Content-Based Filtering and Hybrid Systems

Francesco Ricci

Content

- Typologies of recommender systems
- Content-based recommenders
- Naive Bayes classifiers and content-based filtering
- Content representation (bag of words, tf-idf)
- Demographic-based recommendations
- Clustering Methods
- Utility-based Methods
- Hybrid Systems
  - Weighted
  - Collaboration via content

Other Recommendation Techniques

- The distinction is not related to the **user interface** – even if this matters a lot - or the properties of the user's interaction but rather the **source of data** used for the recommendation
- **Background data:** the information of the system before the recommendation process starts
- **Input data:** the information that the user must communicate to the system to get a recommendation
- **The algorithm:** that combines background and input data to build a recommendation.

"Core" Recommendation Techniques

Technique	Background	Input	Process
Collaborative	Ratings from **U** of items in **I**.	Ratings from **u** of items in **I**.	Identify users in **U** similar to **u**, and extrapolate from their ratings of **i**.
Content-based	Features of items in **I**	**u**'s ratings of items in **I**	Generate a classifier that fits **u**'s rating behavior and use it on **i**.
Demographic	Demographic information about **U** and their ratings of items in **I**.	Demographic information about **u**.	Identify users that are demographically similar to **u**, and extrapolate from their ratings of **i**.
Utility-based	Features of items in **I**.	A utility function over items in **I** that describes **u**'s preferences.	Apply the function to the items and determine **i**'s rank.
Knowledge-based	Features of items in **I**. Knowledge of how these items meet a user's needs.	A description of **u**'s needs or interests.	Infer a match between **i** and **u**'s need.

[Burke, 2007]

Content-Based Recommendation

- In ***content-based*** recommendations the system tries to recommend items “similar” to those a given user has liked in the past (*general idea*)
  - **It builds a predictive model of the user preferences**
- In contrast with ***collaborative*** recommendation where the system identifies users whose tastes are similar to those of the given user and recommends items *they* have liked
- A **pure** content-based recommender system makes recommendations for a user based solely on the profile built up by analyzing the content of items **which that user has rated in the past.**

Simple Example

- I saw yesterday “Harry Potter and and the Sorcerer's Stone”

- The recommender system suggests:
  - Harry Potter and the Chamber of Secrets
  - Polar Express

Content-Based Recommender

- Has its root in **Information Retrieval** (IR)
- It is mainly used for recommending **text-based products** (web pages, usenet news messages) – products for which you can find a textual description
- The items to recommend are "described" by their associated **features** (e.g. keywords)
- The **User Model** can be structured in a "similar" way as the content: for instance the features/keywords more likely to occur in the preferred documents
  - Then, for instance, text documents can be recommended based on a comparison between their content (words appearing in the text) and a user model (a set of preferred words)
- The user model can also be a **classifier** based on whatever technique (e.g., Neural Networks, Naive Bayes, C4.5).

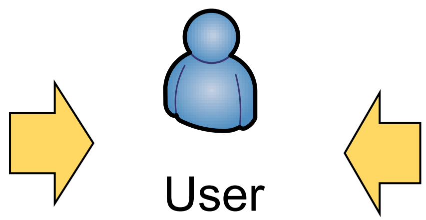
Long-term and Ephemeral Preferences

- The user model **typically** describes **long-term preferences** – since it is build by mining (all) previous user-system interactions (ratings or queries)
  - This is common to collaborative filtering – they have difficulties in modeling the "context" of the decision process
- But one can build a content-based recommender system, more similar to an IR system, **acquiring on-line the user model** (query)
- Or **stable preferences and short-term ones can be combined:**
  - E.g. a selection of products satisfying some short-term preferences can be sorted according to more stable preferences.

Example: Book recommendation

Ephemeral

- I’m taking two weeks off
- Novel
- I’m interested in a Polish writer
- Should be a travel book
- I’d like to reflect on the meaning of life



