Big Data Computing

Master's Degree in Computer Science

2019-2020

Gabriele Tolomei

Department of Computer Science

Sapienza Università di Roma

tolomei@di.uniroma1.it

SAPIENZA
UNIVERSITÀ DI ROMA

Recap from Last Lectures

- So far, we have talked about **2 main tasks** which are pretty common in the context of "Big Data":

Recap from Last Lectures

- So far, we have talked about **2 main tasks** which are pretty common in the context of "Big Data":
  - **Clustering** (unsupervised learning)

Recap from Last Lectures

- So far, we have talked about **2 main tasks** which are pretty common in the context of "Big Data":
  - **Clustering** (unsupervised learning)
  - **Regression**/**Classification** (supervised learning)

Recap from Last Lectures

- So far, we have talked about **2 main tasks** which are pretty common in the context of "Big Data":
  - **Clustering** (unsupervised learning)
  - **Regression**/**Classification** (supervised learning)
- We have discussed a number of techniques to solve those tasks:
  - **K-means**, **PCA**
  - **Linear Regression**, **Logistic Regression**, **Decision Trees**, and ensembles

Information Overload

- With the explosion of content and services available on the Web, consumers are inundated with choices

Information Overload

- With the explosion of content and services available on the Web, consumers are inundated with choices
- This phenomenon is usually referred to as **information overload**

Information Overload

- With the explosion of content and services available on the Web, consumers are inundated with choices
- This phenomenon is usually referred to as **information overload**
- To actually take advantage of such a huge pile of information we need techniques that narrow down it to **human-manageable** figures:

Information Overload

- With the explosion of content and services available on the Web, consumers are inundated with choices
- This phenomenon is usually referred to as **information overload**
- To actually take advantage of such a huge pile of information we need techniques that narrow down it to **human-manageable** figures:
  - **Searching/Filtering**

Information Overload

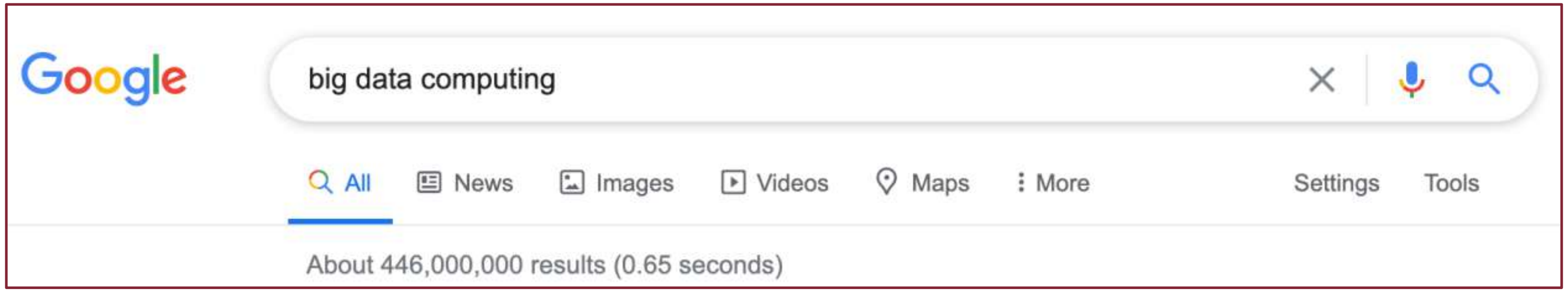
- With the explosion of content and services available on the Web, consumers are inundated with choices
- This phenomenon is usually referred to as **information overload**
- To actually take advantage of such a huge pile of information we need techniques that narrow down it to **human-manageable** figures:
  - **Searching/Filtering**
  - **Recommending**

Why Do We Need Recommendation?

We are constantly moving from scarcity to abundance

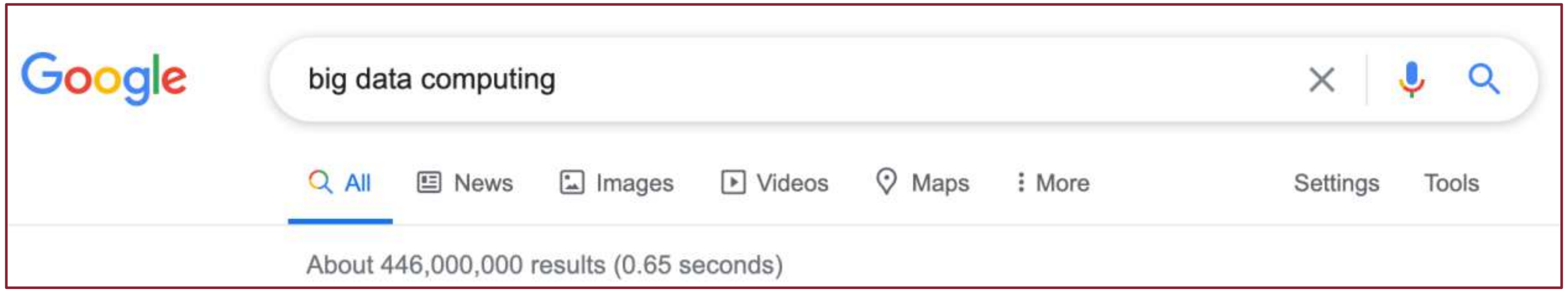
Why Do We Need Recommendation?

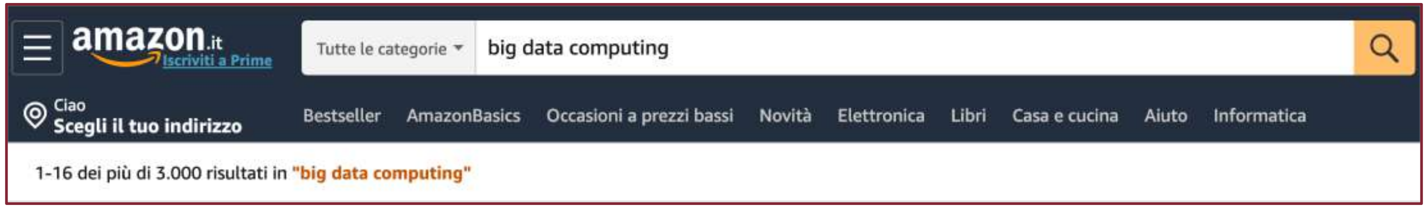
We are constantly moving from scarcity to abundance



Why Do We Need Recommendation?

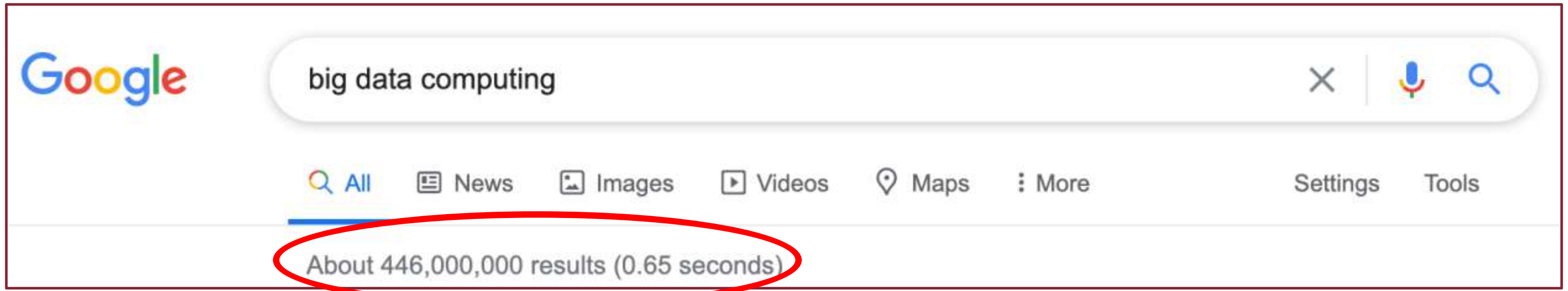
We are constantly moving from scarcity to abundance

