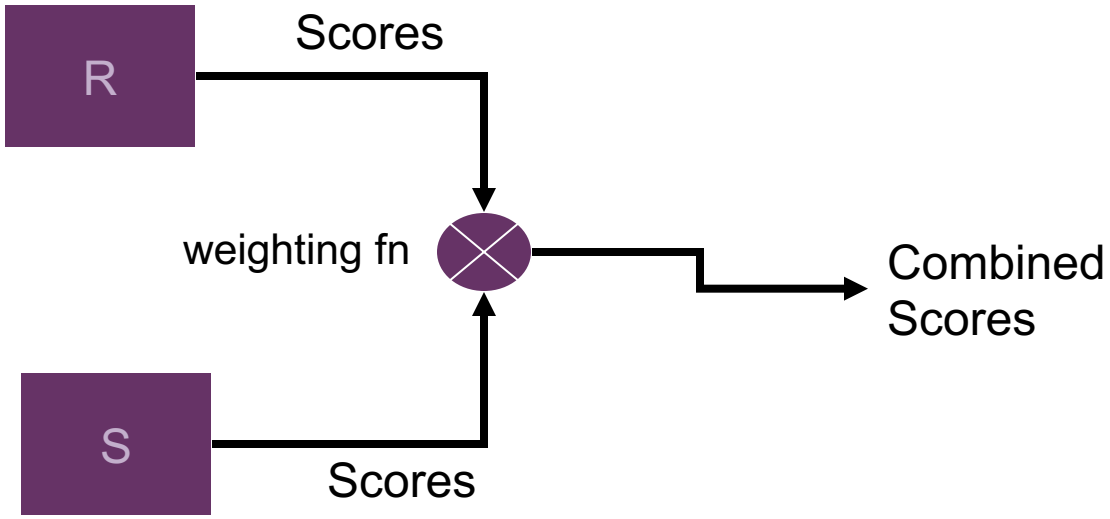
Recommender Systems Hybrid Methods

Professor Robin Burke
Spring 2019

Combination Methods

- Simplest
  - "keep modules separate"
  - weighted
  - mixed
  - switching
  - cascade
- More integrated
  - feature combination
  - feature augmentation
  - metalevel

Weighted (Homework 3)



Weighted Hybrid

- like the semantically-enhanced example
- Two recommendation modules
  - R and S
  - $Pred(u, i)=\alpha\ Pred_R(u,i) + (1- \alpha)Pred_S(u,i)$

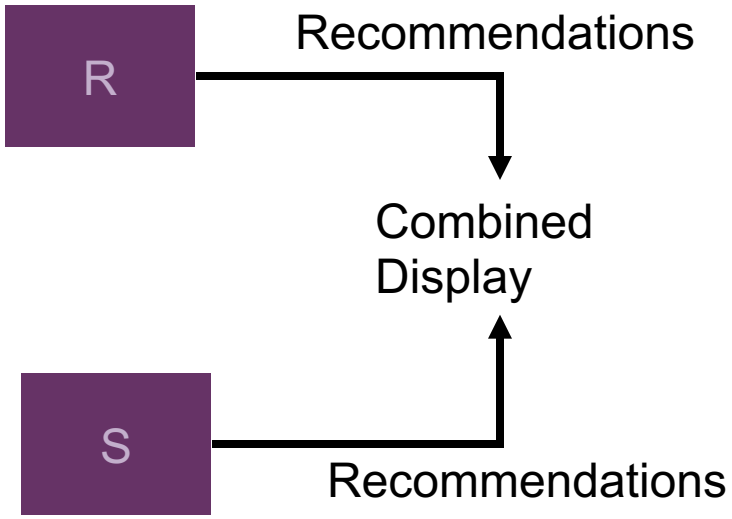
Issues

- Very simple to implement
- Can combine multiple components
- Easy to adjust the reliance on each algorithm
- Assumes uniform reliability across user/item space
  - often not the case
- How to pick the items to be rated?
  - collaborative can only use items rated by neighbors
  - options
    - Items rateable by R $\cap$ Items rateable by S
      - might be empty
    - Items rateable by R $\cup$ Items rateable by S
      - might contain items not rateable by both algorithms
    - Or have a ”fall back” option

Mixed Hybrid

- Also simple
- Display the recommendation together
  - in separate lists
  - or mixed together
- PTV (Smyth and Cotter 2000)
  - collaborative and content-based recommendations
  - mixed in recommendation list

