

# CS-4053 Recommender System

Fall 2023

## Lecture 3: Content-based Recommender System

**Course Instructor:** Syed Zain Ul Hassan

National University of Computer and Emerging Sciences, Karachi

*Email: [zain.hassan@nu.edu.pk](mailto:zain.hassan@nu.edu.pk)*



# Content-based Recommendations:

## Preamble

- ❑ The basic idea of a content-based recommender system is to match user with items that are similar to what they rated highly in the past
- ❑ The similarity is not on the basis of rating correlations across users
- ❑ The attributes of the items are used to find similarity

# Content-based Recommendations:

## Preamble

- ❑ These type of recommendation systems can work well in cases where personalized recommendations are needed
  - ❑ Movie recommendations e.g. a user liking a movie of a certain director more
  - ❑ Websites or blogs *e.g.* recommending similar news or article to the user to the one he is querying or interested in

# Content-based Recommendations:

## Preamble

- ❑ What exactly do we mean by **content**?
- ❑ **Content** are explicit attributes or characteristics of the item e.g. for a movie:
  - ❑ **Genre**: Action / Superhero
  - ❑ **Actor**: Robert Downey Jr.
  - ❑ **Year**: 2019
- ❑ It can also be textual content (title, description, table of content, *etc.*)
  - ❑ Several techniques to compute the distance between two textual documents
  - ❑ Can use NLP techniques to extract content features
- ❑ Can be extracted from the signal itself (audio, image)

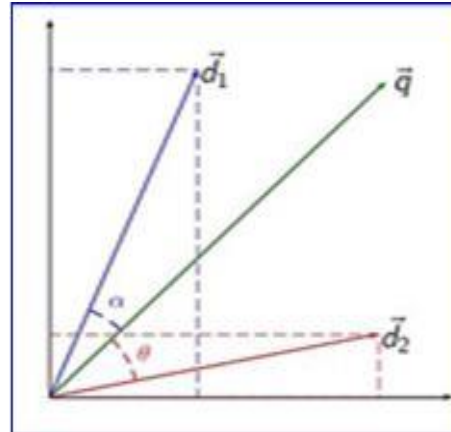
# Content-based Recommender System

- ❑ Formally, a **content-based recommender systems** try to match users to items that are *similar* to what they have liked in the past
- ❑ This *similarity* is not necessarily based on rating correlations across users but on the basis of the attributes of the objects liked by the user

# Vector Space Model

- ❑ In a vector space model, each item is saved as a vector of its attributes (features) in n-dimensional space
- ❑ The angle between two vectors give similarity between the contents of these items
- ❑ Each user is saved as a user profile vector and the similarity is calculated the same way

# Vector Space Model



*User profile and Item feature vectors in 2-D vector space*

# Content-based Filtering: **Approach**

- ❑ Content-based systems are common for recommending text-based products  
e.g. news articles, wiki pages
- ❑ A lazy approach is to recommend an item containing content that is most similar to that present in user's highly rated items
  - ❑ Also called **lazy approach**
- ❑ More novel approaches involve training a classifier (*decision trees, neural networks etc.*) to predict a rating given a set of features as input



# Content-based Filtering: **Approach**

## ❑ **Input**

- ❑ A vector containing attributes for each unrated (unseen) item
- ❑ A user profile vector expressing their taste using already rated items

## ❑ **Output**

- ❑ A weighted matrix depicting user's measure of *liking* of each attribute for all items
- ❑ A recommendation vector containing predicted ratings for unseen items





# Content-based Filtering: Example

- ❑ We have an input of user ratings for a set of movies and their attributes






# Content-based Filtering: Example




- ❑ We first create a user profile based on already rated items (and attributes)