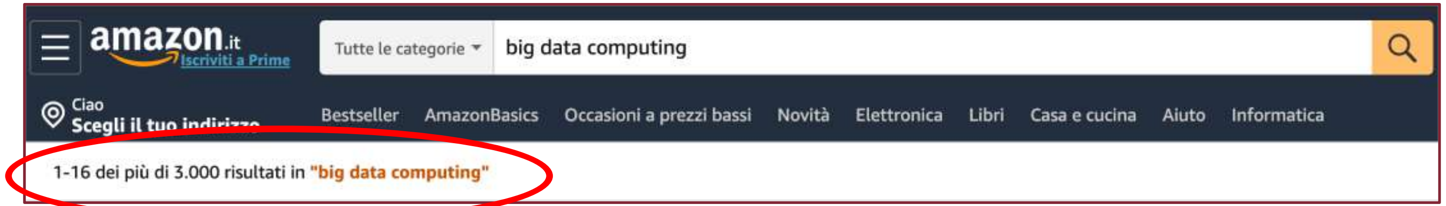




Why Do We Need Recommendation?

We are constantly moving from scarcity to abundance

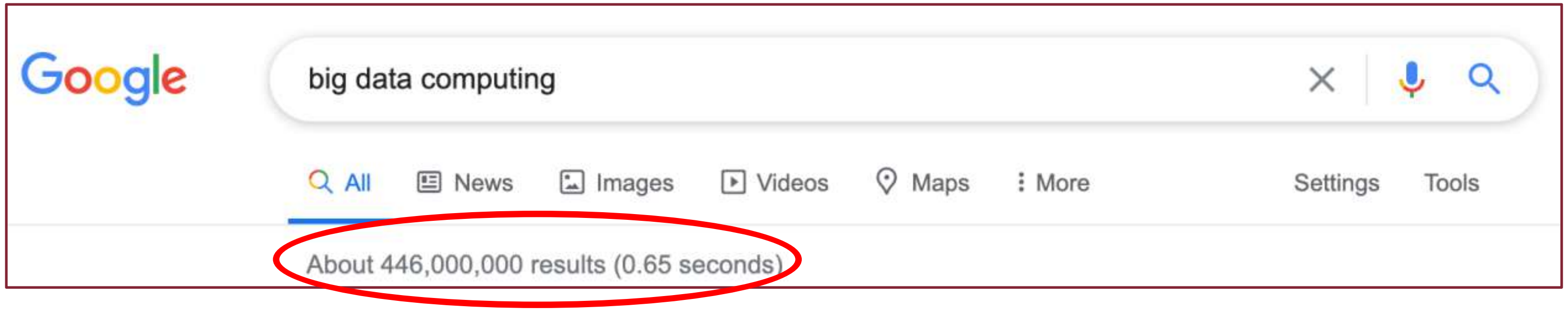


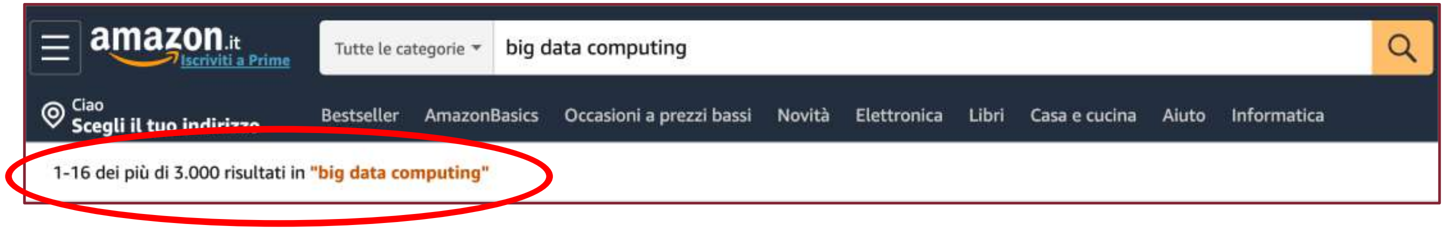


The number of relevant "items" of interest is huge

Why Do We Need Recommendation?

We are constantly moving from scarcity to abundance





The number of relevant "items" of interest is huge

How could we even possibly think of exhaustively explore all of them?

Why Do We Need Recommendation?

Compared to physical stores, online catalogs of product/services are almost unlimited

Why Do We Need Recommendation?

Compared to physical stores, online catalogs of product/services are almost unlimited

Matching consumers with the most appropriate products/services is key to enhancing **user satisfaction** and loyalty

Why Do We Need Recommendation?

Compared to physical stores, online catalogs of product/services are almost unlimited

Matching consumers with the most appropriate products/services is key to enhancing **user satisfaction** and loyalty

More choices need **better filters**!

Why Do We Need Recommendation?

Compared to physical stores, online catalogs of product/services are almost unlimited

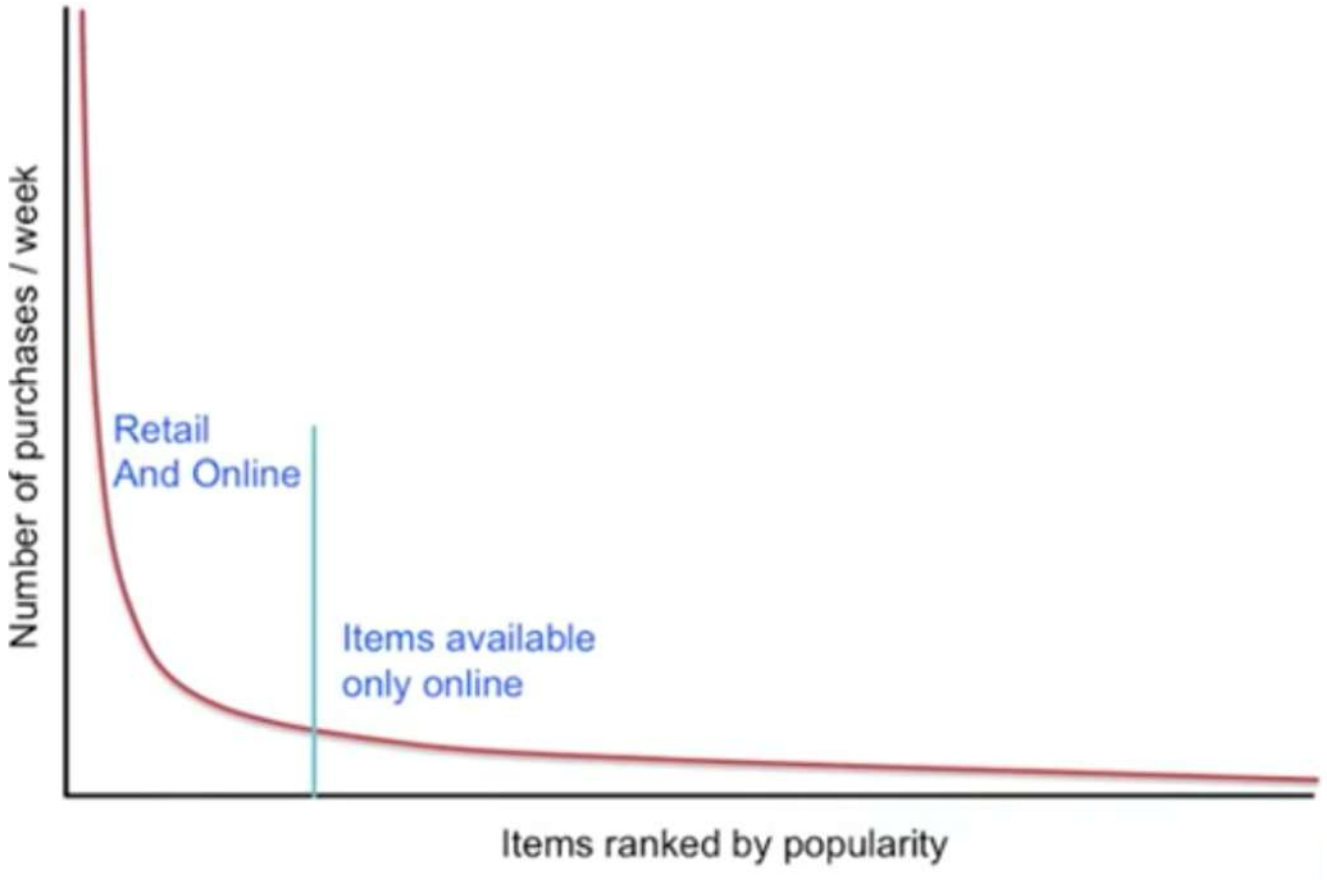
Matching consumers with the most appropriate products/services is key to enhancing **user satisfaction** and loyalty

