# On-the-Fly News Recommendation Using Sequential Patterns

**Mozhgan Karimi**, Boris Cule, and Bart Goethals
ADReM Research Group, University of Antwerp, Belgium

# Challenges of News Recommender Systems

- News articles are published **constantly**
- Recommended items need to be **fresh**
- Users are typically not logged in
  - Thus, there is often **no long-term history**
- News articles are often consumed in a **specific order**

# Research Goal

- How to use the community's **news consumption patterns** to predict what the current user wants to read next?
- Traditional Sequential Pattern Mining (SPM) approaches could not deal with breaking news or recent community click trends due to long training times

- An SPM approach is needed that can **incrementally update its recommendation model** with every incoming click

# Sequential Pattern Mining

- For a news reading session

  A | B | C

  we want to **extract every (sub-) pattern** like so:

  A    B    A | B    C    A | C    B | C    A | B | C

- Equivalent to calculating the **power set** of the session
- How to **do this incrementally** based on session snapshots?

# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Power Set Fragment |
| --- | --- | --- | --- |
| A | A | | |

# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Power Set Fragment |
|---|---|---|---|
| A | A | | |

# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Partial Power Set Fragment |
| --- | --- | --- | --- |
| A | A | | A |

# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Power Set Fragment |
|---|---|---|---|
| A | A | | A |
| B | A B | | |

# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Power Set Fragment |
|---|---|---|---|
| A | A | | A |
| B | A B | A | |

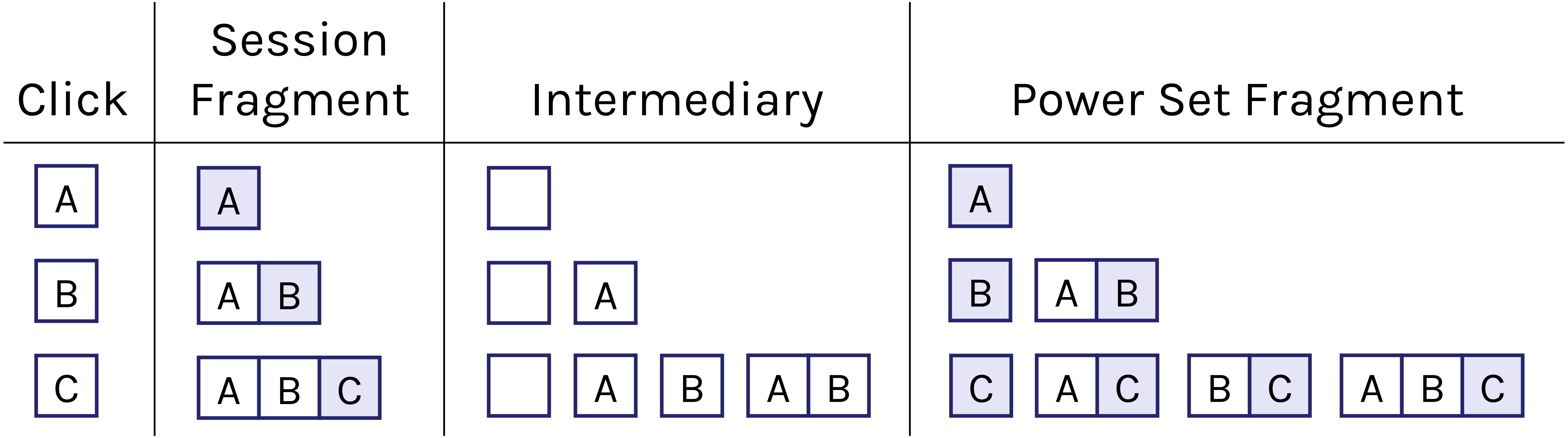
# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Power Set Fragment |
|---|---|---|---|
| A | A | | A |
| B | A B | A | B, A B |

