COMP9727 Project Design

Raymond Li
COMP9727
UNSW Computer Science and Engineering
Kensington, NSW, Australia
z5360323@ad.unsw.edu.au

***Abstract*—This project presents the design and evaluation plan for an e-commerce recommender system that integrates social trust, content semantics, and sentiment analysis. Using the Epinions and Ciao datasets—which provide user–item ratings, text reviews, and directed trust graphs—we implement and compare four approaches: content-based, trust-aware (TrustSVD), a basic weighted-sum based hybrid model, and a more advanced and integrated hybrid (MR3). Recommendations are delivered via a 5×3 panel UI that highlights products endorsed by a user's trustees. System performance will be assessed using top-N metrics (F1@N, HR@N), user-centric KPIs (CTR, conversion, diversity), and insights from a controlled user study.**

***Index Terms*—product-recommendation, trust-aware, social-aware, collaborative filtering, content-based, hybrid, sentiment-aware, e-commerce.**

I. Scope

A. Recommender Brief

A product recommendation system is proposed for a new E-commerce website. This new E-commerce business differentiates itself from existing merchants such as *Amazon* and *Ebay* through the integration of a social, trust-based network. The team is hoping that the addition of a social network will influence users' purchase decisions, as well as provide data to drive recommendation mediums such as trust-aware systems that are unavailable to traditional E-commerce merchants. Our technical team believes that the additional data will allow the business to provide more accurate recommendations faster, thus securing sales and boosting revenue. [1]

The users will be presented with a panel-based UI that displays products according to their rank - that is, the items identified as most likely to be purchased by the user will be presented first. The panel-based UI will be reminiscent of existing merchant UI, presenting an image of the product, the price, the name of the product, and aggregate over other users' ratings. We will enhance this UI further by highlighting when a product is recommended by the user's trustees. Each page will consist of 15 panels arranged in a 5 x 3 grid, each containing one item.

B. System Mechanics

The technical team recognises that such a system may face challenges when the platform is relatively new, as user data, including their social networks, may be sparse. When this is the case, the recommender system will be able to fall back hidden factors and topics (HFT) discovered in content-based data. For a cold start, this method is more effective than purely content-based recommendation systems. [2] Our proposed hybrid model will have access to all information, and thus be able to leverage the better cold start properties of content-based data. [3]

The deployed model will retrained periodically as new data is available, and target to predict the next period's interactions.

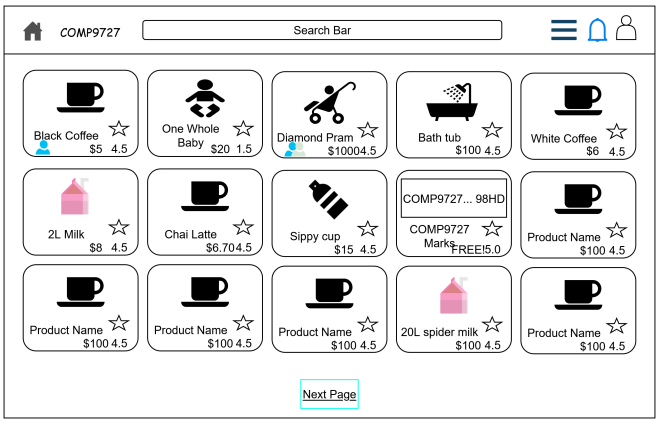

Fig. 1. UI mock-up of the main page. 15 items will be presented to the user in a 5x3 grid format. Key information is presented to the user.

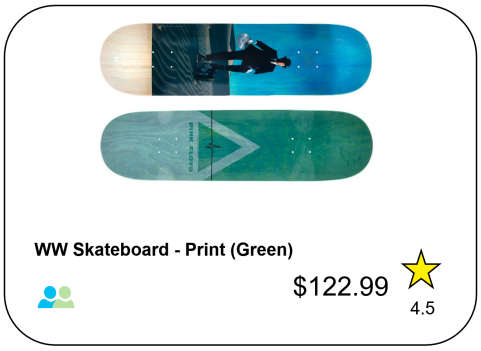

Fig. 2. Close-up of an item panel. Includes key information such as an image of the product, the product name, the price, the rating (given by the star) and also which trusted users recommend the product (where avatars of trusted users are displayed).

