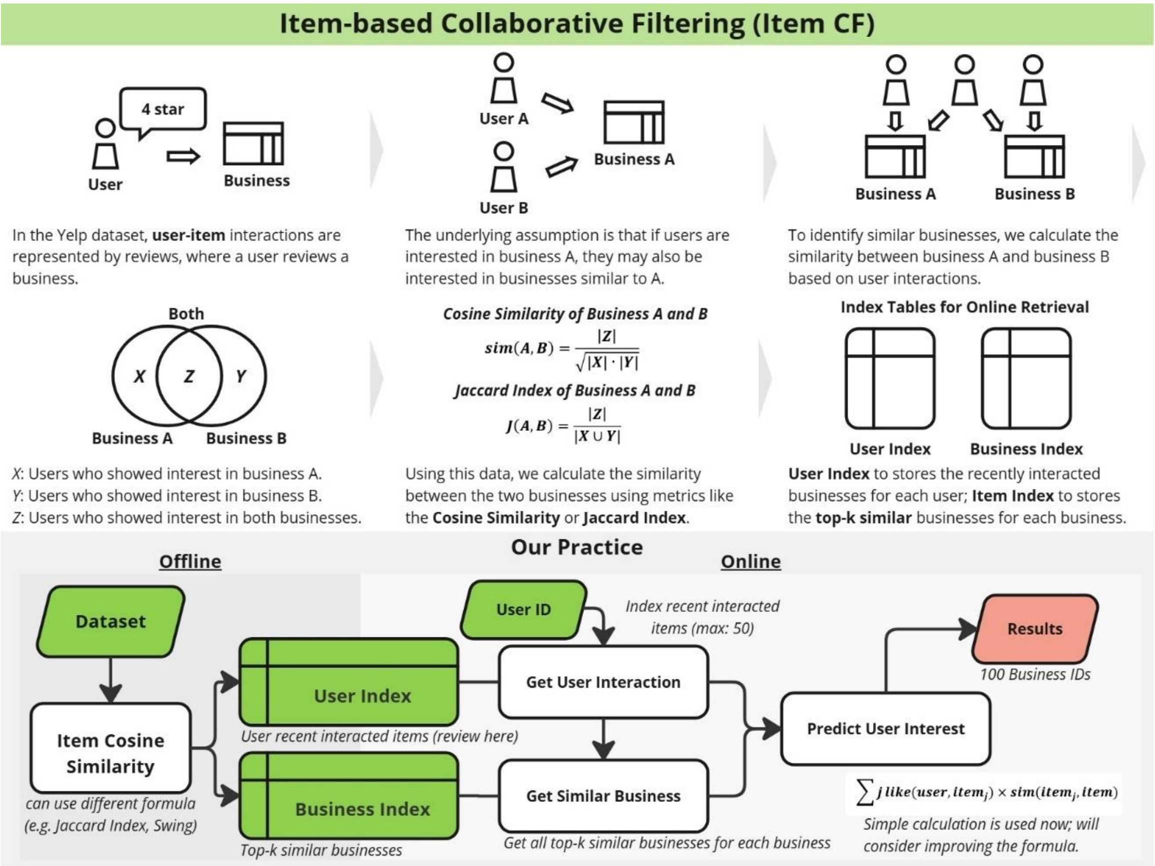
$$Effort = \frac{\sum_{i=1}^{N} Position\ of\ relevant\ item}{N}$$

6. Fairness

Assesses whether the recommendations are equitably distributed among different user groups or item categories, ensuring that the system does not favor certain groups over others.

