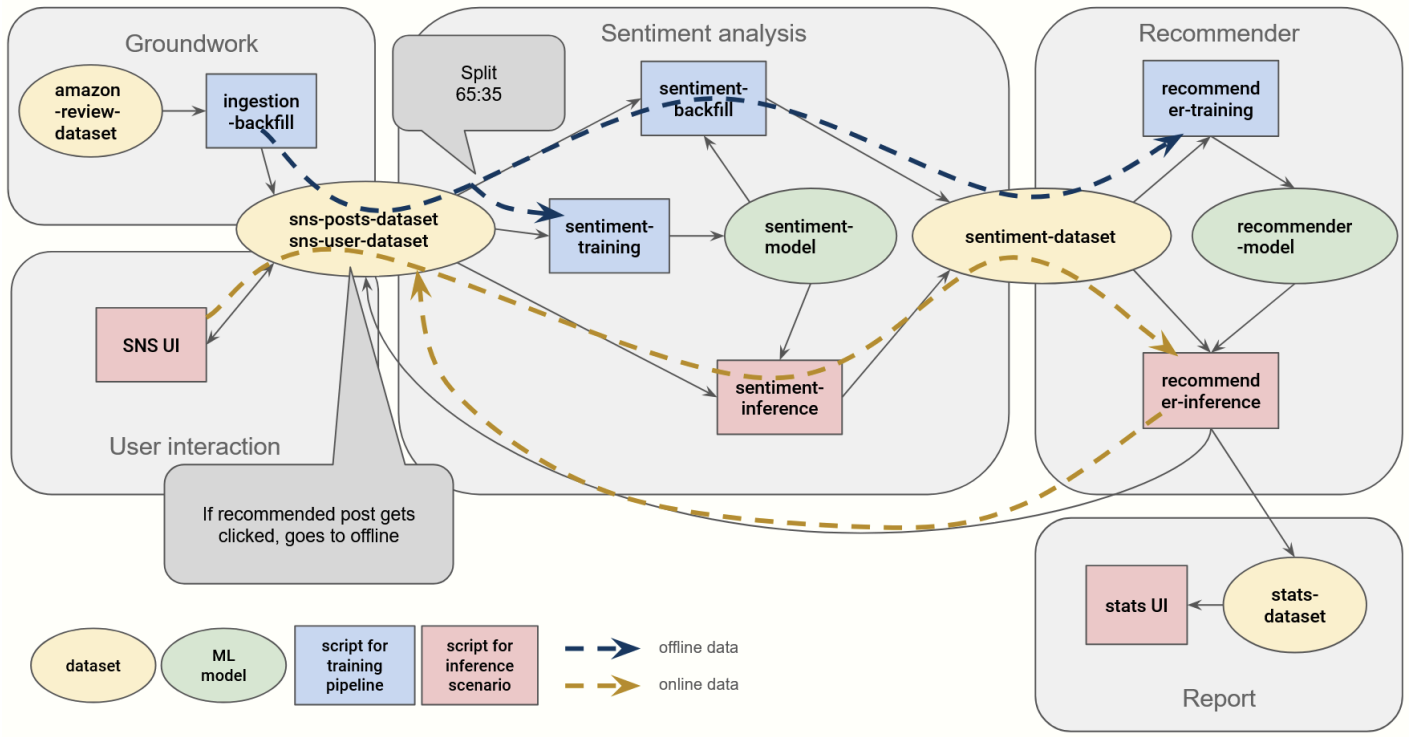
CS 410 FA24: How the software is implemented

Ad recommendation with sentiment analysis
Junyoung Lee(jl298, Coordinator)

System overview



1. Data Generation and Preparation [Groundwork]
    - `ingestion_backfill.py`: Convert review data to SNS format and split into sentiment/recommender datasets

2. Sentiment analysisModel Training [Sentiment analysis] [Recommender]
    - `sentiment_training.py`: DistilBERT-based 3-class sentiment classification training
    - `recommender_training.py`: Matrix Factorization-based recommendation system training

