Intent-Aware Contextual Recommendation System

Biswarup Bhattacharya [1], Iftikhar Burhanuddin [2], *Abhilasha Sancheti* [2], & Kushal Satya [3]

[1] University of Southern California, [2] Adobe Research, [3] Adobe India

bbhattac@usc.edu, sancheti@adobe.com

November 18, 2017

5^{th} DSBDA Workshop @ 17^{th} IEEE ICDM 2017

Overview

Introduction

Need for an effective recommendation system which can identify what the user *wants to do* and *needs to do*.

- What the user **wants** to do:

- What the user **needs** to do:

Introduction

Need for an effective recommendation system which can identify what the user *wants to do* and *needs to do*.

- What the user **wants** to do:
  - Does the user like what he is doing?
  - Does the user want to do what he did the last time?
  - Does the user want to do something new that he never did before?
- What the user **needs** to do:

Introduction

Need for an effective recommendation system which can identify what the user *wants to do* and *needs to do*.

- What the user **wants** to do:
  - Does the user like what he is doing?
  - Does the user want to do what he did the last time?
  - Does the user want to do something new that he never did before?
- What the user **needs** to do:
  - Get aid for job-related work.
  - Function of user accessing the system - e.g. technical or sales.

Introduction - Recommender Systems

Types of recommender systems:

- Frequency-based
- Content-based

Introduction - Recommender Systems

Types of recommender systems:

- Frequency-based
  - Remembers historical visits.
  - Does not consider the content of the page.
- Content-based

Introduction - Recommender Systems

Types of recommender systems:

- Frequency-based
  - Remembers historical visits.
  - Does not consider the content of the page.
- Content-based
  - Observe *usefulness* of a page.
  - Concept of *relevance score* is involved.

Introduction - Novelty

Our system:

- Combines both aspects: *frequency* and *content*.

Introduction - Novelty

Our system:

- Combines both aspects: *frequency* and *content*.
  - Identify user *intent*.
  - Rank the *content*.
  - Take *historical browsing* into account.

Problem Definition

Business Intelligence tasks:

- User has to go through multiple reports (pages).
- There is usually an end goal or a target (report/page) in mind.
  - We define this as the *intent* of the user.
  - Very difficult to solve.
- New users find it hard to comprehend the complex system of reports.
  - Historical information for such users is low.
  - Collaborative/group-based approach makes sense.

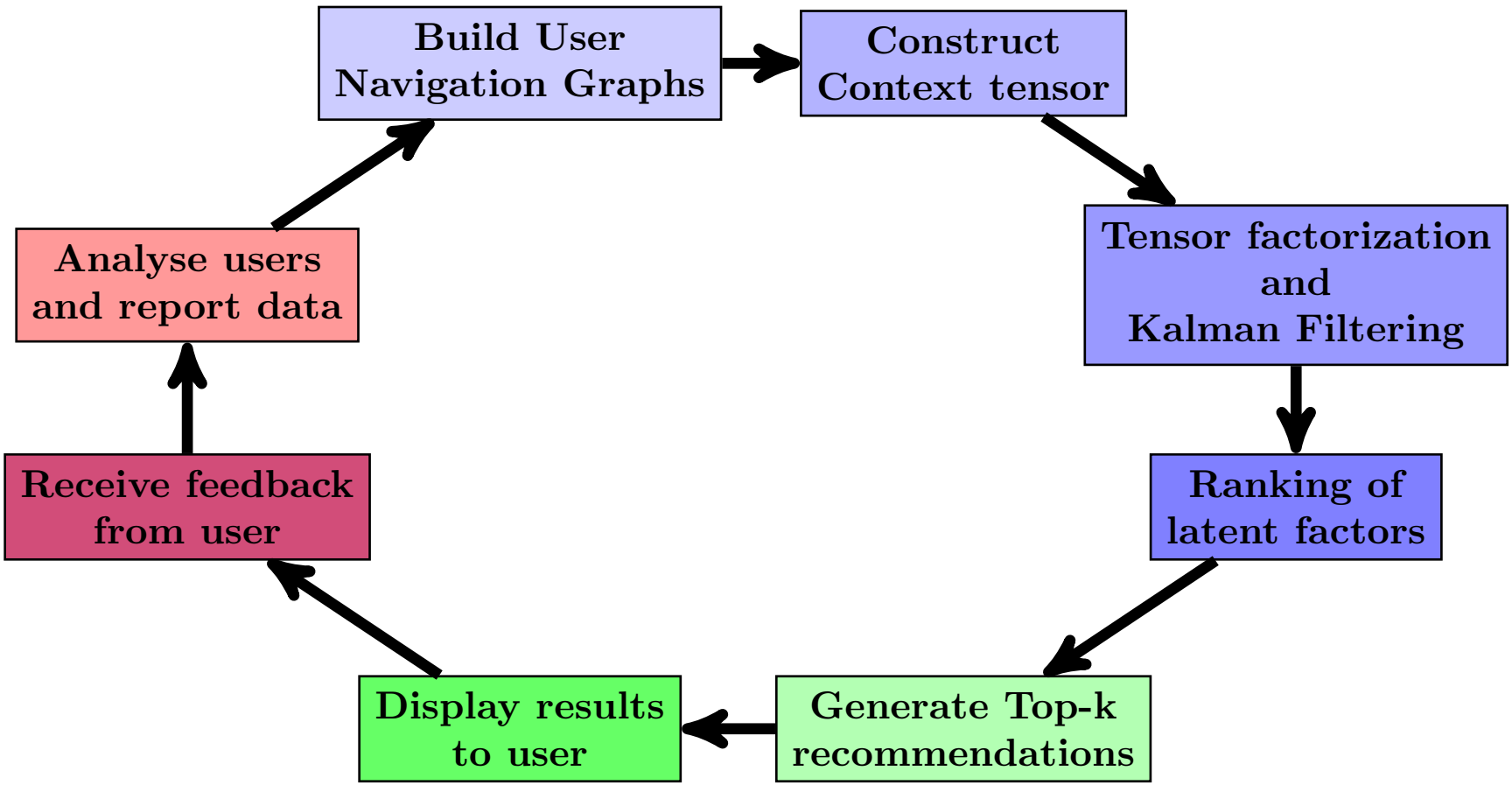
Problem Definition

Our system addresses these questions:

