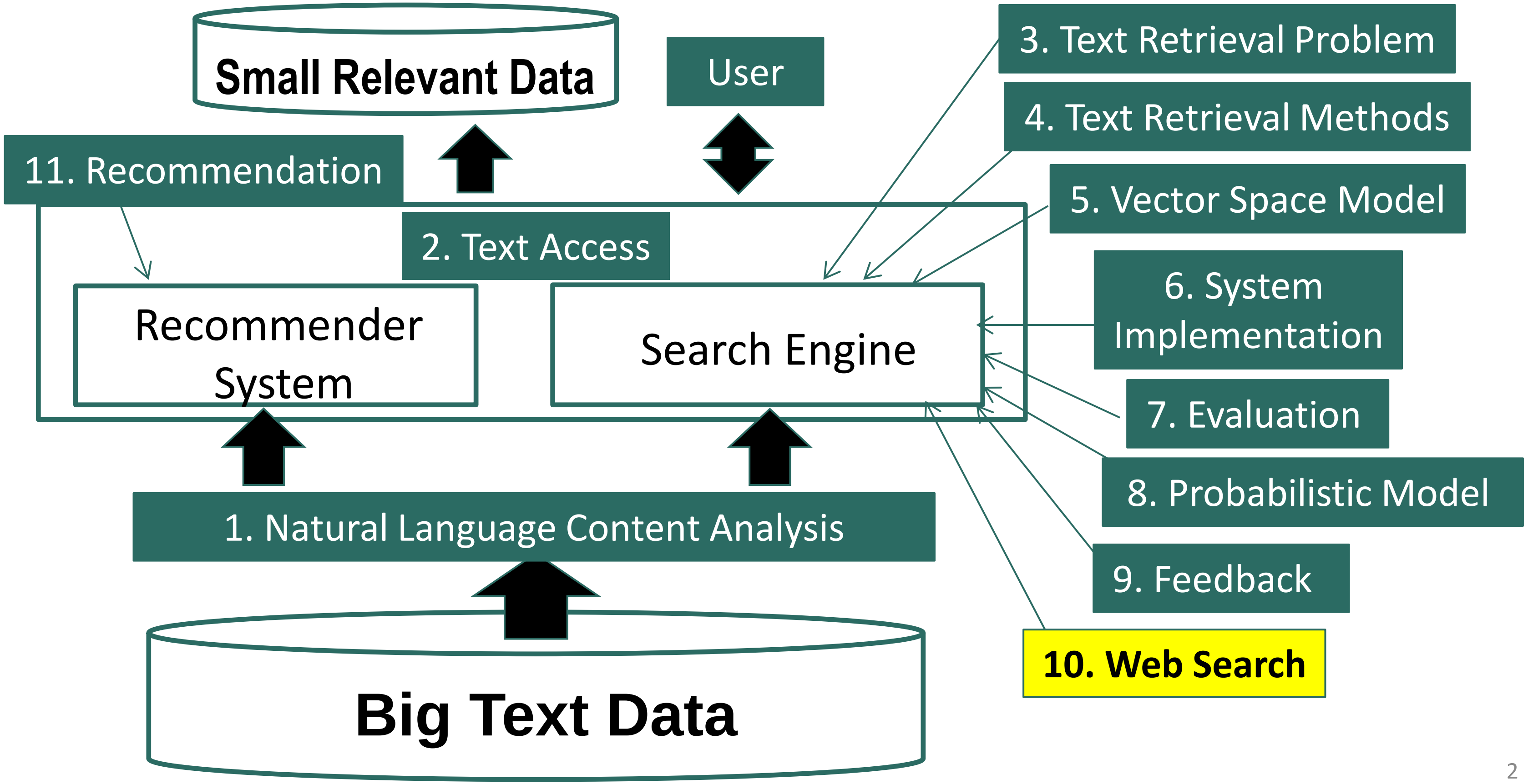
Text Retrieval and Search Engines

Web Search: Learning to Rank - Part 1 - 3

ChengXiang "Cheng" Zhai
Department of Computer Science
University of Illinois at Urbana-Champaign