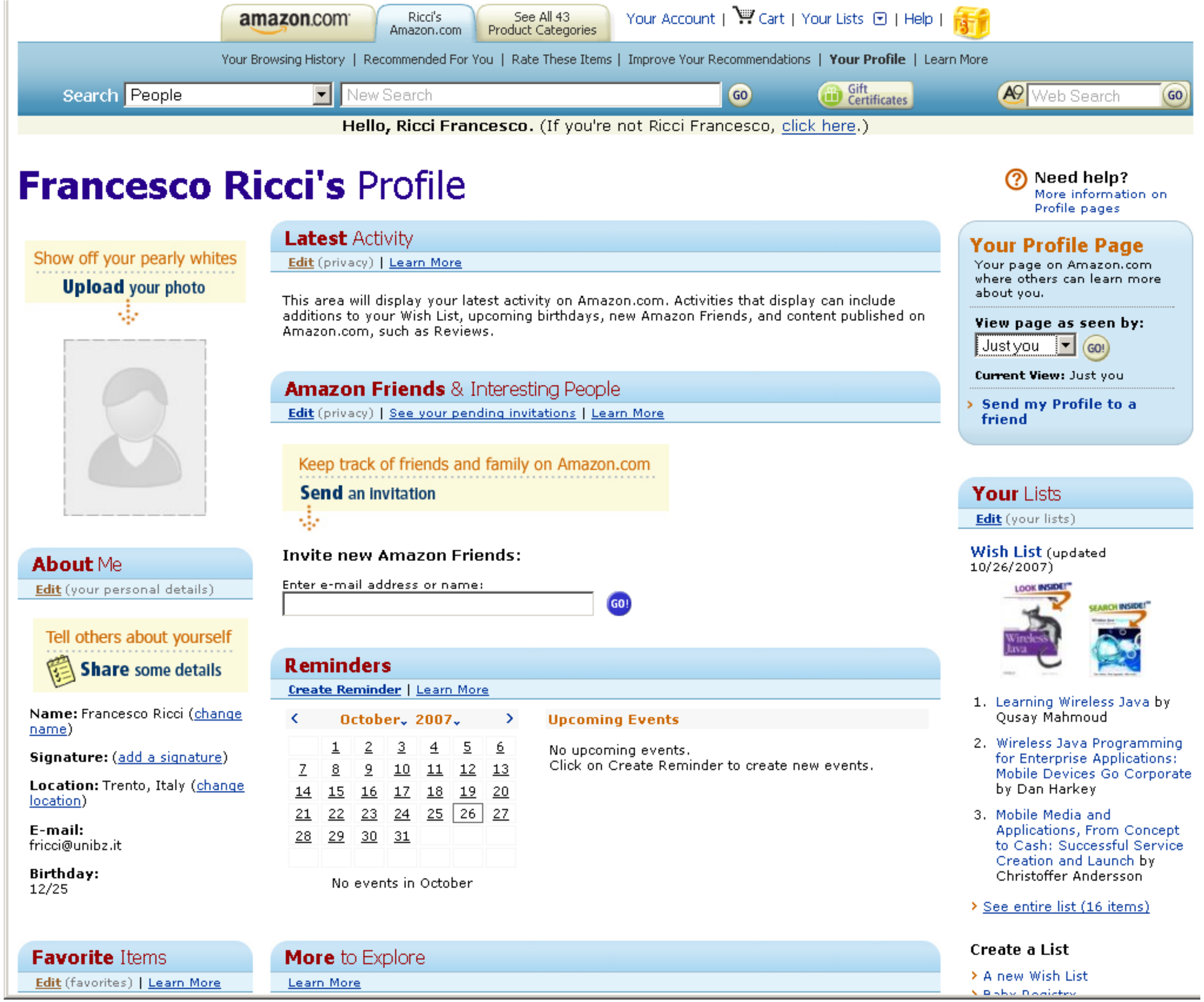
Long Term

- Dostoievsky
- Stendhal
- Checov
- Musil
- Pessoa
- Sedaris
- Auster
- Mann

Recommendation

Joseph Conrad, Hearth of darkness

Amazon User Profile



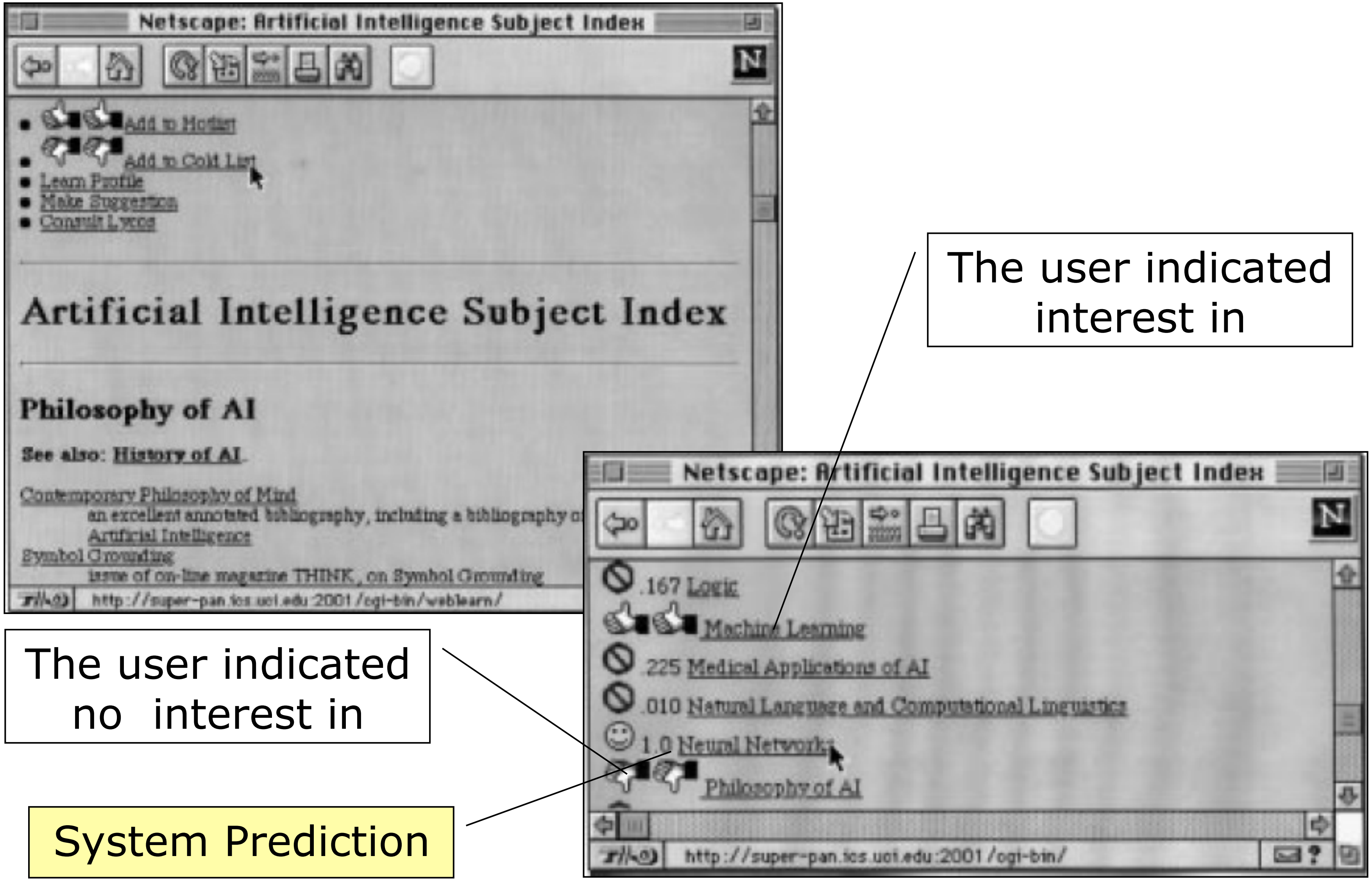
Syskill & Webert [Pazzani &Billsus, 1997]

- Assisting a person to **find information that satisfies long-term, recurring goals** (e.g. digital photography)
- Feedbacks on the “interestingness” of a set of **previously visited sites** is used to **learn a profile**
- The **profile** is used to predict interestingness of **unseen sites**.

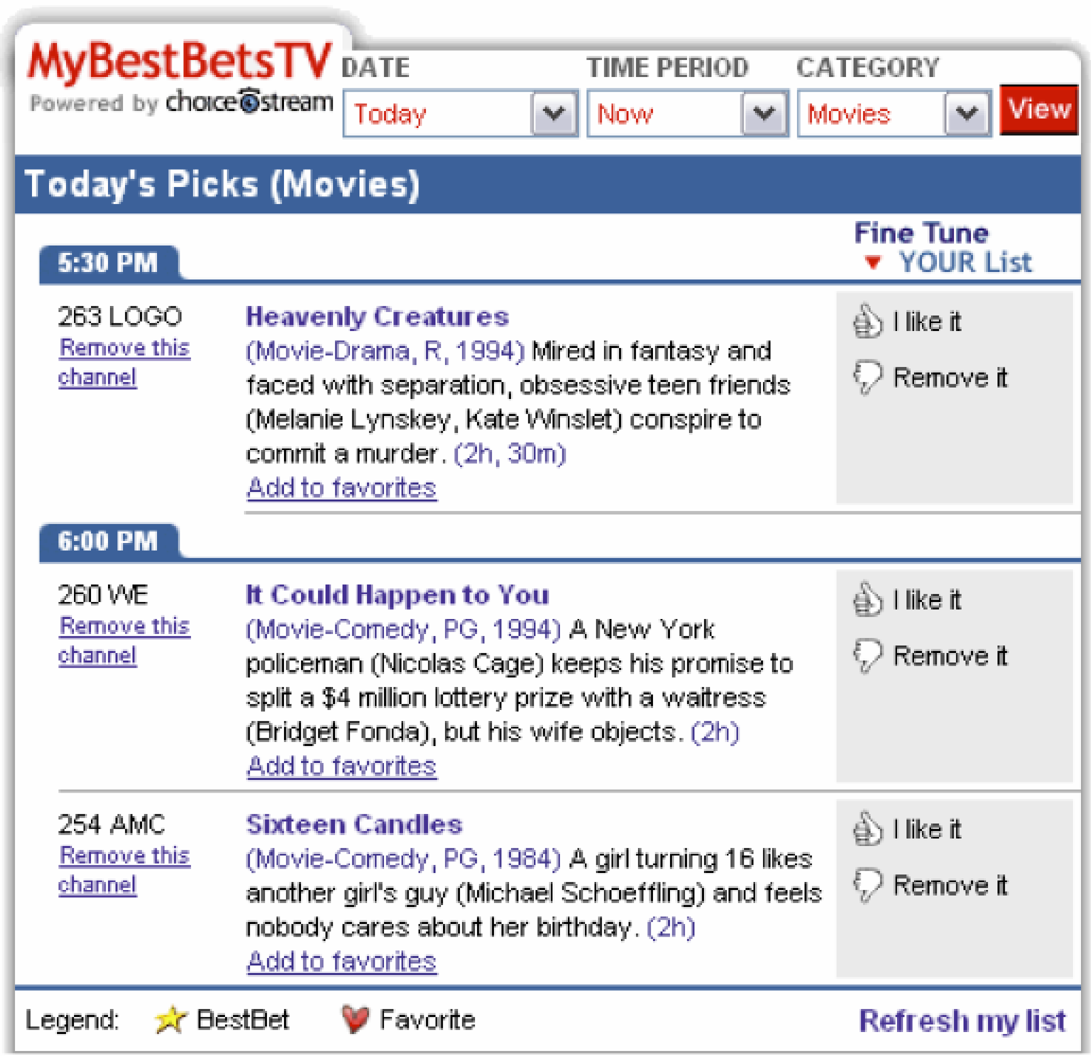
Supported Interaction

- The user identifies a topic (e.g. Biomedical) and a page with many links to other pages on the selected topic (index page)
  - Kleinberg would call this page a "Hub"
- The user can then explore the Web with a browser that in addition to showing a page:
  - Offers a tool for collecting user ratings on displayed pages
  - Suggests which links on the current page are (estimated) interesting
- It is supporting the "recommendation in context" user's task (but not using the context!).

Syskill & Webert User Interface



Explicit feedback example



Content Model: Syskill & Webert

- A document (HTML page) is described as a set of Boolean features (a word is present or not)
- A feature is considered important for the prediction task **if the Information Gain is high**
- **Information Gain:** G(S,W) = E(S) –[P((W is present)*E $(S_{W\text{ is present}})$ + P(W is absent)*E$(S_{W\text{ is absent}})$]

$$E(S) = \sum_{c \in \{hot, cold\}} -p(S_c)\log_2(p(S_c))$$

- E(S) is the Entropy of a labeled collection (how randomly the two labels are distributed)
- W is a word – a Boolean feature (present/not-present)
- S is a set of documents, S_{hot} is the subset of interesting documents
- They have used the 128 most informative words (highest information gain).