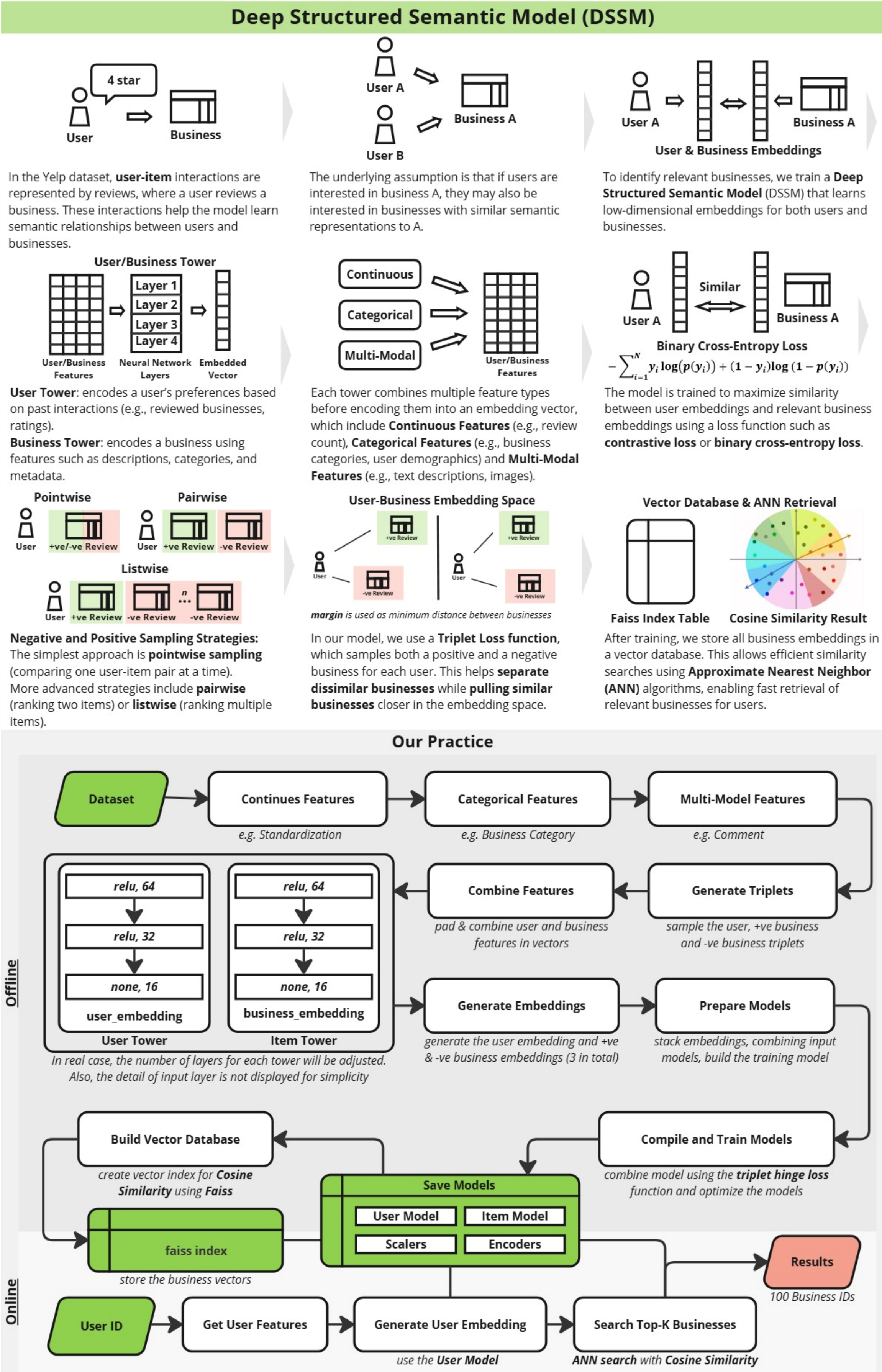
Models in the Retrieval Stage

Item-based Collaborative Filtering (Item CF)



[Detail Description of ItemCF Model]

Deep Structured Semantic Model (DSSM)