Web Search: Learning to Rank



1. Learning to Rank (Parts 1, 2, 3)
How Can We Combine Many Features?

- General idea:
  - Given a query-doc pair (Q,D), define various kinds of features Xi(Q,D)
  - Examples of feature: the number of overlapping terms, BM25 score of Q and D, p(Q|D), PageRank of D, p(Q|Di), where Di may be anchor text or big font text, “does the URL contain ‘~’?”….
  - Hypothesize $p(R=1|Q,D)=s(X1(Q,D),…,Xn(Q,D), \lambda)$ where λ is a set of parameters
  - Learn λ by fitting function s with training data, i.e., 3-tuples like (D, Q, 1) (D is relevant to Q) or (D,Q,0) (D is non-relevant to Q)

Regression-Based Approaches

Logistic Regression: Xi(Q,D) is feature; β's are parameters

$$\log \frac{P(R=1|Q,D)}{1-P(R=1|Q,D)} = \beta_0 + \sum_{i=1}^{n} \beta_i X_i$$

Estimate β's by maximizing the likelihood of training data

$$P(R=1|Q,D) = \frac{1}{1+\exp(-\beta_0 - \sum_{i=1}^{n} \beta_i X_i)}$$

	X1(Q,D) BM25	X2 (Q,D) PageRank	X3(Q,D) BM25Anchor
D1 (R=1)	0.7	0.11	0.65
D2 (R=0)	0.3	0.05	0.4

$$p(\{(Q,D_1,1),(Q,D_2,0)\}) = \frac{1}{1+\exp(-\beta_0 - 0.7\beta_1 - 0.11\beta_2 - 0.65\beta_3)} * (1 - \frac{1}{1+\exp(-\beta_0 - 0.3\beta_1 - 0.05\beta_2 - 0.4\beta_3)})$$

$$\vec{\beta}^* = \arg\max_{\vec{\beta}} \, p(\{(Q_1,D_{11},R_{11}),(Q_1,D_{12},R_{12}),....,(Q_n,D_{m1},R_{m1}),...\})$$

Once β's are known, we can take Xi(Q,D) computed based on a new query and a new document to generate a score for D w.r.t. Q.

More Advanced Learning Algorithms

- Attempt to directly optimize a retrieval measure (e.g. MAP, nDCG)
  - More difficult as an optimization problem
  - Many solutions were proposed [Liu 09]
- Can be applied to many other ranking problems beyond search
  - Recommender systems
  - Computational advertising
  - Summarization
  - …

Summary

- Machine learning has been applied to text retrieval since many decades ago (e.g., Rocchio feedback)
- Recent use of machine learning is driven by
  – Large-scale training data available
  – Need for combining many features
  – Need for robust ranking (again spams)
- Modern Web search engines all use some kind of ML technique to combine many features to optimize ranking
- Learning to rank is still an active research topic

Additional Readings

- Tie-Yan Liu. Learning to Rank for Information Retrieval. Foundations and Trends in Information Retrieval 3, 3 (2009): 225-331.
- Hang Li. A Short Introduction to Learning to Rank, IEICE Trans. Inf. & Syst. E94-D, 10 (Oct. 2011): n.p.

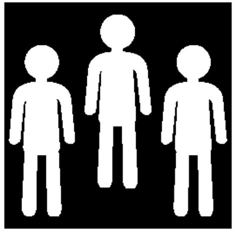
4. The Future of Web Search
Next Generation Search Engines

- More specialized/customized (vertical search engines)
  - Special group of users (community engines, e.g., Citeseer)
  - Personalized (better understanding of users)
  - Special genre/domain (better understanding of documents)
- Learning over time (evolving)
- Integration of search, navigation, and recommendation/filtering (full-fledged information management)
- Beyond search to support tasks (e.g., shopping)
- Many opportunities for innovations!