3. Real-time Processing [User interaction] [Sentiment analysis] [Recommender]
    - `sns.py`: Manages user interactions, including browsing, posting, and commenting on products. Additionally, oversees product catalog updates and real-time synchronization with user activities.
    - `sentiment_inference.py`:
        - --backfill option: Process all historical data
        - No option: Real-time sentiment analysis for new comment on a SNS post

    - `recommender_inference.py`: Sentiment-based product recommendations

4. Performance Monitoring [Report]
    - `stats.py`: Analyze and visualize overall system performance
        - Sentiment analysis accuracy
        - Recommender system metrics (NDCG, Precision, Recall, etc.)
        - Gold standard dataset analysis for validating model accuracy and benchmarking system performance against predefined metrics.
        - Time-based performance trends
        - User satisfaction analysis

Dataset ingestion

In this project, I've designed the system and tested with Amazon product review dataset to create SNS advertising posts. Each unique product from the review data is converted into a single SNS advertisement post, where the product's review text becomes post comments, and the product ratings (ranging from 1 to 5) are transformed into user reactions.

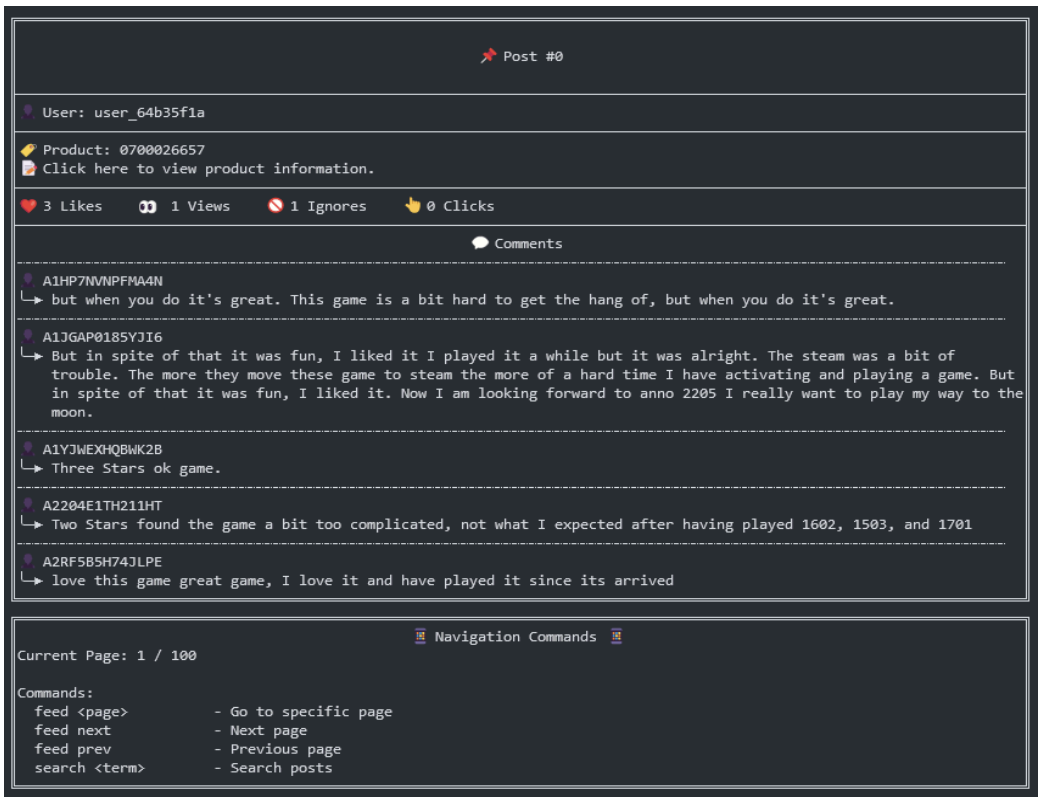
Ratings of 4 and 5 are converted to "Like" button clicks, a rating of 3 indicates the user viewed the product, and ratings of 1 and 2 are translated to ignored reactions. By running the ingestion_backfill.py script, the Amazon review dataset from the specified location is converted into an SNS post dataset, which can then be viewed through the SNS UI.


```
{"overall": 5.0, "verified": true, "reviewTime": "10 17, 2015", "reviewerID": "A1HP7NVNPFMA4N", "asin": "0700026657", "reviewerName": "Ambrosia075", "reviewText": "This game is a bit hard to get the hang of, but when you do it's great.", "summary": "but when you do it's great.", "unixReviewTime": 1445040000}
```

```
{"overall": 4.0, "verified": false, "reviewTime": "07 27, 2015", "reviewerID": "A1JGAP0185YJI6", "asin": "0700026657", "reviewerName": "travis", "reviewText": "I played it a while but it was alright. The steam was a bit of trouble. The more they move these game to steam the more of a hard time I have activating and playing a game. But in spite of that it was fun, I liked it. Now I am looking forward to anno 2205 I really want to play my way to the moon.", "summary": "But in spite of that it was fun, I liked it", "unixReviewTime": 1437955200}
```