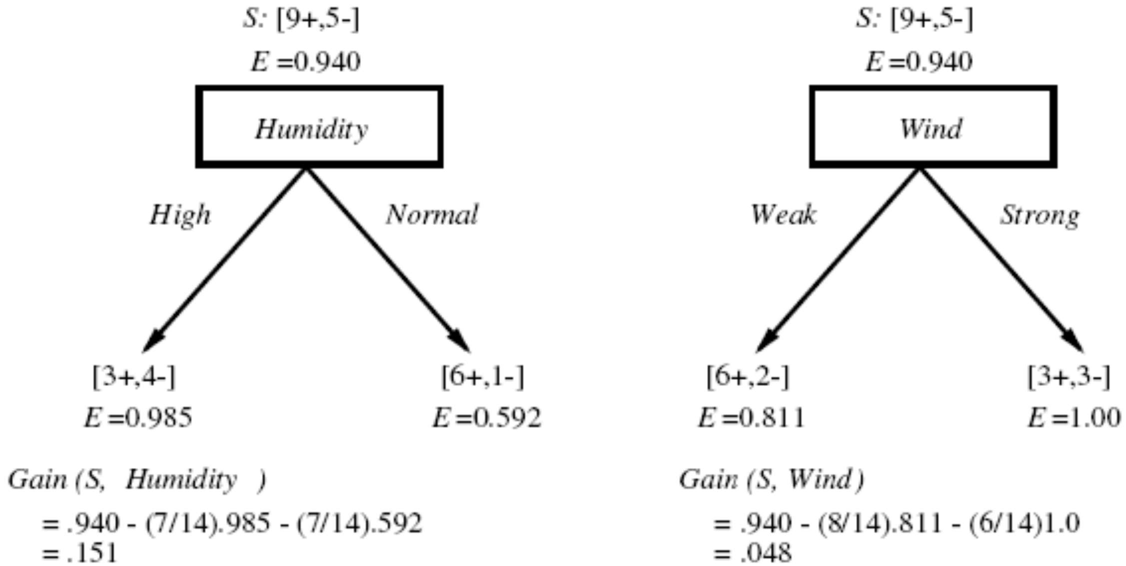
Example

5 yes and 9 no
E(S) = -(9/14)log(9/14) – (5/14)log(5/14)= 0.9429 …

outlook	temperature	humidity	windy	Play/CLASS
sunny	85	HIGH	WEAK	no
sunny	80	HIGH	STRONG	no
overcast	83	HIGH	WEAK	yes
rainy	70	HIGH	WEAK	yes
rainy	68	NORMAL	WEAK	yes
rainy	65	NORMAL	STRONG	no
overcast	64	NORMAL	STRONG	yes
sunny	72	HIGH	WEAK	no
sunny	69	NORMAL	WEAK	yes
rainy	75	NORMAL	WEAK	yes
sunny	75	NORMAL	STRONG	yes
overcast	72	HIGH	STRONG	yes
overcast	81	NORMAL	WEAK	yes
rainy	71	HIGH	STRONG	no

Would the entropy be larger with 7 yes and 7 no?

Entropy and Information Gain example



Smaller Entropy
Higher Information Gain

- 9 positive and 5 negative examples → E(S)=0.940
- Using the “humidity” attribute – the entropy of the split produced is:
  - P((Humidity is high)E($S_{\text{hum. is high}}$) + P(Humidity is normal) E($S_{\text{hum. is normal}}$)=(7/14)*0.985 + (7/14)*0.592 = 0.789
- Using the “wind” attribute – the entropy of the split produced is:
  - P((wind is weak)E($S_{\text{wind. is weak}}$) + P(wind is strong)E($S_{\text{wind is strong}}$)=(8/14)*0.811 + (6/14)*1.0 = 0.892

Learning

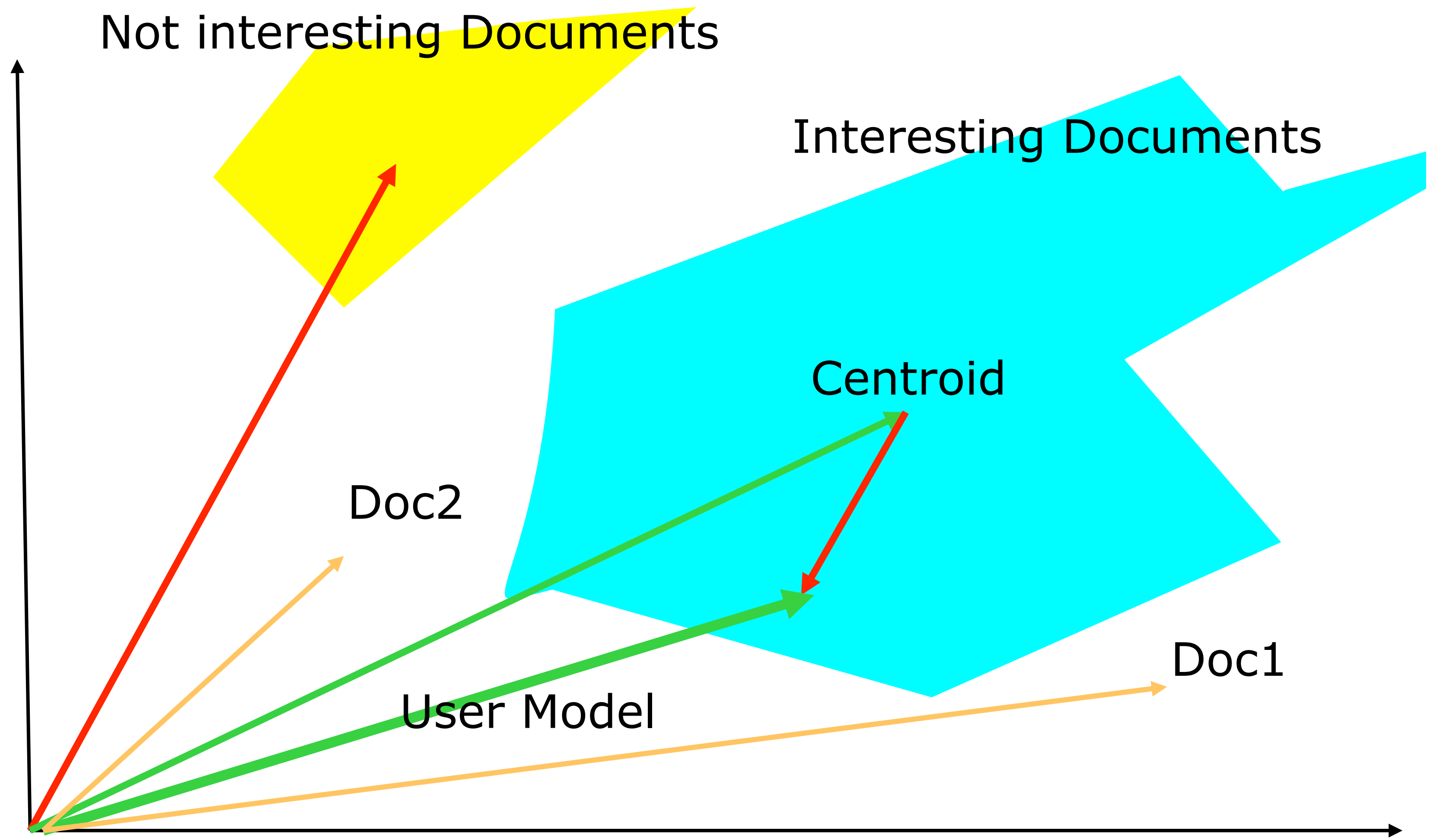
Multinomial or
Multivariate?

- They used a Naïve Bayesian classifier (**one for each user**), where the probability that a document $w_1=v_1$, …, $w_n=v_n$ (e.g. car=1, story=0, …, price=1) belongs to a class (cold or hot) is

$$P(C = hot \mid w_1 = v_1, \ldots, w_n = v_n) \cong P(C = hot) \prod_j P(w_j = v_j \mid C = hot)$$

- Both $P(w_j = v_j|C=hot)$ (i.e., the probability that in the set of the documents liked by a user the word w_j is present or not) and P(C=hot) is estimated from the training data (Bernoulli model)
- After training on 30/40 examples it can predict hot/cold with an accuracy between 70% and 80%

Content-Based Recommender with Centroid



Doc1 is estimated more interesting than Doc2

Problems of Content-Based Recommenders

- A very shallow analysis of certain kinds of content can be supplied
- **Some kind of items are hardly amenable to any feature extraction methods** with current technologies (e.g. movies, music)
- Even for texts (as web pages) the IR techniques cannot consider multimedia information, aesthetic qualities, download time
  - Any ideas about how to use them?
  - Hence if you rate positively a page it could be not related to the presence of certain keywords!

Problems of Content-Based Recommenders (2)

- **Over-specialization:** the system can only recommend items scoring high against a user's profile – the user is recommended with items similar to those already rated
- **Requires user feed-backs:** the pure content-based approach (similarly to CF) **requires user feedback** on items in order to provide meaningful recommendations
- **It tends to recommend expected items** – this tends to **increase trust** but could make the recommendation not much useful (no serendipity)
- Works better in those situations where the **"products" are generated dynamically** (news, email, events, etc.) and there is the need to check if these items are relevant or not.