Mixed



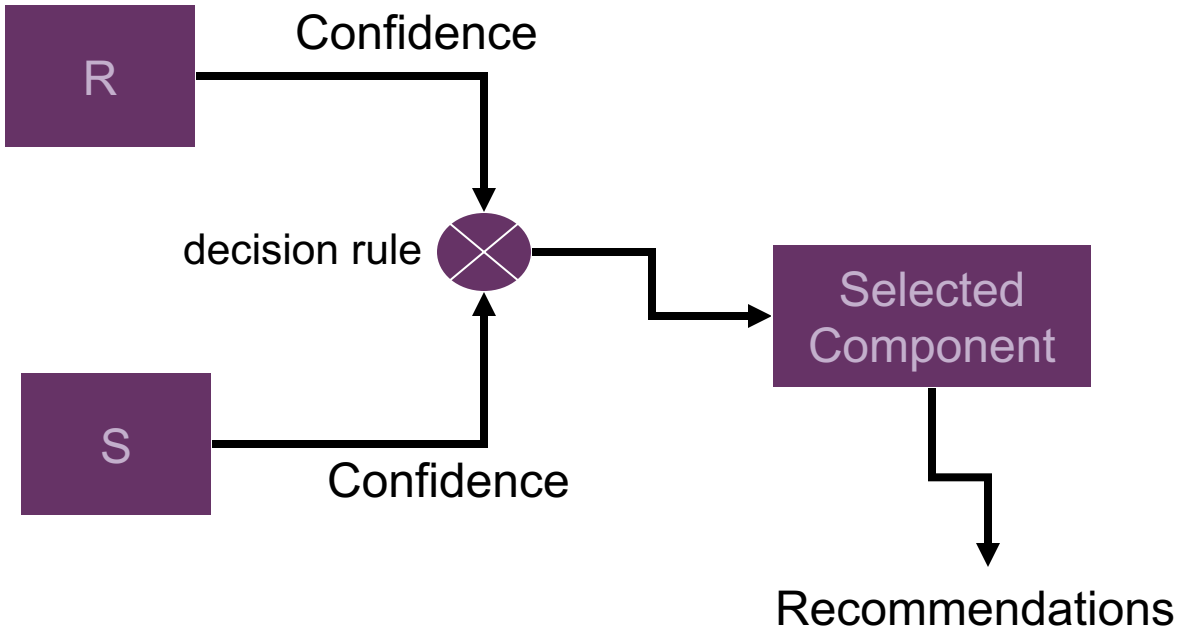
Issues

- UI Issues
  - Do customers need to distinguish between different kinds of recommendations?
- No synergy between knowledge sources

Switching

- Switch between recommendation components
  - (van Setten 2005)
- Needed
  - confidence measure
    - c(R,u,i)
    - measures how confident the component R is in predicting a rating for user u for item i
    - usually this requires making a prediction
  - decision rule
    - given the confidence results which recommender should be selected

Switching


R
Confidence
decision rule
S
Confidence
Selected Component
Recommendations

Confidence

- Ideally it should be sensitive to user and item
- Collaborative recommendation
  - # of useful neighbors
  - distance of closest useful neighbor
  - density of ratings for item
- Content-based recommendation
  - naive Bayes output has probabilistic interpretation
    - however, these values are very unreliable
- Knowledge-based recommendation
  - some work by Cheetham (2005) in learning a content-based decision tree to predict accuracy of recommender

Issues

- Confidence measures on different algorithms may not be comparable
  - fine-tuning of the decision rule may be necessary
- Sufficient data to learn decision threshold
- Could be treated as a discrete optimization version of the weighted hybrid
  - where the weights can only be 1 or 0
  - Dynamic programming solutions