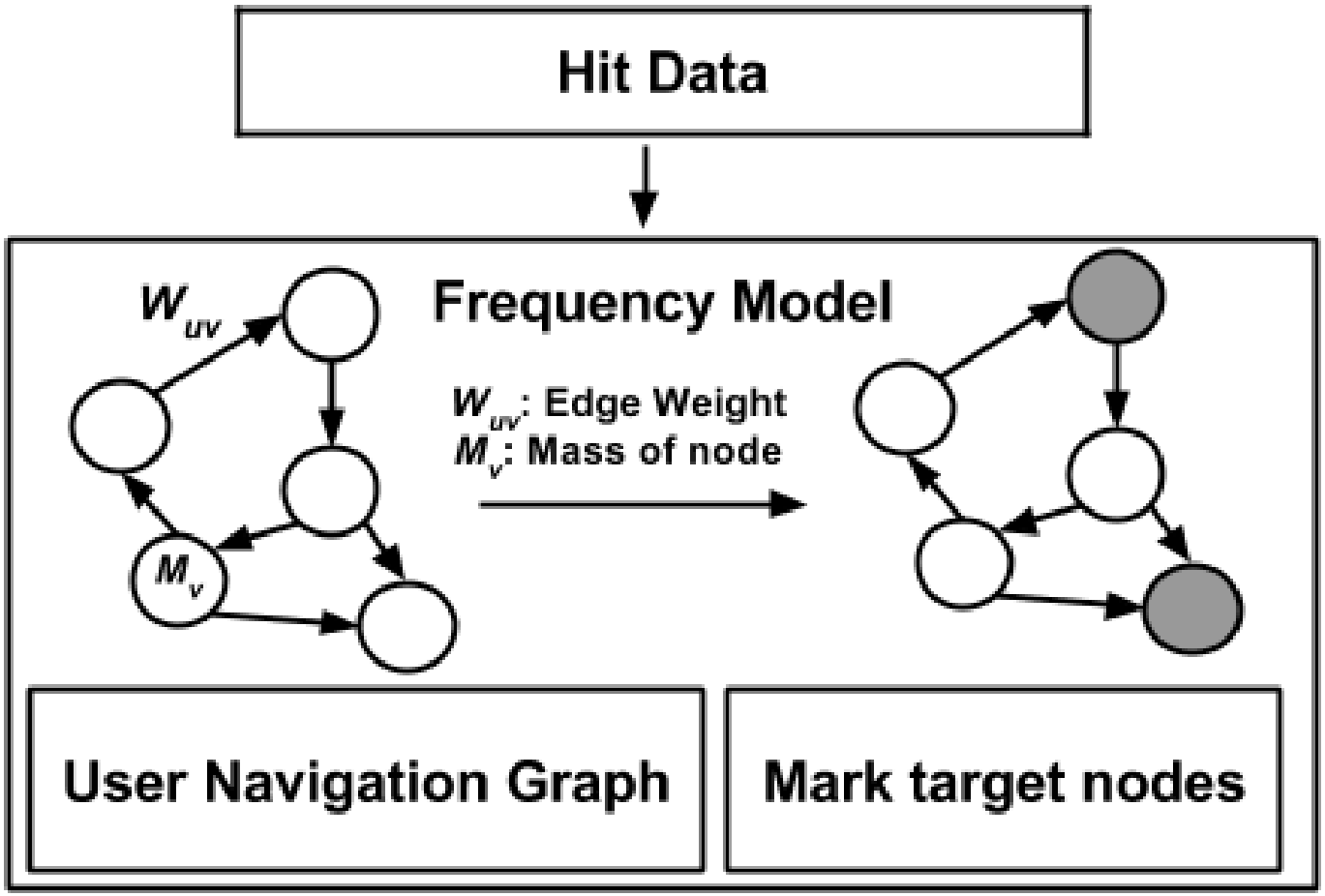
- Predict user *intent* which is the end goal (report/page) in our scenario, from context and frequency.
- Determine the right content, data and representation based on the type and *expertise* of the user.
- Find the most suitable recommendation *scoring* system.

The terms “report”, “node” and “page” are used interchangeably in both the paper and these slides.

System Design - Workflow



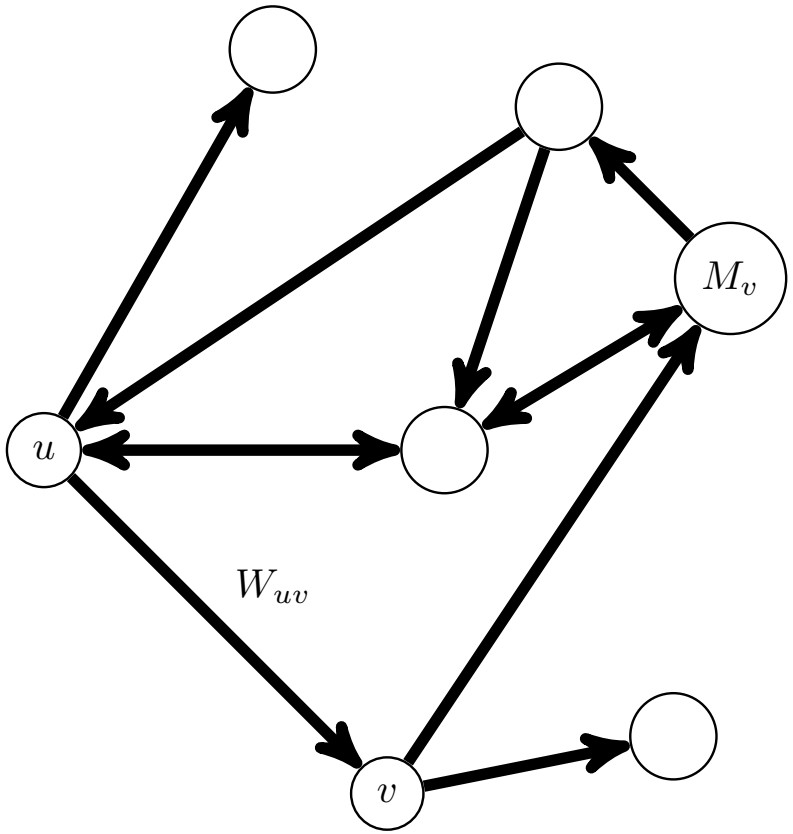
User Navigation Graph



User Navigation Graph

Graph description

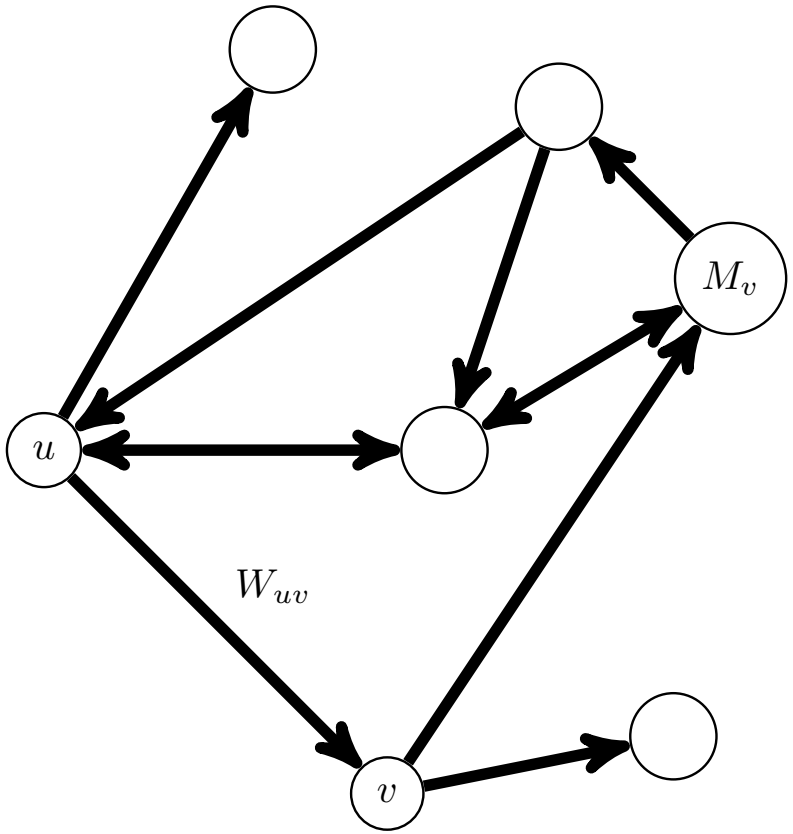
- Nodes (u and v) - Unique reports seen by the user.
- Edge Weights (W_{uv}) - Probability that the user goes from node u to node v.