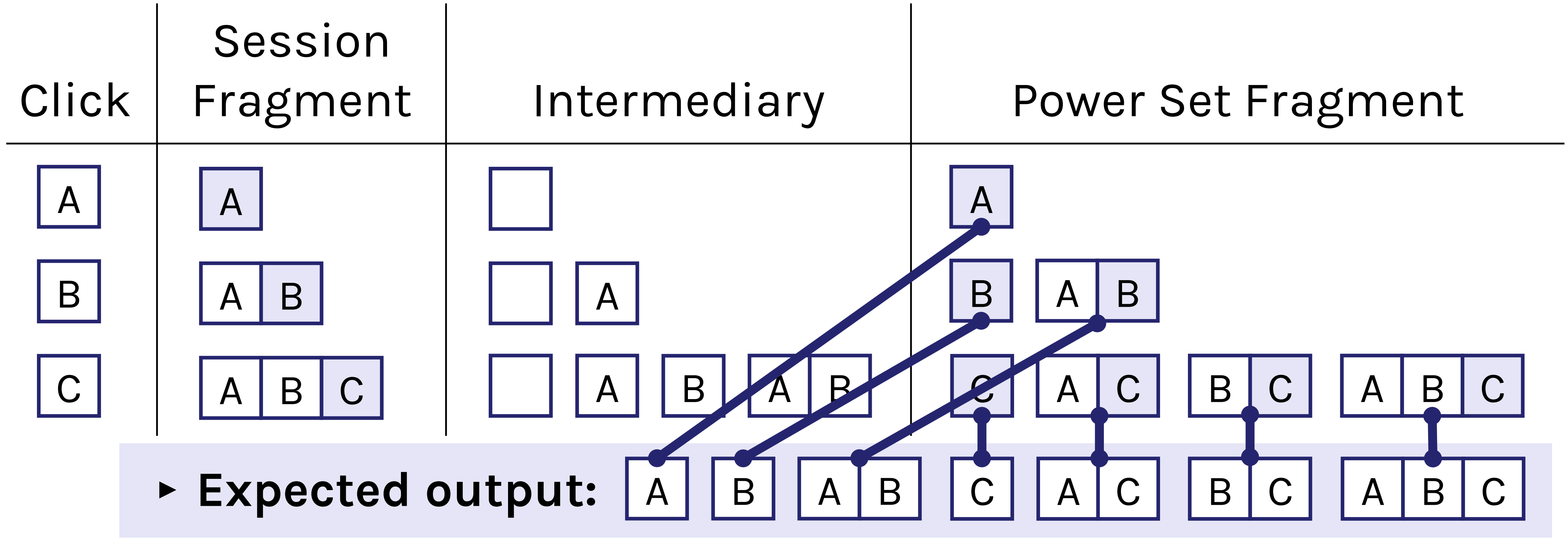
# Incremental Power Set Creation

| Click | Session Fragment | Intermediary | Power Set Fragment |
|---|---|---|---|
| A | A | □ | A |
| B | A B | □   A | B   A B |
| C | A B C | □   A   B   A B | |

# Incremental Power Set Creation



# Incremental Power Set Creation

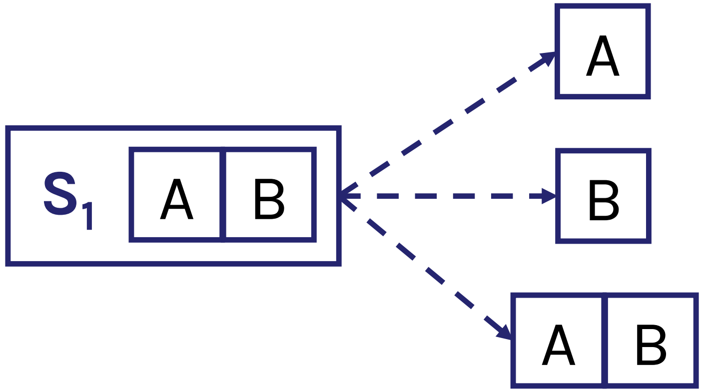


# Recommendation Model

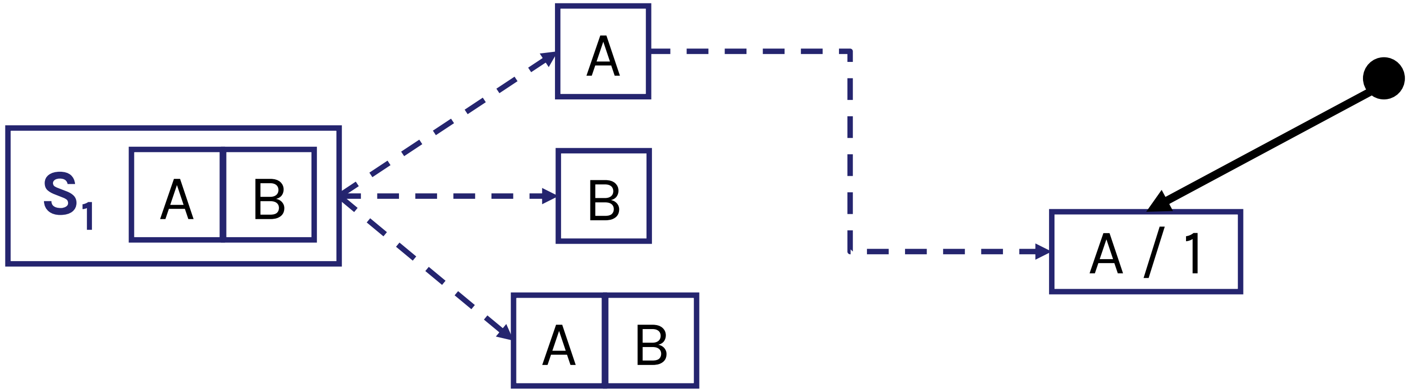
- To generate recommendations from extracted patterns efficiently, patterns need to be stored in a model that allows:
  - … quick updates
  - … quick calculation of recommendation scores