Serendipity

- **Serendipity**: to make discoveries, by accident and sagacity, of things not in quest of.
- Examples:
  - **Velcro by Georges de Mestral.** The idea came to him after walking through a field and observing the hooks of burdock attached to his pants
  - **Post-it Notes by Spencer Silver and Arthur Fry.** They tried to develop a new glue at 3M, but it would not dry. So they devised a new use for it.
  - **Electromagnetism, by Hans Christian Oersted.** While he was setting up his materials for a lecture, he noticed a compass needle deflected from magnetic north when the electric current from the battery he was using was switched on and off.

"Core" Recommendation Techniques

Technique	Background	Input	Process
Collaborative	Ratings from **U** of items in **I**.	Ratings from **u** of items in **I**.	Identify users in **U** similar to **u**, and extrapolate from their ratings of **i**.
Content-based	Features of items in **I**	**u**'s ratings of items in **I**	Generate a classifier that fits **u**'s rating behavior and use it on **i**.
Demographic	Demographic information about **U** and their ratings of items in **I**.	Demographic information about **u**.	Identify users that are demographically similar to **u**, and extrapolate from their ratings of **i**.
Utility-based	Features of items in **I**.	A utility function over items in **I** that describes **u**'s preferences.	Apply the function to the items and determine **i**'s rank.
Knowledge-based	Features of items in **I**. Knowledge of how these items meet a user's needs.	A description of **u**'s needs or interests.	Infer a match between **i** and **u**'s need.

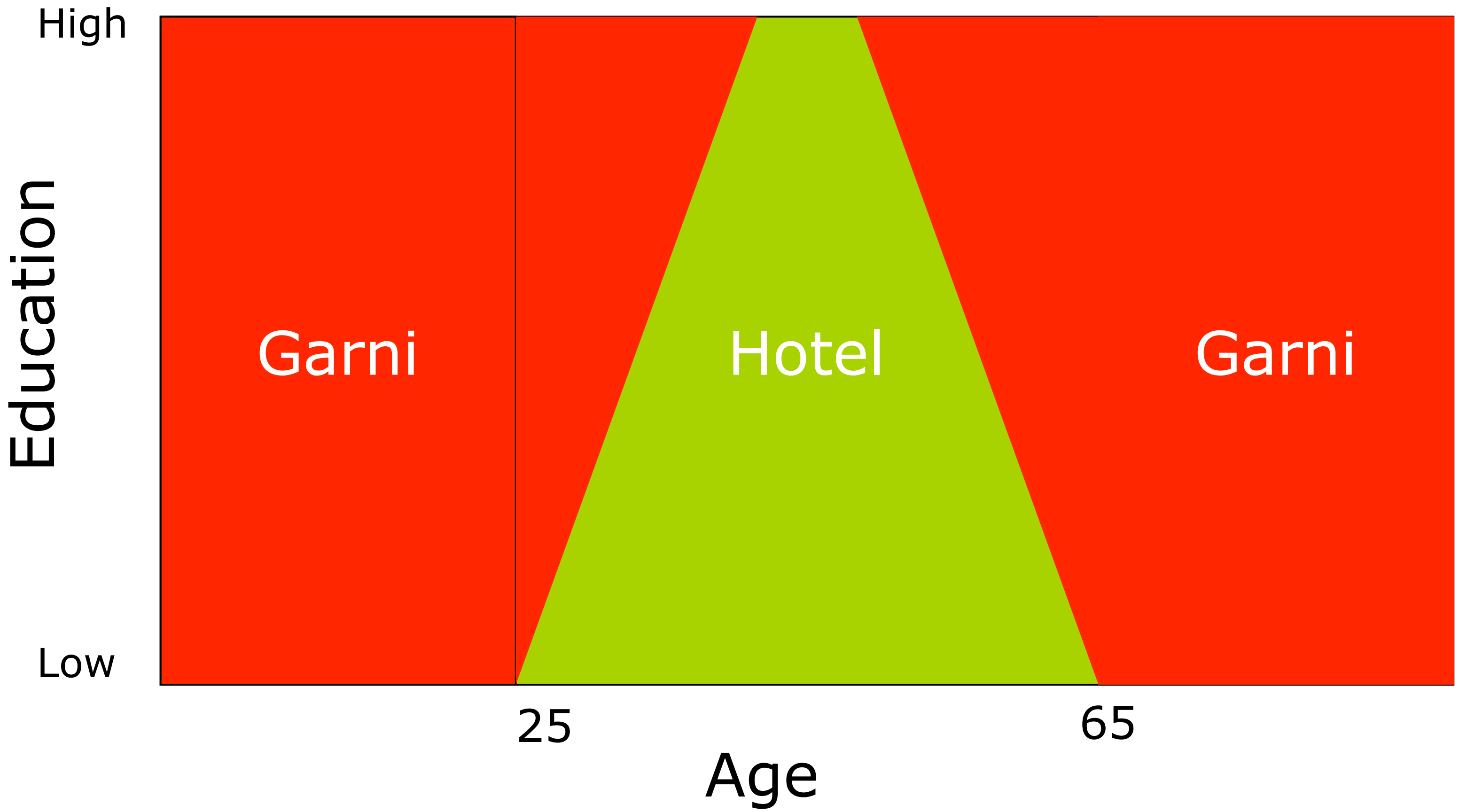
Demographic Methods

- Aim to **categorize** the user based on **personal attributes** and make recommendation based on demographic classes
- Demographic groups can come from marketing research – hence experts decide how to model the users
- Demographic techniques form people-to-people correlations using their demographic descriptions
- Tend to be similar to *collaboration via content* (we shall discuss it later) but demographic techniques do not use explicit ratings.