C. Business Risk Management

In order to assure shareholders that our recommender system (III) will work well prior to deployment, our team will

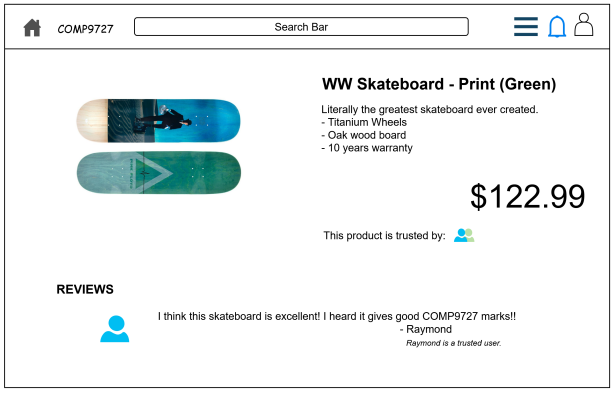

Fig. 3. Clicking into a panel will give additional information, including a more detailed description of the product, as well as a review space where the user is presented with product reviews and may leave a review of their own.

initially test our methodology on similar datasets - namely the product review datasets Ciao and Epinions which both feature not only product ratings and reviews, but also a relevant social network (II). [4] [5] Our recommender system will then be thoroughly evaluated against relevant metrics (IV) and user studies to ensure performance targets are met. A user trial will also be conducted to further validate the model.

II. Dataset

This project proposes to use two datasets:
(i) *Ciao*
(https://www.kaggle.com/datasets/aravindaraman/ciao-data?select=rating.txt)
and
(ii) *Epinions*
(https://www.kaggle.com/datasets/pypiahmad/social-recommendation-data).
These datasets were selected due to their relevance and similarity to our site's expected data attributes.

A. Datset: Epinions

1) Dataset Overview: Epinions was a popular product review site that has been scraped. *Epinions* allowed users to rate products and add other members to their trust networks.

The *Epinions* dataset details the ratings users gave to items, with metadata including (i) user, (ii) item name, (ii) price paid, (iii) time of review, (iv) rating (in stars) and (V) a text-based review. This enables the usage of content-based and sentiment-based analysis. Additionally, the presence of timestamps allows back-testing on temporally informed purchase decisions.

An additional directed graph is provided to map out the trust-based relations between users. This information is core to building and testing a trust-aware recommender system.

This dataset is available on *Kaggle*, which provides easy access, and the entire raw dataset is available with no predefined train-test split.

TABLE I
Statistics of the *Epinions* Dataset

Statistic	Value
Number of Users	116,260
Number of Items	41,269
Number of Ratings	181,394
Rating Density	0.0038%
Social Relations	181,304
Social Density	0.0027%

2) Dataset Limitations: The *Epinions* dataset exists in many forms, this version found on Kaggle was chosen due to its rich metadata which also includes the word-based reviews given by users. [5] [6] Unfortunately, this dataset seems to be less comprehensive when compared to other scrapings of the *Epinions* website. For example, this dataset only consists of 181,394 ratings whilst other scrapings have more than 1 million ratings. As this dataset seems to only be a small subset of the total data generated by the site, both the rating and social density are quite low.

In addition, although time information is available for the users' reviews, the time at which users made 'trust' connections is not available. This limits its usefulness as a temporally aware back-tester. The time stamps are also only available on product reviews, and not when users actually bought the item.

B. Dataset: Ciao

1) Dataset Overview: Ciao was an European online-shopping portal launched in February 2008. It also hosted a forum where users could rate and discuss products. Users could also rate other ratings - from not helpful to very helpful. In a similar fashion to *Epinions*, *Ciao* allowed its users to form their own trust networks.

Each review in the *Ciao* dataset included metadata for: (i) user, (ii) product, (iii) category, (iv) rating, (v) helpfulness, (vi) time of review and (vii) a text-based review. This setup supports both content- and sentiment-based analysis, and the inclusion of timestamps makes it possible to back-test purchase decisions with temporal context.

The *Ciao* dataset also included a directed graph representing the trust network between users.

As with *Epinions*, a version on *Kaggle* was selected which provides for easy access. The entire raw dataset is available, with no predefined train-test split.

In comparison with the *Epinions* dataset, the *Ciao* dataset is much larger subset of the site's overall data. This is reflected in the greater rating and social density present in the dataset.