- Solution: An incrementally updateable **pattern tree model**

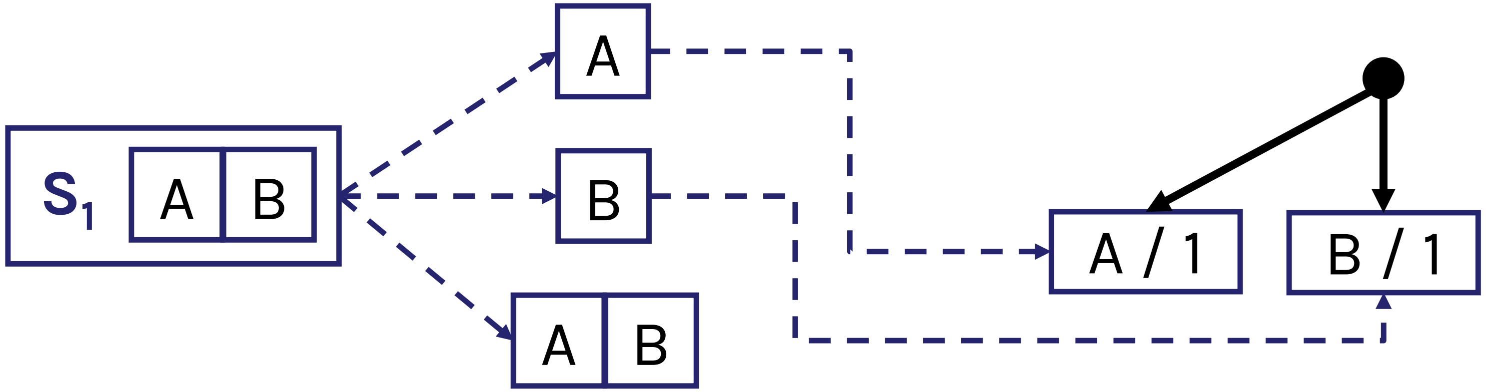
# Pattern Tree Model



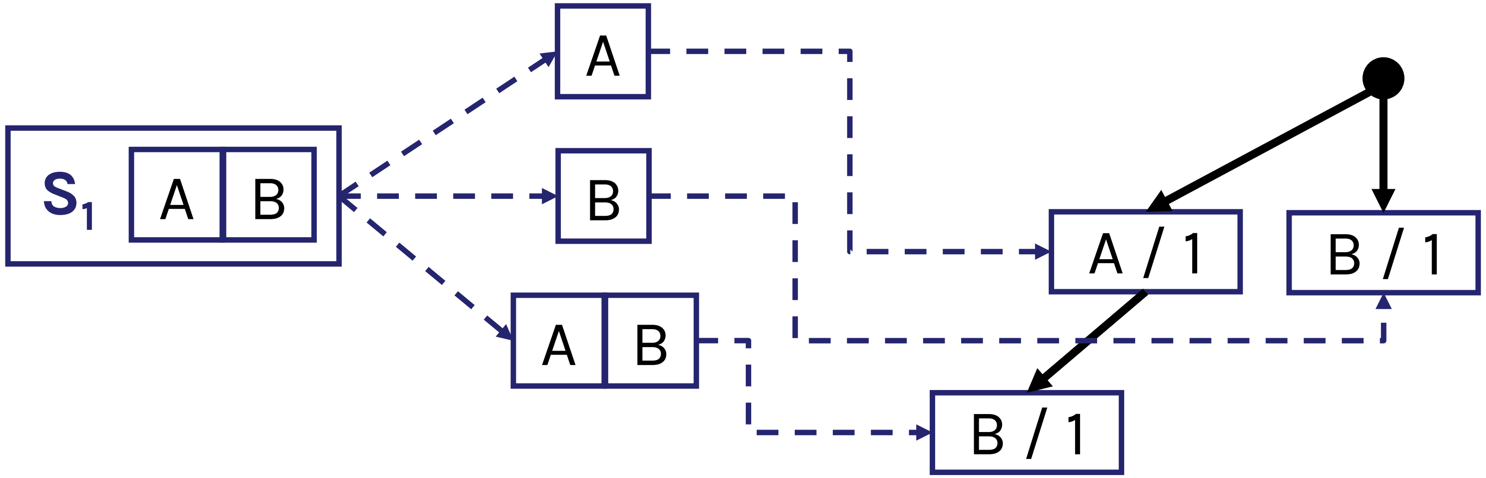
# Pattern Tree Model



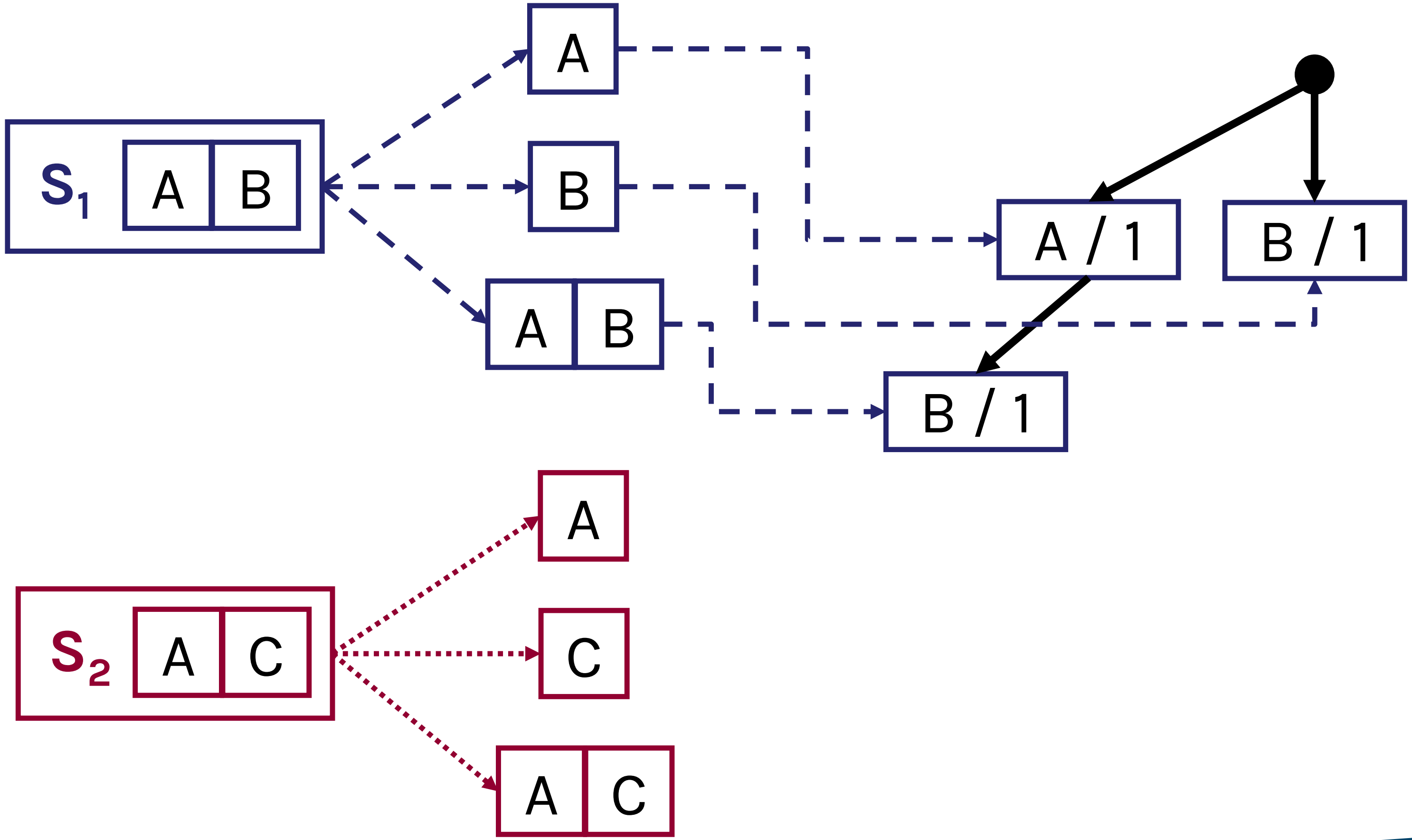
# Pattern Tree Model



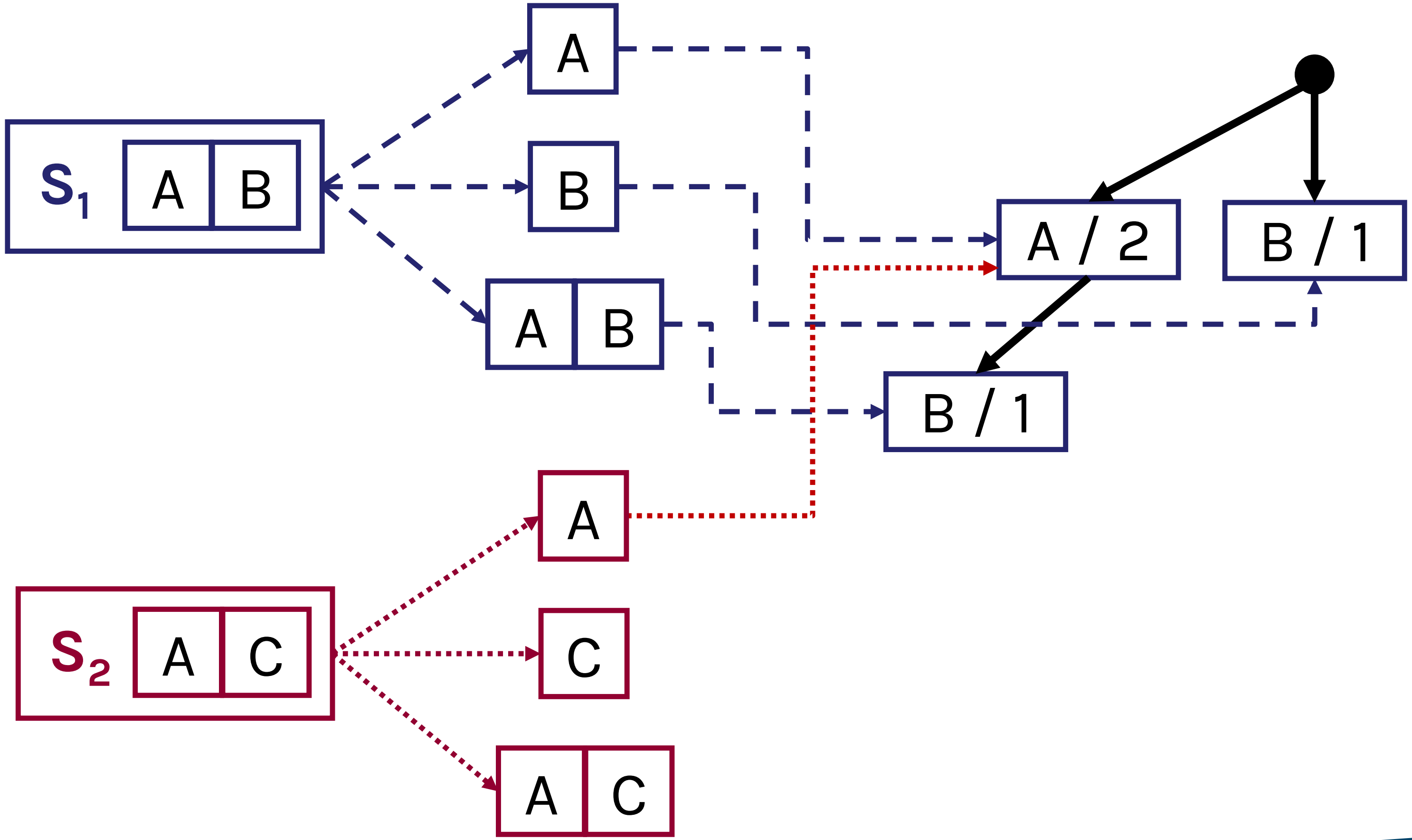
# Pattern Tree Model



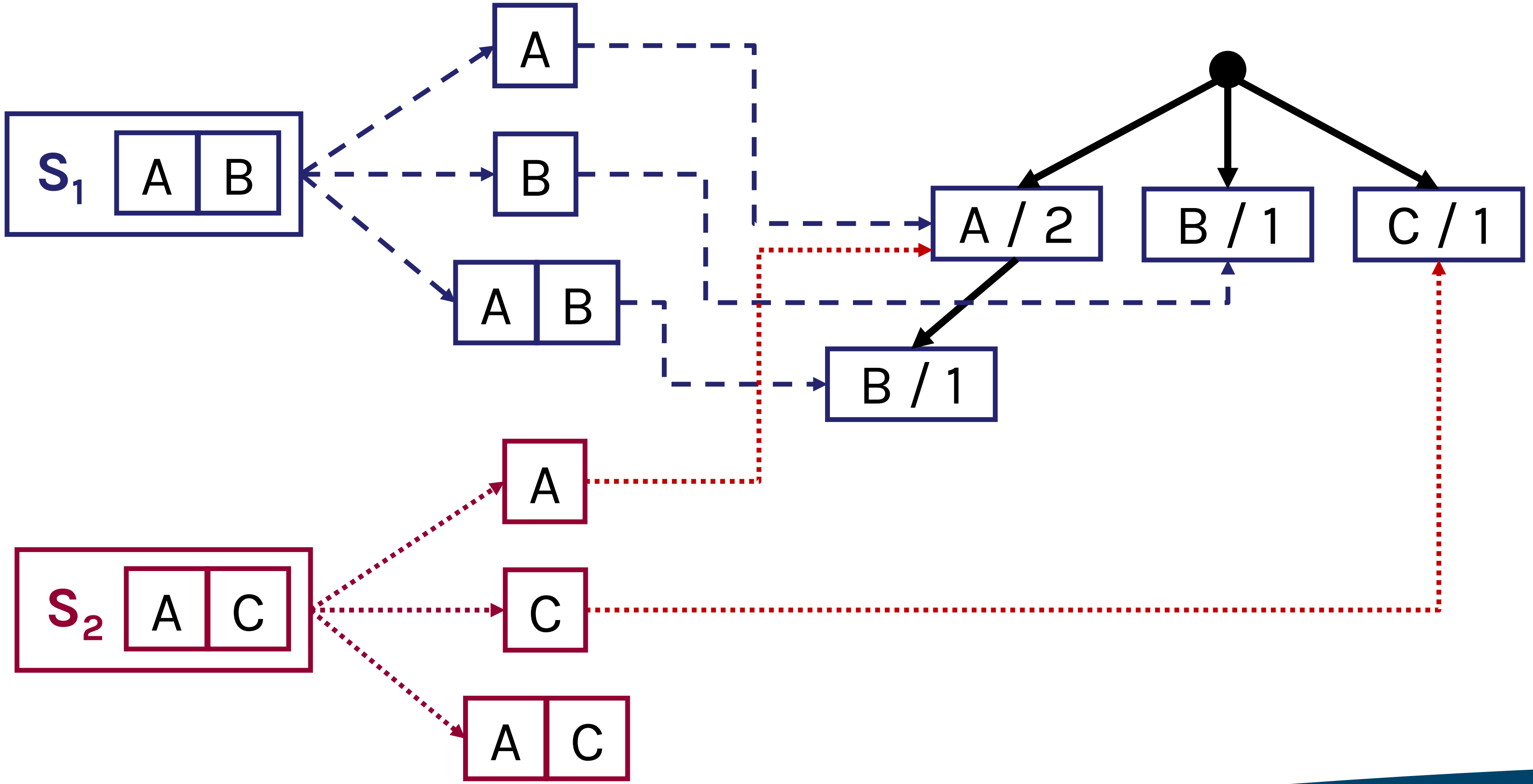