More choices need **better filters**!

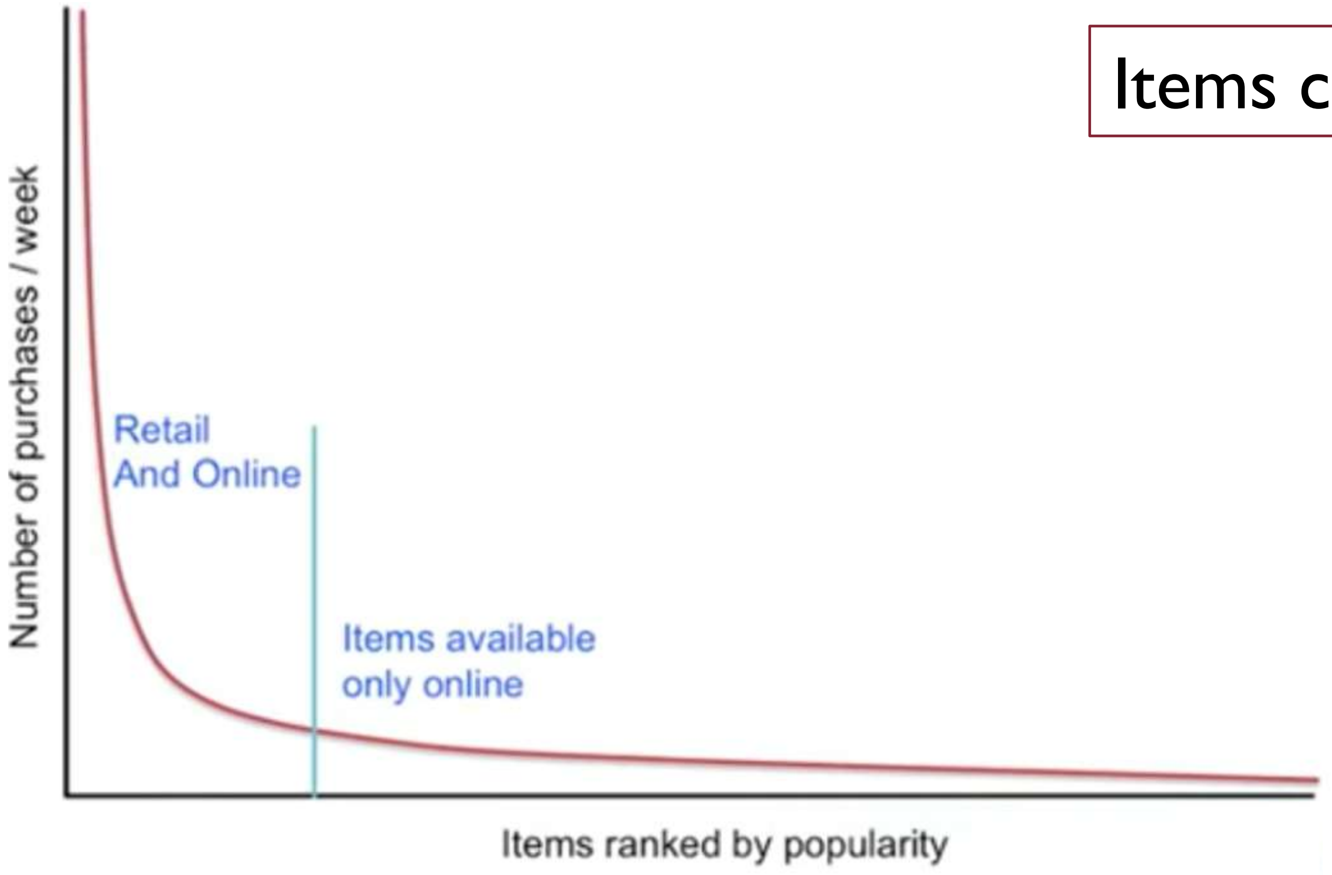
↓

Recommender Systems

The Economics of Abundance: The Long Tail

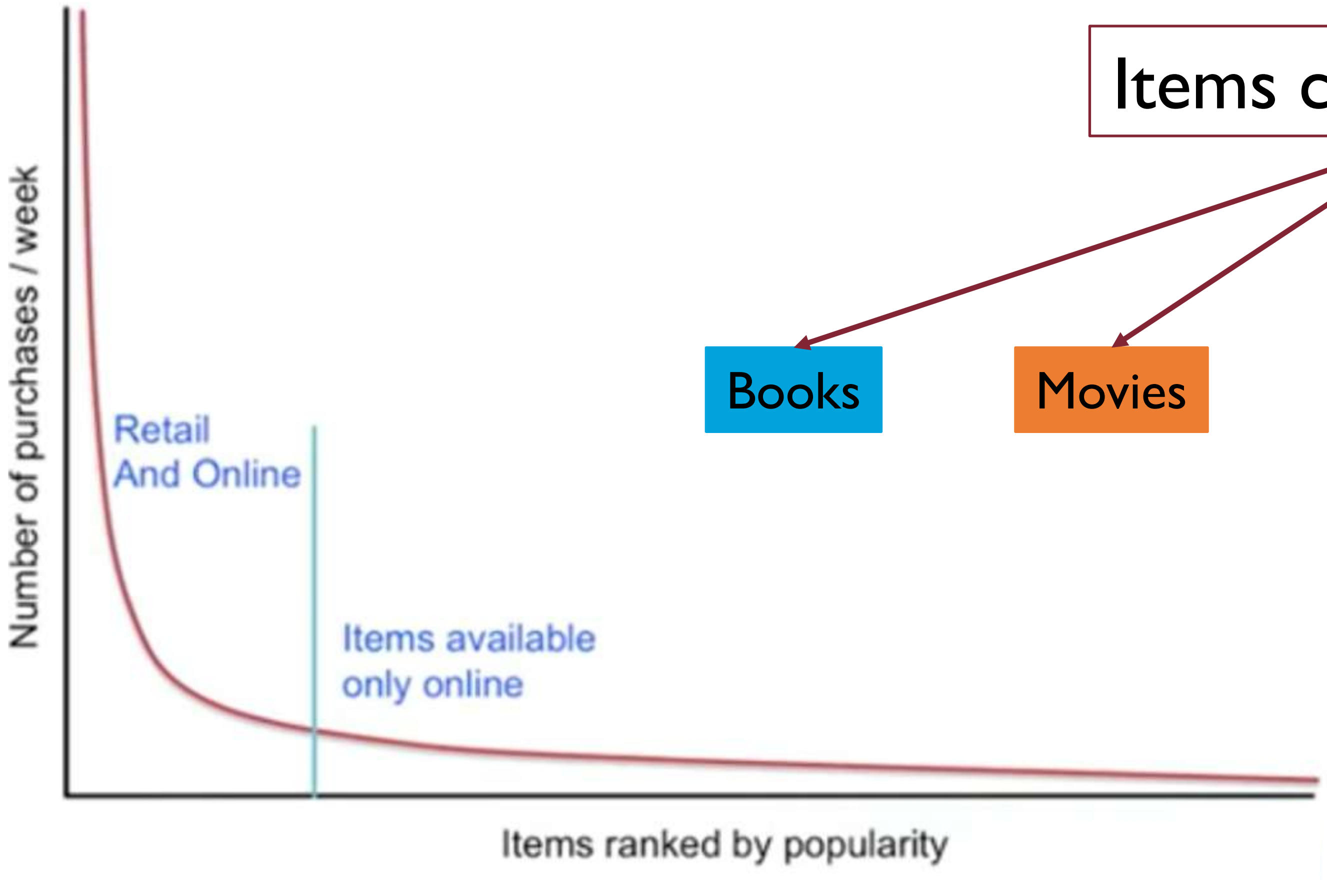