2) Dataset Limitations: Like the *Epinions* dataset, a critical flaw is the lack of data regarding *when* users formed their trust-based interactions. This makes it difficult to utilise the dataset in a temporally aware manner whilst incorporating trust information. The time stamps are also only available on product reviews, and not when users actually bought the item.

TABLE II
STATISTICS OF THE *Ciao* DATASET

Statistic	Value
Number of Users	10,877
Number of Items	103,889
Number of Ratings	268,907
Rating Density	0.024%
Social Relations	129,839
Social Density	0.22%

III. METHODS

It is proposed to construct a content-based, sentiment-aware recommendation system (III-A) and a trust-aware recommender system (III-B) as baselines. Then, two hybrid recommender systems combining the two will be constructed (III-C, III-D), which is hypothesised to perform better than either baselines.

There is a temporal gap between when a user *buys* an item, compared to the time they *review* it. As information is not available in the dataset regarding the time of purchase, we will simply assume that the time order that reviews occur roughly corresponds to the order that the items would have been purchased anyways, and thus will approximate review time to item purchase time.

To handle the issue that there is no temporal data for trust networks, we will assume that only the users who have made a review prior to the training cut-off period T_c are part of the trust network, and will be given the relevant directed edges to all other users who satisfy the criteria. Although this is not perfect, it does allow us to also simulate sparse social networks semi-realistically.

A. Content-based Recommender

Content-based recommenders are a suitable application, given that the product name and reviews gives information on not only the product, but also the semantic buyer opinions. However, given the opportunities and limitations of our dataset, namely that there is additional information available via social circles and that extended descriptions of the items are not provided, it is believed that this approach will *work* with the dataset, but not be the best performing method.

To accomplish this, the following pipeline will be used:

(i) Pre-process the data through methods such as removing stop words, lemmatisation and tokenisation.
(ii) Utilise word2vec to vectorise the combined corpus of product name and reviews. This method is superior to other methods such as TF-IDF in our application as it can better encode semantic and syntactical regularities. [7]
(iii) Aggregate word2vec embeddings using TF-IDF-weight into a single *d*-dimensional vector for each product name.
(iv) Utilise K-means clustering to determine products in similar categories in an unsupervised manner (as this data is not available in the dataset). This will be run on the product name corpus only.
(v) Utilise similarity indexes (such as cosine similarity) to determine similar items.
(vi) Recommend a top-N (where N = 15 per page) mix of items that are in a similar (v) or similar in category (iv).

To tune the mix between K-means and similarity indexes, the following formula will be used.

$$\begin{aligned}\text{score}(i) = \alpha\Big(1 - \tfrac{\text{dist}_{\cos}(u,i)}{2}\Big) \\ + (1-\alpha)\big[\text{cluster}(u) = \text{cluster}(i)\big]\end{aligned}$$

where α is a tuneable parameter that can be used to balance the likelihood of showing a similar product, or one from a similar cluster whilst still effectively handling the case where an item satisfies both categories.

We expect for this method to form part of a hybrid recommender, as well as serving as a useful baseline.

B. Trust-aware/Social-aware Recommender

To leverage the trust networks between users, a trust-aware recommender system based on collaborative filtering (CF) will also be used. A reliable and thoroughly researched method *TrustSVD* is suitable for this task, which extends the popular SVD++ matrix factorisation method to trust-aware systems. This approach is especially advantageous as it also offers a degree of performance even on "cold starts" and is able to operate on sparse trust networks, both of which are issues we face especially in the *Epinions* dataset. [8]

TrustSVD will be utilised with the following pipeline.

(i) Gather inputs: rating matrix R, trust network T, implicit feedback F (purchase/review time-stamped logs).
(ii) Pre-process data by normalising rating matrix R and building fast lookup data structures.
(iii) Initialise model parameters, including latent factors P_u, Q_i representing users and item factor matrices, item implicit-feedback vectors Y_i, explicit and implicit trust vectors X_v, Z_v and bias terms.
(iv) Train the SVD by using stochastic gradient descent (SGD) to minimise the loss function

$$\min_{\Theta} \mathcal{L}(\Theta) = \sum_{(u,i)\in\mathcal{K}} \big(r_{u,i} - \hat{r}_{u,i}\big)^2 + \lambda \|\Theta\|^2,$$

where

$$\begin{aligned}\hat{r}_{u,i} = \mu + b_u + b_i \\ + Q_i^\top \Big(P_u + |N_u|^{-\frac{1}{2}} \sum_{j\in N_u} Y_j\Big) \\ + Q_i^\top \sum_{v\in T(u)} t_{u,v} X_v \\ + Q_i^\top \Big(|T(u)|^{-\frac{1}{2}} \sum_{v\in T(u)} Z_v\Big)\end{aligned}$$

and

$$\Theta = \{P, Q, Y, X, Z, b_u, b_i\},$$

representing the latent factors.