User Navigation Graph

Attributes of each node

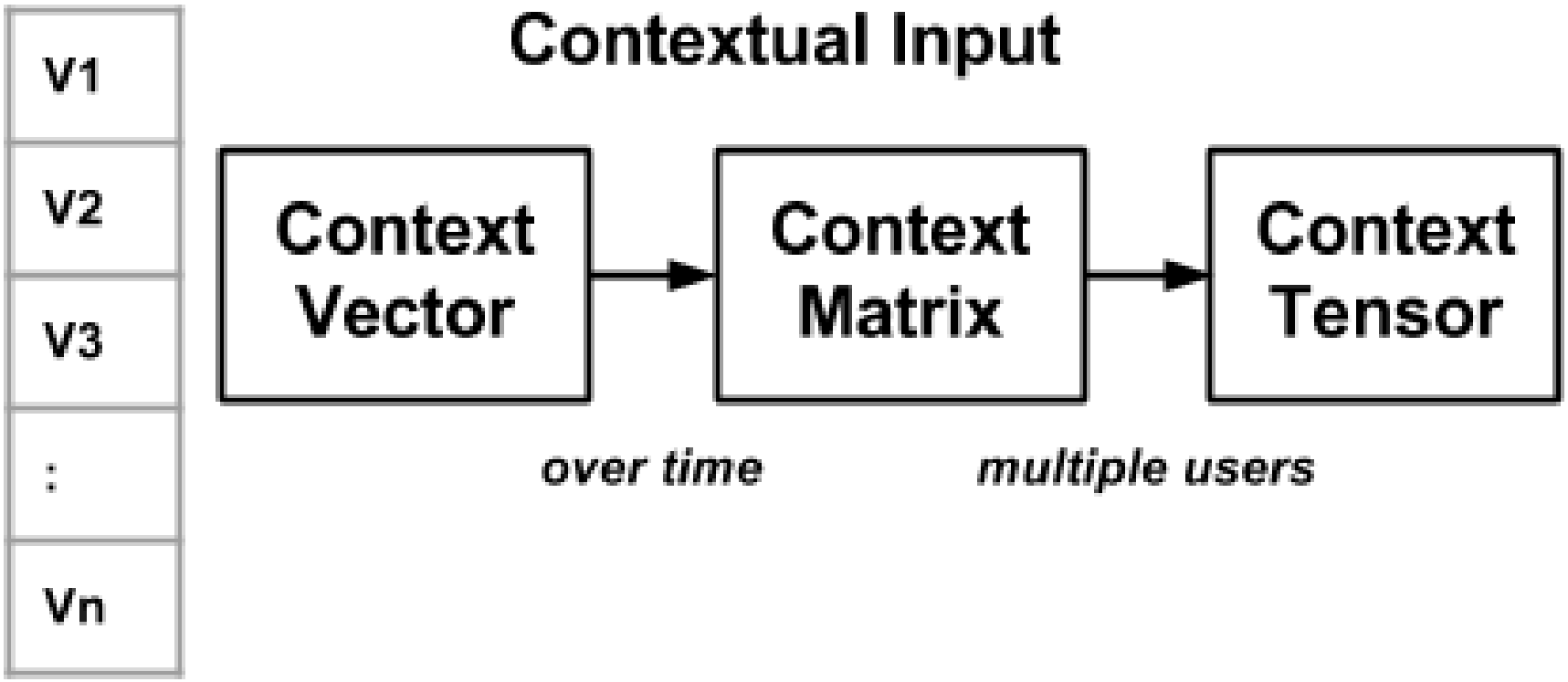
- Unique ID, Content information
- Mass (M_v): Fraction of total time spent on a node (value between 0 and 1)



Target Nodes

- Target: 0 or 1
- Target = 1 implies possible intent node.
  - High in-degree nodes
  - Domain knowledge gained by discussing with users
  - Specific or important reports

Contextual Input



Contextual Input

- Two kinds of reports seen by users:
  1. Time series
  2. Histograms

Contextual Input

- Two kinds of reports seen by users:
  1. Time series
     - Aggregate Value
     - Maximum and minimum value of the time series
     - Location of maximum and minimum observation
     - Longest positive and negative runs
     - Length of time series
     - Average absolute change in consecutive
  2. Histograms

Contextual Input

- Two kinds of reports seen by users:
  1. Time series
     - Aggregate Value
     - Maximum and minimum value of the time series
     - Location of maximum and minimum observation
     - Longest positive and negative runs
     - Length of time series
     - Average absolute change in consecutive
  2. Histograms
     - Aggregate Value
     - Rest of the 5 values set to 0

Context Vector & Context Matrix

- For a given user we have a set of different dimension elements for which a metric is calculated. The cardinality of this set for each metric m is denoted as d_m.
- The **context vector** is of size $N^u = 6 \times D_u$, where $D_u = \sum_{m=1}^{M_u} d_m$, and M_u is the cardinality of the set of all metrics seen by the user.

Context Vector & Context Matrix

- For a given user we have a set of different dimension elements for which a metric is calculated. The cardinality of this set for each metric m is denoted as d_m.
- The **context vector** is of size $N^u = 6 \times D_u$, where $D_u = \sum_{m=1}^{M_u} d_m$, and M_u is the cardinality of the set of all metrics seen by the user.
- The aggregation of many such context vectors over the number of reports seen forms the **context matrix**.
- The dimension of the context matrix is hence $N^u \times T$, where N^u signifies the context variables which varies from user to user and T signifies the number of reports seen.

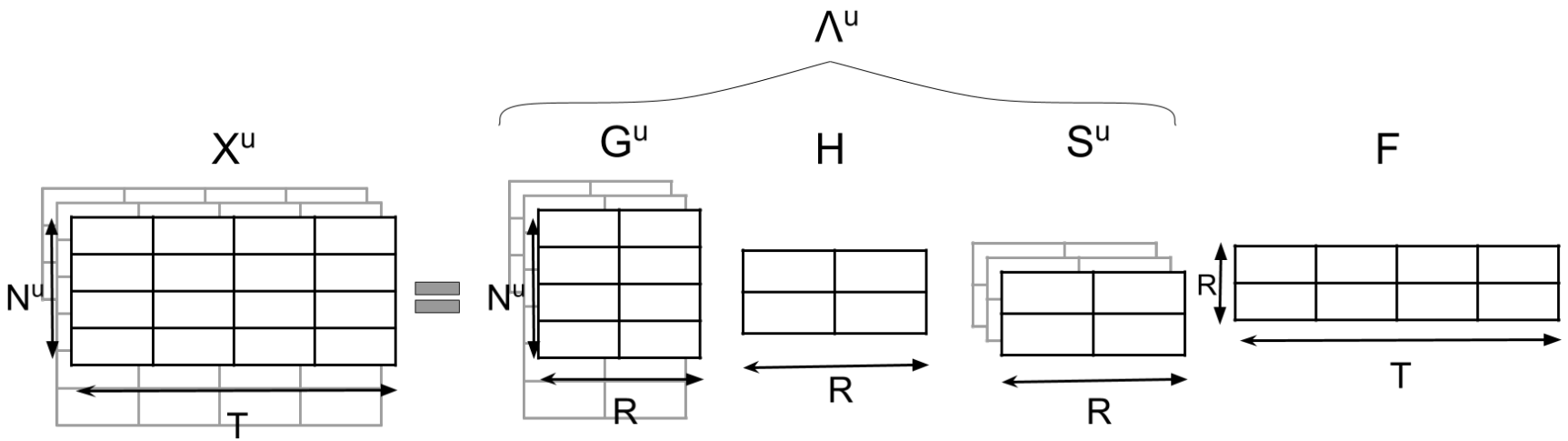
Context Tensor

- The activity of the users are analyzed and consequently the users are clustered into 4 categories depending on their exposure and competence level of using the tool using k-means clustering algorithm.
- We form the **context tensor** consisting of context matrices of U users from the same cluster.
- The dimension of the 3-way tensor thus formed is: $N^u \times T \times U$.