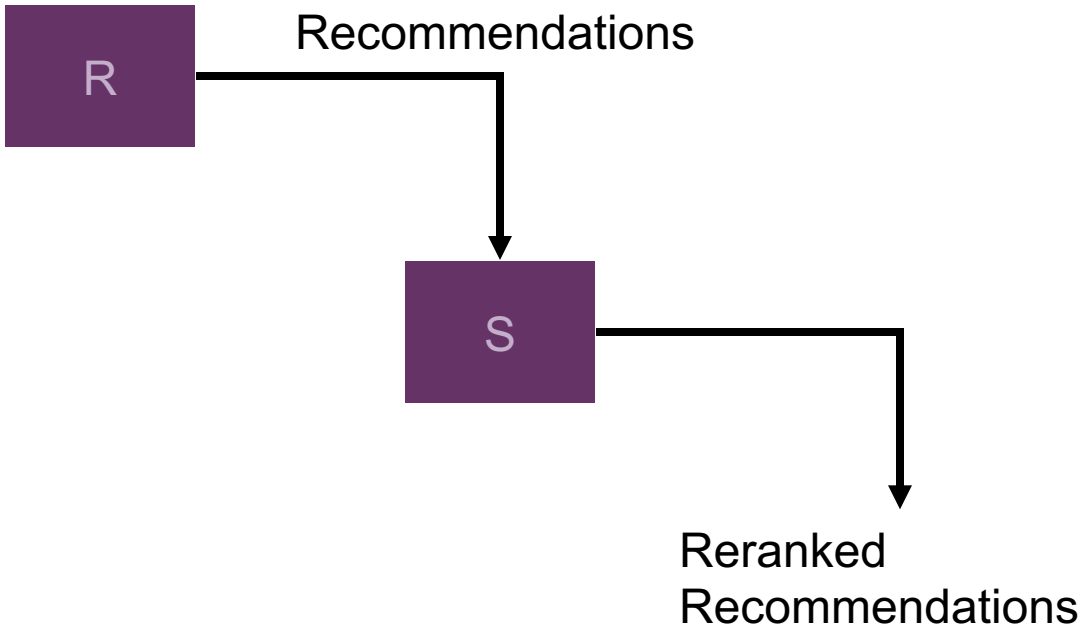
You are trying to decide whether to use a weighted or a switching hybrid for recommendations. Your collaborative component works well for users with a lot of ratings but poorly on cold-start users. Your content-based component has about the same performance across the users. Choose:

- A. Weighted
- B. Switching
- C. A different hybrid would be better

Cascade

- Recommendation hierarchy
  - use 2nd recommender to break ties
- Useful if one recommender is powerful but coarse
  - results can be fine-tuned
- Example
  - knowledge-based recommendation