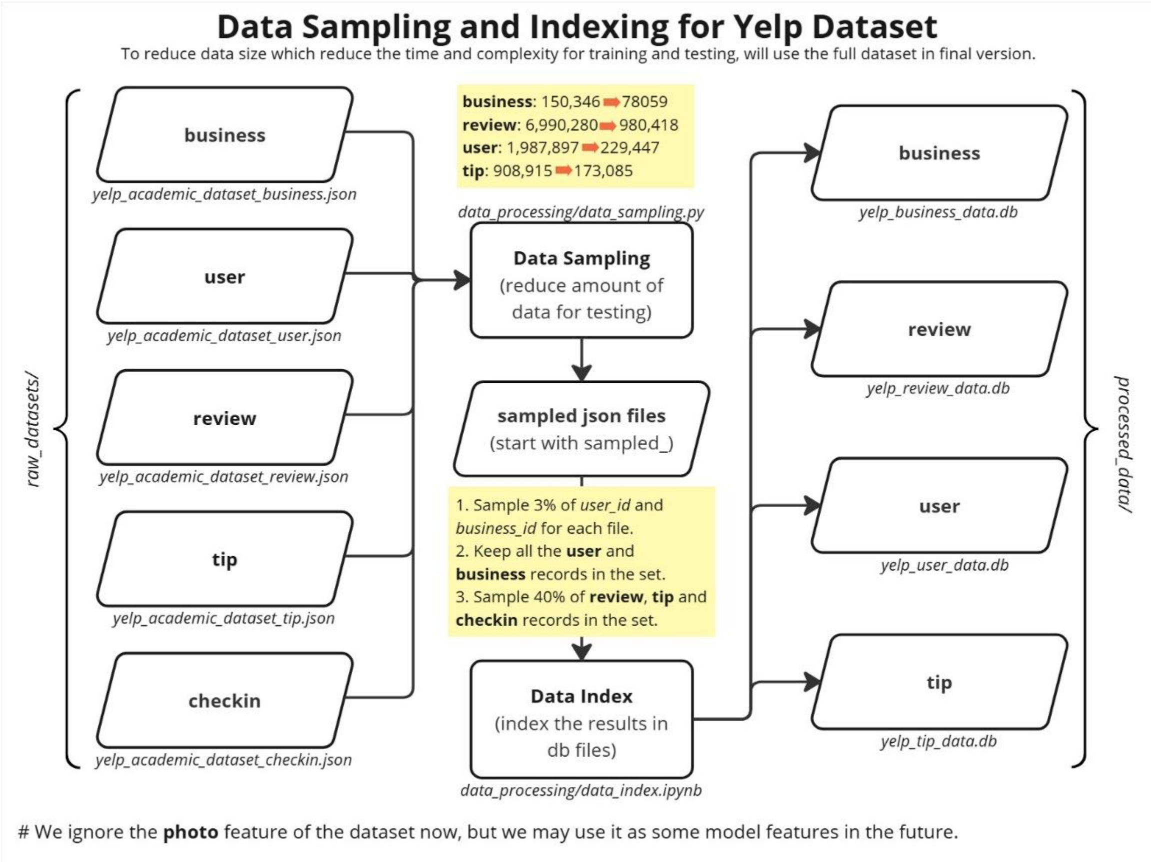


[Detail Description of DSSM Model]

Testing of the Solution

Dataset

To reduce training time, we down-sample the dataset during the training phase. This approach enables more efficient model development while still capturing the essential patterns present in the full dataset.



Improving Models in the Retrieval Stage

Our improvements target both traditional similarity-based methods (e.g., Item CF) and more advanced retrieval models, including neural embedding and hybrid approaches. The following strategies are applicable across these models:

1. Improving Item Similarity Metric

For models that rely on similarity calculations or embedding representations:

- **Cosine Similarity**: Serves as the baseline by measuring the angular similarity between item vectors.

- **Jaccard Index**: Emphasizes the overlap of interaction sets, making it particularly effective for binary interaction data.
- **Pearson Correlation Coefficient**: Evaluates the linear correlation between numerical ratings.
- **Combine Multiple Metrics**: Integrates various similarity measures by weighting each according to its performance contribution.
- **Embedding Distance Metrics**: For neural retrieval models, consider alternative distance metrics (e.g., Euclidean or Manhattan distance) and evaluate their impact on candidate selection.