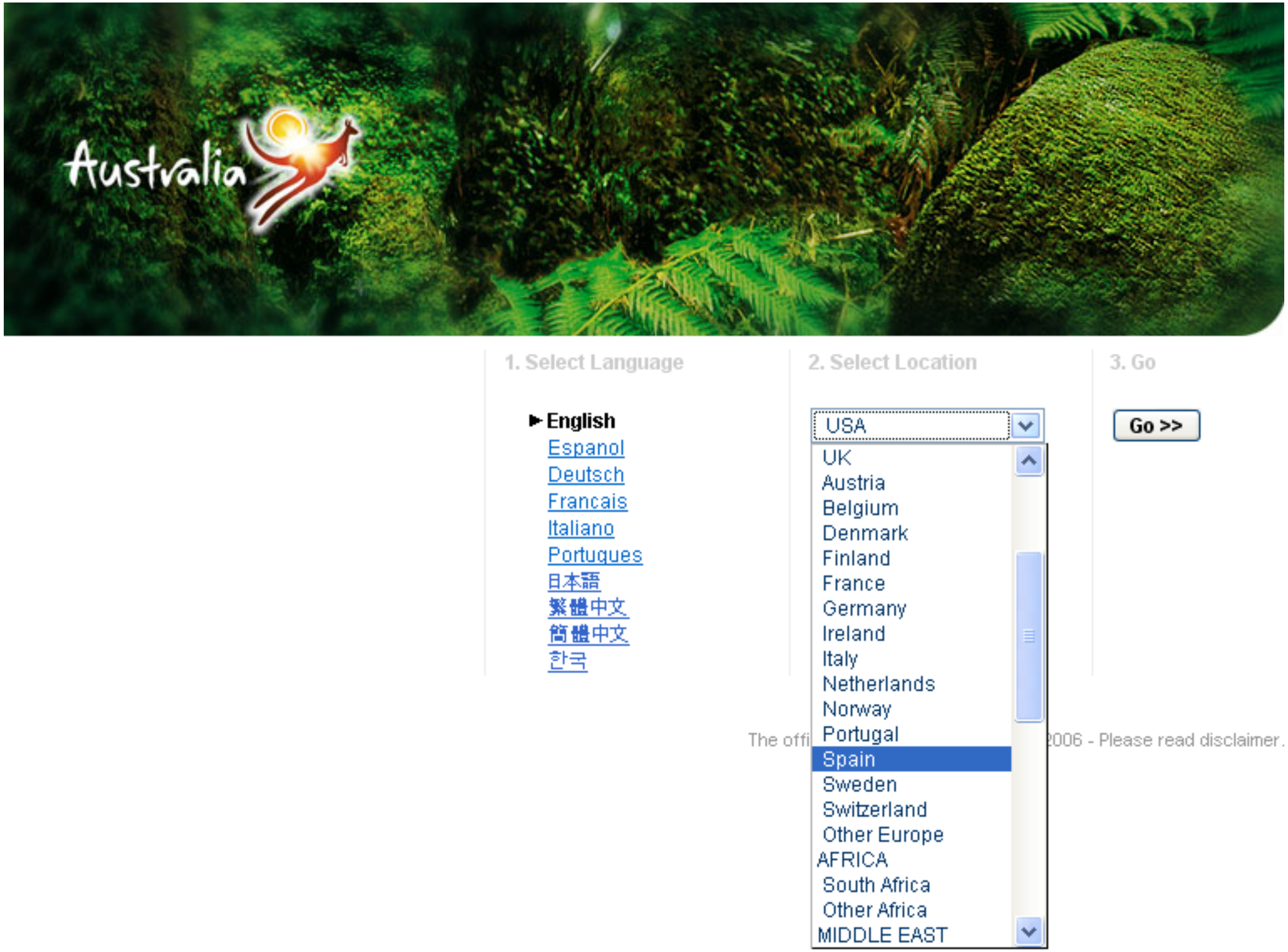
Simple Demographic Method

- The marketer knows how to separate the demographic classes and exploits this knowledge to define the **personalization rules**
- This is the method used by many commercial (expensive) personalization engines (e.g. ATG) [Fink & Kobsa, 2000]
- It is very efficient but:
  - Do not tracks the changes in the population (user products)
  - Rely on the rules inserted by an “expert”
  - Suffers of all the classical problems of Expert Systems (e.g. brittle).

Example



Demographic-based personalization



Demographic-based personalization

Demographic Methods (more sophisticated)

Product (e.g., restaurant)

	gender	age	area code	education	employed	Dolce
Karen	F	15	714	HS	F	+
Lynn	F	17	714	HS	F	–
Chris	M	35	714	C	T	+
Mike	F	40	714	C	T	–
Jill	F	10	714	E	F	?

- Demographic features in general are asked to user
- But can also be induced by classifying a user using other user descriptions (e.g. the home page) – you need some users (training) whose class is known (e.g. male/female)
- Prediction can use whatever learning mechanism we like (nearest neighbor, naive Bayes classifier, etc.)
- **A classifier for each product!**

[Pazzani, 1999]

Clustering Methods

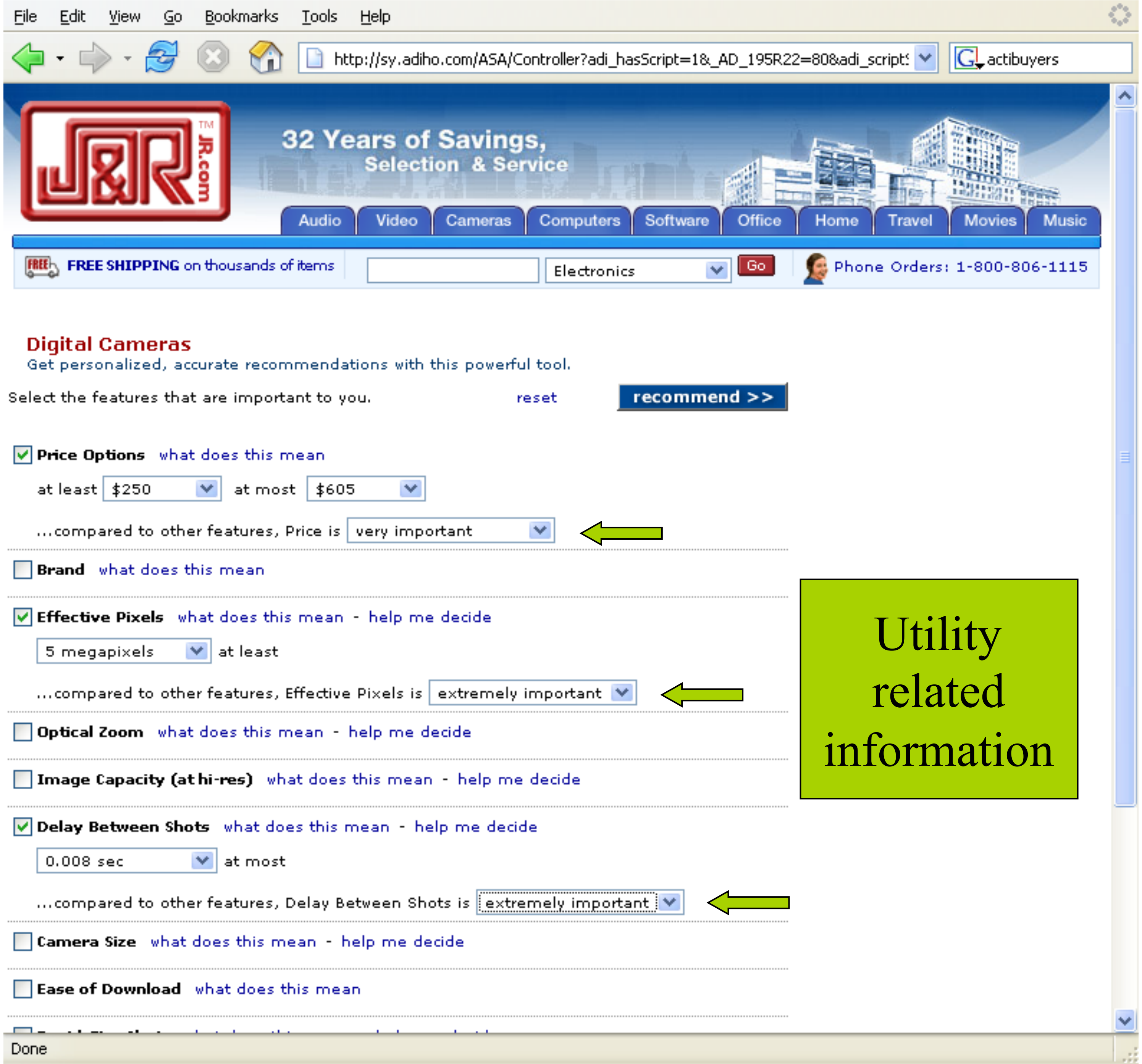
- Use a **clustering** method to **divide** the customers' base into **segments**
  - Unsupervised method
  - It needs a **similarity** measure between customers
  - Typically it exploits a greedy algorithm
- It assigns each user to a cluster – the one that contains the most similar users
- Use purchases or ratings of customers in the segment to generate recommendations
- Many different user models can be considered for the similarity computation.

"Core" Recommendation Techniques

Technique	Background	Input	Process
Collaborative	Ratings from **U** of items in **I**.	Ratings from **u** of items in **I**.	Identify users in **U** similar to **u**, and extrapolate from their ratings of **i**.
Content-based	Features of items in **I**	**u**'s ratings of items in **I**	Generate a classifier that fits **u**'s rating behavior and use it on **i**.
Demographic	Demographic information about **U** and their ratings of items in **I**.	Demographic information about **u**.	Identify users that are demographically similar to **u**, and extrapolate from their ratings of **i**.
Utility-based	Features of items in **I**.	A utility function over items in **I** that describes **u**'s preferences.	Apply the function to the items and determine **i**'s rank.
Knowledge-based	Features of items in **I**. Knowledge of how these items meet a user's needs.	A description of **u**'s needs or interests.	Infer a match between **i** and **u**'s need.

Utility methods

- A **utility** function is a **map** from **states** onto **real numbers**, which describes the degree of happiness (utility) associated to the state
- A state could be an item but could also be a state of the human-computer interaction – for now it is a selected item
- One can try to build (or predict) a long term utility function but more often the systems using this approach try to acquire a **short term** utility function (ephemeral)
- These methods must hypothesize the user utility function, or the parameters defining such a function
  - How can you *hypothesize a function*?