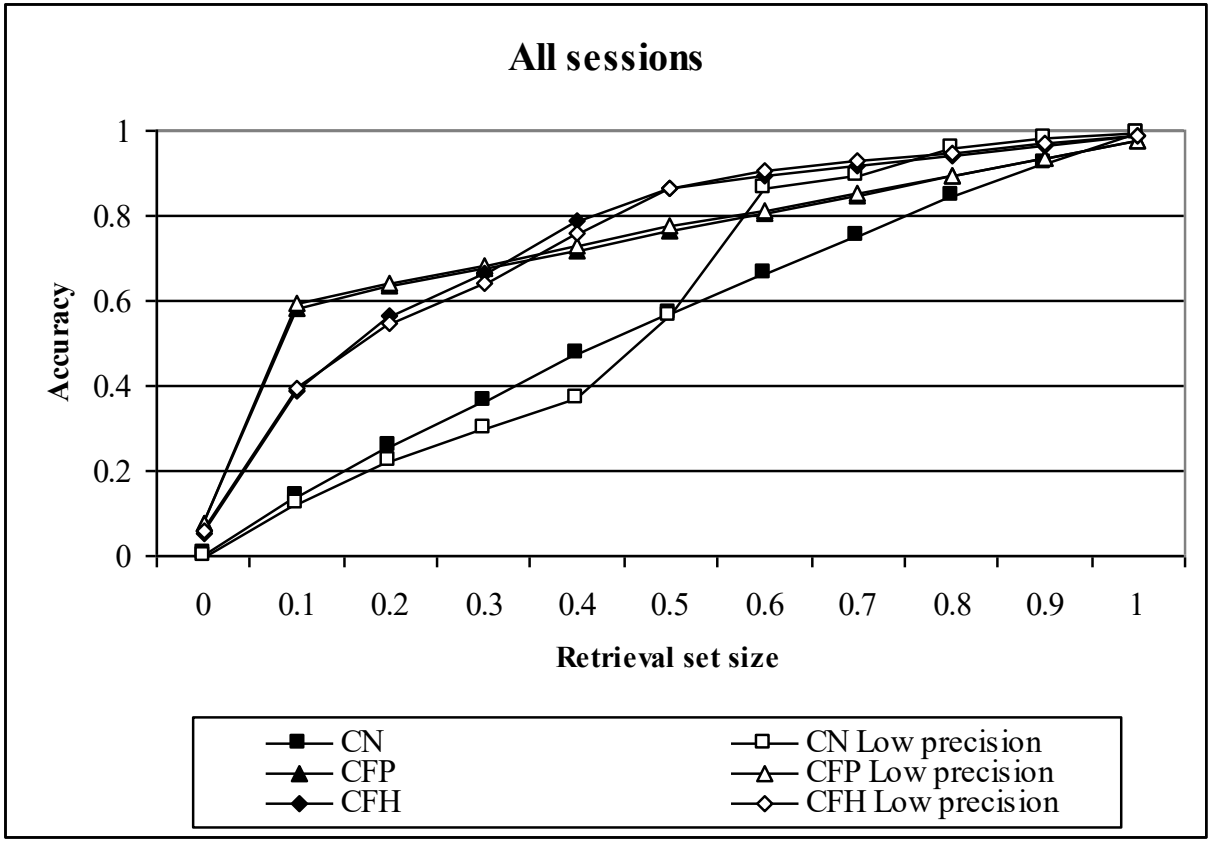
Cascade



Confidence Interval

- Ratings are real-valued
  - ties might be rare
- But what is the appropriate precision of a predicted rating?
  - obviously not the full 32-bit floating point number

Cost of imprecision



Issues

- May require de-precisioning ratings
- Best if one recommender is always stronger
- Re-ranking idea will return again in Week 14 (Fairness)

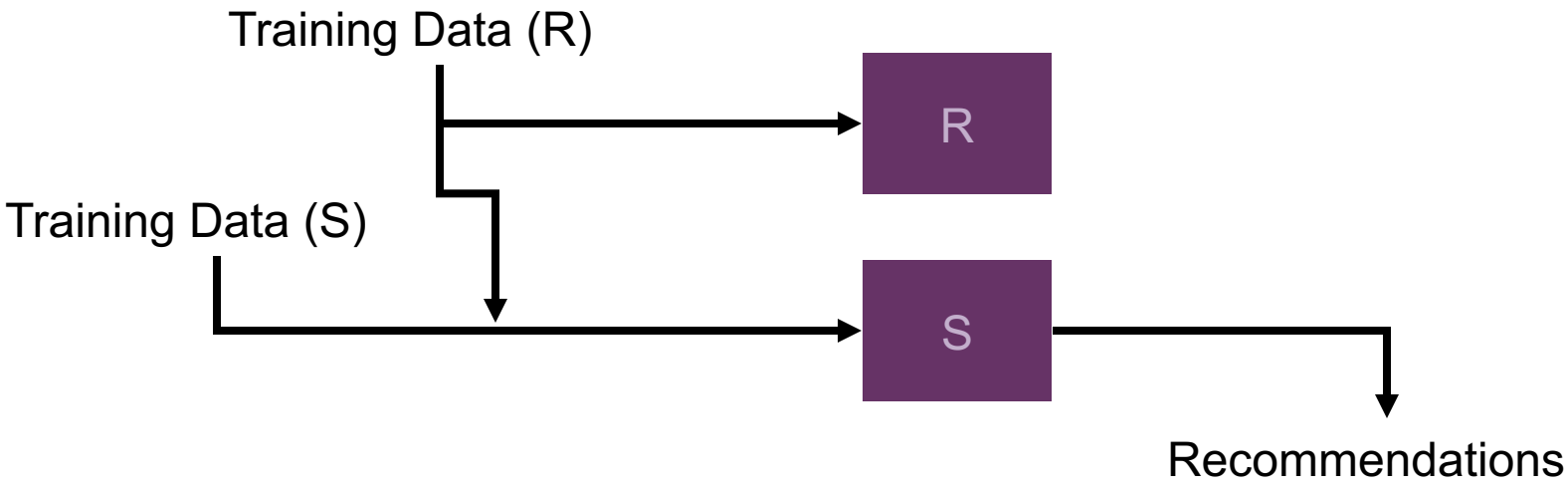
Feature Combination

- We can combine recommendation logics
  - without having separate components
- Take features of one knowledge type
  - use as input to recommender of another type