# Pattern Tree Model



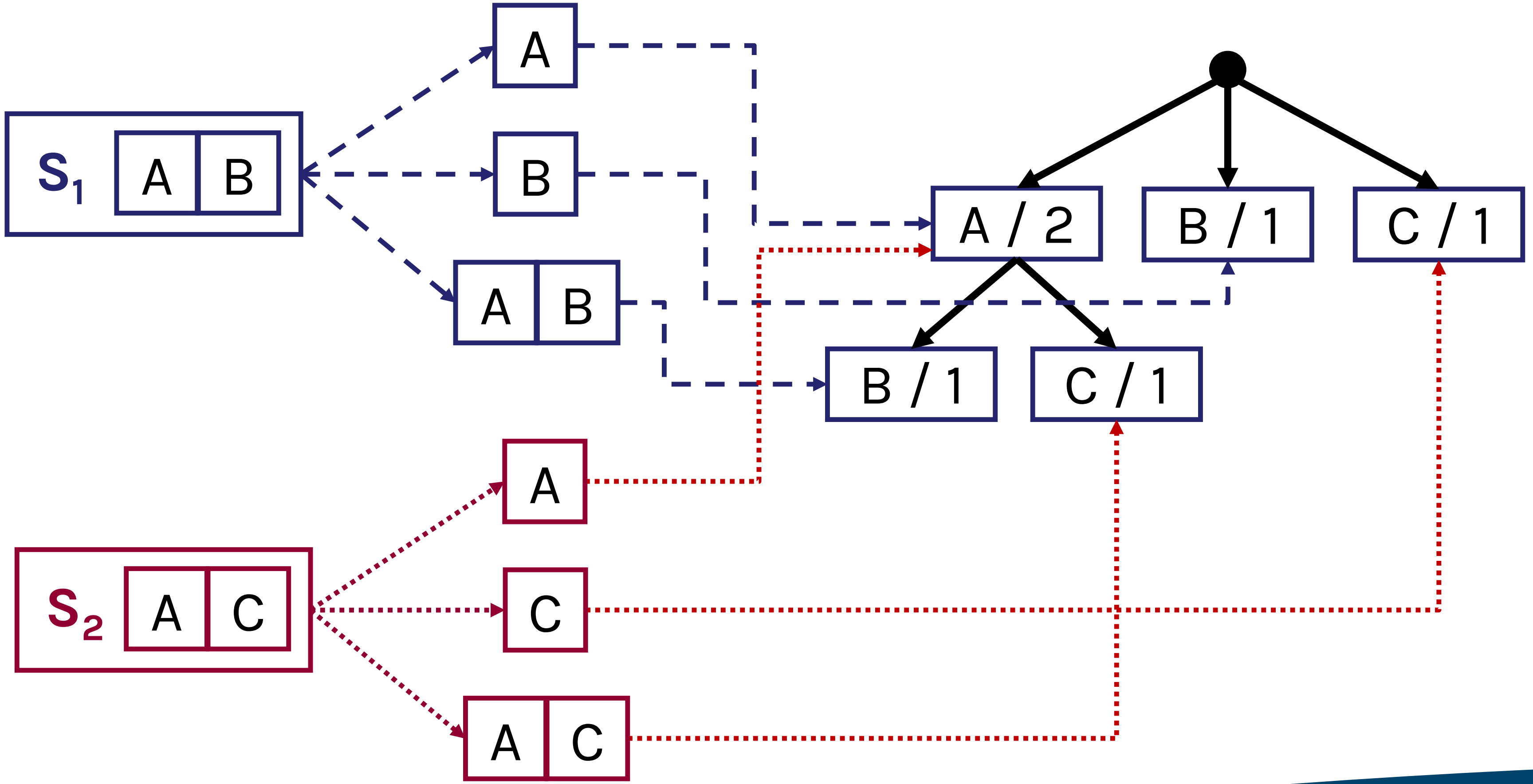
# Pattern Tree Model



# Pattern Tree Model

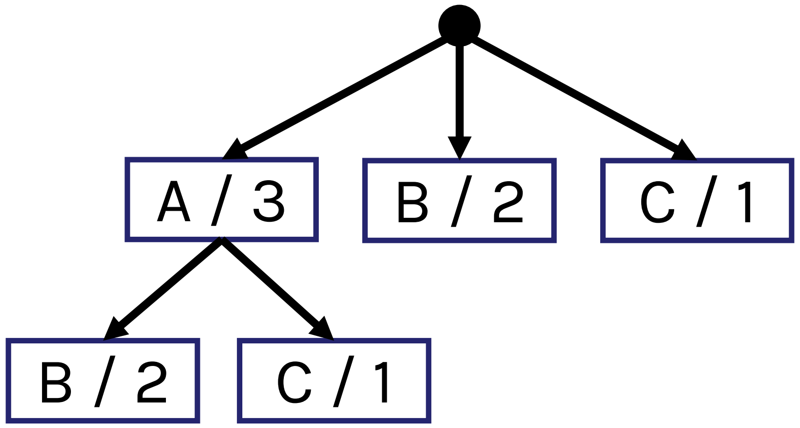


# Pattern Tree Model

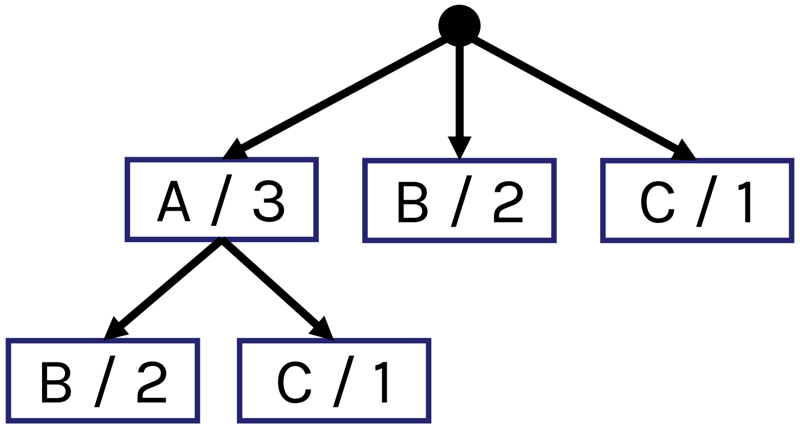


# Recommendation Scoring

- Given a session (e.g., $\boxed{A}$),
  build the power set
  (in this case also $\boxed{A}$)