2. Features Engineering

Enhancing the feature set is critical regardless of the underlying retrieval model:

- **Enhanced Weighting Schemes**: Assign higher weights to more recent interactions or to interactions associated with higher ratings.
- **Incorporate Additional Parameters**: Enrich the feature set by including user demographics, business categories, and sentiment analysis of reviews. Consider also incorporating contextual factors such as time of interaction, user location, or session data.
- **Timeliness and Time Decay**:
  Exclude older reviews or apply decay factors to reduce their influence over time. For example, the weight of an interaction can be computed as:

$$Weight = e^{-\lambda \cdot Age\ of\ Interaction}$$

  where λ controls the rate of decay (e.g., 0.01 for gradual decay).

3. Negative Sampling

Negative sampling improves training efficiency and model robustness:

- **Rationale**: Emphasizes harder-to-classify items during training.
- **Approach**: Dynamically generate negative samples during both training and testing phases. This method helps reduce bias and ensures that the model learns to differentiate between relevant and irrelevant items effectively.

4. Loss Functions

Optimize the training objective to better reflect retrieval performance:

- **Ranking-Specific Loss Functions**: Experiment with loss functions designed for ranking tasks, such as Bayesian Personalized Ranking (BPR) or Listwise Loss. These losses are applicable to both traditional CF models and modern neural retrieval systems.
- **Hybrid Loss Functions**: Consider combining classification losses with ranking losses to better balance between prediction accuracy and ranking quality.

5. Hyperparameter Tuning

- **Optimization Strategies**: Optimize key parameters such as learning rate, embedding dimensions, and regularization factors via grid search or Bayesian optimization. This systematic approach is vital for both similarity-based and neural retrieval models.
- **Model-Specific Adjustments**: For neural models, adjust network architectures (e.g., number of layers, dropout rates) to improve generalization. For traditional models, fine-tune similarity weighting and feature selection parameters.

6. Model Architecture Enhancements

Beyond feature engineering and loss functions, consider improvements specific to model architectures:

- **Deep Neural Embeddings**: Experiment with different network architectures to generate more discriminative embeddings for items and users.
- **Attention Mechanisms**: Incorporate attention layers to dynamically weight the importance of various features, which can help capture complex user-item interactions.
- **Hybrid Approaches**: Combine traditional collaborative filtering with neural-based representations to leverage the strengths of both approaches.

Results

Overall Performance

Performance Across Different Models

Models in Retrieval Stage				
	Metric	ItemCF	DSSM	UserCF
Retrieval	Accuracy	0.56	0.54	
	Precision	0.63	0.66	
	Recall	0.27	0.13	
	F1 Score	0.37	0.21	
	Fβ (B=2)	0.32	0.15	
	MRR	0.07	0.00	
Prediction	Accuracy	0.43	0.60	
	Precision	0.79	0.69	
	Recall	0.26	0.76	
	F1 Score	0.39	0.73	
	Fβ (B=2)	0.30	0.75	
	Unratted	88.72%	0.00%	

[Place Holder for Models in Ranking]

Factors Affecting Model Performance

Item CF Discovered Factors				
	Metric	1001	1002	1003
Retrieval	Accuracy	0.56	0.50	0.50
	Precisior	0.63	0.67	0.33
	Recall	0.27	0.01	0.00
	F1 Score	0.37	0.01	0.00
	Fβ (B=2)	0.32	0.01	0.00
	MRR	0.07	0.01	0.70
Prediction	Accuracy	0.43	0.42	0.38
	Precisior	0.79	0.80	0.68
	Recall	0.26	0.25	0.15
	F1 Score	0.39	0.38	0.25
	Fβ (B=2)	0.30	0.29	0.18
	Unratted	88.72%	88.76%	89.83%

Model	Objective
1001	Baseline Model
1002	Add Time-Decay Feature
1003	Apply Jaccard Similarity