The Data-User-Service (DUS) Triangle

Lawyers
Scientists
UIUC employees
Online shoppers
…

Users



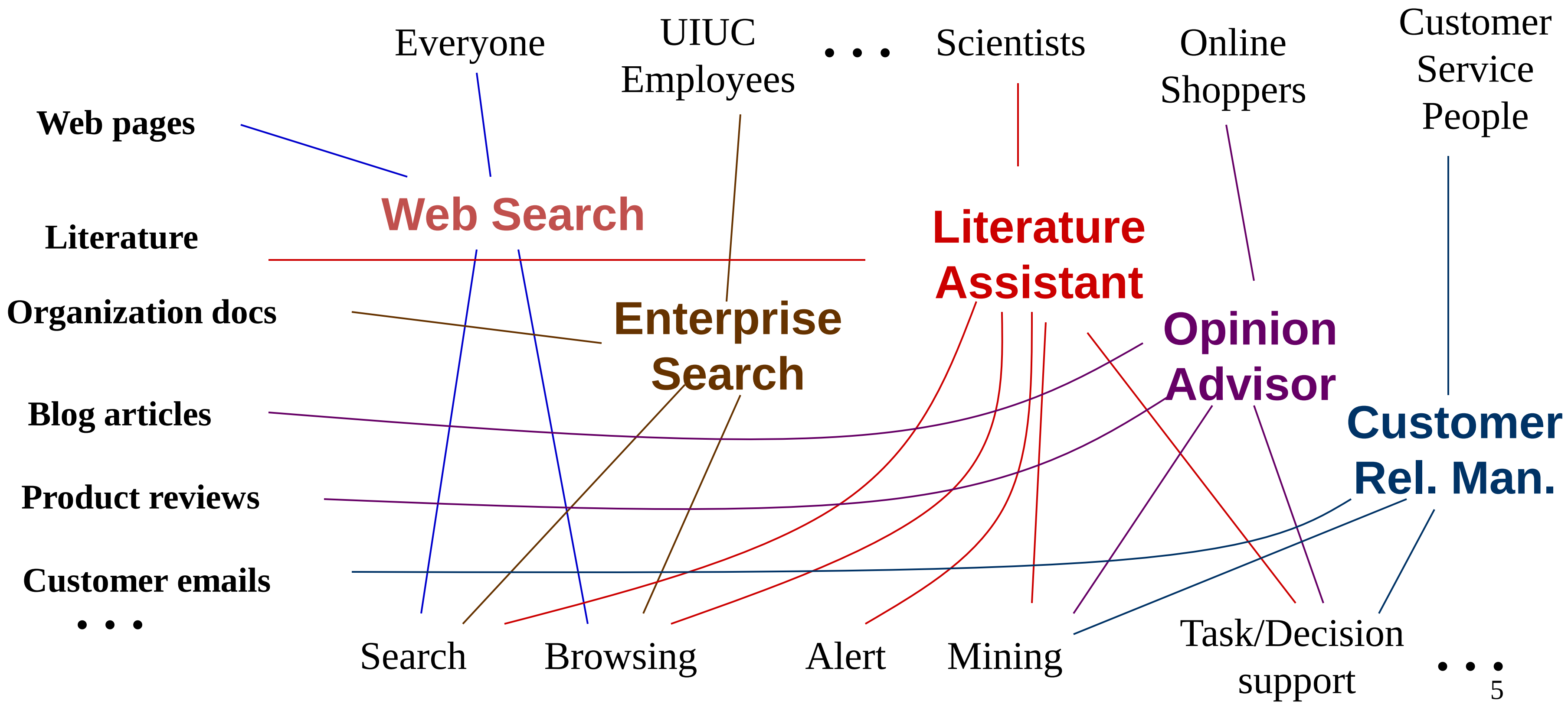