(v) Generate recommended items by predicting $\hat{r}_{u,i}$ for each user, and recommending top-N (where N = 15) items per page.

This trust-aware system will form another important baseline, this time from the perspective of leveraging social networks. To tune the model, a grid search can be run on model parameters.

C. Basic Hybrid Recommender

Building on the content-based recommender (refcontent) and the trust-aware recommender (III-B), we construct a hybrid model that fuses semantic item similarity with social influence. This model will rank items based on a weighted sum of content-based and trust-aware recommender scores, incorporating information from both sources.

The following pipeline can implement this basic hybrid recommender:

(i) Collect rating $\hat{r}_c(u,i)$ from content-based recommender III-A.

(ii) Collect rating $\hat{r}_t(u,i)$ from trust-aware recommender III-B.

(iii) Create new rating $\hat{r}_h(u,i)$ through a weighted sum between $\hat{r}_c(u,i)$ and $\hat{r}_t(u,i)$.

$$\hat{r}_h(u,i) = \alpha(\hat{r}_c(u,i)) + (1-\alpha)(\hat{r}_t(u,i))$$

where α is a tuneable parameter.

This method presents a simple way of fusing information from two different sources, but it is believed that a tighter integration of information at the training level can yield better performance, which we present in III-D.

D. Integrated Hybrid Recommender

Another method of constructing a hybrid recommender is to integrate both content and social features into a CF factorisation matrix. To do this, we will leverage *Hu*'s *MR3* CF model which captures both the latent semantics of product names and the collaborative power of user trust networks. This approach more closely integrates different information aspects in the training process, which would improve recommendations especially in sparse or cold-start scenarios where it can fall back on the integrated LDA information. [3]

The pipeline to implement the hybrid method will be the following:

(i) Gather inputs: Rating set R, trust network T, review corpus D and product names N.

(ii) Pre-process data:
- Clean text (lowercasing, stop words, lemmatise, tokenise),
- normalise ratings,
- build fast lookup structures for the input data.

(iii) Fit Latent Dirichlet Allocation (LDA) model over combined corpus to uncover hidden factors and topics (HFT) in the textual data. [1]

(iv) Initalise MR3 parameters (latent parameters)
- User factors P and item factors Q
- content map W
- bias terms $b_u,\ b_i$

(v) We learn parameters $\Theta = \{P,\, Q,\, W,\, b_u,\, b_i\}$ by minimising the loss function

$$\begin{aligned}\mathcal{L}(\Theta) = &\sum_{(u,i)\in R} \left(\tilde{r}_{u,i} - P_u^\top Q_i - b_u - b_i\right)^2 \\ &+ \alpha \sum_{(u,v)\in T} \|P_u - P_v\|^2 \\ &+ \beta \sum_i \|Q_i - W\,\theta_i\|^2 \\ &+ \lambda \|\Theta\|^2 .\end{aligned}$$

where:
- $r_{u,i}$ is the observed rating given by user u to item i.
- $\tilde{r}_{u,i} = r_{u,i} - \mu$ is the rating centered by the global mean μ.
- $R = \{(u,i)\}$ indexes all observed ratings, and $T = \{(u,v)\}$ indexes all trust relations.
- $\theta_i \in \mathbb{R}^K$ is the LDA topic distribution for item i.
- α, β, λ are regularisation hyperparameters.

Optimisation is performed via SGD.

(vi) Generate recommended items by predicting $\hat{r}_{u,i}$ for each user, and recommending top-N (where N = 15) items per page.

The hypothesis is that this hybrid method, with access to more information, will outperform both baseline methods and thus provide a case for making a social network integrated E-commerce platform. To tune the model, a grid search can be run on model parameters.