Examples

- Collaborative+Content
  - for each feature
    - create a "pseudo-user"
      - likes all of the items with that feature
      - dislike all without
  - collaborative algorithm works as before
    - but now feature similarity has a role
- Content+Collaborative
  - for each user
    - create a feature ("liked by u")
    - or two features ("liked by u", "disliked by u")
  - add these features to descriptions of each item
  - (can't use naive Bayes, though)
    - Winnow algorithm handles more features

Combination



Issues

- Doesn't require modifying the original algorithm
  - cheap way to overcome new item problem
- Addition of features will (usually) increase sparsity
- Doesn’t (easily) work with knowledge-based

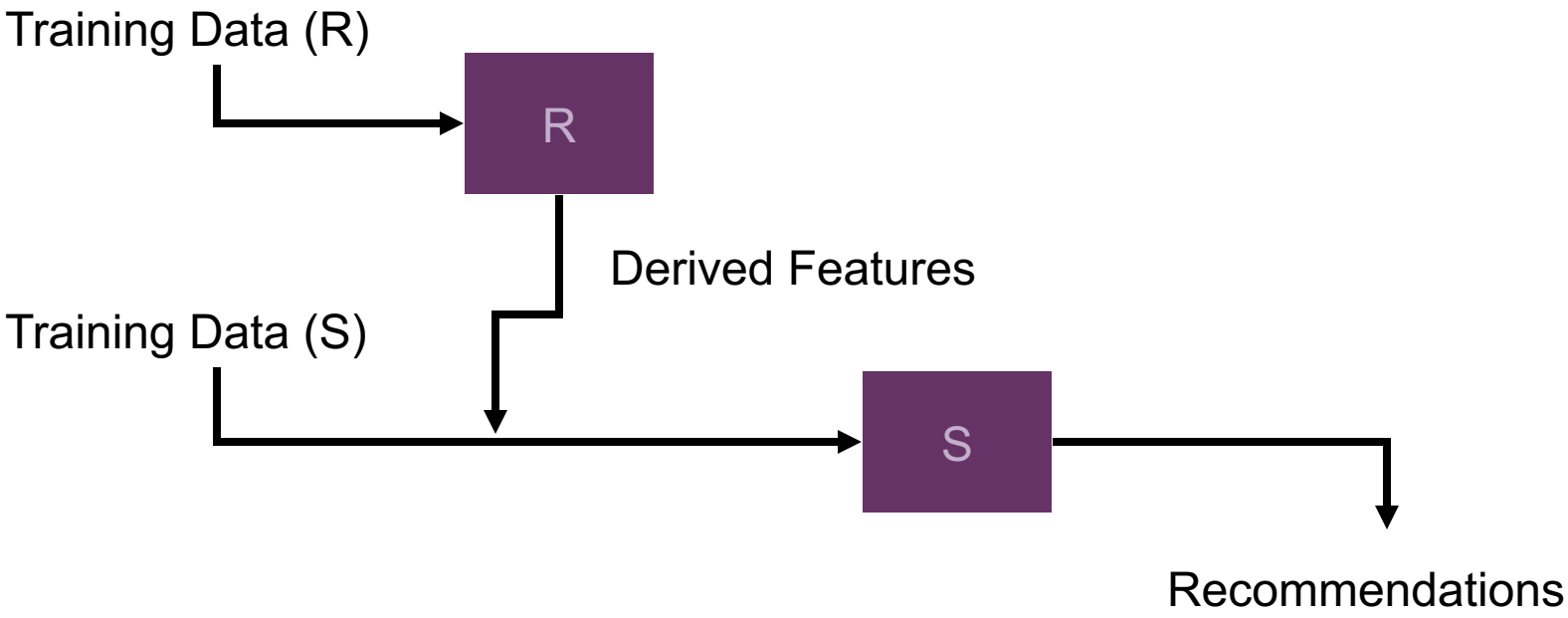
Feature Augmentation

- Feature combination
  - simple but can generate great sparsity
- Feature augmentation
  - use recommendation logic of R
    - to produce a feature (or features)
    - input to S
    - augmenting its normal input

Example

- Content+Collaborative
  - Cluster items based on features
  - Consider each cluster as a psuedo-user
    - likes each item in the cluster
  - Not as sparse as feature combination
- Collaborative+Knowledge-based
  - Cluster items based on ratings
  - Create binary similarity rule
    - prefer items in same cluster