Utility related information

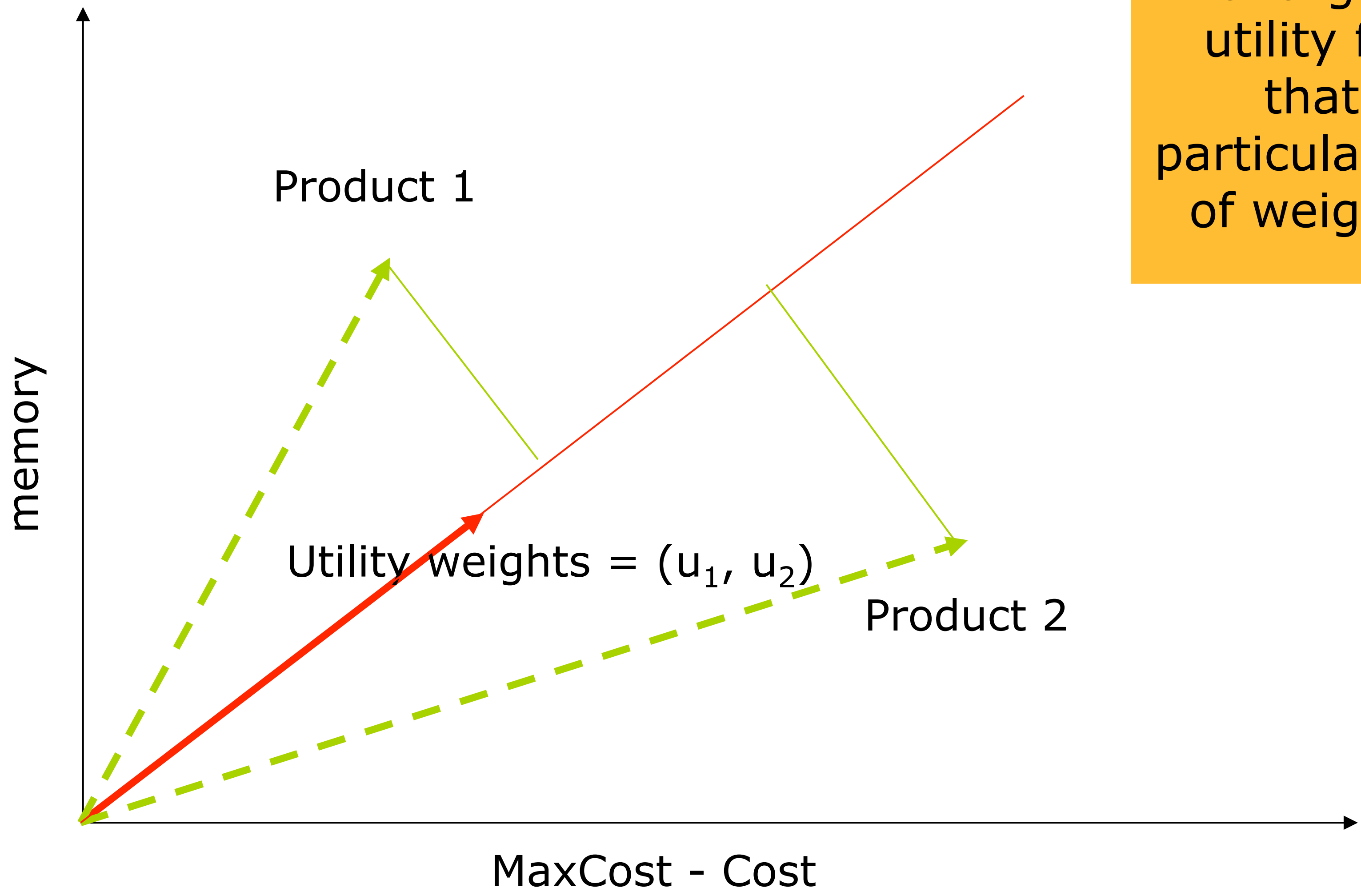
Utility: Linear Combination

- The item is described by a list of attributes: $p_1, \ldots p_m$, e.g., number of rooms, square meters, levels, (MaxCost – Cost), …
- It is generally assumed that higher values of the attribute correspond to higher utilities
- Or, in a simpler way, p_i is a Boolean value – 1 if the product has the required i-th feature
- The user utility function is modeled with a set of weights, $u_1, \ldots, u_m$ (in [0,1]) on the same attributes (user model)

$$U(u_1, \ldots, u_m, p_1, \ldots, p_m) = \sum_{j=1}^{m} u_j p_j$$

- The objective is to find (retrieve) the products with larger utility (maximal) – maximization of a linear function (trivial?!)
- *The problem is the* ***elicitation*** *or learning of the user model* $u_1, \ldots, u_m$.

Example



Product 2 has a larger utility for that particular set of weights

Hybrid Methods

- Try to address the **shortcomings** of both content-based and collaborative-based approaches, and produce recommendations using a combination of those techniques
- There is a **large variability** on these hybrid methods – there is no standard hybrid method
- We shall present some of them here but many will come later
- More in general, **hybrid methods** could be devised by combining two (or more) elementary methods: ex. Utility+Demographic.

Hybridization Methods

Hybridization method	Description
Weighted	The scores (or votes) of several recommendation techniques are combined together to produce a single recommendation.
Switching	The system switches between recommendation techniques depending on the current situation.
Mixed	Recommendations from several different recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm.
Cascade	One recommender refines the recommendations given by another.
Feature augmentation	Output from one technique is used as an input feature to another.
Meta-level	The model learned by one recommender is used as input to another.

Weighted Hybrid

- A simple approach for building hybrid systems - **weighted**:
  - $S_A(p)$ is the predicted rating for product p computed by algorithm A
  - $S_B(p)$ is the predicted rating for product p computed by algorithm B
  - $S_H(p) = aS_A(p) + (1\text{-}a)S_B(p)$ hybrid rating.

Weighted Ranking

Product	Rank for Score1	Score1	Score2	Rank for Score2	alpha	beta	Compound score	Hybrid Rank
					0,4	0,6		
a	1	0,9	0,5	6			0,66	2
b	2	0,7	0,6	4			0,64	3
c	3	0,65	0,95	1			0,83	1
d	4	0,6	0,58	5			0,59	4
f	5	0,4	0,46	7			0,44	6
g	6	0,2	0,3	8			0,26	8
h	7	0,1	0,88	2			0,57	5
i	8	0,04	0,1	10			0,08	10
l	9	0,03	0,66	3			0,41	7
m	10	0,02	0,23	9			0,15	9

$$\text{Compound Score} = \alpha * \text{Score1} + \beta * \text{Score2}$$

Weighted

- **The score of a recommended item is computed from the results of all of the available recommendation techniques present in the system**
  - Example 1: a linear combination of recommendation scores
  - Example 2 – many recommender systems: treats the output of each recommender (collaborative, content-based and demographic) as a set of votes, which are then combined in a consensus scheme
- The implicit assumption in this technique is that **the relative value of the different techniques is more or less uniform** across the space of possible items
- Not true in general: e.g. a collaborative recommender will be weaker for those items with a small number of raters.

Weighted Example

- Movie recommendations that integrates item-to-item collaborative filtering and information retrieval [Park et al., 2006]
- **Information retrieval component:** Web(i, q)= $(N+1-k_i)/N$ where N are the items returned by the query q and k_i is the position of movie i in the results set (example q = “arnold swarzenegger”)
  - Movies highly ranked by the IR component (low k_i) have a Web(i,q) value close to 1
- **Item-to-item collaborative filtering:** Auth(i, u) is the score of item i for user u
  - Movies similar to those highly ranked by the user in the past get a high Auth(i, u) score
- **Final rank:** MADRank(i,q,u)= a Auth(i,u) +(1-a)Web(i,q)
- If Auth(i,u) cannot be computed (not enough ratings for u or i) then Auth(i,u) can be a non personalized score (e.g. item popularity) or simply not used (also some switching!)

Switching

- The system uses some criterion to **switch** between recommendation techniques
- Example: The DailyLearner [Billsus and Pazzani, 2000] system uses a content/collaborative hybrid in which a content-based recommendation method is employed first
- If the content-based system cannot make a recommendation with sufficient confidence (how?), then a collaborative recommendation is attempted
  - *We need a method to measure the confidence of a prediction*
- This switching hybrid does not completely avoid the ramp-up problem, since both the collaborative and the content-based systems have the "new user" problem
- The main problem of this technique is to identify a GOOD switching condition.