					
	2				
	10				
	8				
Input User Ratings					


X

	Comedy	Adventure	Super Hero	Sci-Fi	
	0	1	1	0	
	1	1	1	1	
	1	0	1	0	
Movies Matrix					

=

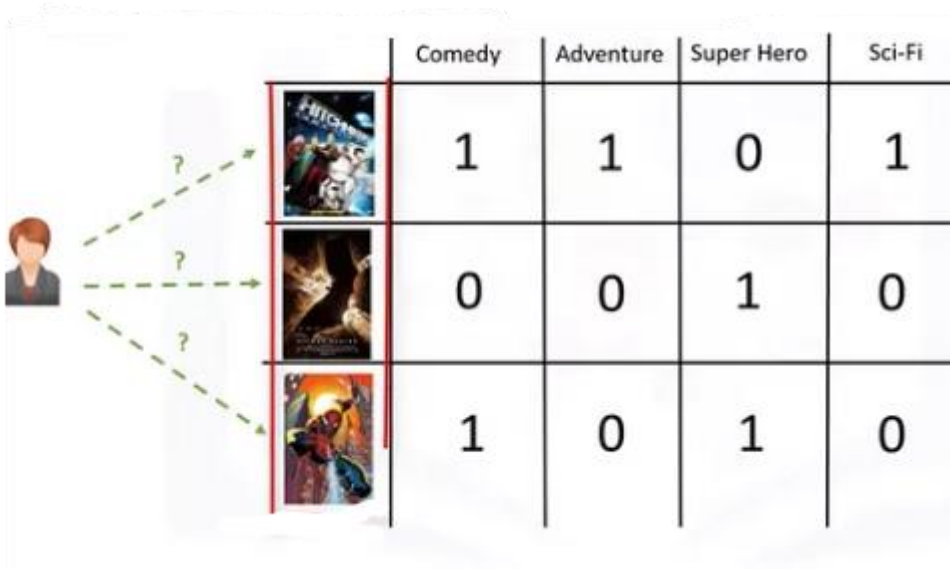
	Comedy	Adventure	Super Hero	Sci-Fi
	0	2	2	0
	10	10	10	10
	8	0	8	0




User Profile

	Comedy	Adventure	Super Hero	Sci-Fi
	0.3	0.2	0.33	0.16
				5

# Content-based Filtering: **Example**

❑ Which one of the yet unrated movies should we be recommending?



	Comedy	Adventure	Super Hero	Sci-Fi
	1	1	0	1
	0	0	1	0
	1	0	1	0

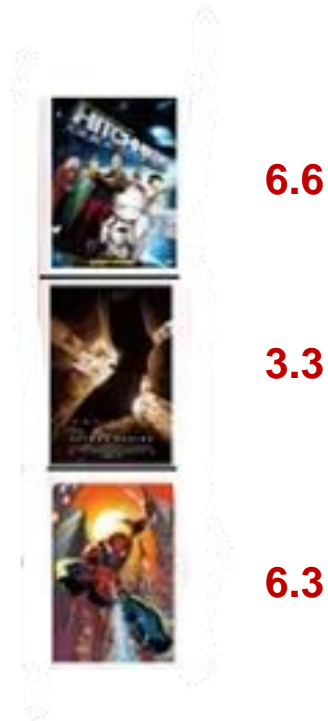
# Content-based Filtering: Example

- ❑ We can recommend the movie that gives us the maximum weighted average



# Content-based Filtering: **Example**

- ❑ Maximum weighted average can also be expressed as raw predicted rating



# Content-based Filtering: **Exercise**

- **Task:** Find Cosine similarity between user profile and movie matrix and compare with previous results.

# Content-based Filtering: **Example**

- ❑ Let us assume that a user clicks and reads a blog on a website
- ❑ The objective is to recommend **two** similar blogs to the user
- ❑ We can calculate the **TF** (term frequency) and **IDF** (inverse document frequency) of the blogs in order to find their relative importance based on how similar they are to the current blog



# Content-based Filtering: **Example**

- ❑ Term Frequency (**TF**) is simply the number of times a term (feature) appears in the document (item)
- ❑ We generally use transformed **TF** in order to de-emphasize the effect of frequency on relevance of the term. Hence:

$$TF_{tk, dj} = \frac{freq(tk)}{max(ti, dj)}$$

- ❑ Inverse Document Frequency (**IDF**) helps us find the overall relevance of a term

$$IDF_{tk} = \log_{10} \frac{N}{n_k}$$

# Content-based Filtering: **Example**

- ❑ Let us assume that **Blog 1** is being read by the user and contains the given three terms with their frequencies

Blog	Business	Invention	Hospital
Blog 1	31	27	0

# Content-based Filtering: **Example**

- ❑ The given table contains retrieved blog along with information for all other blogs (from which two are to be recommended)