Augmentation



Issues

- Must be possible to use recommendation logic to produce features
  - clustering can work
  - "embedding" methods are another example
- Augmentation can be done off-line
  - efficient
- Reduce dimensionality of recommendation features
- Assumes S is more reliable
  - just needs help

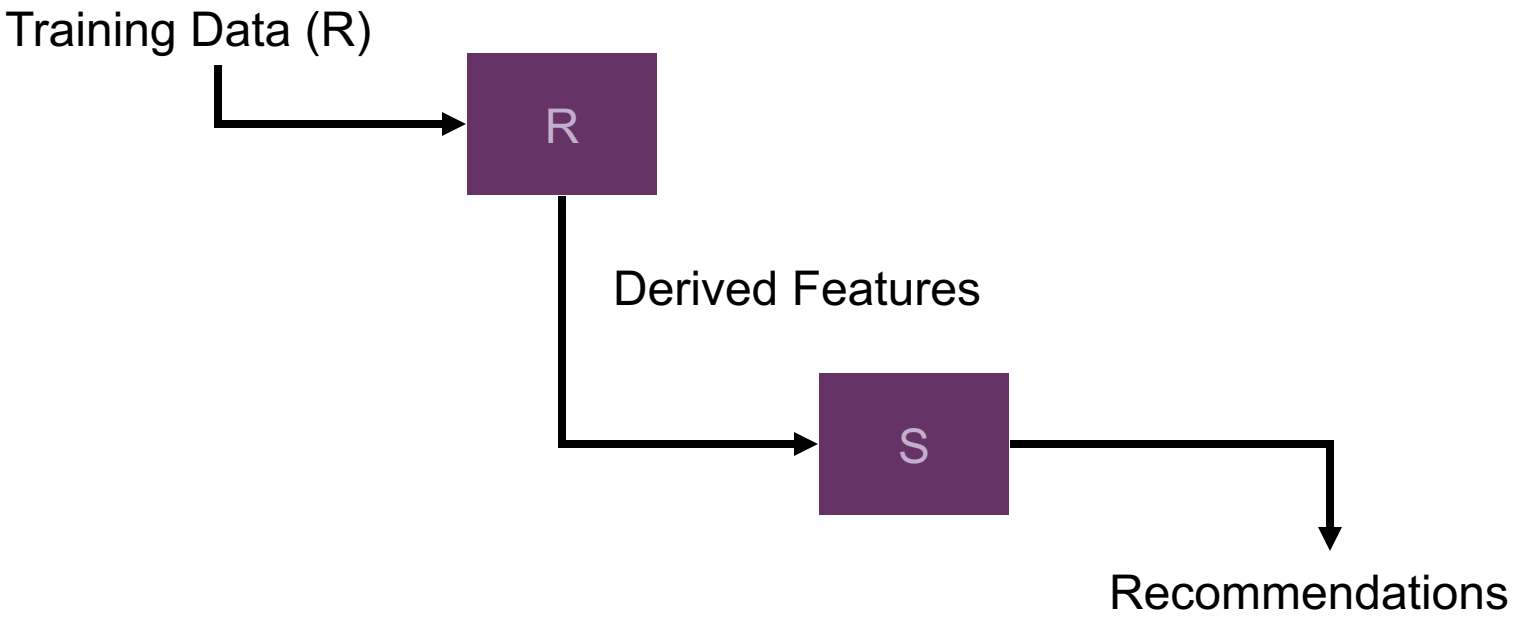
Meta-Level

- Push feature augmentation to its limit
  - what if we replace the input to S entirely
- S no longer has raw training data
  - but data processed by R's recommendation logic
  - "Change of basis" in recommendation space

Examples

- Content-Based+Collaborative
  - Content-based system learns a profile
    - naive Bayes / PLSA
  - Collaborative system compare these profiles
    - rather than raw ratings
- Content-Based+Knowledge-Based
  - Content-based system learns a profile
  - For a given user
    - used top features for a query

Meta-level



Issues

- Assumes strong components
  - weakness in R will cripple S
- Assumes R produces a usable representation
  - may be incompatible with some algorithms

You implement a content-based deep learning model that learns a representation of items based on their content. The output layer of this model maps to the latent factors in a collaborative factorization model, so the model basically learns the mapping from content features to the latent factor representation of each item. You could use this component as part of which type of hybrid:

- A. Switching hybrid for cold-start items
- B. Meta-hybrid with a second collaborative component
- C. Feature combination hybrid