Data

Search
Browsing
Mining
Task support
…

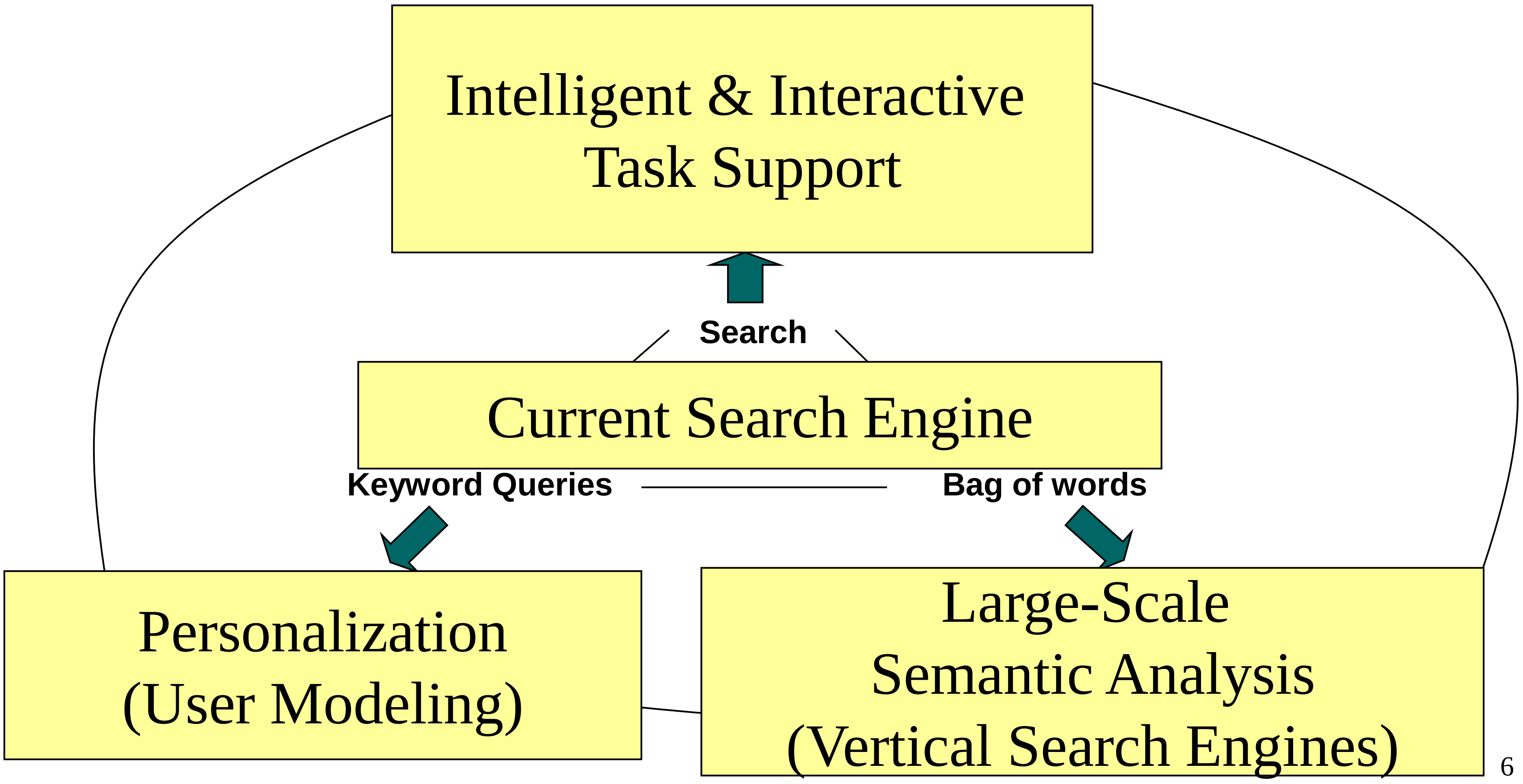
Web pages
News articles
Blog articles
Literature
Email
…