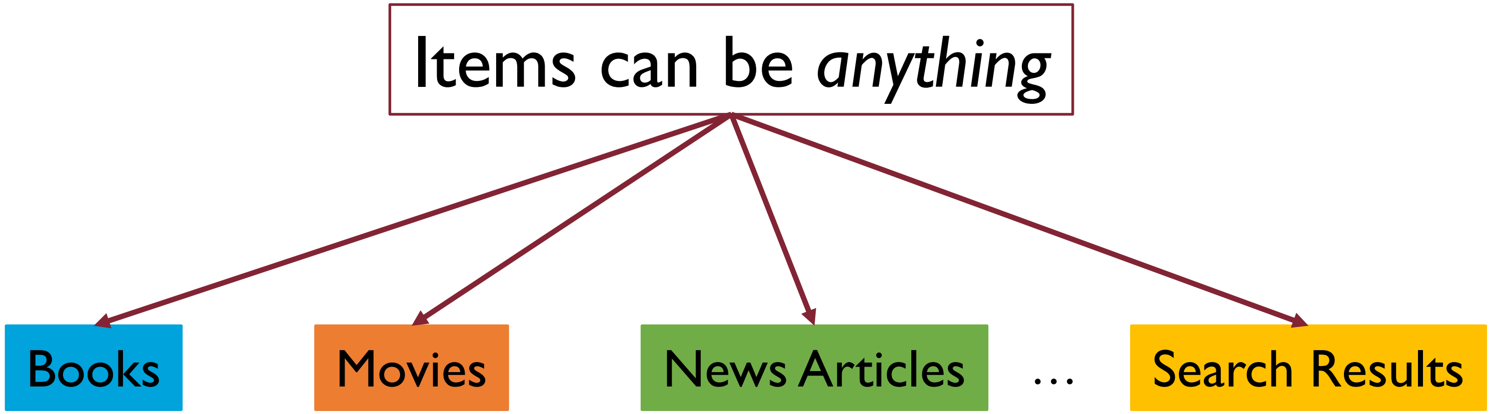
The Economics of Abundance: The Long Tail



Items can be *anything*

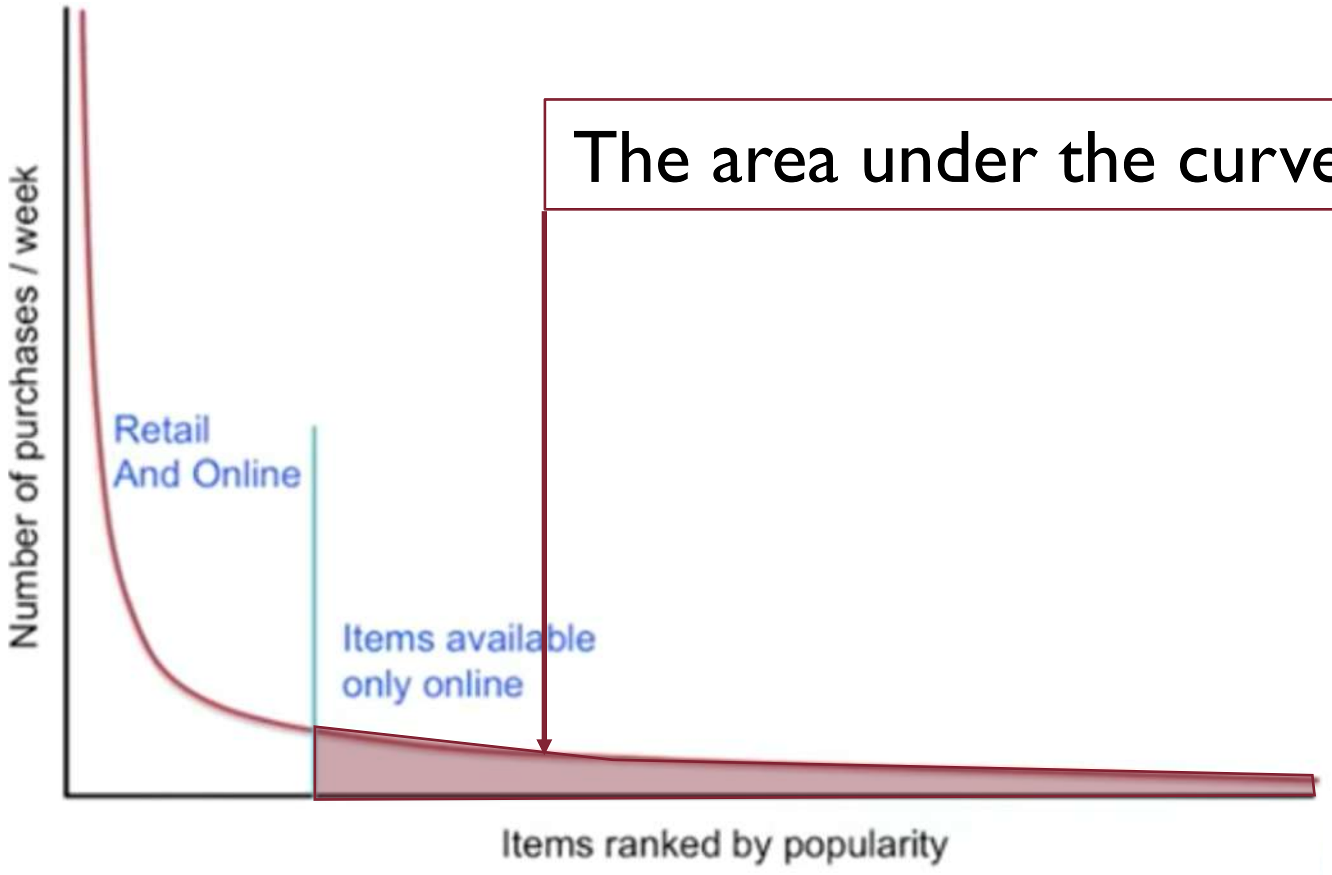
The Economics of Abundance: The Long Tail