IV. Evaluation

A. Temporal Train-Test Split

To ensure an accurate reflection of user behaviour, training and testing data will be temporally dependent. A rolling window evaluation technique will be used to determine the train-test split. That is, training data will begin from $t = 0$ to a train-cut time $t_{train} = 0 \ldots T_c$. Then, a fixed rolling window will be used to determine the test data $t_{test} = T_c \ldots T_c + T_w$. This method allows us to prevent look-ahead bias by ensuring the model only ever "sees" past interactions when predicting future ones, and to measure performance stability over time. By sliding the cut-off T_C forward (for example, in steps of the window length T_w), we generate multiple train–test splits to calculate our metrics on, before aggregating them to judge model performance. Although T_w is tuneable, for each experiment we keep T_w constant to ensure that metrics remain fair across the cross-validation process.

This method additionally allows us to simulate both cold-start situations, and when information is plentiful. By setting T_c to a low number (approaching the initial start time $t = 0$), it effectively simulates a low-information, cold-start situation. As recommender performance will differ based on available information, a split between cold-start situations (low T_c) and information-rich situations (high T_c) will allow for judgement of the model performance under different situations.

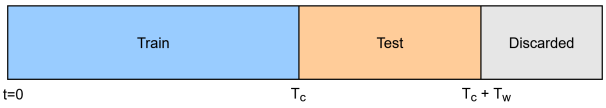

Fig. 4. Diagrammatic representation of the train-test split formation. Data is sorted by chronological order and data from times $t = 0 \dots T_c$ are used as training data, and a fixed window $t = T_c \dots T_c + T_w$ is used as testing data. This aims to better reflect real use behaviour and prevent bias due to chronological data leaks. $t = 0$ represents the first interaction on the website (start time).

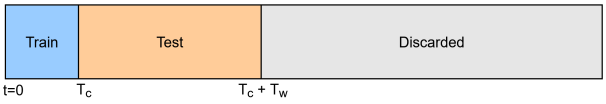

Fig. 5. Train-test split simulating a cold-start situation. In this case, by setting T_C to be close or equal to 0, it emulates the situation where limited information is available to the model. A consistent window size T_w is used to ensure fairness in quantitative metric evaluation.

B. Training Metrics

a) Mean Squared Error (MSE): To train the matrix-based recommenders, namely the trust-aware (III-B) and the integrated hybrid recommender (III-D). Evaluation of the recommender's performance, without consideration for the context of the recommender (i.e. our recommender provides top-N choices), is typically done by taking the MSE loss between the predicted and actual user ratings of a product. As ratings are sparse and most products have no rating (to be assumed to be equal to a 0 rating), this loss is only calculated over products that are rated by a user.

$$\text{MSE} = \frac{1}{|\mathcal{K}|} \sum_{(u,i)\in\mathcal{K}} \left(\hat{r}_{u,i} - r_{u,i}\right)^2$$

This metric is useful to evaluate and tune model performance for *TrustSVD* and *MR3*, but not the performance of the recommender as a whole. This is because they do not directly measure the quality of top-N lists. Other similar alternatives include root mean squared error (RMSE) which retains the original units of the subject, but otherwise serves the same purpose as MSE.

C. Evaluation Metrics

Metrics to evaluate recommender performance. We define true positives (TP) to be rel_i, a relevant item to a user. A relevant item is defined to be any item "purchased (rated)" by the user with a rating or 3 or above. We filter out low ratings as these products are likely to result in returns, which is an extra cost and will dig into the dividends that we can pay our shareholders.

a) Precision@N: measures the fraction of the top-N recommended items that are actually relevant to the user. It is defined as

$$\text{Precision@}N = \frac{1}{N}\sum_{i=1}^{N} \text{rel}_i,$$

where $\text{rel}_i = 1$ if the ith item in the ranked list is relevant (and 0 otherwise). Precision@N focuses on the accuracy of your top-N list, but does not account for how many relevant items the user has in total.

b) Recall@N: easures the fraction of all of a user's relevant items that appear in the top-N recommendations. It is defined as

$$\text{Recall@}N = \frac{\sum_{i=1}^{N} \text{rel}_i}{|\mathcal{R}_u|},$$

where $\mathcal{R}_u$ is the set of all relevant items for user u. Recall@N captures how well you cover the user's interests, but does not penalise extra non-relevant items in the list.

c) F1@N: The F1 score at N is the harmonic mean of Precision@N and Recall@N:

$$F1@N = 2 \times \frac{\text{Precision@}N \times \text{Recall@}N}{\text{Precision@}N + \text{Recall@}N}.$$