Services

Millions of Ways to Connect the DUS Triangle



Future Intelligent Information Systems

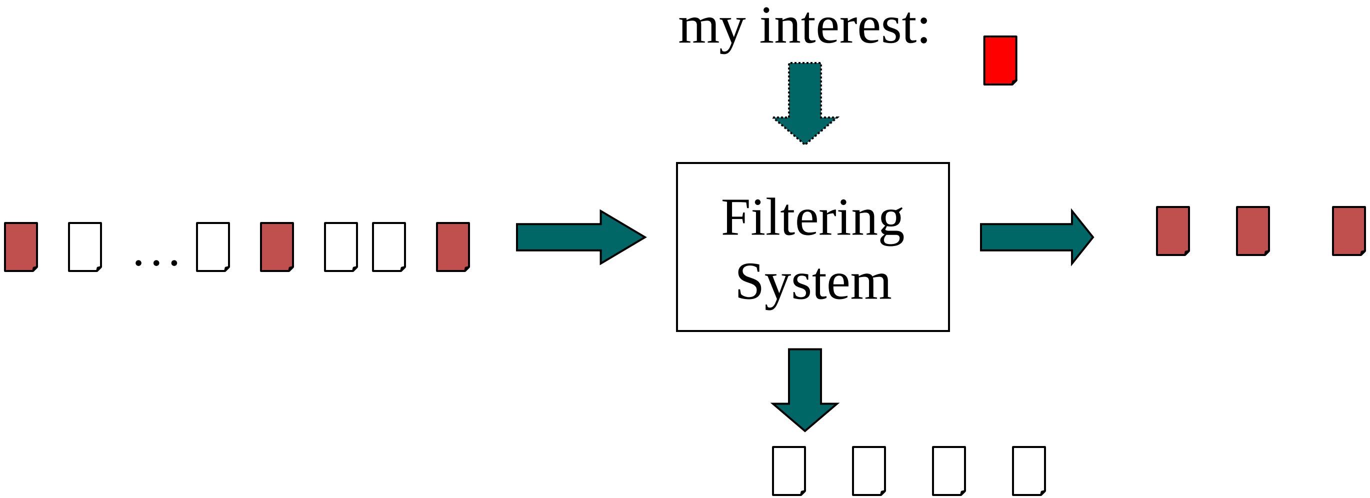


5-6. Recommender Systems: Content-Based Filtering
Two Modes of Text Access: Pull vs. Push

- **Pull** Mode (**search engines**)
  - Users take initiative
  - Ad hoc information need
- **Push** Mode (**recommender systems**)
  - Systems take initiative
  - Stable information need or system has good knowledge about a user's need

Recommender ≈ Filtering System

- Stable & long term interest, dynamic info source
- System must make a delivery decision immediately as a document “arrives”



Basic Filtering Question: Will User *U* Like Item *X*?

- Two different ways of answering it
  - Look at what items U likes, and then check if X is similar

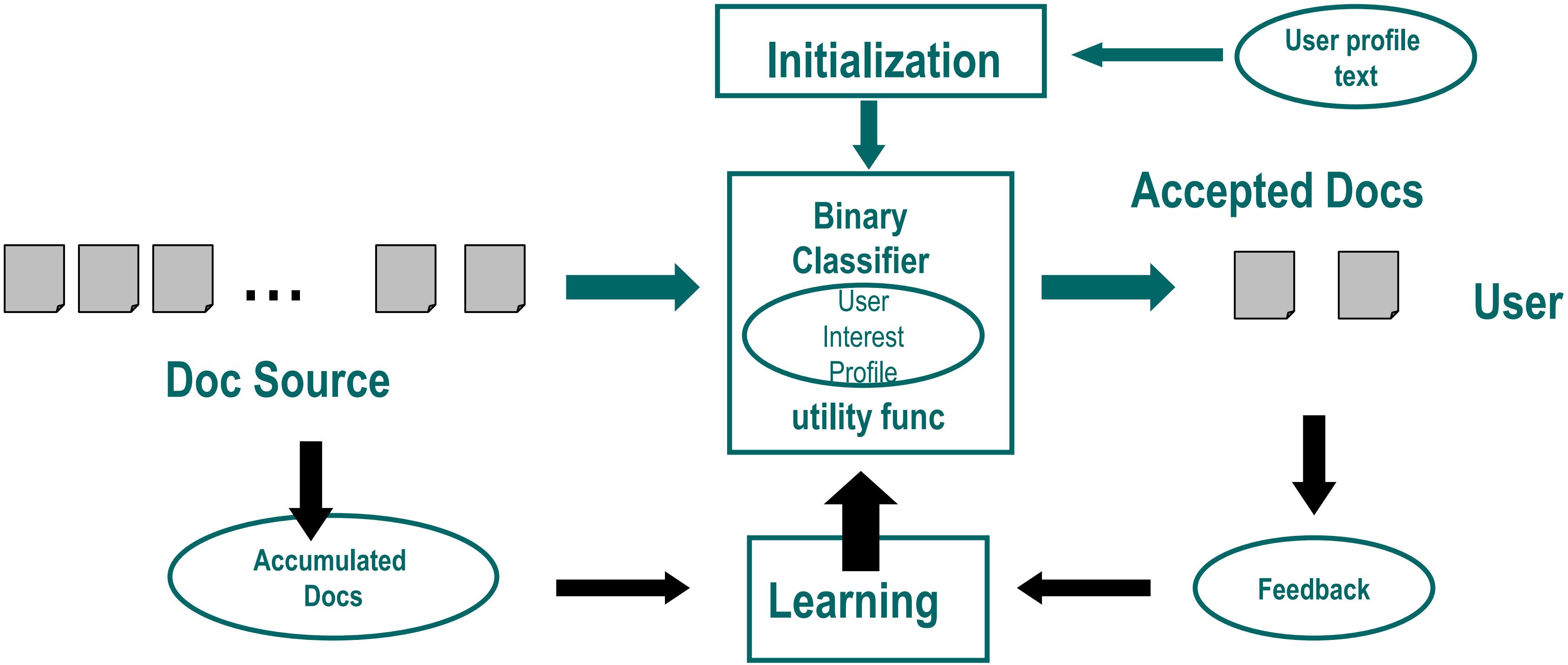
Item similarity => content-based filtering