Blog	Business	Invention	Hospital
Blog 1	31	27	0
Blog 2	29	61	3
Blog 3	52	99	6
Blog 4	10	29	5
Blog 5	13	41	7

- ❑ Assume that the corpus contains **50000** documents (blogs) on the website and **2500** of them contain the term **Business**, **1200** contain **Invention** and **800** contain **Hospital**

# Content-based Filtering: **Example**

- ❑ The given table contains retrieved blog along with information for all other blogs (from which two are to be recommended)

Blog	Business	Invention	Hospital
Blog 1	31	27	0
Blog 2	29	61	3
Blog 3	52	99	6
Blog 4	10	29	5
Blog 5	13	41	7
Doc. Freq.	2500	1200	800

- ❑  $N = 50000$

# Content-based Filtering: Example

□ We first calculate TF for each attribute of every blog e.g.

$$TF_{Business, Blog2} = \frac{freq(Business, Blog2)}{\max(freq(\text{any other term}, Blog2))} = \frac{29}{61} = 0.47$$

Blog	Business	Invention	Hospital
Blog 1	31	27	0
Blog 2	0.47	61	3
Blog 3	52	99	6
Blog 4	10	29	5
Blog 5	13	41	7
Doc. Freq.	2500	1200	800

# Content-based Filtering: **Example**

□ We calculate TF for all other attributes in the same manner

$$TF_{Business, Blog2} = \frac{freq(Business, Blog2)}{\max(freq_i, Blog2)} = \frac{29}{61} = \mathbf{0.47}$$

Blog	Business	Invention	Hospital
Blog 1	1.14	0.87	0
Blog 2	0.47	2.10	0.04
Blog 3	0.52	1.90	0.06
Blog 4	0.34	2.9	0.17
Blog 5	0.31	3.15	0.17
Doc. Freq.	2500	1200	800

# Content-based Filtering: **Example**

❑ We also need to calculate **IDF** for each attribute e.g.

$$IDF_{Business} = \log_{10} \left( \frac{50000}{2500} \right) = 1.30$$

Blog	Business	Invention	Hospital
Blog 1	1.14	0.87	0
Blog 2	0.47	2.10	0.04
Blog 3	0.52	1.90	0.06
Blog 4	0.34	2.9	0.17
Blog 5	0.31	3.15	0.17
Doc. Freq.	2500	1200	800
IDF	1.30	1.61	1.79

# Content-based Filtering: **Example**

□ Let us calculate the TF-IDF weights for each term in every blog e.g.

$$\textbf{TF-IDF}_{tk, dj} = \text{TF-IDF}_{\text{Business}, \text{Blog2}} = 0.47 * 1.30 = \textbf{0.61}$$

Blog	Business	Invention	Hospital
Blog 1	1.14	0.87	0
Blog 2	<b>0.61</b>	2.10	0.04
Blog 3	0.52	1.90	0.06
Blog 4	0.34	2.9	0.17
Blog 5	0.31	3.15	0.17
Doc. Freq.	2500	1200	800
IDF	1.30	1.61	1.79



# Content-based Filtering: **Example**

□ Let us calculate the TF-IDF weights for each term in every blog e.g.

$$\textbf{TF-IDF}_{tk, dj} = \text{TF-IDF}_{\text{Business}, \text{Blog2}} = 0.47 * 1.30 = \textbf{0.61}$$

Blog	Business	Invention	Hospital
Blog 1	1.48	1.40	0
Blog 2	0.61	3.38	0.07
Blog 3	0.67	3.05	0.10
Blog 4	0.44	4.66	0.30
Blog 5	0.40	5.07	0.30
Doc. Freq.	2500	1200	800
IDF	1.30	1.61	1.79

# Content-based Filtering: **Example**

- Let us first find the length of each vector and then use it calculate the normalized TF-IDF weights e.g.