It aims to strike a balance between precision and recall. Typically, optimising for precision will incur a cost to recall, and vice versa.

d) Hit Rate (HR@N): measures the fraction of users for whom at least one relevant item appears in their top-N recommendations. Formally,

$$\text{HR@}N = \frac{1}{|U|}\sum_{u\in U} \mathbf{1}\Big(\sum_{i=1}^{N} \text{rel}_{u,i} > 0\Big),$$

where $\mathbf{1}(\cdot)$ is the indicator function. HR@N tells you whether the system "hits" any relevant item at all, but ignores how many hits or where they occur. This was judged to be an important metric.

e) Mean Reciprocal Rank (MRR@N): evaluates how early the first relevant item appears. For each user u, let rank_u be the position of their first relevant recommendation (up to N). Then

$$\text{MRR@}N = \frac{1}{|U|}\sum_{u\in U} \frac{1}{\text{rank}_u}.$$

MRR@N heavily rewards placing a relevant item at the very top but does not consider any additional relevant items or their positions. This was judged as not important as our page format evenly represents each item.

f) nDCG@N: Normalized Discounted Cumulative Gain at N accounts both for graded relevance and position. First,

$$\text{DCG@}N = \sum_{i=1}^{N} \frac{2^{\text{rel}_i} - 1}{\log_2(i+1)},$$

and then

$$\text{nDCG@}N = \frac{\text{DCG@}N}{\text{IDCG@}N},$$

where $\text{IDCG@}N$ is the ideal (maximum) DCG@N under a perfect ranking. nDCG@N rewards placing highly relevant items early and gracefully handles non-binary relevance scores. However, it was not felt that this was the most important as our page format evenly represents the items and there is near to no difference between the 1st item on a page and the 15th.

D. User-Centric KPIs

The metrics here are primarily to be used in user studies, and not applicable to the existing datasets.

For these metrics, as they are user facing, the amount of items "recommended" is defined as $P * N$, where $N = 15$ which is the amount of items per page, and P is the amount of pages that were recommended to the user. An assumption is made that different pages yield non-duplicate items.

- Click-Through Rate (CTR): fraction of items "clicked".
- Conversion Rate: fraction of recommended items purchased.
- Diversity: How many distinct item categories your system recommends. As this would be part of a user study, the sample size is likely to be quite small and so item categories can be manually labelled. In this case, a target "diversity" score would have to be additionally defined as we want neither no diversity nor be too diverse in our recommendations.

E. Metrics Discussion

For training and tuning the matrix-based CF techniques (*TrustSVD*, *MR3*), MSE loss will be utilised. However, when evaluating the recommender performance, we will primarily utilise **F1@N**, **HR@N**. **F1@N** was chosen as it struck a balance between precision@N and recall@N. It was judged that both precision@N is important in encouraging shopping sprees as more relevant items are recommended, and recall@N is also important as we would like to milk the customer of as much money as possible. To balance the precision-recall tradeoff, the **F1@N** metric is suitable. **HR@N** was chosen as a metric as it was determined that consistently recommending users at least one relevant item was important in retaining users on the platform. Users who remain on the platform not only enhance our trust network and thus our data, but also present future business opportunities. In the case that a choice between the two metrics presents itself, **HR@N** was judged as being the most important, as we see the retainment of customers for future profits to be of utmost importance.

F. Computational Requirements

To ensure a smooth experience for patrons, the algorithm is required to run near instantaneously. The matrix technique featured in our hybrid recommender system is able to run in milliseconds, once it has been trained. The trade-off for this fast inference speed is in memory. Two dense matrices of size $U \times f, \quad I \times f$, are required, where U is the number of users, I is the number of items, and f is the latent dimension.

Although inference is near instantaneous, training is not, incurring a cost of

$$O(|K|f)$$

per epoch, where K is the number of observed ratings. This can get quite slow, especially as the dataset gets large. However, training can occur periodically and as a background process, and thus does not affect the user experience. Periodic re-training is required to ensure that the model is able to perform at its best using the most amount of available data.

It was judged that both trade-offs can be mitigated through the usage of modern hardware (such as server-grade GPUs) that have both large memories and are able to accelerate training times. Most importantly, if hosted on servers, these trade-offs do not hurt the user experience, which is paramount.