  - Look at who likes X, and then check if U is similar

User similarity => collaborative filtering

- Can be combined

A Typical Content-Based Filtering System



Linear Utility = 3* #good - 2 *#bad

#good (#bad): number of good (bad) documents delivered to user

Are the coefficients (3, -2) reasonable? What about (10, -1) or (1, -10)?

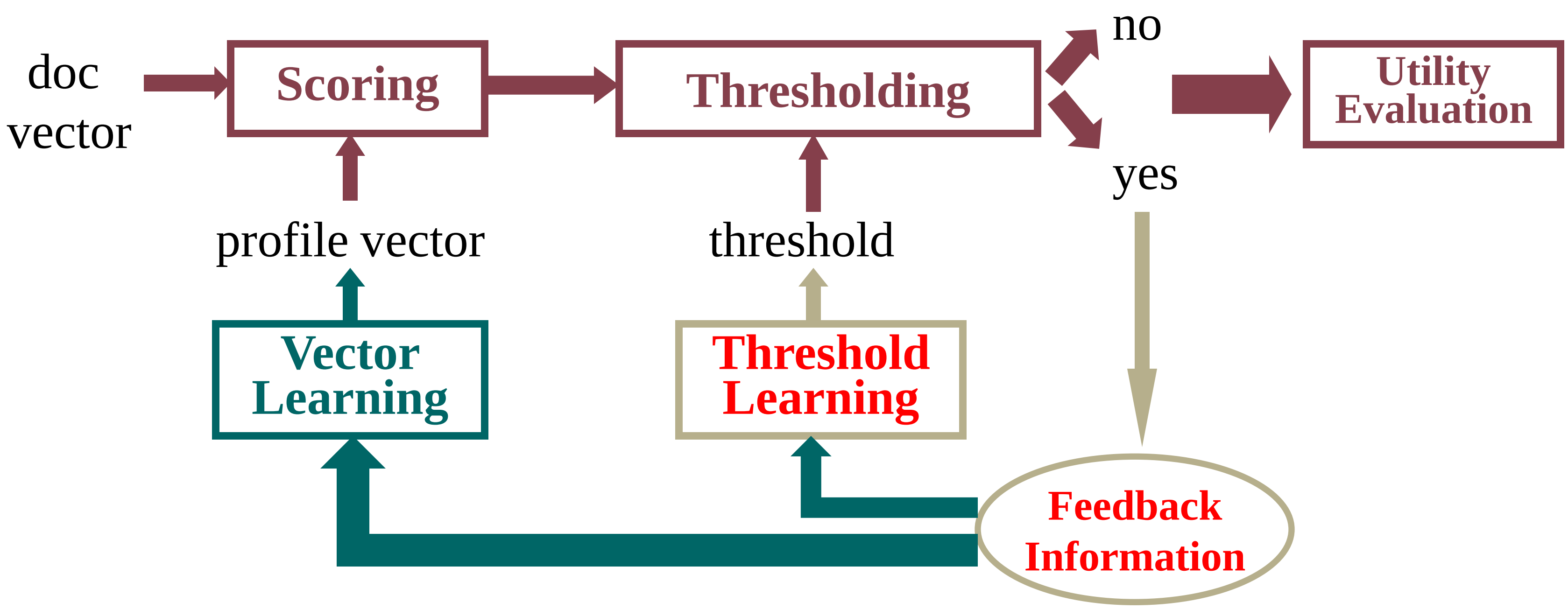
Three Basic Problems in Content-Based Filtering

