- How Can we combine Many Features? (Learning to Rank)
  - 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 lambda is a set of parameters
    - Learn lambda by fitting function s with training data, i.e., 3-tuples like
- **Learning to Rank Part 2**
- Regression-Based Approaches
  - Logistic Regression: Xi(Q,D) is feature; B’s are parameters

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

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

  - Estimate B’s by maximizing the likelihood of training data

	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

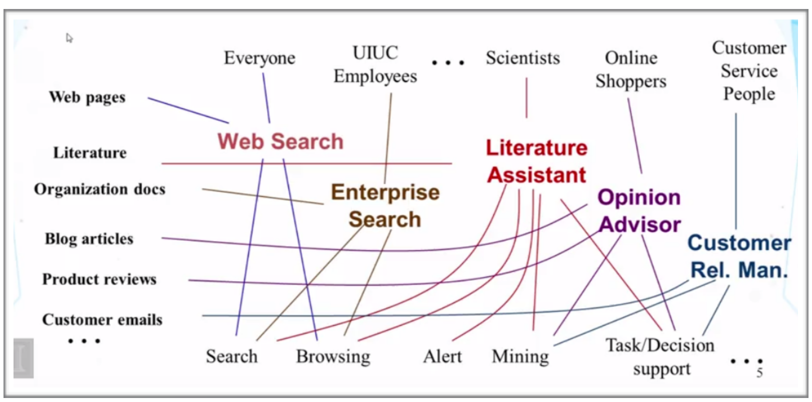
$$p(\{(Q,D_1,1),(Q,D_2,0)\}) = \frac{1}{1+\exp(-\beta_0 - 0.7\beta_1 - 0.11\beta_2 - 0.65\beta_3)} * (1 - \frac{1}{1+\exp(-\beta_0 - 0.3\beta_1 - 0.05\beta_2 - 0.4\beta_3)})$$

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

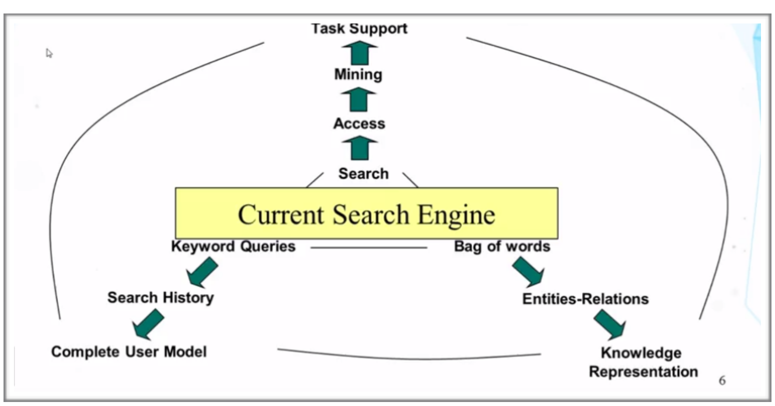
Once β’s are known, we can take Xi(Q,D) computed based on a new query and a new document to generate a score for D w.r.t. Q. 4

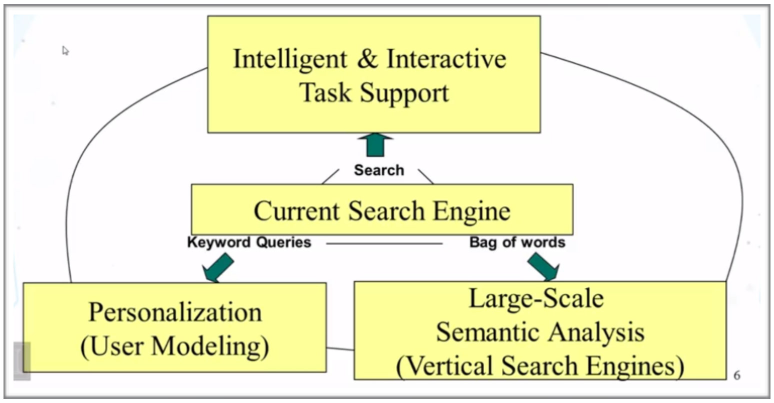
- **Learning to Rank Part 3**
- Mode Advanced Learning Algorithms
  - Attempt to directly optimize a retrieval measure (e.g. MAP, nDCG)
    - More difficult as an optimization problem
    - Many solutions were proposed
  - Can be applied to many other ranking problems beyond search
    - Recommender systems
    - Computational advertising
    - Summarization
    - …

- Summary
  - Machine learning has been applied to text retrieval since many decades ago
  - 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
- **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 understand 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
  - Data, Users, Services
- Millions of Ways to Connect the DUS Triangle