Context Model - Tensor Factorization

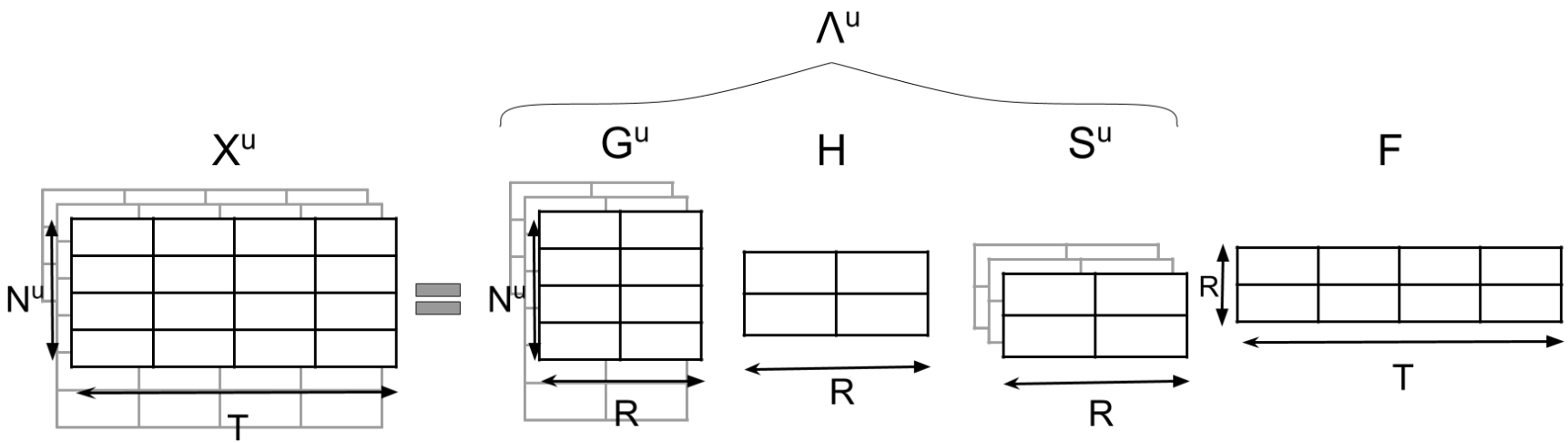
- Since the tensor is highly sparse and the number of context features varies from user to user, PARAFAC2 tensor decomposition is used to obtain latent factors for each report seen by the user.
- Equivalent to solving this optimization problem:

$$(\tilde{\mathbf{F}}, \tilde{\mathbf{\Lambda}}^u)_{u=1,2,\ldots|U|} = \min_{\mathbf{F},\mathbf{\Lambda}^u} \sum_{u=1}^{|U|} ||\mathbf{X}^u - \mathbf{\Lambda}^u \mathbf{F}||^2_F$$



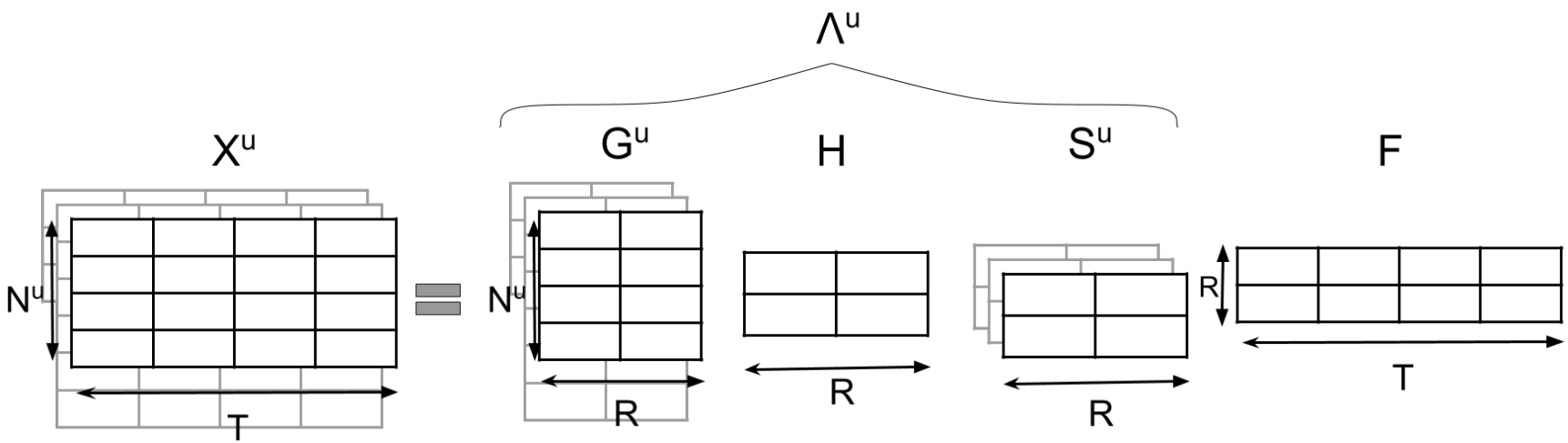
Context Model - Tensor Factorization

- $\mathbf{X}^u$ - context panel of the u^{th} user
- $\mathbf{G}^u$ - orthonormal matrix
- $\mathbf{H}$ - matrix invariant to u
- $\mathbf{S}^u$ - diagonal matrix
- $\mathbf{\Lambda}^u$ - factor loading matrix
- $\mathbf{F}$ - matrix containing the collaborative latent factors at T time instances.



Context Model - Tensor Factorization

- N^u, T - previously described
- R - optimally chosen so that latent factors evolve faster.