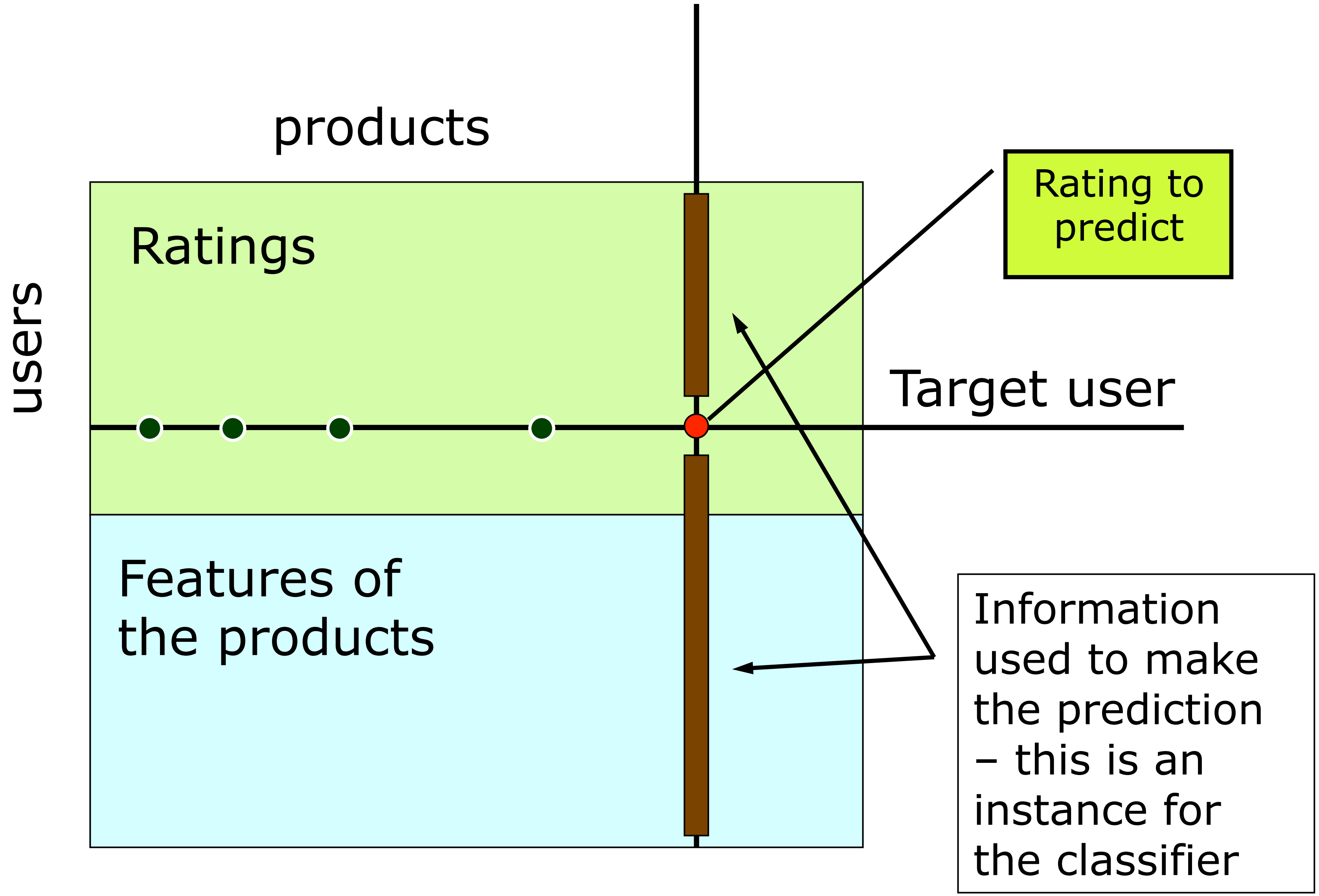
Mixed

- **Recommendations from more than one technique are presented together**
- The mixed hybrid **avoids the “new item” start-up problem**:
  - since the content-based approach can be used for new items
- It does not get around the “new user” start-up problem:
  - both the content and collaborative methods need some data about user preferences to start up
- But it is a good idea for hybridizing two different kind of recommender (e.g. demographic and collaborative)
- It introduces DIVERSITY in the recommendation list.

Feature Combination

- Achieves the content/collaborative merger **treating collaborative information (ratings of users) as simply additional feature** data associated with each example and use content-based techniques over this augmented data set
- [Basu, Hirsh & Cohen 1998] apply the inductive rule learner Ripper to the task of recommending movies using both users' ratings and content features
- The feature combination hybrid lets the system consider collaborative data without relying on it exclusively, so it reduces the sensitivity of the system to the number of users who have rated an item
- The system have information about the inherent similarity of items that are otherwise opaque to a collaborative system.

Feature Combination



● Known Ratings of the target user

Cascade

- *One recommendation technique is employed first to produce a coarse ranking of candidates and a second technique refines the recommendation from among the candidate set*
- Example: EntreeC uses its knowledge of restaurants to make recommendations based on the user's stated interests:
  - The recommendations are placed in buckets of equal preference (equal utility)
  - and the collaborative technique is employed to break ties
- Cascading allows the system to avoid employing the second, lower-priority, technique on items that are already well-differentiated by the first
- But requires a meaningful and constant ordering of the techniques.

Feature Augmentation

- *Produce a rating or classification of an item and that information is then incorporated into the processing of the next recommendation technique*
- Example: Libra system [Mooney & Roy 1999] makes content-based recommendations of books based on data found in Amazon.com, using a naive Bayes text classifier
- In the text data used by the system is included “related authors” and “related titles” information that Amazon generates using its internal collaborative systems
- Very similar to the feature combination method:
  - **Here** the output of a recommender system is used for a second RS
  - In **feature combination** the representations used by two systems are combined.

Meta-level

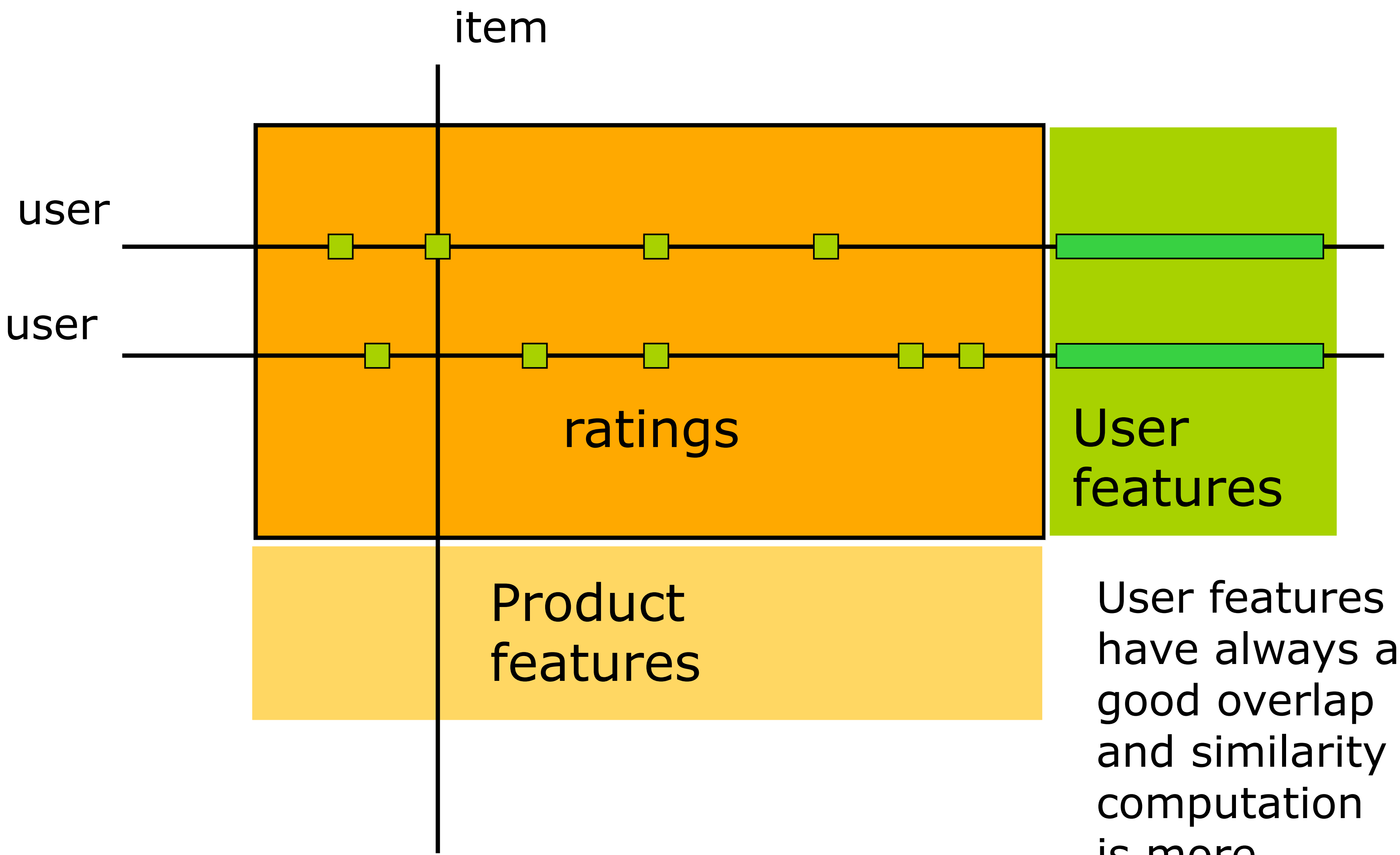
- Using the model generated by one as the input for another
- Example: FAB system
  - user-specific selection agents perform content-based filtering using Rocchio's method to maintain a term vector model that describes the user's area of interest (*Model 1*)
  - Collection agents, which gather new pages from the web, **use the models from all users** in their gathering operations (*Model 2*)
  - Documents are first collected on the basis of their interest to the community as a whole (*Model 2*) and then distributed to particular users (*Model 1*)
- Example: [Pazzani 1999] collaboration via content: **the model generated by the content-based approach (winnow –model 1) is used for representing the users in a collaborative filtering approach (model 2).**

Collaboration via Content

- **Problem addressed:** in a collaborative-based recommender, products co-rated by a pair of users may be very few – hence in this case correlation between two users is not reliable
- In **collaboration via content** a *content-based profile* of each user is exploited to detect similarities among users
- Main problems to solve are:
  - How to build a content-based profile for each user?
  - What kind of knowledge must be used?
  - How to measure user-to-user similarity?

