# Goodreads Recommender Systems

A three-task pipeline: Read, Rating, and Category Prediction

Emily Chen

April 17, 2026

M.S. Data Science, UC San Diego
CSE 158/258

# Outline

# Project Overview

## What was built

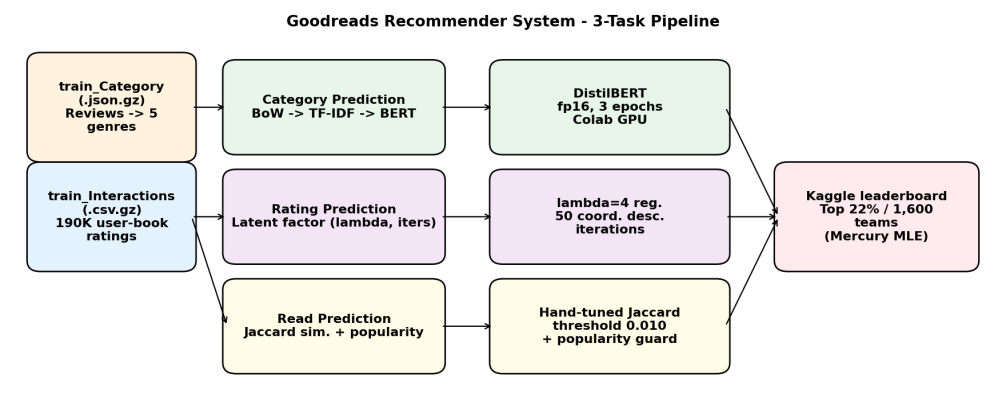
A quarter-long sequence of recommender-system tasks on the **Goodreads / UCSD-Book-Graph** subset (190K rated user–book interactions), plus a final independent project on Food.com recipe ratings.

**Three Kaggle tasks (Mercury MLE leaderboard).**

- **Read** (binary): Jaccard sim. + popularity guard.
- **Rating** (regression, 1–5): biased latent-factor model with $L_2$ reg.
- **Category** (5-class): BoW $\rightarrow$ TF-IDF + Linear SVC $\rightarrow$ DistilBERT.

**Headline outcome**
Top **22%** of **1,600** participants on the Mercury MLE leaderboard; category accuracy lifted **0.69** $\rightarrow$ **0.78** (+13pp absolute) across three iterations.

# Three-Task Pipeline




Each task ran end-to-end — from raw `train_*.gz` files through model training to a Kaggle-format prediction CSV.

# Task 1 — Read Prediction

# Read Prediction — Item–Item Jaccard with a Popularity Guard

**Setup.** Negatives sampled per validation row from books not in the user's history.

**Predictor.**

$$\text{read}(u, b) = \mathbf{1}\Big[\max_{b' \in \text{history}(u)} J\big(\text{readers}(b), \text{readers}(b')\big) > 0.010 \;\text{ or }\; |\text{readers}(b)| > 30\Big]$$

where $J$ is the Jaccard index over reader sets.

**Why it works.** Personalised similarity keeps a user-specific tilt (a fantasy reader is not predicted to have read every popular romance), while the popularity guard catches cold-start books that have no overlap with the user's history. Both thresholds were hand-tuned on validation.