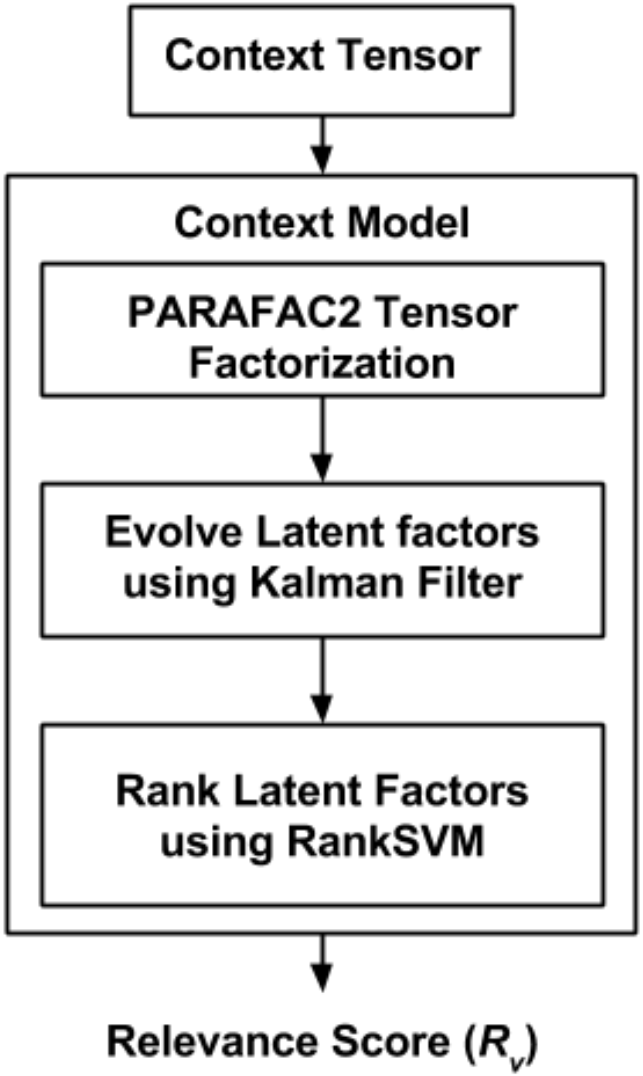
Context Model - Kalman Filtering

Kalman Filter is used to enforce the dynamics and sequential correlations in the latent factors. This enables real-time (fast) processing by rapidly evolving the latent factors.

$$\mathbf{x}_t^u = \mathbf{\Lambda}^u \mathbf{f}_t + \zeta_t^u$$
$$\tilde{\mathbf{f}}_t = \mathbf{A}^u \hat{\mathbf{f}}_{t-1} + \omega_t^u$$
$$\hat{\mathbf{f}}_t = \tilde{\mathbf{f}}_t + \mathbf{K}_t^u (\mathbf{x}_t^u - \hat{\mathbf{A}} \tilde{\mathbf{f}}_t)$$

- $\mathbf{x}_t^u$ - t^{th} column of $\mathbf{X}^u$, $\mathbf{f}_t$ - t^{th} column of $\mathbf{F}$
- $\mathbf{K}_t^u$ - Kalman gain (involves apriori error covariance matrices)
- $\mathbf{A}^u$ - transition matrix $(R \times R)$
- $\mathbf{\Lambda}^u$ - factor loading matrix $(N^u \times R)$
- ζ_t^u, ω_t^u - mutually independent Gaussian random variables with known positive definite covariance matrices

Context Model



Ranking of Latent Factors

Based on the training data:

- Set $\mathbf{R}_1$ = latent factors $\mathbf{f}_i$ such that the user eventually ends up at target node l in the session.
- Set $\mathbf{R}_2$ = latent factors $\mathbf{f}_j$ such that the user does not eventually end up at target node l in the session.

Ranking of Latent Factors

Based on the training data:

- Set $\mathbf{R}_1$ = latent factors $\mathbf{f}_i$ such that the user eventually ends up at target node I in the session.
- Set $\mathbf{R}_2$ = latent factors $\mathbf{f}_j$ such that the user does not eventually end up at target node I in the session.
- We define the ranking function $g(I, \mathbf{f}_i)$:

$$g(I, \mathbf{f}_i) = \begin{cases} 1, & \mathbf{f}_i \in \mathbf{R}_1, \\ 0, & \mathbf{f}_i \in \mathbf{R}_2. \end{cases}$$

Ranking of Latent Factors

Based on the training data:

- Set $\mathbf{R}_1$ = latent factors $\mathbf{f}_i$ such that the user eventually ends up at target node I in the session.
- Set $\mathbf{R}_2$ = latent factors $\mathbf{f}_j$ such that the user does not eventually end up at target node I in the session.
- We define the ranking function $g(I, \mathbf{f}_i)$:

$$g(I, \mathbf{f}_i) = \begin{cases} 1, & \mathbf{f}_i \in \mathbf{R}_1, \\ 0, & \mathbf{f}_i \in \mathbf{R}_2. \end{cases}$$

- We also define a set P:

$$P = \{(i, j) : \mathbf{f}_i \in \mathbf{R}_1, \mathbf{f}_j \in \mathbf{R}_2\}$$

Ranking of latent factors - RankSVM

- Objective of RankSVM: Learn the ranking function $g(I, \mathbf{f}_i)$ such that $g(I, \mathbf{f}_i) > g(I, \mathbf{f}_j)$.
- Thus, $g(I, \mathbf{f}_i)$ can be defined as:

$$g(I, \mathbf{f}_i) = < \vec{w}, \mathbf{f}_i >$$

Ranking of latent factors - RankSVM

- Objective of RankSVM: Learn the ranking function $g(I, \mathbf{f}_i)$ such that $g(I, \mathbf{f}_i) > g(I, \mathbf{f}_j)$.
- Thus, $g(I, \mathbf{f}_i)$ can be defined as:

$$g(I, \mathbf{f}_i) = < \overrightarrow{w}, \mathbf{f}_i >$$

- Large margin approach $\rightarrow$ optimization problem:

$$\min_{\overrightarrow{w}, \epsilon_{ij} \geq 0} \quad < \overrightarrow{w}, \overrightarrow{w} > + \lambda \sum_{ij} \epsilon_{ij}$$

$$\text{s.t.} \quad \forall (i,j) \in P, < \overrightarrow{w}, \mathbf{f}_i > \geq < \overrightarrow{w}, \mathbf{f}_j > + 1 - \epsilon_{ij}$$

- $\lambda > 0$ determines the trade-off between margin maximization and error minimization

Ranking of Latent Factors - Intent Score

- Once optimal $\overrightarrow{w}$ has been learned they can be used to induce ranking of new latent factors for each intent.