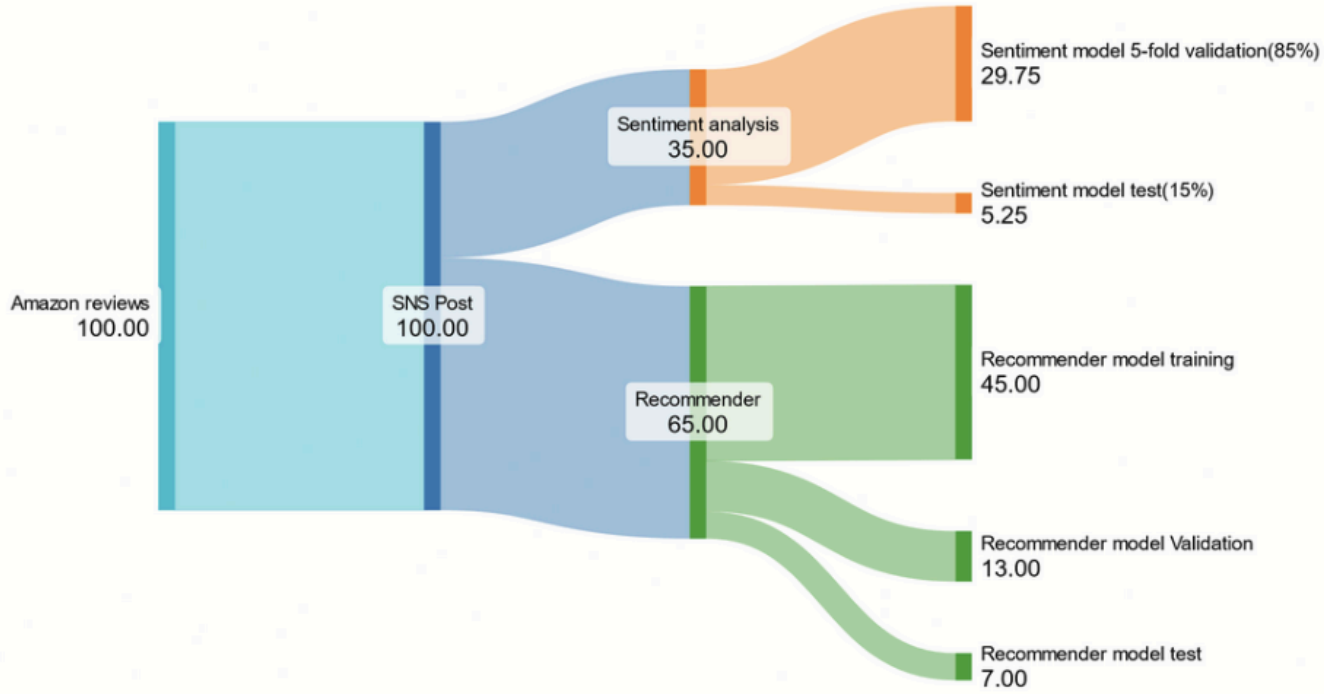
```
{"overall": 3.0, "verified": true, "reviewTime": "02 23, 2015", "reviewerID": "A1YJWEXHQBWK2B", "asin": "0700026657", "reviewerName": "Vincent G. Mezera", "reviewText": "ok game.", "summary": "Three Stars", "unixReviewTime": 1424649600}
```


For example, the data entries above are converted into comments on a post in the SNS UI, with each post being created based on a unique ASIN (Amazon's product ID), as shown in the screenshot below.