$$\text{Length}_{\text{Blog1}} = \sqrt{1.48^2 + 1.40^2 + 0^2} = \mathbf{2.03}$$

Blog	Business	Invention	Hospital	Magnitude
Blog 1	1.48	1.40	0	<b>2.03</b>
Blog 2	0.61	3.38	0.07	3.43
Blog 3	0.67	3.05	0.10	3.12
Blog 4	0.44	4.66	0.30	4.69
Blog 5	0.40	5.07	0.30	5.09
Doc. Freq.	2500	1200	800	
IDF	1.30	1.61	1.79	

# Content-based Filtering: **Example**

- Let us first find the length of each vector and then use it calculate the normalized TF-IDF weights e.g.

$$W_{Business, Blog1} = \frac{TF-IDF_{Business, Blog1}}{Length_{Blog1}} = \mathbf{0.73}$$

Blog	Business	Invention	Hospital	Magnitude
Blog 1	<b>0.73</b>	0.68	0	2.03
Blog 2	0.17	0.98	0.02	3.43
Blog 3	0.21	0.97	0.03	3.12
Blog 4	0.09	0.99	0.06	4.69
Blog 5	0.07	0.99	0.05	5.09
Doc. Freq.	2500	1200	800	
IDF	1.30	1.61	1.79	

# Content-based Filtering: **Example**

- ❑ Calculate the Cosine Similarity between the weighted user profile (Blog1) and each weighted item profile (all other blogs) to recommend blogs

Blog	Business	Invention	Hospital	Magnitude
Blog 1	0.73	0.68	0	2.03
Blog 2	0.17	0.98	0.02	3.43
Blog 3	0.21	0.97	0.03	3.12
Blog 4	0.09	0.99	0.06	4.69
Blog 5	0.07	0.99	0.05	5.09
Doc. Freq.	2500	1200	800	
IDF	1.30	1.61	1.79	

# Content-based Filtering: Example

- ❑ Calculate the Cosine Similarity between the weighted user profile (Blog1) and each weighted item profile (all other blogs) to recommend blogs

$$\text{Cosine}_{(\text{Blog1}, \text{Blog2})} = \frac{(0.73 * 0.17) + (0.68 * 0.98) + (0 * 0.02)}{\sqrt{0.73^2 + 0.68^2 + 0^2} \cdot \sqrt{0.17^2 + 0.98^2 + 0.02^2}} = 0.79$$

Blog	Business	Invention	Hospital	Magnitude	Cosine Similarity
Blog 1	0.73	0.68	0	2.03	
Blog 2	0.17	0.98	0.02	3.43	0.79
Blog 3	0.21	0.97	0.03	3.12	
Blog 4	0.09	0.99	0.06	4.69	
Blog 5	0.07	0.99	0.05	5.09	
Doc. Freq.	2500	1200	800		
IDF	1.30	1.61	1.79		

# Content-based Filtering: **Example**

- ❑ Calculate the Cosine Similarity between the weighted user profile (Blog1) and each weighted item profile (all other blogs) to recommend blogs

Blog	Business	Invention	Hospital	Magnitude	Cosine Similarity
Blog 1	0.73	0.68	0	2.03	
Blog 2	0.17	0.98	0.02	3.43	0.79
Blog 3	0.21	0.97	0.03	3.12	0.82
Blog 4	0.09	0.99	0.06	4.69	0.74
Blog 5	0.07	0.99	0.05	5.09	0.73
Doc. Freq.	2500	1200	800		
IDF	1.30	1.61	1.79		

# Content-based Filtering: **Example**

- ❑ Based on Cosine Similarities, **Blog 2** and **Blog 3** will be recommended to the user for reading

Blog	Business	Invention	Hospital	Magnitude	Cosine Similarity
Blog 1	0.73	0.68	0	2.03	
Blog 2	0.17	0.98	0.02	3.43	0.79
Blog 3	0.21	0.97	0.03	3.12	0.82
Blog 4	0.09	0.99	0.06	4.69	0.74
Blog 5	0.07	0.99	0.05	5.09	0.73
Doc. Freq.	2500	1200	800		
IDF	1.30	1.61	1.79		

# Content-based Filtering: **Pros** and **Cons**

## **Pros**

- Scalable in terms of new users
- Highly personalized recommendations
- Partially solves cold start problem *i.e.* solves for new item
- Recommendations are explainable

## **Cons**

- Not easy to hand-engineer meaningful attributes
- Has little to no serendipity
- Suffers from cold start user problem