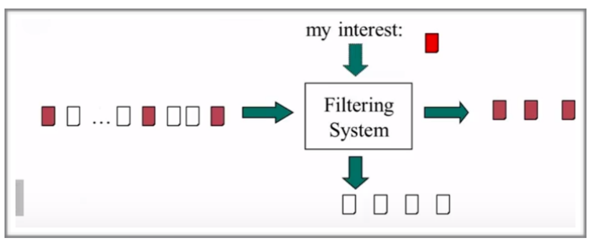


- Future Intelligent Information Systems

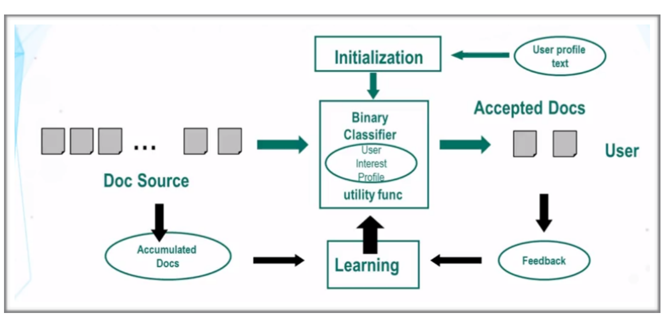




- **Recommender Systems Content-Based Filtering - Part 1**
- Two Modes of Text Access: Pull vs. Push
  - Pull Mode (search engines)
    - Users take initiative
    - Ad hoc information need
  - Push Mode (recommender systems)
    - Systems take initiative
    - Stable information need or system has good knowledge about a user’s need
- Recommender equivalent to Filtering System
  - Stable and long term interest, dynamic info source
  - System must make a delivery decision immediately as document “arrives”

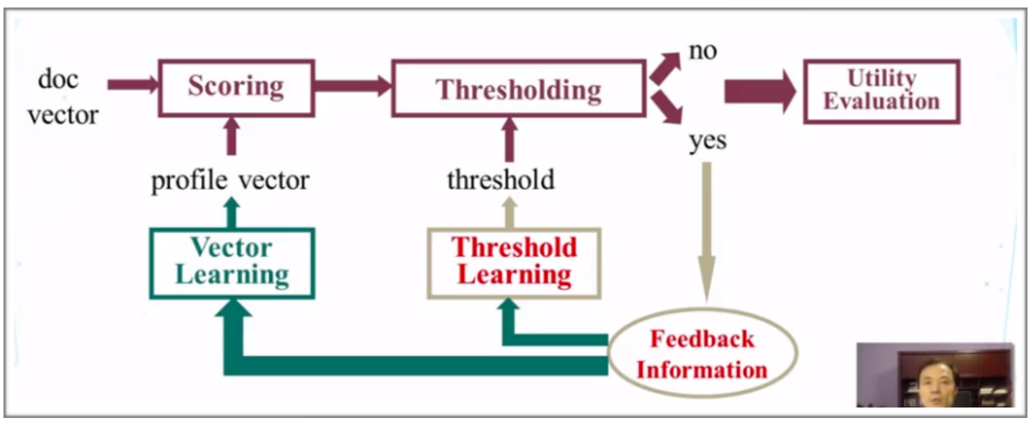


- Basic Filtering Question: Will User U Like Item X?
  - Two different ways of answering it
    - Look at what items U likes, and the check if X is similar
      - Item similarity => content-based filtering
    - Look at who likes X, and then check if U is similar
      - User similarity => collaborative filtering
  - Can be combined
- A Typical Content-Based Filtering System



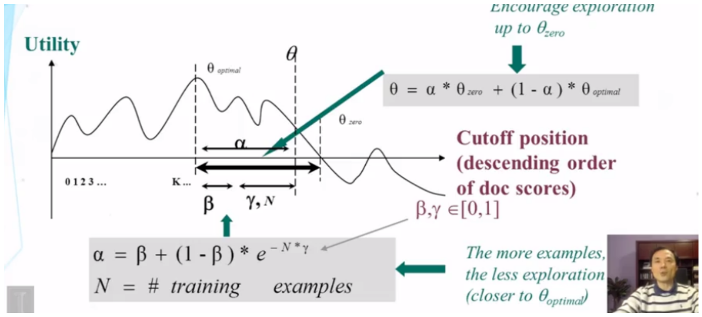
Linear Utility = 3* #good - 2 *#bad
#good (#bad): number of good (bad) documents delivered to user
Are the coefficients (3, -2) reasonable? What about (10, -1) or (1, -10)?