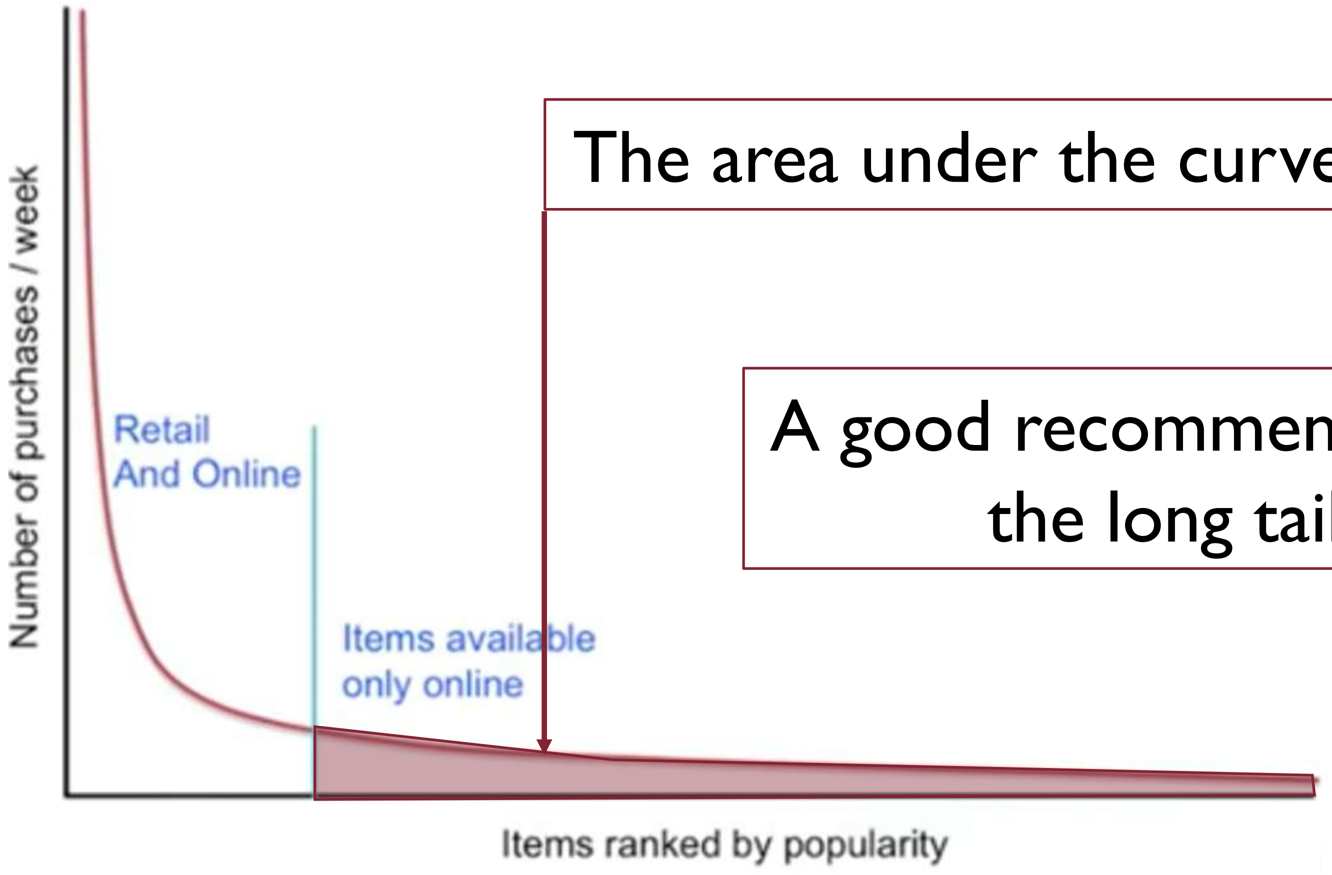


The Economics of Abundance: The Long Tail

The area under the curve for the long tail is quite **large**!



The Economics of Abundance: The Long Tail



The area under the curve for the long tail is quite **large**!

A good recommendation system should expose the long tail/hidden items to users

Recommender Systems: Formalism

$\mathcal{U} = \{u_1, \ldots, u_m\}$ Set of **users**

Recommender Systems: Formalism

$\mathcal{U} = \{u_1, \ldots, u_m\}$ Set of **users**

$\mathcal{I} = \{i_1, \ldots, i_n\}$ Set of **items**

Recommender Systems: Formalism

$\mathcal{U} = \{u_1, \ldots, u_m\}$ Set of **users**

$\mathcal{I} = \{i_1, \ldots, i_n\}$ Set of **items**

$r : \mathcal{U} \times \mathcal{I} \mapsto \mathcal{R}$ **utility function** (**user-item matrix**)

Recommender Systems: Formalism

$\mathcal{U} = \{u_1, \ldots, u_m\}$ Set of **users**

$\mathcal{I} = \{i_1, \ldots, i_n\}$ Set of **items**

$r : \mathcal{U} \times \mathcal{I} \mapsto \mathcal{R}$ **utility function** (**user-item matrix**)

$\mathcal{R} \subseteq \mathbb{R}$ Set of **ratings** (totally ordered)

Recommender Systems: Formalism

$\mathcal{U} = \{u_1, \ldots, u_m\}$ Set of **users**

$\mathcal{I} = \{i_1, \ldots, i_n\}$ Set of **items**

$r : \mathcal{U} \times \mathcal{I} \mapsto \mathcal{R}$ **utility function** (**user-item matrix**)

$\mathcal{R} \subseteq \mathbb{R}$ Set of **ratings** (totally ordered)

$\mathcal{R} = \{0, 1, \ldots, v-1\}$ Discrete ratings (e.g., 0-5 stars)

$\mathcal{R} = [0, 1]$ Continuous ratings

The Utility Function (User-Item Matrix)

USERS	
	Alice
	Bob
	Carl
	…
	Zoe

The Utility Function (User-Item Matrix)

MOVIES

USERS

 Alice

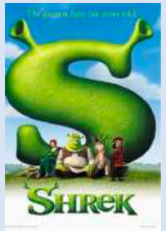
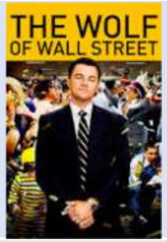
 Bob

 Carl

…

 Zoe

The Utility Function (User-Item Matrix)

		MOVIES							
									
USERS	 Alice	2		5	4	5	4		4
	 Bob	4					3		3
	 Carl	5	5	3	4	5	4		5
	…	…	…	…	…	…	…	…	…
	 Zoe		1	3				5	4

The Utility Function (User-Item Matrix)

3 key problems for a recommender system

The Utility Function (User-Item Matrix)

3 key problems for a recommender system

Data Collection

Gathering known ratings to populate the utility matrix

The Utility Function (User-Item Matrix)

3 key problems for a recommender system

Data Collection

Gathering known ratings to populate the utility matrix

Rating Prediction

Extrapolate unknown ratings from the known ones

The Utility Function (User-Item Matrix)

3 key problems for a recommender system

Data Collection

Gathering known ratings to populate the utility matrix

Rating Prediction

Extrapolate unknown ratings from the known ones

Recommendation Evaluation

Measure the performance of recommender methods

Data Collection

Data Collection

Gathering known ratings to populate the utility matrix

Data Collection

Data Collection

Gathering known ratings to populate the utility matrix

Explicit

Ask people to rate items

Doesn't scale: only few users leave ratings

Data Collection

Data Collection

Gathering known ratings to populate the utility matrix

Explicit

Ask people to rate items

Doesn't scale: only few users leave ratings

Implicit

Learn ratings from user actions

Click/purchases implies positive feedback
What about negative ones?