Ranking of Latent Factors - Intent Score

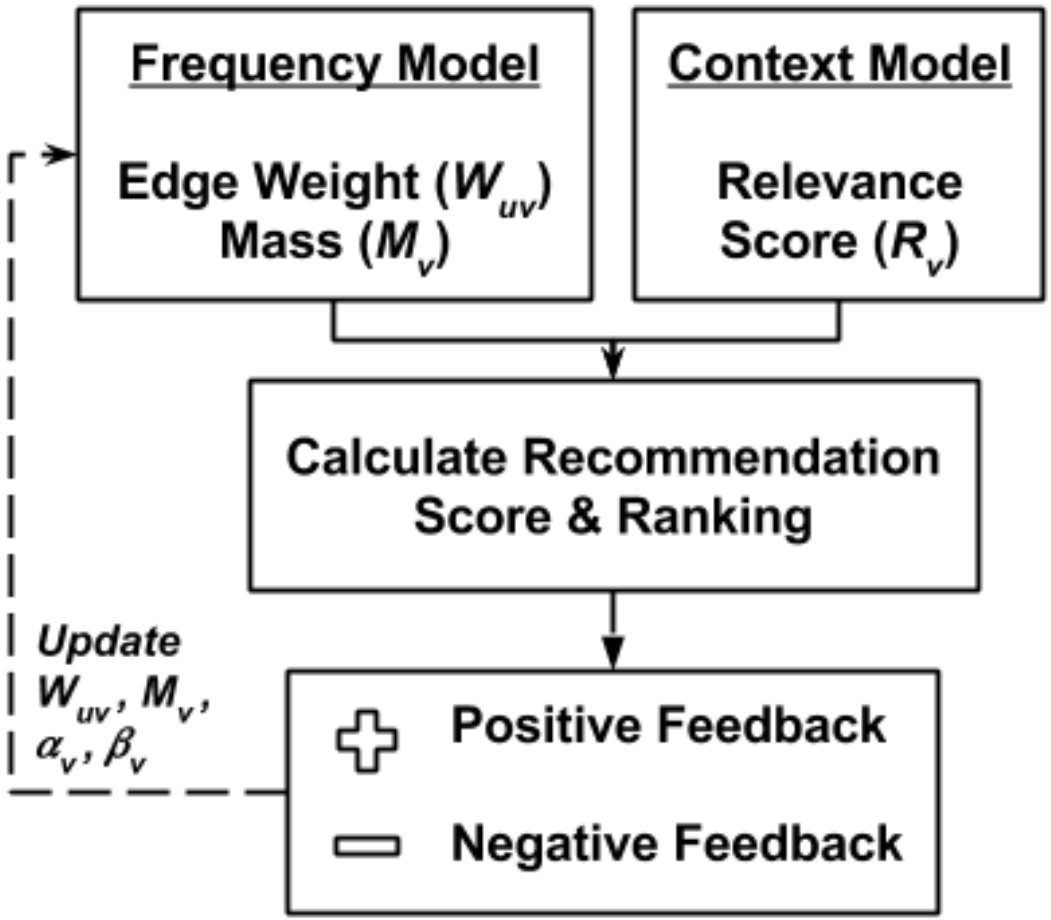
- Once optimal $\overrightarrow{w}$ has been learned they can be used to induce ranking of new latent factors for each intent.
- $\overrightarrow{w} \cdot \overrightarrow{\mathbf{f}}$ = distance between the training point and plane separating positive training and negative training points.
- Latent factors normalized to have norm 1.

Ranking of Latent Factors - Intent Score

- Once optimal $\overrightarrow{w}$ has been learned they can be used to induce ranking of new latent factors for each intent.
- $\overrightarrow{w} \cdot \overrightarrow{\mathbf{f}}$ = distance between the training point and plane separating positive training and negative training points.
- Latent factors normalized to have norm 1.
- For every test data, we calculate the latent factor ($\overrightarrow{\mathbf{f}}$) for the report seen and take its dot product with the weights for each intent to get the intent score (S), normalized to be between 0 and 1:

$$S_I(\overrightarrow{\mathbf{f}}) = \frac{(4 + \overrightarrow{w_I} \cdot \overrightarrow{\mathbf{f}})}{8} \quad (1)$$

Recommendation Scoring



Recommendation Scoring

$$K_{uv} = (\alpha_v \times W_{uv} \times R_v) + (\beta_v \times M_v) \quad (2)$$

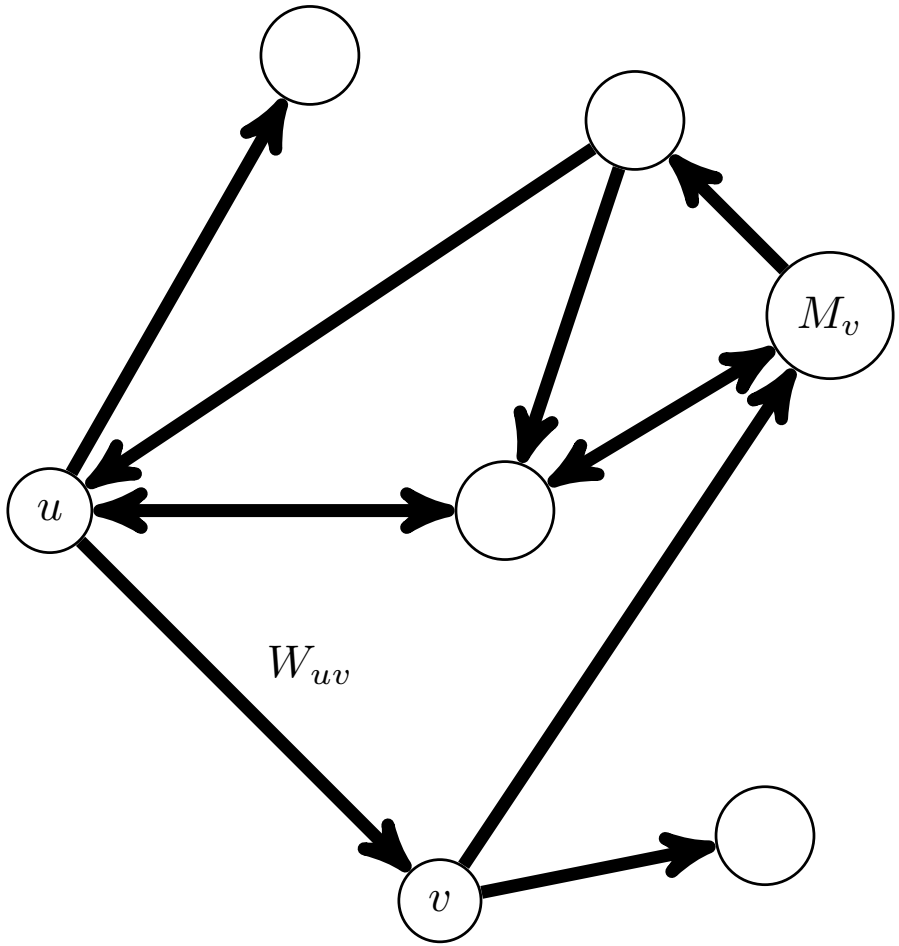
Recommendation Scoring

$$K_{uv} = (\alpha_v \times W_{uv} \times R_v) + (\beta_v \times M_v) \quad (2)$$

- W_{uv} = Historical probability of user going from node u to node v.
- R_v = Relevance score of the recommendable node v.
- M_v = Fraction of time spent on node v.
- α_v, β_v = Feedback factors (initialized to 1.0)

Recommendation Scoring - R_v

- Node v has path to target nodes $(l_1, \ldots, l_k)$.
- $R_1, \ldots, R_k$ are the intent scores.
- $D_1, \ldots, D_k$ are the probabilistic Dijkstra distances to the respective target nodes from node v.



Recommendation Scoring - R_v

$$K_{uv} = (\alpha_v \times W_{uv} \times R_v) + (\beta_v \times M_v)$$

- $R_v = f(S, D)$, where S are the intent scores and D is the the probabilistic distance from the node v to the set of target (intent) nodes.