Data partitioning

The dataset ingestion process starts with converting Amazon reviews into SNS posts, splitting them into sentiment analysis (35%) and recommender system (65%) datasets. The diagram visually represents these proportions and data processing stages for better understanding.



1. Sentiment Analysis (35 %)
    - Split into validation (29.75% of total, 85% of Sentiment analysis)
    - Test set (5.25% of total, 15% of Sentiment analysis)
2. Recommender System (65 %)
    - Training (45 % of total)
    - Validation (13% of total)
    - Test set (7% of total)

The width of each flow represents the relative proportion of data being processed at each stage, showing how the initial dataset is systematically divided for different modeling purposes.

Sentiment Analysis details

The code implements a comprehensive sentiment analysis system for analyzing social media comments using the DistilBERT model. The system consists of two main components: training (`sentiment_training.py`) and inference (`sentiment_inference.py`).

The training component focuses on building a robust sentiment classification model

- It uses DistilBERT, a lighter and faster version of BERT, pre-trained on general text data
- The model is fine-tuned to classify comments into three categories: 'ignored' (0), 'viewed' (1), and 'liked' (2)
- Implements 5-fold cross-validation to ensure model robustness and prevent overfitting
- Uses a batch size of 16 and trains for 3 epochs in each fold
- Employs macro F1 score as the primary metric for model evaluation
- Includes warmup steps and weight decay for optimization
- Saves the best performing model based on validation performance
- Finally trains a model on the complete training dataset for deployment

Key features of the training process

- Stratified train-test split (85% training/validation, 15% test)
- Tokenization with max sequence length of 512
- Detailed logging of performance metrics for each fold
- Comprehensive evaluation including accuracy and F1 scores for each class
- Test results are saved alongside the model for future reference

The inference component provides flexible deployment options

1. Backfill Mode
    - Processes all existing offline comments in bulk
    - Excludes the most recent comment to maintain consistency
    - Creates a new sentiment dataset from scratch
2. Single User Mode
    - Processes only the most recent comment from a specific user
    - Appends results to existing sentiment dataset
    - Useful for real-time sentiment analysis of new comments

The inference process includes

- Efficient batch processing of comments
- Converts model predictions into human-readable sentiment labels ('good', 'moderate', 'bad') using softmax probabilities to calculate confidence scores
- Confidence score calculation using softmax probabilities
- Detailed result logging and error handling
- Records a timestamp for each inference to ensure chronological consistency in data processing and facilitate error traceability
- JSON output format for easy integration with other systems

Both components include robust error handling and logging

- Detailed progress tracking and error messages
- Input validation and data integrity checks
- Performance metrics logging
- Exception handling for model loading and prediction
- Resource cleanup and proper file handling

The system is designed for both batch processing and real-time analysis, making it suitable for various use cases in social media sentiment analysis. The modular design allows for easy updates and maintenance, while the comprehensive logging system helps in monitoring and debugging the analysis process.

Recommender System details

The code implements a hybrid recommender system that combines collaborative filtering using matrix factorization with sentiment analysis scores. The system consists of two main components: training (`recommender_training.py`) and inference (`recommender_inference.py`).

The training component focuses on building a recommendation model

1. Score Calculation

    - Combines reaction scores (liked=1.0, viewed=0.5, ignored=0.0) and sentiment scores (good=1.0, moderate=0.5, bad=0.0)
    - Uses weights: 70% for reaction and 30% for sentiment
    - Adds a consistency bonus (0.1) when reaction and sentiment align
    - Final scores are normalized between 0 and 1

2. Matrix Factorization

    - Creates a user-product interaction matrix from the combined scores
    - Applies Singular Value Decomposition (SVD) using scipy's svds
    - Uses 50 latent factors by default
    - Implements train-test split (85 %/15 %) for evaluation
    - Scales predictions using MinMaxScaler

3. Evaluation Metrics

    - RMSE (Root Mean Square Error)
    - Precision@K and Recall@K
    - NDCG@K (Normalized Discounted Cumulative Gain)