[Place Holder for DSSM]

Addressing the Problems

Conclusions

Appendixes

1. Detail of Yelp Dataset

Yelp Dataset

business_details (business)

Field Name	Data Type	Description
business_id	TEXT	Unique identifier for each business
name	TEXT	Name of the business
address	TEXT	Street address of the business
city	TEXT	City where the business is located
state	TEXT	State where the business is located
postal_code	TEXT	Postal code of the business
business_id	TEXT	Foreign key referencing business_details
category	TEXT	Business category

business_categories (business)

Field Name	Data Type	Description
business_id	TEXT	Foreign key referencing business_details
category	TEXT	Business category

checkin_data (business)

Field Name	Data Type	Description
business_id	TEXT	Foreign key referencing business_details
checkin_date	TEXT	Date of check-in (format: YYYY-MM-DD HH:MM:SS)

tip_data (tip)

Field Name	Data Type	Description
user_id	TEXT	Foreign key referencing user_data
business_id	TEXT	Foreign key referencing business_details
text	TEXT	Tip content
date	TEXT	Date of the tip (YYYY-MM-DD HH:MM:SS)
compliment_count	INTEGER	Number of compliments received for the tip

review_data (review)

Field Name	Data Type	Description
review_id	TEXT	Unique identifier for each review
user_id	TEXT	Foreign key referencing user_data
business_id	TEXT	Foreign key referencing business_details
stars	REAL	Star rating given in the review
date	TEXT	Date of the review (YYYY-MM-DD HH:MM:SS)
text	TEXT	Review content
useful	INTEGER	Useful votes received
funny	INTEGER	Funny votes received
cool	INTEGER	Cool votes received

user_data (user)

Field Name	Data Type	Description
user_id	TEXT	Unique identifier for each user
name	TEXT	Name of the user
review_count	INTEGER	Number of reviews written by the user
yelping_since	TEXT	Date the user joined Yelp (YYYY-MM)
useful	INTEGER	Number of useful votes received
funny	INTEGER	Number of funny votes received
cool	INTEGER	Number of cool votes received
fans	INTEGER	Number of fans
average_stars	REAL	Average star rating given by the user
friends	TEXT	List of friends stored as a string
elite	TEXT	Years user was elite stored as a string
compliment_*	INTEGER	Counts of specific compliments received

2. Item CF Model Training Log

	Basic Information			Model Performance						
Code	Objective	Description	Strategy	Accruacy	Precision	Recall	F1 Score	F-beta (B=2)	MRR	Unrated
1001	setting up baseline model	using the basic model, using the balace data (50% +ve)	Retrieval	0.56	0.63	0.27	0.37	0.32	0.07	N/A
1001	setting up baseline model	using the basic model	Prediction	0.43	0.79	0.26	0.39	0.30	N/A	88.72%
1002	test the time-decay feature	Apply time weighting to the rating	Retrieval	0.504	0.667	0.007	0.014	0.009	0.009	N/A
1002	test the time-decay feature	Apply time weighting to the rating	Prediction	0.42	0.80	0.25	0.38	0.29	N/A	88.76%
1003	test the Jaccard Similarity	Using Jaccard similarity	Retrieval	0.50	0.33	0.00	0.00	0.00	0.70	N/A
1003	test the Jaccard Similarity	Using Jaccard similarity	Prediction	0.38	0.68	0.15	0.25	0.18	N/A	89.83%

	Model Result			Parameters			Testing Dataset		Storage
Code	TN	FP	FN	Beta	K	i (user)	+ve sample	-ve sample	
1001	712	128	611	1.5	300	1000	832	840	ItemCF 1001.ipynb
1001	74	14	155	2.00	N/A	1000			ItemCF 1001.ipynb
1002	873	3	826	2.00	10	1000	832	840	
1002	72	13	158	2.00	N/A	1000	1821	813	
1003	836	4	830				832	840	
1003	73	13	154				1821	813	

For training detail, please explore the ***Model Optimization*** Excel file.