G. User Study

a) Objective: To validate offline evaluation and KPIs with real users, we will conduct a controlled user study using a click-through prototype of our e-commerce interface. The study will measure (i) perceived relevance, (ii) trust and satisfaction, and (iii) behavioural engagement under each recommendation algorithm.

b) Participants: We will recruit $N = 25$ shoppers, with both a spread of individual users and users who already know each other. This allows us to have predefined trust networks. Users will be asked to follow their existing "friends" on the site, and to make a base number of recommendations on items they have already seen before and like.

c) Apparatus: A web-based prototype reflecting our 5×3 product panel UI (image, name, price, average rating, "recommended by your trustee" badge) will be deployed. Three back-ends (content, trust, hybrid) will be instrumented to log page views, panel clicks, "add to cart" events, and timestamps. Only a subset of item categories (e.g. electronics, household) will be used to ensure data is not too sparse.

d) Procedure:

(i) **Introduction & Training**: demo of UI and one practice task with a control algorithm.

(ii) **Item recommendations**: Users will be asked to "buy" and rate items they find on the site that they already use in real life and like. The model will be trained on this.

(iii) **Shopping Round**: Users will be recommended products by the model. They can click around on the interface and allowed to click off after 5 minutes of "shopping" if they like. Users will be asked to click "buy" on any products they would actually buy. Their activity will be tracked, including click data.

(iv) **Final Questionnaire** (5 min): overall ranking of the three systems and System Usability Scale (SUS).

e) Evaluation: The user trial will be evaluated both based on the questionnaire, and the KPIs discussed in IV-D.

V. Conclusion

We have outlined a comprehensive framework for building and evaluating an e-commerce recommendation engine that leverages multiple information sources—textual reviews, social trust networks, and latent content topics. By developing content-based, trust-aware, and tightly integrated hybrid models, and by deploying them through a consistent panel-based UI, we ensure that each method can be directly compared both

offline (via temporal train–test splits and top-N metrics) and in a live user study. Together, our quantitative metrics and qualitative feedback loop will inform the next stages of model refinement, with the ultimate goal of delivering more accurate, trustworthy, and user-preferred product recommendations that drive higher conversions and long-term customer retention.

REFERENCES

[1] J. J. McAuley and J. Leskovec, "Hidden factors and hidden topics: understanding rating dimensions with review text," in *Proceedings of the Seventh ACM Conference on Recommender Systems (RecSys '13)*, Hong Kong, China, 2013, pp. 165–172. doi:10.1145/2507157.2507163

[2] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, "Methods and metrics for cold-start recommendations," in *SIGIR Forum (ACM Special Interest Group on Information Retrieval)*, 2002, pp. 253–260. doi:10.1145/564418.564421

[3] G.-N. Hu, X.-Y. Dai, Y. Song, S.-J. Huang, and J.-J. Chen, "A Synthetic Approach for Recommendation: Combining Ratings, Social Relations, and Reviews," in *Proceedings of the 24th International Joint Conference on Artificial Intelligence (IJCAI'15)*, Buenos Aires, Argentina, July 25–31, 2015, pp. 1756–1762.

[4] J. Tang, H. Gao, H. Liu, and A. Das Sarma, "eTrust: Understanding trust evolution in an online world," in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, 2012, pp. 253–261. ACM.

[5] J. Tang, H. Gao, H. Liu, and A. D. Sarma, *eTrust: Understanding Trust Evolution in an Online World*, in *Proceedings of the 18th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD '12)*, Beijing, China, August 12–16, 2012, pp. 253–261, ACM.

[6] M. R. Hamedani, I. Ali, J. Hong, and S.-W. Kim, "TrustRec: An Effective Approach to Exploit Implicit Trust and Distrust Relationships along with Explicit ones for Accurate Recommendations," *Computer Science and Information Systems*, vol. 18, no. 1, pp. 93–114, Jan. 2021, doi:10.2298/CSIS200608039H. :contentReference[oaicite:0]index=0

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality," in *Advances in Neural Information Processing Systems*, vol. 26, pp. 3111–3119, 2013.

[8] G. Guo, J. Zhang, and N. Yorke-Smith, "TrustSVD: Collaborative Filtering with Both the Explicit and the Implicit Influence of User Trust and of Item Ratings," in *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence*, vol. 29, no. 1, 2015, pp. 123–129. doi:10.1609/aaai.v29i1.9153