Recommendation Scoring - R_v

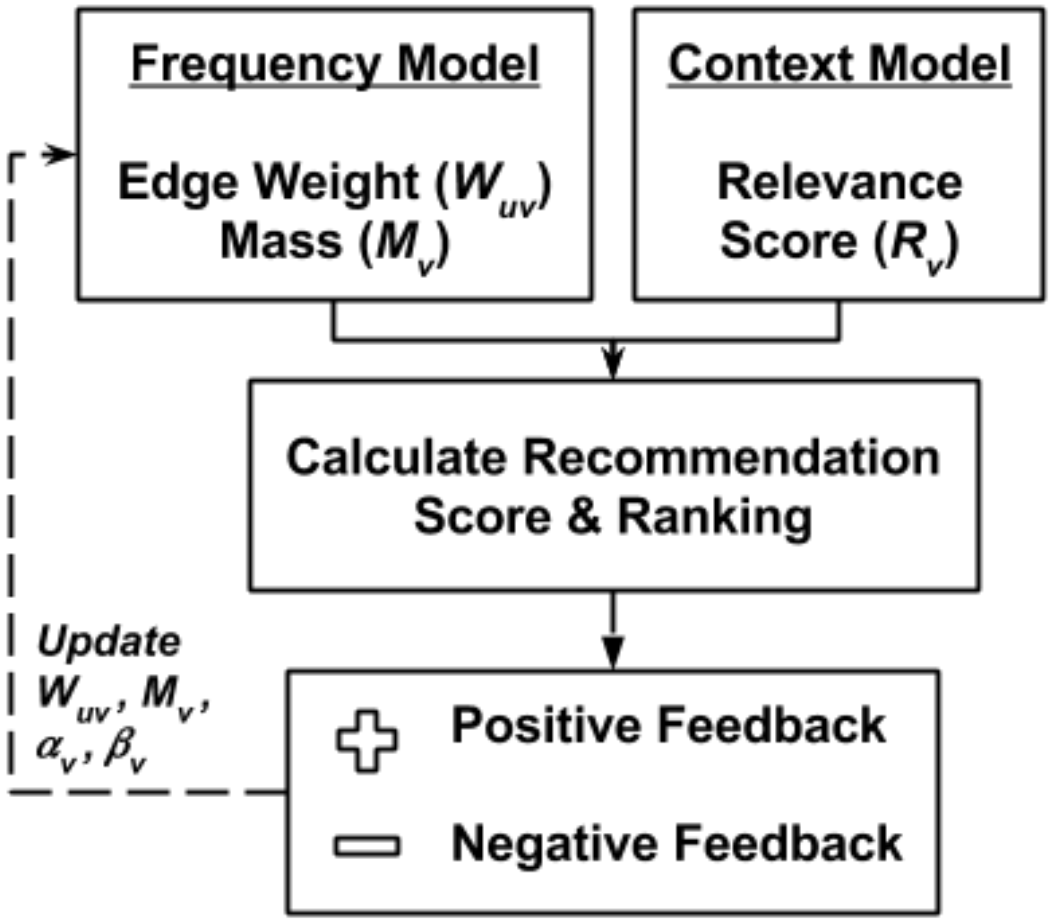
$$K_{uv} = (\alpha_v \times W_{uv} \times R_v) + (\beta_v \times M_v)$$

- $R_v = f(S, D)$, where S are the intent scores and D is the the probabilistic distance from the node v to the set of target (intent) nodes.
  - **Max-IxD**: Maximum of (intent scores $\times$ distance to target node) $\rightarrow R_v = \max(R_1 \times D_1, \ldots, R_k \times D_k)$
  - **Sum-IxD**: Dot product between intent score and distances of node to those intents $\rightarrow R_v = \sum_{i=1}^{k} R_i \times D_i$
  - **Max-I**: Maximum of intent scores $\rightarrow R_v = \max(R_1, \ldots, R_k)$
  - **Sum-I**: Sum of intent scores (proposed) $\rightarrow R_v = \sum_{i=1}^{k} R_i$

Group-based Recommendation

- For providing recommendations to user u, use the graphs of user u as well as other users.
- Clustered all users according to past activity: "experienced" and "new" users.
- For "new" users, graphs of "experienced" users were considered.
  - R_v sourced from the "new" (current) user, all other parameters sourced from the "experienced" user.
  - Scores of only "novel" nodes were considered (nodes not present in the current user's graph).

Feedback

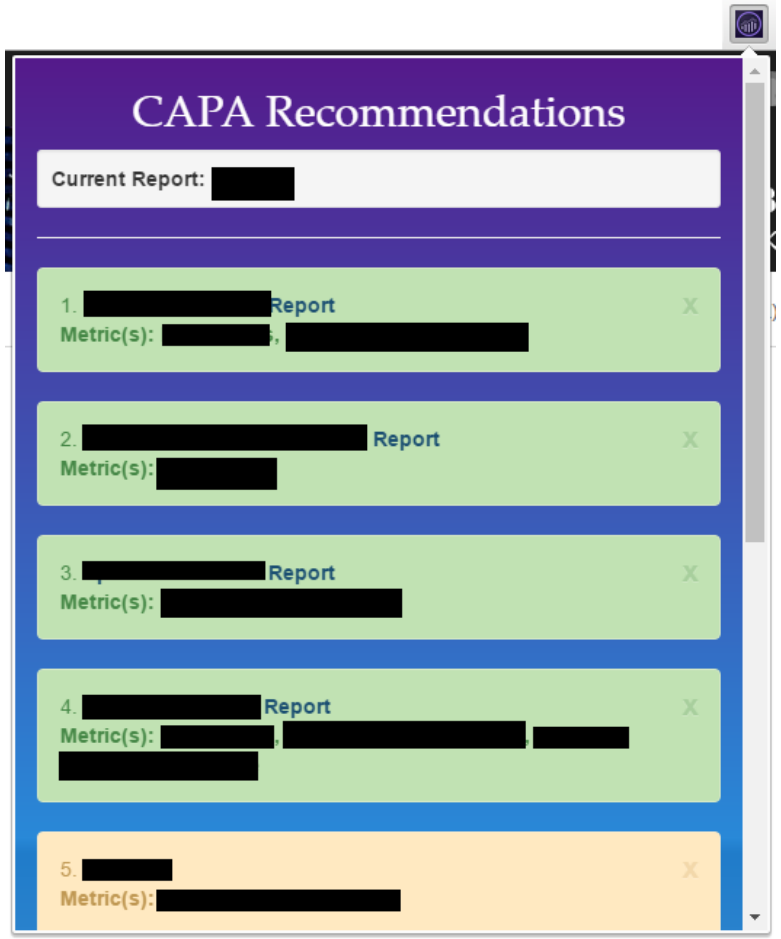

Frequency Model
Edge Weight (W_{uv})
Mass (M_v)
Context Model
Relevance Score (R_v)
Calculate Recommendation Score & Ranking
Update W_{uv}, M_v, α_v, β_v
Positive Feedback
Negative Feedback

Displaying Recommendations

Ranking recommendations

1. K_{uv} (Descending)
2. Prefer collaborative
3. R_v (Descending)
4. W_{uv} (Descending)
5. M_v (Descending)

- Maximum 10 recommendations shown
- Color coded to show system's confidence



* Proprietary material has been redacted in the image

Feedback (Idea)

Four types of feedback:

- Explicit Feedback: User interacted with the recommendations list.
- Implicit Feedback: User did not interact with any of the recommended items.

Feedback (Idea)

Four types of feedback:

- Explicit Feedback: User interacted with the recommendations list.
  - Positive feedback: User clicked on a recommendation.
  - Negative feedback: User clicked '×' (close) on a recommendation.
- Implicit Feedback: User did not interact with any of the recommended items.