# Recommendation Scoring

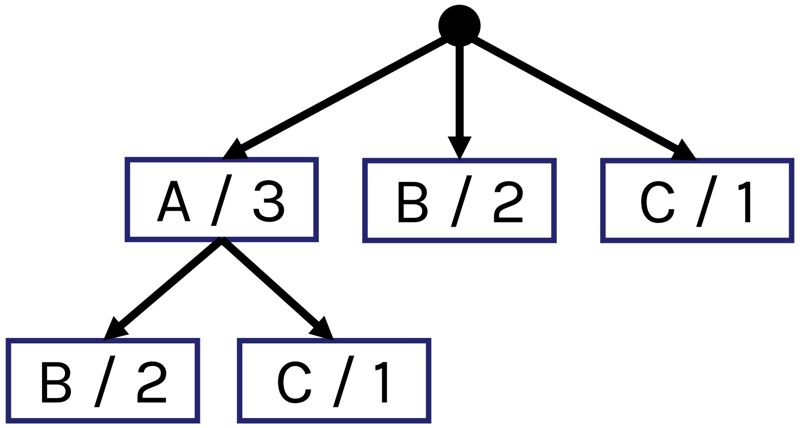



- Given a session (e.g., $\boxed{A}$),
  build the power set
  (in this case also $\boxed{A}$)
- Then, traverse the tree based on
  every sub-pattern of the user's current session and aggregate
  confidence scores for each candidate item $i$ calculated like so:

# Recommendation Scoring

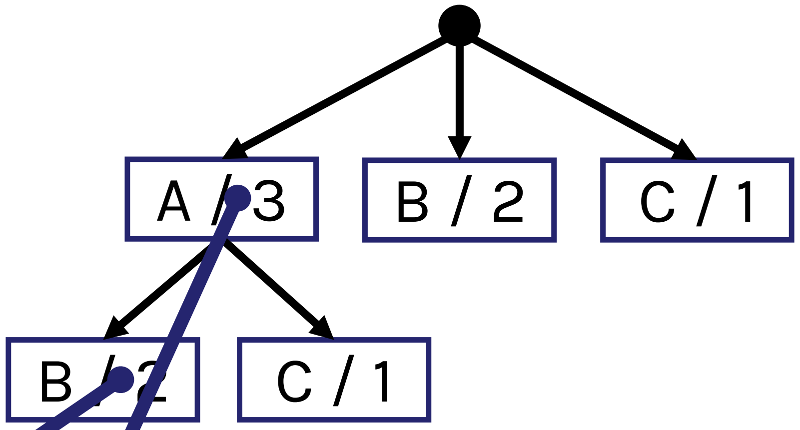
- Given a session (e.g., $\boxed{A}$), build the power set (in this case also $\boxed{A}$)
- Then, traverse the tree based on every sub-pattern of the user's current session and aggregate confidence scores for each candidate item $i$ calculated like so:

$$conf(i \in P) = {frequency(P)} \Big/ {frequency(P \setminus i)}$$