- Various distribution and correlation metrics

The inference component provides multiple recomendation strategies

1. Primary Strategy (Matrix Factorization)

- Creates a user vector from their interaction history
- Projects the vector into latent space using trained SVD components
- Generates predictions for unrated products
- Ranks products by predicted scores

2. Fallback Strategies

- Similar Products
  - Activates when MF recommendations are insufficient
  - Finds similar products using latent factor cosine similarity
  - Uses product vectors from SVD's Vt matrix
- Popularity-based
  - Final fallback when user has no history
  - Recommends products based on average scores across all users

Key Features

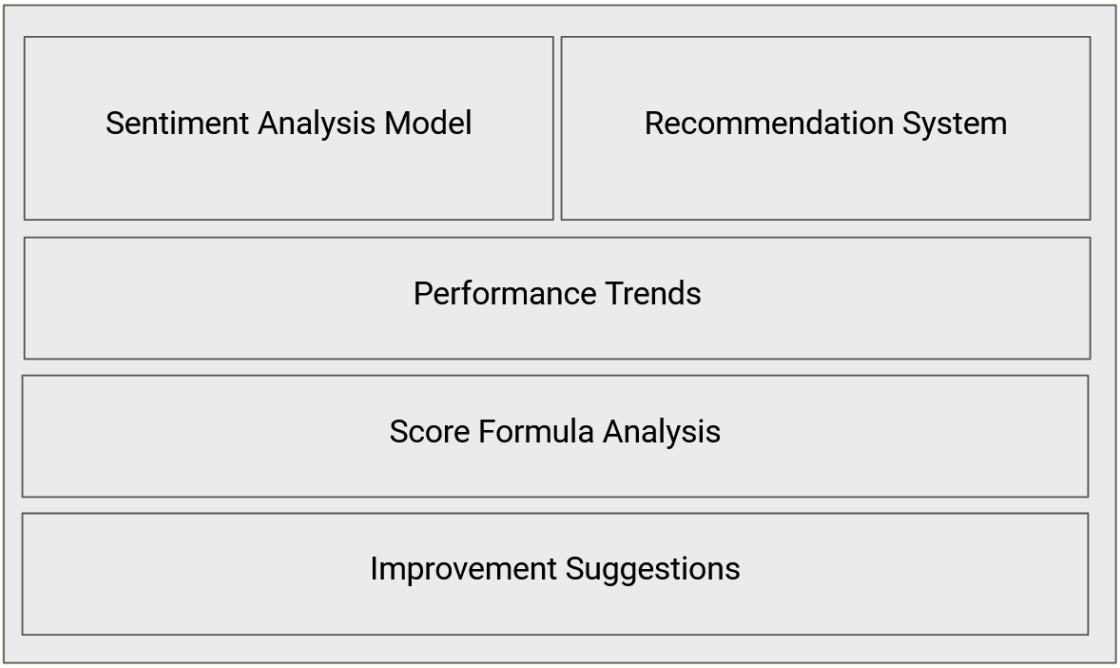
- Mitigates cold-start problems by employing fallback strategies, such as recommending popular products or finding similar items based on cosine similarity in latent space
- Incorporates both explicit (reactions) and implicit (sentiment) feedback
- Generates confidence scores for recommendations by calculating a weighted sum of reaction and sentiment alignment, with an additional consistency bonus for aligned scores
- Maintains product catalog integration
- Implements efficient batch processing
- Includes detailed logging and error handling
- Supports real-time recommendations for new interactions

The system generates recommendations by

1. Loading the latest user interaction
2. Converting sentiment and reaction to a combined score
3. Creating a user preference vector
4. Generating primary recommendations using matrix factorization
5. Falling back to similarity or popularity recommendations if needed
6. Saving recommendations with confidence scores and metadata

Stats UI

The box below shows the UI layout.