Rating Prediction

Rating Prediction

Extrapolate unknown ratings from the known ones

Rating Prediction

Rating Prediction

Extrapolate unknown ratings from the known ones

The utility matrix R is sparse!

Most people have not rated most items

Rating Prediction

Rating Prediction

Extrapolate unknown ratings from the known ones

The utility matrix R is sparse!

Most people have not rated most items

Cold Start

New users/items have no history

Recommendation Evaluation

Recommendation Evaluation

Measure the performance of recommender methods

Recommendation Evaluation

Recommendation Evaluation

Measure the performance of recommender methods

RMSE

Mean Average Precision/Recall at K (MAP@K/MAR@K)

Personalization

Serendipity

Recommendation Strategies

3 approaches to recommender systems

Recommendation Strategies

3 approaches to recommender systems

Content-based filtering

Recommendation Strategies

3 approaches to recommender systems

Content-based filtering

Collaborative filtering

Recommendation Strategies

3 approaches to recommender systems

Content-based filtering

Collaborative filtering

Neighborhood-based
(a.k.a. memory-based)

Recommendation Strategies

3 approaches to recommender systems

Content-based filtering

Collaborative filtering

Neighborhood-based
(a.k.a. memory-based)

Latent-factor-based
(a.k.a. model-based)

Recommendation Strategies

3 approaches to recommender systems

Content-based filtering

Collaborative filtering

- **Neighborhood-based** (a.k.a. memory-based)
- **Latent-factor-based** (a.k.a. model-based)
- **Hybrid** (memory-model-based)

Recommendation Strategies

3 approaches to recommender systems

Content-based filtering

Collaborative filtering

Hybrid

Neighborhood-based
(a.k.a. memory-based)

Latent-factor-based
(a.k.a. model-based)

Hybrid
(memory-model-based)

CONTENT-BASED FILTERING

Content-based Filtering

Idea

Recommend items to user *u* similar to previous items rated highly by *u*

Content-based Filtering

Idea

Recommend items to user *u* similar to previous items rated highly by *u*

Core concept: Item/User **Profiles**

Content-based Filtering

Idea

Recommend items to user *u* similar to previous items rated highly by *u*

Core concept: Item/User **Profiles**

Steps

1. Build **item profiles** (i.e., a description of items using metadata information)

Content-based Filtering

Idea

Recommend items to user *u* similar to previous items rated highly by *u*

Core concept: Item/User **Profiles**

Steps

1. Build **item profiles** (i.e., a description of items using metadata information)
2. Based on the item profiles, build **user profiles**: user profile says what the user likes

Content-based Filtering

Idea

Recommend items to user *u* similar to previous items rated highly by *u*

Core concept: Item/User **Profiles**

Steps

1. Build **item profiles** (i.e., a description of items using metadata information)
2. Based on the item profiles, build **user profiles**: user profile says what the user likes
3. Match the user profile with the item catalog

Building Item Profiles

Goal

For each item i create a **profile**, i.e., a set of features

Building Item Profiles

Goal

For each item *i* create a **profile**, i.e., a set of features

Movies

```
- Author
- Title
- Director
- Genre
```

…

Images/Videos

```
- Width
- Height
- Framerate
- Tags
```

…

…

People

```
- Age
- Sex
- Job
- Friends
```

…

Building Item Profiles

Goal

For each item *i* create a **profile**, i.e., a set of features

Movies

```
- Author
- Title
- Director
- Genre
```
…

Images/Videos

```
- Width
- Height
- Framerate
- Tags
```
…

…

People

```
- Age
- Sex
- Job
- Friends
```
…

Think of each profile as a vector of numerical/categorical features

Item Profile: Example

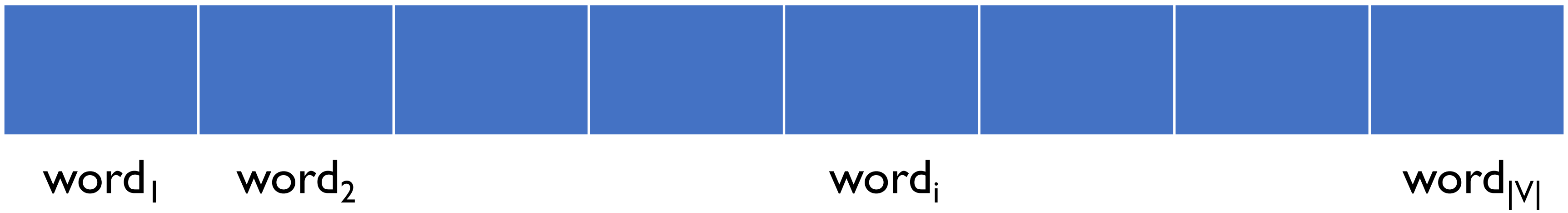
- Suppose we want to build a news recommender system

Item Profile: Example

- Suppose we want to build a news recommender system
- Items are news article (i.e., text documents)

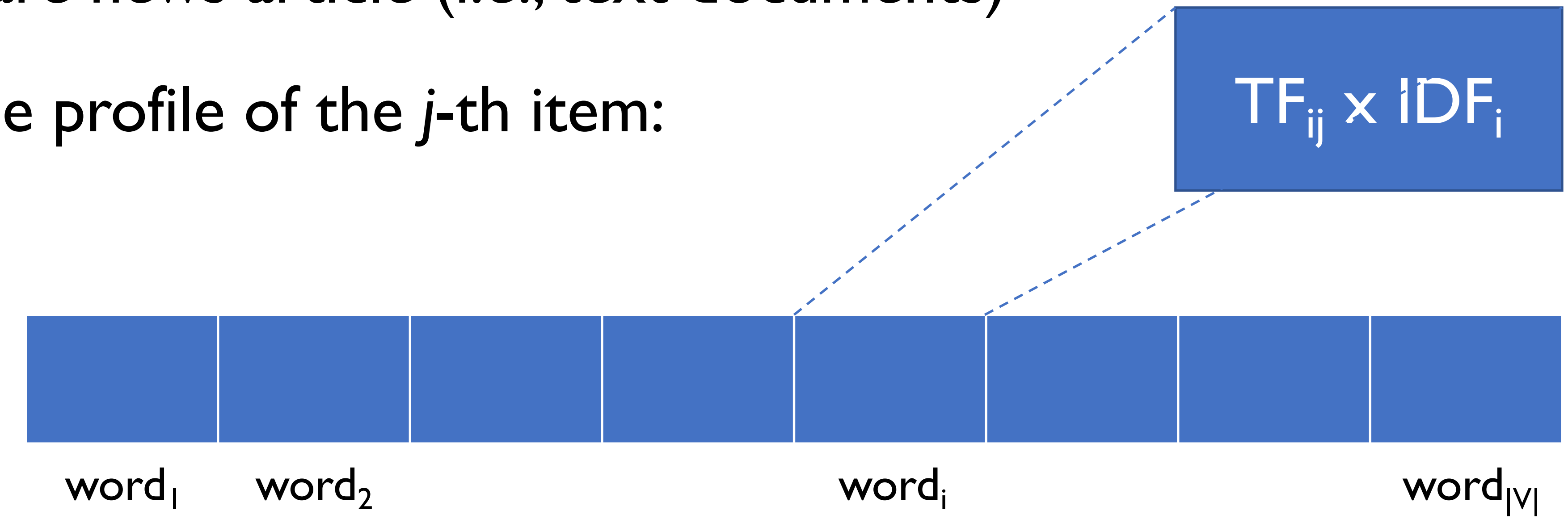
Item Profile: Example

- Suppose we want to build a news recommender system
- Items are news article (i.e., text documents)
- Possible profile of the *j*-th item:



Item Profile: Example

- Suppose we want to build a news recommender system
- Items are news article (i.e., text documents)
- Possible profile of the *j*-th item:



Building User Profiles

Goal

For each user *u* create a **profile**, from the profiles of the items she rated

Building User Profiles

Goal

For each user *u* create a **profile**, from the profiles of the items she rated

$\mathcal{I}_u \subseteq \mathcal{I}$ The set of items rated by user *u*

Building User Profiles

Goal

For each user *u* create a **profile**, from the profiles of the items she rated

$\mathcal{I}_u \subseteq \mathcal{I}$ The set of items rated by user *u*

$\mathbf{i}_j$ The vector profile of item *j*

Building User Profiles

Goal

For each user *u* create a **profile**, from the profiles of the items she rated

$\mathcal{I}_u \subseteq \mathcal{I}$ The set of items rated by user *u*

$\mathbf{i}_j$ The vector profile of item *j*

The simplest solution to build the user profile is to take the average of item profiles rated

$$\mathbf{u}_i = \frac{1}{|\mathcal{I}_u|} \sum_{\mathbf{i}_j \in \mathcal{I}_u} \mathbf{i}_j$$

Building User Profiles

Goal

For each user *u* create a **profile**, from the profiles of the items she rated

$\mathcal{I}_u \subseteq \mathcal{I}$ The set of items rated by user *u*

$\mathbf{i}_j$ The vector profile of item *j*

The simplest solution to build the user profile is to take the average of item profiles rated

$$\mathbf{u}_i = \frac{1}{|\mathcal{I}_u|} \sum_{\mathbf{i}_j \in \mathcal{I}_u} \mathbf{i}_j$$

All the items are treated equally, independently of the rating

Simple User Profile: Example

Items = Movies

Simple User Profile: Example

Items = Movies

Movie Profile = List of Actors

Simple User Profile: Example

Items = Movies

Movie Profile = List of Actors

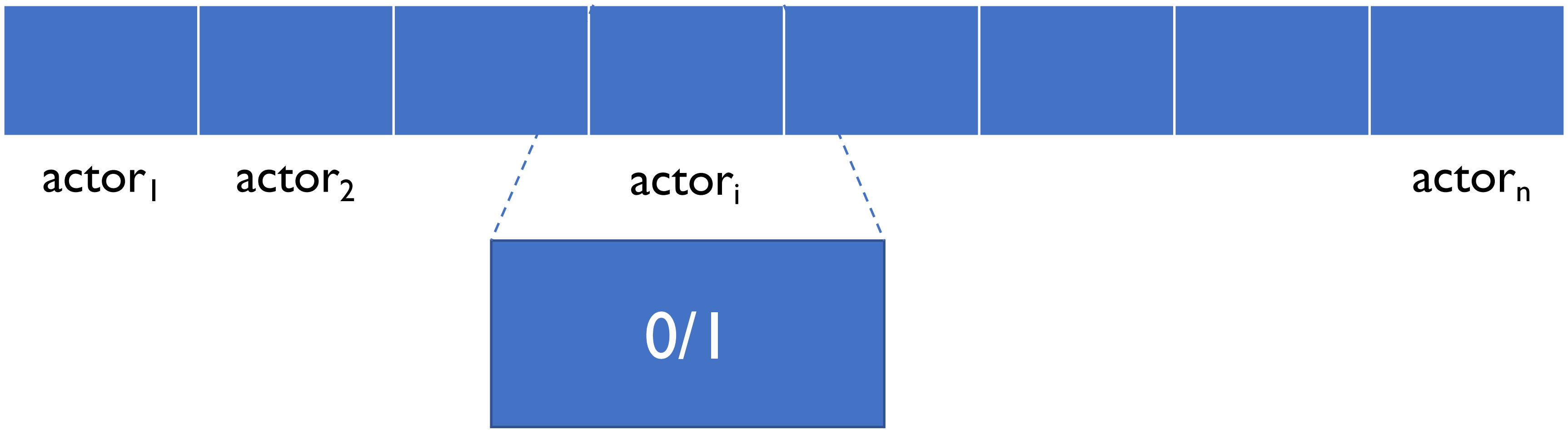
j-th movie

Simple User Profile: Example

Items = Movies

Movie Profile = List of Actors

j-th movie



Binary feature indicating if $actor_i$ appears in $movie_j$

Simple User Profile: Example

Suppose user *u* has watched **5** movies, each movie represented by **2** actors

1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

Simple User Profile: Example

Suppose user *u* has watched **5** movies, each movie represented by **2** actors

1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

2 movies feature actor 1

Simple User Profile: Example

Suppose user *u* has watched **5** movies, each movie represented by **2** actors

1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

3 movies feature actor 2

Simple User Profile: Example

Suppose user *u* has watched **5** movies, each movie represented by **2** actors

1	0	0	1	0	1	1	0	0	1
actor$_1$	actor$_2$	actor$_1$	actor$_2$	actor$_1$	actor$_2$	actor$_1$	actor$_2$	actor$_1$	actor$_2$

User profile is the **mean** of item profiles

Simple User Profile: Example

Suppose user *u* has watched **5** movies, each movie represented by **2** actors

1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

User profile is the **mean** of item profiles

$(1+1)/5$

0.4	
$actor_1$	$actor_2$

Simple User Profile: Example

Suppose user *u* has watched **5** movies, each movie represented by **2** actors

1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

User profile is the **mean** of item profiles

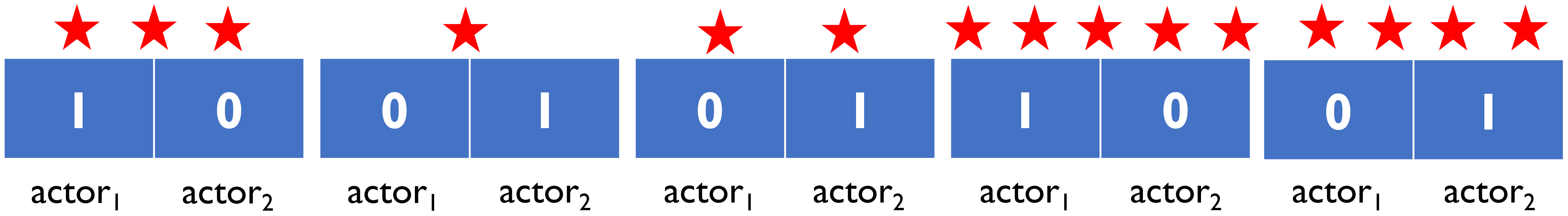
$(1+1)/5$

0.4	0.6
$actor_1$	$actor_2$

$(1+1+1)/5$

Simple User Profile: Example With Ratings

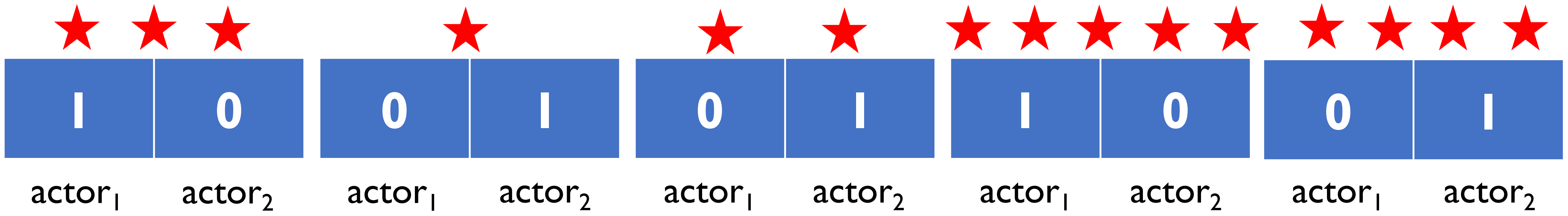
Suppose user *u* has watched (and rated) **5** movies