- Three Basic Problems in Content-Based Filtering
  - Making filtering decision (Binary classifier)
    - Doc text, profile text -> 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



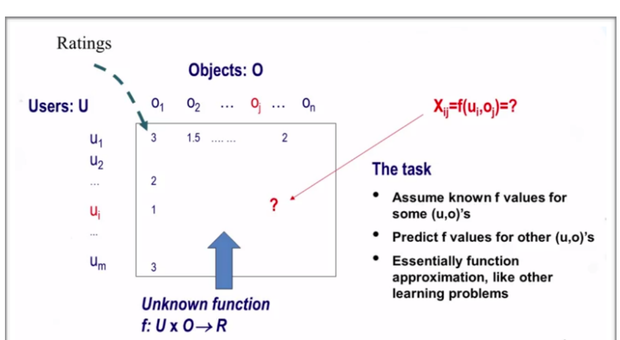
- **Recommender Systems: Content-Based Filtering - Part 2**
- Difficulties in Threshold Learning
  - Censored data (judgments only available on delivered documents)
  - Little/none labeled data
  - Exploration vs. Exploitation
- Empirical Utility Optimization
  - Basic idea
    - Compute the utility 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 et 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



- 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
- **Recommender Systems Collaborative filtering Part - 1**
- 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 {u1,…,um}
    - Predict u’s preferences based on the preferences of u1,…,um
    - 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



- **Recommender Systems: Collaborative Filtering - Part 2**
- Memory based Approaches
  - General ideas:
    - Xij: rating of object oj by user ui
    - ni: average rating of all objects by user ui
    - Normalized ratings: Vij = Xij - ni
    - Prediction of rating of object oj by user ua

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

      - weights control the inference on the prediction
      - formula
        - prediction of object by user is the sum of all the possible user normalized ratings multiplied by the weights of similarity
          - similarity rating - controls the inference of the user on the prediction
          - the weight is related to the similarity of user ua and ui
          - the more similar they are the more contribution the user will make to the prediction of ua
        - k is a normalizer = 1 over the sum of all the weights
  - Specific approaches differ in w(a,i) —- the distance/similarity between usera and ui
- User Similarity Measures
  - Pearson correlation coefficient (sum over commonly rated items)

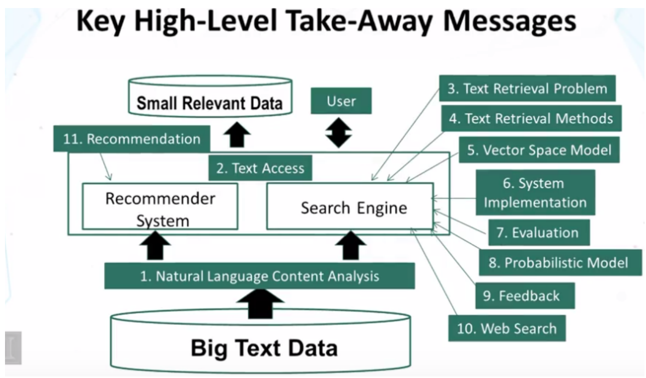
$$w_p(a,i) = \frac{\sum_j (x_{aj} - n_a)(x_{ij} - n_i)}{\sqrt{\sum_j (x_{aj} - n_a)^2 \sum_j (x_{ij} - n_i)^2}}$$

  - Cosine measure
    - treat the rating vectors in vector space and compute the cosine of the angle of the two measures

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

  - Many other possibilities!
  - User similarity is based on the preference of their items
- Improving User Similarity Measures
  - Dealing with missing values: set to default ratings (e.g. average ratings)
  - Inverse User Frequency (IUF): similar to IDF
    - emphasizes more on similarity on items that are not viewed by many users

- **Recommender Systems: Collaborative Filtering - Part 3**
- 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
- **Course Summary**



  - Natural Language Content Analysis
    - NLP is foundation for TR, but current NLP isn't robust enough: BOW is sufficient for most search tasks
  - Text Access
    - Push vs Pull; Querying vs Browsing
  - Text Retrieval Problem
    - TR -> Ranking Problem
  - Text Retrieval Methods / Vector Space Model
    - VSM, LM, TF-IDF, Length Norm
  - System Implementation
    - Inverted Index + Fast Search
  - Evaluation
    - Cranfield Eval, Method, MAP, nDCG, Prac.Recall
  - Probabilistic Model / Feedback
    - Rocchio, Mixture Model
  - Web Search
    - MapReduce for parallel indexing PageRank, HITS, Learning to Rank, Future of Web search
  - Recommendation:
    - Content-based + collaborative filtering
-