Sentiment Analysis Model Section

- Accuracy Gauge
    - Circular gauge visualization showing overall model accuracy from 0 to 1
    - Features a blue progress arc and centered numerical display
    - Displays both percentage and absolute values for precise reading
    - Includes title and current value with 1 decimal place precision
- Class-wise F1 Scores
    - Bar visualization for three sentiment classes (ignored, viewed, liked)
    - Each class shows
        - Progress bar with blue fill indicating F1 score
        - Class name label
        - Exact F1 score displayed as percentage with 1 decimal precision
    - Allows direct comparison between class performances
- Class Balance Score
    - Measures how evenly the model performs across all classes
    - Calculated as: 1 - (sum of deviations from ideal 0.33 distribution) / 2
    - Displayed as circular gauge similar to accuracy
    - Higher values (closer to 1) indicate better balance between classes

Recommender System Section

- Key Metrics
    - RMSE
        - Card display with icon and title
        - Value shown as percentage with 1 decimal place
        - Lower values indicate better prediction accuracy

        - NDCG@10
            - Card display with icon and title
            - Value shown as percentage with 1 decimal place
            - Measures ranking quality of top 10 recommendations
- Precision-Recall Trade-off
    - Interactive scatter plot with
        - X-axis: Recall@10 (0 to 1)
        - Y-axis: Precision@10 (0 to 1)
        - Single point showing current model's position
        - Hover tooltip showing exact values
        - Axis labels and grid lines for readability
- Item Coverage
    - Circular gauge showing proportion of total items being recommended
    - Calculated as: min(10 / total_items, 1.0)
    - Blue progress arc with centered percentage display
    - Indicates how well the system explores the full item catalog

Performance Trends section

- Time series graphs showing model performance changes
- Sentiment Analysis Trends
    - Line chart showing temporal performance changes
    - Two metrics plotted
        - Accuracy (blue line)
        - Macro F1 (red line)
    - Features
        - X-axis: Time periods
        - Y-axis: Score values (0.6 to 1.0 range)
        - Interactive tooltip
        - Legend for metric identification
- Recommender System Trends
    - Line chart tracking recommender performance
    - Two metrics plotted
        - NDCG@10 (blue line)
        - F1@10 (red line)
    - Features
        - X-axis: Time periods
        - Y-axis: Score values (0.6 to 1.0 range)
        - Interactive tooltip
        - Legend for metric identification

Score Formula Analysis section

- Score Distribution
    - Distribution metrics display
        - Entropy gauge showing score diversity
        - Skewness value with 2 decimal precision
        - Kurtosis value with 2 decimal precision
    - Warning system

            - Alerts for entropy < log(5)
            - Flags for skewness > 1
            - Warnings for kurtosis > 3
- Component Contribution
    - Interactive pie chart showing
        - Reaction weight (70%)
        - Sentiment weight (20%)
        - Consistency bonus (10%)
    - Features
        - Color-coded segments
        - Hover tooltips with exact values
        - Warning for components > 60% contribution
- Component Correlation
    - Bar chart showing correlation coefficients between
        - Reaction-Sentiment
        - Reaction-Consistency
        - Sentiment-Consistency
    - Features
        - Value labels above bars
        - 0-1 scale on Y-axis
        - Warning for correlations > 0.8
- Consistency Metrics
    - Bar chart displaying
        - Overall consistency rate
        - Good-Liked alignment rate
        - Bad-Ignored alignment rate
    - Features
        - Value labels
        - 0-1 scale
        - Warning for rates < 0.3

Improvement Suggestions section

- Automated analysis system checking:
    - Score distribution concentration: Evaluate the concentration of score distribution by calculating metrics such as entropy and kurtosis. Highlight abnormally skewed or concentrated distributions with warning messages
    - High correlation between components: Identify and monitor correlations, such as reaction-sentiment or reaction-consistency, to ensure balanced contributions and avoid over-reliance on a single factor
    - Consistency rates for alignment issues
- Generates specific improvement recommendations based on:
    - Entropy thresholds
    - Correlation thresholds
    - Consistency rate minimums
- Displays recommendations as info alerts with actionable suggestions