- Making **filtering decision** (Binary classifier)
  - Doc text, profile text $\rightarrow$ yes/no
- **Initialization**
  - Initialize the filter based on only the profile text or very few examples
- **Learning** from
  - Limited relevance judgments (only on "yes" docs)
  - Accumulated documents
- All trying to maximize the utility

Extend a Retrieval System for Information Filtering

- “Reuse” retrieval techniques to score documents
- Use a score threshold for filtering decision
- Learn to improve scoring with traditional feedback
- New approaches to threshold setting and learning

A General Vector-Space Approach



Difficulties in Threshold Learning

36.5	Rel
33.4	NonRel
32.1	Rel
29.9	?
27.3	?
…	
...	

θ=30.0

No judgments are available for these documents

- **Censored data (judgments only available on delivered documents)**
- **Little/none labeled data**
- **Exploration vs. Exploitation**

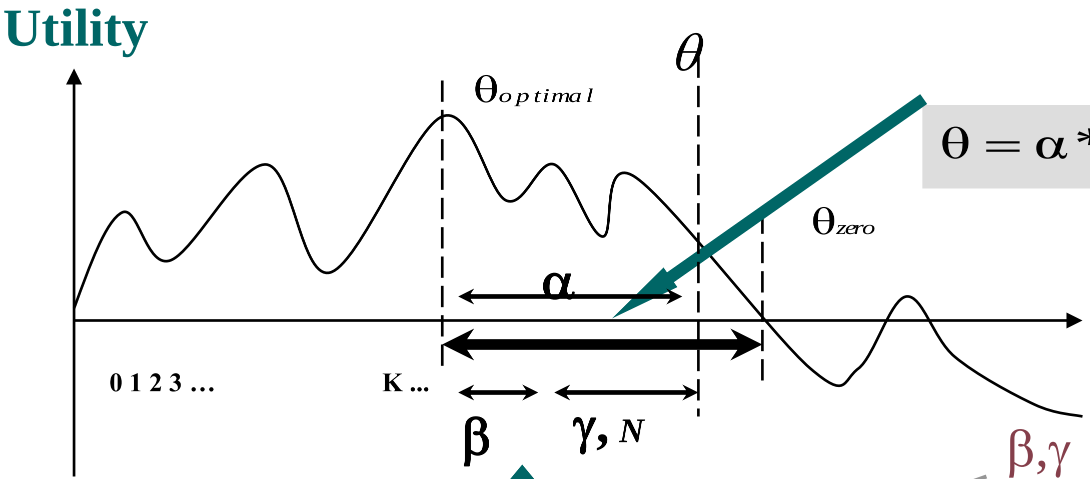
Empirical Utility Optimization

- Basic idea
  - Compute the utility on the training data for each candidate score threshold
  - Choose the threshold that gives the maximum utility on the training data set
- Difficulty: Biased training sample!
  - We can only get an upper bound for the true optimal threshold
  - Could a discarded item be possibly interesting to the user?
- Solution:
  - Heuristic adjustment (lowering) of threshold

Beta-Gamma Threshold Learning

Encourage exploration up to θ_{zero}

$$\theta = \alpha * \theta_{zero} + (1-\alpha) * \theta_{optimal}$$



Cutoff position (descending order of doc scores)

$\beta,\gamma \in [0,1]$

$$\alpha = \beta + (1-\beta) * e^{-N*\gamma}$$

$$N = \#\ training\ \ examples$$

The more examples, the less exploration (closer to $\theta_{optimal}$)

Beta-Gamma Threshold Learning (cont.)