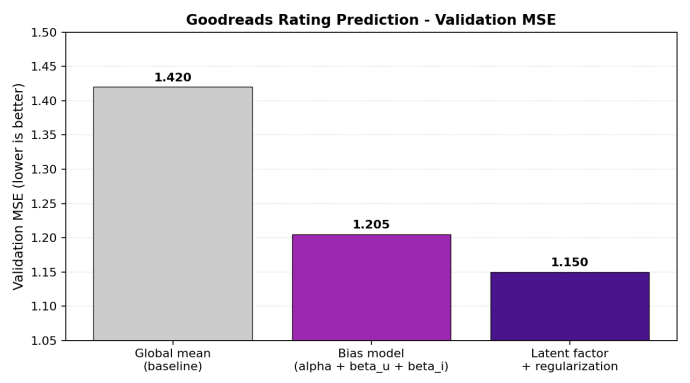
# Task 2 — Rating Prediction

# Rating Prediction — Biased Latent-Factor (From Scratch)

**Estimator.** $\hat{r}(u,i) = \alpha + \beta_u + \beta_i$. **Closed-form coordinate descent.**

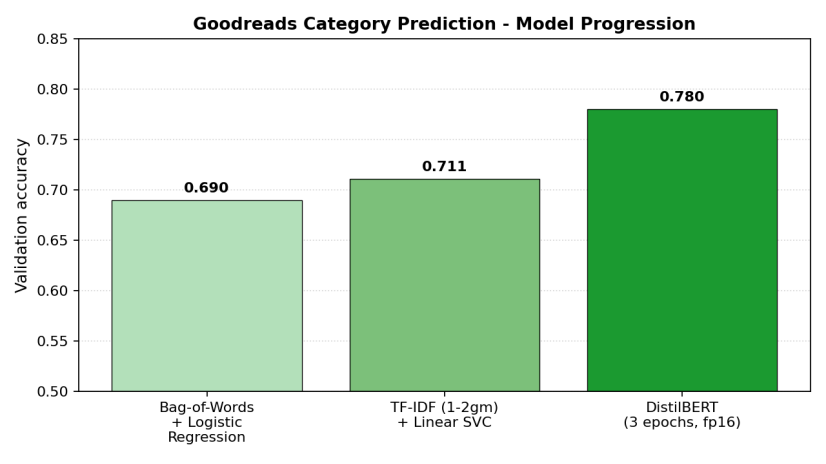
$$\beta_u \leftarrow \frac{\sum_{(i,r)\in D_u}(r - \alpha - \beta_i)}{\lambda + |D_u|}, \quad \beta_i \leftarrow \frac{\sum_{(u,r)\in D_i}(r - \alpha - \beta_u)}{\lambda + |D_i|}.$$

$\lambda = 4$, 50 iterations, train/val 90/10 with seed 4.



# Task 3 — Category Prediction

# Category Prediction — Three Iterations of Modelling




| Model | Key Choices | Val. Acc |
|---|---|---|
| BoW + Logistic Reg. | Top-10K vocab, lowercased, $C = 1$ | 0.69 |
| TF-IDF + Linear SVC | 1–2-grams, 150K features, sublinear TF, $C = 0.5$ | 0.711 |

# Why DistilBERT Won

- Captures *contextual* word meaning (e.g. “magic” near “school” vs. near “murder”) that BoW and TF-IDF cannot.
- Pretrained on web text $\rightarrow$ massive head start over training a 5-class classifier on $\sim$10K labelled reviews.
- Mixed-precision (`fp16=True`) cut training time roughly in half on Colab T4 without measurable accuracy loss.

# Final Independent Project — Food.com

# Final Independent Project — Food.com Recipe Rating Prediction

**Dataset.** Kaggle Food.com (700K+ user–recipe interactions, joined with recipe metadata). 30 % sample after cleaning (filter $\geq 3$ steps, 4–20 ingredients, drop nulls); 80/10/10 stratified split.

**Sequenced models:**

1. **Bias model:** closed-form coord. descent, $\lambda = 5$, 1,000 iters.
2. **Latent factor:** adds $\gamma_u^\top \gamma_i$. Captures hidden interactions.
3. **Sentiment + LF hybrid:** `review_text` sentiment as auxiliary feature to mitigate cold-start.

**EDA insight.** All raw feature–rating correlations are weak ($|r| \leq 0.04$), which is exactly why a bias / latent-factor approach beats a regression on raw features.

# Closing

# Quantified Impact

- Top **22%** of **1,600** on the Mercury MLE leaderboard.
- Category accuracy: **0.69** $\rightarrow$ **0.711** $\rightarrow$ **0.78** (BoW $\rightarrow$ TF-IDF $\rightarrow$ DistilBERT).
- From-scratch biased latent-factor models for both Goodreads ratings and Food.com recipe ratings, with closed-form coordinate-descent updates and $L_2$ regularisation tuning.
- Final independent project on Food.com extended the Goodreads rating model with sentiment features to mitigate cold start.

## Model-and-iteration, end to end.

Baseline $\rightarrow$ similarity $\rightarrow$ latent factor $\rightarrow$ deep model; each step justified by the previous step's failure mode.

# Thank you.

## Questions?