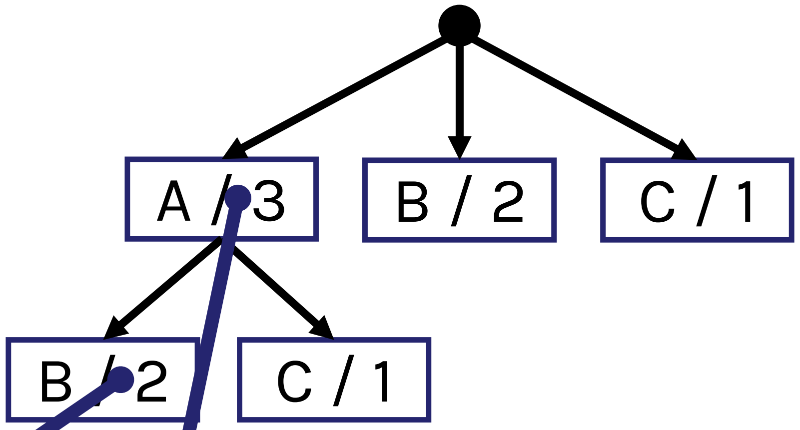
# Recommendation Scoring

- Given a session (e.g., $\boxed{A}$), build the power set (in this case also $\boxed{A}$)
- Then, traverse the tree based on every sub-pattern of the user's current session and aggregate confidence scores for each candidate item $i$ calculated like so:




$$conf(i \in P) = {frequency(P)}/{frequency(P \setminus i)}$$

# Recommendation Scoring




- Given a session (e.g., $\boxed{\text{A}}$), build the power set (in this case also $\boxed{\text{A}}$)
- Then, traverse the tree based on every sub-pattern of the user's current session and aggregate confidence scores for each candidate item $i$ calculated like so:

$$conf(i \in P) = {frequency(P)} \Big/ {frequency(P \setminus i)}$$

- Thus, $conf(B) = 2/3$ and $conf(C) = 1/3$

# Tweaks

- Default approach: **Seq**
- Approach that reduces the support values of “stale” patterns based on a click queue implementation: $\mathbf{Seq_r}$
- Approach with a slightly modified scoring method that penalizes candidate items from longer patterns: $\mathbf{Seq_p}$
- Combination of recency and penalization varieties: $\mathbf{Seq_{pr}}$

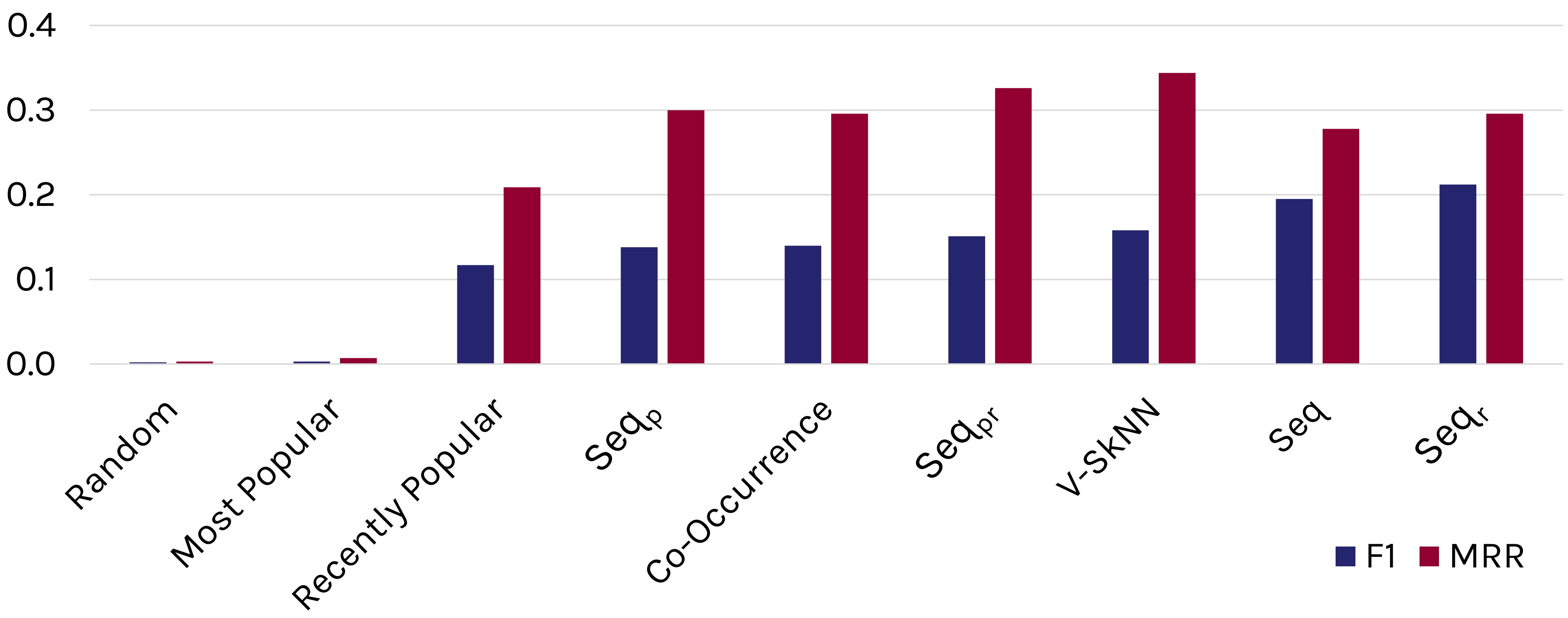
# Evaluation

- The experiments are designed based on the StreamingRec framework, which simulates real-time recommendation and allows algorithms to learn incrementally

  Michael Jugovac, Dietmar Jannach, and Mozhgan Karimi. 2018. StreamingRec: A Framework for Benchmarking Stream-based News Recommenders. In *Proceedings of the 12th Conference on Recommender Systems*

- Evaluation on two real-world data sets from Plista and Outbrain

# Results


0.4
0.3
0.2
0.1
0.0
Random
Most Popular
Recently Popular
Seq$_p$
Co-Occurrence
Seq$_{pr}$
V-SkNN
Seq
Seq$_r$
F1
MRR

# Conclusions

- The proposed approach is **more effective in terms of F1** than previous NRS methods
  - Also: Much **more efficient** (~0.2 ms per request)
- For more in-depth results, come to the poster!

- Future work
  - Weighted scoring for candidate items
  - Personalized variant that considers user history