Feedback (Idea)

Four types of feedback:

- Explicit Feedback: User interacted with the recommendations list.
  - Positive feedback: User clicked on a recommendation.
  - Negative feedback: User clicked '$\times$' (close) on a recommendation.
- Implicit Feedback: User did not interact with any of the recommended items.
  - Positive feedback: But user navigated to a page which was recommended.
  - Negative feedback: And user did not navigate to any of the recommended pages.

Feedback (Idea)

Four types of feedback:

- Explicit Feedback: User interacted with the recommendations list.
  - Positive feedback: User clicked on a recommendation.
  - Negative feedback: User clicked '$\times$' (close) on a recommendation.
- Implicit Feedback: User did not interact with any of the recommended items.
  - Positive feedback: But user navigated to a page which was recommended.
  - Negative feedback: And user did not navigate to any of the recommended pages.

Feedback leads to changes in the $W_{uv}, M_v, \alpha_v, \beta_v$ values.

Experiments

- 10 days worth of real-world hit data.
- Each hit $\rightarrow$ page access.
- Data was sessionized based on the access timestamps.
- Training: 70% (first 7 days)
- Testing: 30% (last 3 days)

Baselines

Four baselines:

1. **Frequency**: Recommendation based upon the probabilistic graphical model i.e. based upon edge weights (W_{uv}) in the user navigation graph.
2. **Mass** (M_v): Recommendation based upon the average time spent on the reports seen.
3. **Context**: Recommendation based upon intent scores obtained from the current context of the user.
4. **Tensor Factorization**: Recommendation based upon obtaining latent factors only from PARAFAC2 tensor decomposition (without Kalman Filter regularization).

Metrics

Four metrics:

1. **NDCG** (Normalized Discounted Cumulative Gain)
2. **Precision@k**
3. **Recall@k**
4. **w-AUC** (weighted average Area Under the Curve)

Results

Table: Comparison of proposed system with baselines & different R_v

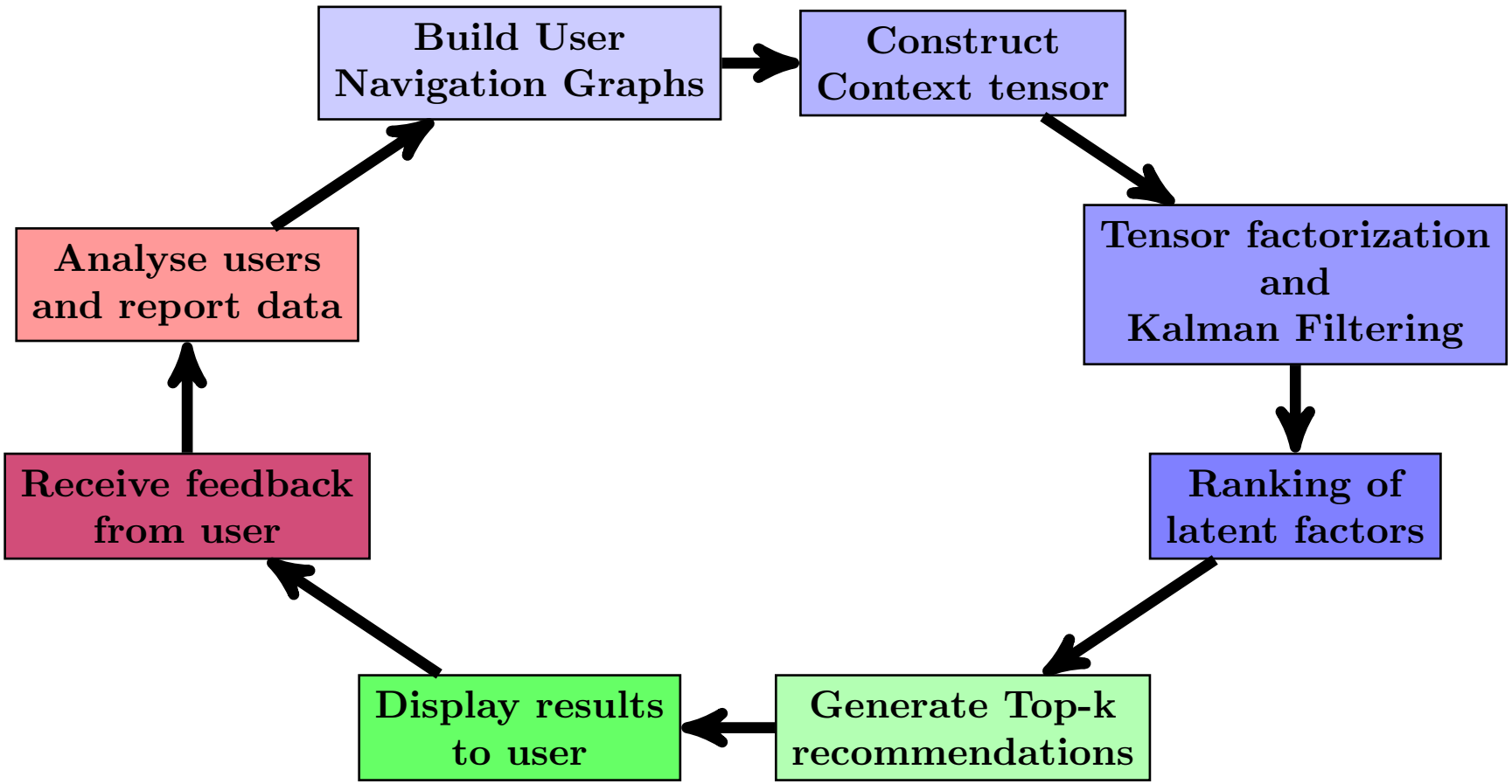
Method	NDCG	Precision	Recall	w-AUC
Mass	0.4297	0.0885	0.6181	0.5867
Frequency	0.4744	0.0853	0.5866	0.5866
Context Based	0.5228	0.1003	0.6730	0.7226
PARAFAC2 Model	0.5470	0.0987	0.6343	0.6250
Max-IxD	0.4908	0.1006	0.6753	0.6858
Dot-IxD	0.5274	0.1021	0.6911	0.7165
Max-I	0.4736	0.0969	0.6390	0.6555
Sum-I (proposed)	**0.5706**	**0.1006**	**0.6753**	**0.7239**

Future Work

- Changing training pattern (alternating days, interleaving etc.)
- Optimizing feedback factors
- Trying the system on other kinds of datasets
- Implementing optimizations to tensor factorization (the most computationally heavy aspect of this system)
- Using qualitative domain knowledge to inform user activity paths thereby enhancing the estimation of user intent

Conclusion

Cycle completed!



References

[1] Sun, Y., Yuan, N. J., Xie, X., McDonald, K., & Zhang, R. (2016, April). Collaborative nowcasting for contextual recommendation. In *Proceedings of the 25th International Conference on World Wide Web* (pp. 1407-1418). International World Wide Web Conferences Steering Committee.

[2] Kiseleva, J., Lam, H. T., Pechenizkiy, M., & Calders, T. (2013, December). Predicting current user intent with contextual markov models. In *Data mining workshops (ICDMW), 2013 IEEE 13th international conference on* (pp. 391-398). IEEE.

[3] Giannopoulos, G., Brefeld, U., Dalamagas, T., & Sellis, T. (2011, October). Learning to rank user intent. In *Proceedings of the 20th ACM international conference on Information and knowledge management* (pp. 195-200). ACM.

Thank You!