Dataset schema

amazon-review-dataset

Outsourced Amazon review dataset

- reviewerID - ID of the reviewer, e.g. A2SUAM1J3GNN3B
- asin - ID of the product, e.g. 0000013714
- reviewerName - name of the reviewer
- vote - helpful votes of the review
- style - a dictionary of the product metadata, e.g., "Format" is "Hardcover"
- reviewText - text of the review
- overall - rating of the product
- summary - summary of the review
- unixReviewTime - time of the review (unix time)
- reviewTime - time of the review (raw)
- image - images that users post after they have received the product

sns-posts-dataset

You can think of a post as an advertisement for one ASIN of aws-review-dataset

- post_id: sequence ID
- user_id: a hashed value with ASIN; NOT a reviewer of amazon-review-dataset.
- product_id: asin of amazon-review-dataset
- post_text: summary and reviewText of amazon-review-dataset
- like_count: the number of comment whose reaction is ‘liked’
- view_count: the number of comment whose reaction is ‘viewed’
- ignore_count: the number of comment whose reaction is ‘ignored’
- comments: comment list; comment attribute is as follows. Basically, One comment is a review entry in the amazon-review-dataset for the product_id of this post.
    - comment_id: sequence ID
    - user_id: reviewerID of amazon-review-dataset
    - comment_text: reviewText of amazon-review-dataset
    - reaction: ‘liked’ if overall is 4 or 5, ‘viewed’ if 3 or 2, ‘ignored’ if 1 or 0

sns-user-dataset

One random reviewer of amazon-review-dataset and the reviewer’s stats of overall distribution, generated when ingestion-backfill is executed.

- user_id: reviewerID of amazon-review-dataset
- user_name: reviwerName of amazon-review-dataset
- liked_list: List of ASINs with overall value of 5 or 4 from this user.
- viewed_list: List of ASINs with overall value of 3 or 2 from this user.
- ignored_list: List of ASINs with overall value of 1 or 0 from this user.
- clicked_list: List of ASINs of posts this user clicked on in the SNS UI

sentiment-dataset

- user_id: the user_id of sns-user-dataset
- product_id: the product_id of sns-posts-dataset
- sentiment_label: {good|moderate|bad} determined by sentiment inference step.

- reaction: {liked|viewed|ignored}, indicating the user's reaction to the post, derived from the corresponding SNS comment. For example, 'liked' corresponds to a product rating of 4 or 5 in the source dataset

Dataflow - Logical View

Big picture

- Comments will help recommendations!
- New comment on a sns post ⇒ sentiment_lable ⇒ ad recommendation ⇒ new sns post

Background pipeline

1. ingestion-backfill: converts the outsourced amazon_review_dataset to sns_dataset for internal use.
2. sentiment-backfill: applies the sentiment analysis on existing sns_dataset and generates sentiment-dataset
3. sentiment-training: trains sentiment analysis model
4. recommender-training: trains recommender model

While presentation

1. SNS: add a new comment that will be stored as an entry of sns_dataset
2. sentiment-inference: script will read the new comment and add a new sentiment_label to sentiment_dataset
3. recommender-inference: read the new sentiment_label and output a recommendation

Dataflow - Physical view

SNS UI: `sns.py`

Input	Output

`dataset/Video_Games_demo.json`	`dataset/sns-user-dataset.json`
	`dataset/sns-posts-dataset.json`
	`dataset/sns-posts-dataset.sentiment.json`
	`dataset/sns-posts-dataset.recommender.json`

Sentiment Analysis: `sentiment-training.py`

Input	Output
`dataset/sns-posts-dataset.sentiment.json`	`model/sns_sentiment_model`

Sentiment Analysis: `sentiment-backfill.py`

Input	Output

`model/sns_sentiment_model` `dataset/sns-posts-dataset.recommender.json`	`dataset/sentiment-dataset.json (overwrite)`

Sentiment Analysis(Inference part): `sentiment-inference.py`

Input	Output
`model/sns_sentiment_model` `dataset/sns-posts-dataset.json`	`dataset/sentiment-dataset.json (append)`

Recommender(Training part): `recommender-training.py`

Input	Output
`dataset/sentiment-dataset.json`	`model/recommender_model`

Recommender(Inference part): `recommender-inference.py`

Input	Output
`model/recommender_model` `dataset/sentiment-dataset.json`	`dataset/recommendations-dataset.json`