- Pros
  - Explicitly addresses exploration-exploitation tradeoff (“Safe” exploration)
  - Arbitrary utility (with appropriate lower bound)
  - Empirically effective
- Cons
  - Purely heuristic
  - Zero utility lower bound often too conservative

Summary

- Two strategies for recommendation/filtering
  - Content-based (item similarity)
  - Collaborative filtering (user similarity)
- Content-based recommender system can be built based on a search engine system by
  - Adding threshold mechanism
  - Adding adaptive learning algorithms

7-8. Recommender Systems: Collaborative Filtering Basic Filtering
Question: Will user *U* like item *X*?

- Two different ways of answering it
  - Look at what items U likes, and then check if X is similar

    Item similarity => content-based filtering

  - Look at who likes X, and then check if U is similar

    User similarity => collaborative filtering

- Can be combined

What is Collaborative Filtering (CF)?

- Making filtering decisions for an individual user based on the judgments of other users
- Inferring individual's interest/preferences from that of other similar users
- General idea
  - Given a user u, find similar users $\{u_1, \ldots, u_m\}$
  - Predict u's preferences based on the preferences of $u_1, \ldots, u_m$
  - User similarity can be judged based on their similarity in preferences on a common set of items

CF: Assumptions

- Users with the same interest will have similar preferences
- Users with similar preferences probably share the same interest
- Examples
  - "interest is information retrieval" => "favor SIGIR papers"
  - "favor SIGIR papers" => "interest is information retrieval"
- Sufficiently large number of user preferences are available (if not, there will be a "cold start" problem)

The Collaboration Filtering Problem

Ratings

Objects: O

Users: U

	o_1	o_2	…	o_j	…	o_n
u_1	3	1.5	…. …		2	
u_2						
…	2					
u_i	1			?		
…						
u_m	3					

$X_{ij}=f(u_i,o_j)=?$

Unknown function

f: U x O→ R

The task

- **Assume known f values for some (u,o)'s**
- **Predict f values for other (u,o)'s**
- **Essentially function approximation, like other learning problems**

Memory-based Approaches

- General ideas:
  - X_{ij}: rating of object o_j by user u_i
  - n_i: average rating of all objects by user u_i
  - Normalized ratings: $V_{ij} = X_{ij} - n_i$
  - Prediction of rating of object o_j by user u_a

$$\hat{v}_{aj} = k\sum_{i=1}^{m} w(a,i)v_{ij} \qquad \hat{x}_{aj} = \hat{v}_{aj} + n_a \qquad k = 1/\sum_{i=1}^{m} w(a,i)$$

- Specific approaches differ in w(a,i) -- the distance/similarity between user u_a and u_i

User Similarity Measures

- Pearson correlation coefficient (sum over commonly rated items)

$$w_p(a,i) = \frac{\sum_j (x_{aj} - n_a)(x_{ij} - n_i)}{\sqrt{\sum_j (x_{aj} - n_a)^2 \sum_j (x_{ij} - n_i)^2}}$$

- Cosine measure

$$w_c(a,i) = \frac{\sum_{j=1}^{n} x_{aj} x_{ij}}{\sqrt{\sum_{j=1}^{n} x_{aj}{}^2 \sum_{j=1}^{n} x_{ij}{}^2}}$$

- Many other possibilities!

Improving User Similarity Measures

- Dealing with missing values: set to default ratings (e.g., average ratings)
- Inverse User Frequency (IUF): similar to IDF

Summary of Recommender Systems

- Filtering/Recommendation is "easy"
  - The user's expectation is low
  - Any recommendation is better than none
- Filtering is "hard"
  - Must make a binary decision, though ranking is also possible
  - Data sparseness (limited feedback information)
  - "Cold start" (little information about users at the beginning)
- Content-based vs. Collaborative filtering vs. Hybrid
- Recommendation can be combined with search ➔ Push + Pull
- Many advanced algorithms have been proposed to use more context information and advanced machine learning

Additional Readings

- Francesco Ricci, Lior Rokach, Bracha Shapira, Paul B. Kantor: Recommender Systems Handbook. Springer 2011.

http://www.cs.bme.hu/nagyadat/Recommender_systems_handbook.pdf