Normalize ratings by subtracting user's mean rating before

Simple User Profile: Example With Ratings

Suppose user *u* has watched (and rated) **5** movies



Normalize ratings by subtracting user's mean rating before

Avg. User Rating = (3 + 1 + 2 + 5 + 4)/5 = 3

Simple User Profile: Example With Ratings

Suppose user *u* has watched (and rated) **5** movies

3-3 = 0		1-3 = -2		2-3 = -1		5-3 = 2		4-3 = 1	
1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

Normalize ratings by subtracting user's mean rating before

Avg. User Rating = (3 + 1 + 2 + 5 + 4)/5 = 3

Simple User Profile: Example With Ratings

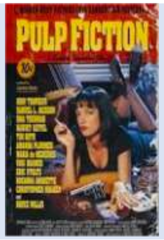
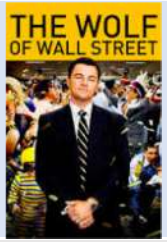
Suppose user *u* has watched (and rated) **5** movies

0		-2		-1		2		1	
1	0	0	1	0	1	1	0	0	1
$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$	$actor_1$	$actor_2$

Normalize ratings by subtracting user's mean rating before

(0+2)/2	1	-2/3	[-2+(-1)+1]/3
	$actor_1$	$actor_2$	

Building Predictions (from Item/User Profiles)

		MOVIES							
									
USERS	 Alice	2		5	4	5	4		4
	 Bob	4	?	?	?	?	3	?	3
	 Carl	5	5	3	4	5	4		5
	…	…	…	…	…	…	…	…	…
	 Zoe		1	3				5	4

Building Predictions (from Item/User Profiles)

How to fill the "?"?

		MOVIES							
USERS	 Alice	2		5	4	5	4		4
	 Bob	4	?	?	?	?	3	?	3
	 Carl	5	5	3	4	5	4		5
	...	...	...	...	...	...	...	...	...
	 Zoe		1	3				5	4

Building Predictions (from Item/User Profiles)

- Given user profile **u** we can **estimate** the missing entries of the utility matrix for *u* (i.e., the ratings *u* would give to unrated items)

Building Predictions (from Item/User Profiles)

- Given user profile **u** we can **estimate** the missing entries of the utility matrix for *u* (i.e., the ratings *u* would give to unrated items)
- For each item unrated by *u*, compute the cosine similarity (or Pearson's correlation) between *u* and the corresponding item profile vectors

Building Predictions (from Item/User Profiles)

- Given user profile **u** we can **estimate** the missing entries of the utility matrix for *u* (i.e., the ratings *u* would give to unrated items)
- For each item unrated by *u*, compute the cosine similarity (or Pearson's correlation) between *u* and the corresponding item profile vectors
- Finally, we pick the top-*k* items with the **highest** similarity score, and we recommend those to *u*

Building Predictions (from Item/User Profiles)

$$A^0 = \emptyset$$

Building Predictions (from Item/User Profiles)

$$A^0 = \emptyset$$
$$A^1 = \text{argmax}_i \{\text{sim}(\mathbf{u}, \mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0\}$$

Building Predictions (from Item/User Profiles)

$$A^0 = \emptyset$$
$$A^1 = \text{argmax}_i \{\text{sim}(\mathbf{u}, \mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0\}$$
$$A^2 = \text{argmax}_i \{\text{sim}(\mathbf{u}, \mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0 - A^1\}$$

Building Predictions (from Item/User Profiles)

$$A^0 = \emptyset$$

$$A^1 = \text{argmax}_i \{\text{sim}(\mathbf{u}, \mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0\}$$

$$A^2 = \text{argmax}_i \{\text{sim}(\mathbf{u}, \mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0 - A^1\}$$

$$\cdots$$

$$A^k = \text{argmax}_i \{\text{sim}(\mathbf{u}, \mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0 - A^1 - \ldots - A^{k-1}\}$$

Building Predictions (from Item/User Profiles)

$$A^0 = \emptyset$$
$$A^1 = \operatorname{argmax}_i\{\operatorname{sim}(\mathbf{u},\mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0\}$$
$$A^2 = \operatorname{argmax}_i\{\operatorname{sim}(\mathbf{u},\mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0 - A^1\}$$
$$\dots$$
$$A^k = \operatorname{argmax}_i\{\operatorname{sim}(\mathbf{u},\mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - A^0 - A^1 - \ldots - A^{k-1}\}$$

$$\boxed{R_{u,k} = \bigcup_{j=1}^{k} A^j = \bigcup_{j=1}^{k} \operatorname{argmax}_i \left\{\operatorname{sim}(\mathbf{u},\mathbf{i}) : i \in \mathcal{I} - \mathcal{I}_u - \{\cup_{l=0}^{j-1} A^l\}\right\}}$$

Content-based Filtering: PROs

- No need for data on other users: only the user and the items rated by her/him are needed

Content-based Filtering: PROs

- No need for data on other users: only the user and the items rated by her/him are needed
- No item cold start problem: when the item is new or unpopular (i.e., no one has rated yet) it can still be recommended to users with highest profile similarity

Content-based Filtering: PROs

- No need for data on other users: only the user and the items rated by her/him are needed
- No item cold start problem: when the item is new or unpopular (i.e., no one has rated yet) it can still be recommended to users with highest profile similarity
- Explainable recommendations using content features that caused an item to be recommended

Content-based Filtering: CONs

- Finding the appropriate item features is hard

Content-based Filtering: CONs

- Finding the appropriate item features is hard
- Overspecialization: never recommends items outside user's profile

Content-based Filtering: CONs

- Finding the appropriate item features is hard
- Overspecialization: never recommends items outside user's profile
- Unable to exploit quality judgments of other users

Content-based Filtering: CONs

- Finding the appropriate item features is hard
- Overspecialization: never recommends items outside user's profile
- Unable to exploit quality judgments of other users
- Cold start problem for new users: If the user hasn't rated any items, then there's no user profile

Content-based Filtering: CONs

- Finding the appropriate item features is hard
- Overspecialization: never recommends items outside user's profile
- Unable to exploit quality judgments of other users
- Cold start problem for new users: If the user hasn't rated any items, then there's no user profile
- May need to create average profiles and gradually improve them overtime

Take-Home Message of Today

- Recommender systems as tools for dealing with information overload

Take-Home Message of Today

- Recommender systems as tools for dealing with information overload
- The main goal of recommender systems is to select items that are likely of interest to users

Take-Home Message of Today

- Recommender systems as tools for dealing with information overload
- The main goal of recommender systems is to select items that are likely of interest to users
- They make use of either explicit (e.g., ratings) or implicit (e.g., clicks) feedback to build a user-item utility matrix

Take-Home Message of Today

- Recommender systems as tools for dealing with information overload
- The main goal of recommender systems is to select items that are likely of interest to users
- They make use of either explicit (e.g., ratings) or implicit (e.g., clicks) feedback to build a user-item utility matrix
- Content-based recommender systems make use of item and user profiles (built in the item space) to come up with top-*k* suggestions