[Pazzani, 1999]

A Bidimensional Model



User features have always a good overlap and similarity computation is more reliable.

Content-Based Profiles

	noodle	shrimp	basil	exotic	salmon	Dolce
Karen	2.5	0	.2	0	0	+
Lynn	1.1	0	1.1	1.5	0	–
Chris	1.5	0	3.5	1.5	.5	+
Mike	1.1	1.1	2.1	2.0	2.5	–
Jill	1.1	2.2	0	0	3.5	?

- The weights can be the average of the TF-IDF vectors of the documents that are highly rated by the user (as in FAB or in Syskill & Webert) – centroid of the documents he likes
  - E.g. in the restaurants liked by Karen the word "noodle" is very frequent (and not much frequent in the entire collection of restaurant descriptions)
- Or you can use **winnow** as in [Pazzani, 1999], to learn the user model (see next slide…)
  - A user is modeled by his/her linear classifier classifying the good and bad restaurants.

Winnow (learning a user model)

- Each word appearing in the item descriptions evaluated by a user is considered as a Boolean feature (present/ not present)
  - Bag-of-Words model
- Winnowing learns (for each user) a weight w_i associated to each word x_i
  - Weights are positive
  - Similar to the factor models in CF – if the factors are the keywords …
- The larger the weight the more important is the corresponding word in the items that the user likes
- The weights together represents a linear classifier for that user.

Weights Learning

- Initially all the weights are set to 1. Then, for each document rated by the user a linear threshold function is computed

$$\sum w_i x_i > \tau$$

- If the above sum is **over the threshold** and the **user did not liked** the document, then the weights associated with each word in the document are **divided** by 2
- If the sum is **below the threshold** and the **user liked** the document then all weights associated with words in the document are **multiplied** by 2
- Otherwise no change is made
- The set of training examples is cycled through adjusting the weights until all the examples are processed correctly and no changes are made to the weights.

[Pazzani, 1999]

Winnow in the general case

- The Winnow algorithm takes as input an initial vector w= $(w_1, \ldots, w_n)$, a promotion factor α, and a threshold τ
- The algorithm requires that:
  - w is positive (i.e., each coordinate of w is positive)
  - $\alpha > 1$ *(previous slide $\alpha = 2$)*
  - $\tau > 0$
- Winnow proceeds in a series of trials and predicts in each trial according to the threshold function (inner product) $w \cdot x > \tau$
- If the prediction is correct, then no update is performed; otherwise the weights are updated as follows:
  - On false positive (erroneously **above** the threshold), for all i, $w_i \leftarrow \alpha^{-x_i} w_i$
  - On false negative (erroneously **below** the threshold), for all i, $w_i \leftarrow \alpha^{x_i} w_i$

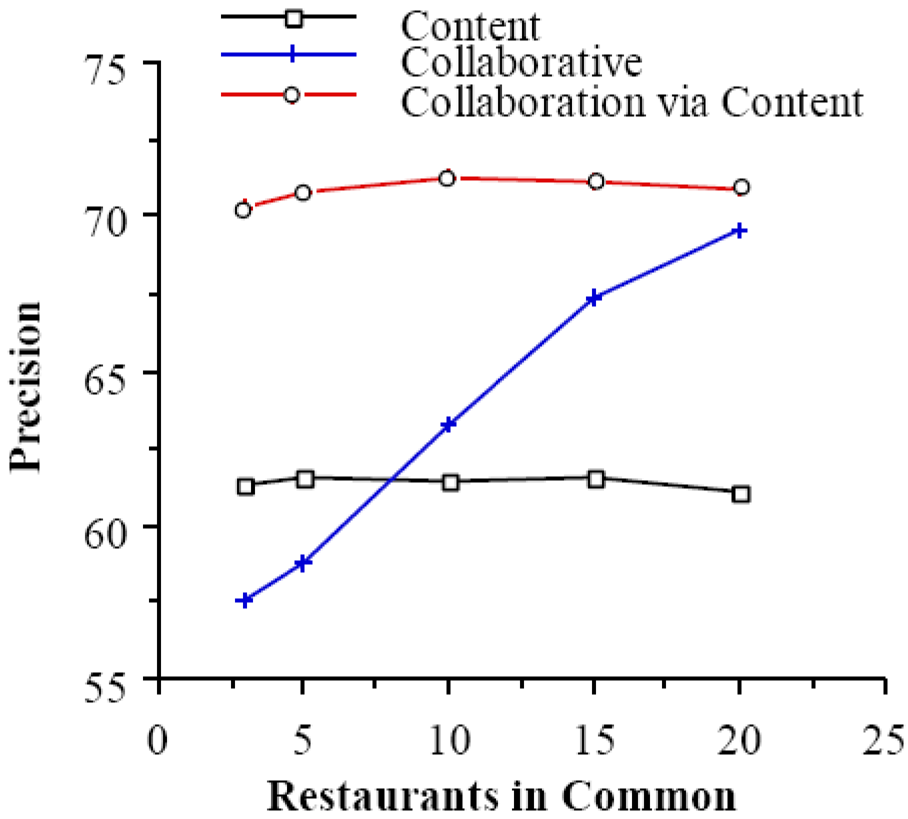
Content-Based using Winnow

- They have built a content-based recommender using
  - The **ratings of a user on a set of restaurants**
  - The user profile is built using the **winnow** technique
  - The recommendation for a new restaurant is based on the threshold function
    - If the inner product of the user model multiplied by the Boolean vector of the restaurant is above the threshold → prediction is +
    - Otherwise the prediction is -

Collaborative-Based

- They have built **two collaborative based systems:**
  - **Standard** collaborative filtering using Pearson correlation
  - **Collaboration via Content**
    - For each user the profile is built as for the content-based recommender system
    - Winnow used for learning the feature weights for each feature (e.g., noodles, shrimps, basil, salmon, etc.)
    - Similarity between two users is performed using Pearson on the two profiles.

Comparison



Averaged on 44 users

Precision is computed in the top 3 recommendations = (# of plus in the recommendation list)/3

- Content–based recommendation is done with winnow
- Collaborative is standard using Pearson correlation
- Collaboration via content uses the content-based user profiles built by winnow.

Discussion

- Collaboration via content is attractive because
  - You do not to have to collect experiences (ratings) of the users on **common products**
  - It may be applicable to recommend products in a product category **even if the user has not rated any product in that category** but there are some other ratings that enable to generate a user model (the weights of the user features)
  - The user profile is built using ratings, but provided that you have **some ratings** in a domain, you can build a "useful" user model
  - It could be used to **bootstrap a collaborative filtering** system (when not enough ratings are available).

Summary

- Content-based methods are well rooted in information retrieval
- A content-based method is a classifier and exploits only knowledge derived from observing the target user
- Examples:
  - Naive Bayes classifier
  - Centroid
- Demographic methods are very simple and could provide limited personalization (sometime can be sufficient)
- Utility-based methods go to the root of the decision problem – but how to acquire the utility function?
- Hybrid methods are the most powerful and popular right now – there are plenty of options for hybridization
- We mostly described content-based and collaborative-based hybrids – but you may build hybrid systems combining any kind of RS
- The simplest and largely used methods are: weighted, switched, and mixed approaches.

Questions

- Can a content-based recommender operate in a not networked environment?
- List a set of attributes of a recommender system and compare a content-based system to a collaborative-based one
- Is the Centroid of the interesting documents a good User Model? What are the problems of this representation? How to exploit ephemeral needs?
- How to build a content-based recommender for music or photography?
- Can a utility function be learned or acquired without explicitly asking?
- Could you imagine different ways (not the sum) to integrate the utility over a single issue to produce the total utility?

Questions

- What are the pros and cons of different hybridization approaches?
- What is the user profile in a collaboration via content approach?
- Can collaboration via content be applied for catalogues containing multiple types of products (e.g., dig. cameras and movies)?
- How it is structured the product model and the user model in a content-based filtering system based on winnow?
- In the “weighted” approach, how the weights could be determined?
- What are the similarities between the “feature combination” approach and item-to-item collaborative filtering?