Lecture 5

Recommender Systems

Personalization, Collaborative Filtering & Content-based recommendation

COMP 474/6741, Winter 2024

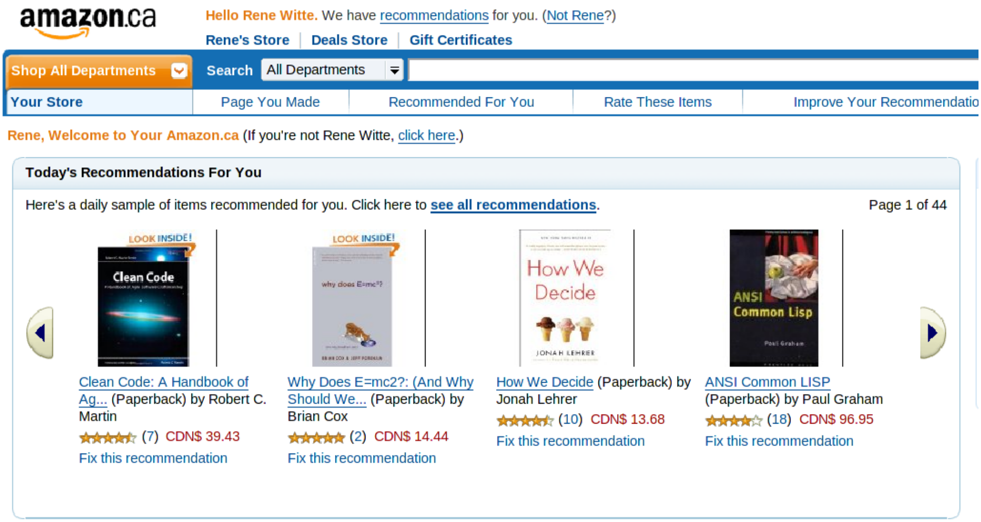
René Witte
Department of Computer Science
and Software Engineering
Concordia University

Outline

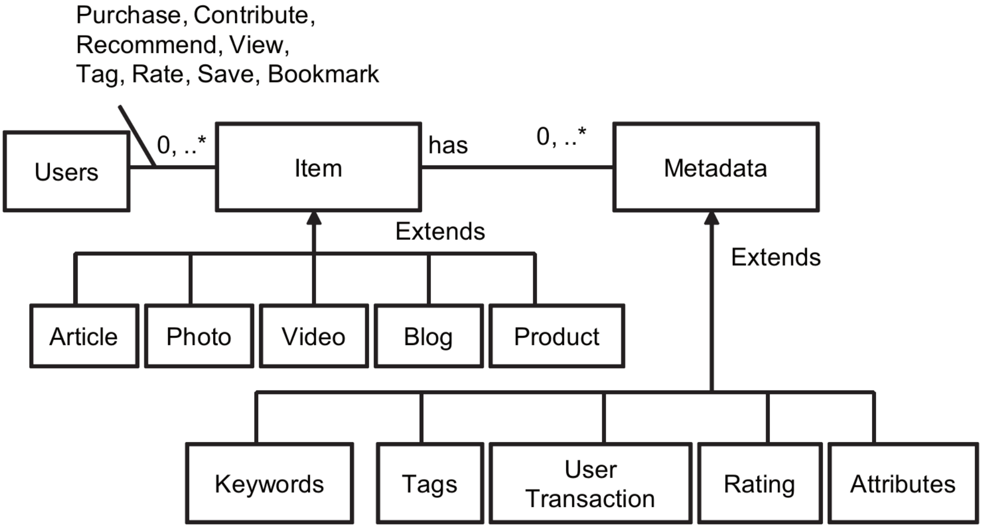
Slides Credit

Includes slides by Christopher D. Manning, Prabhakar Raghavan and Hinrich Schütze [MRS08]

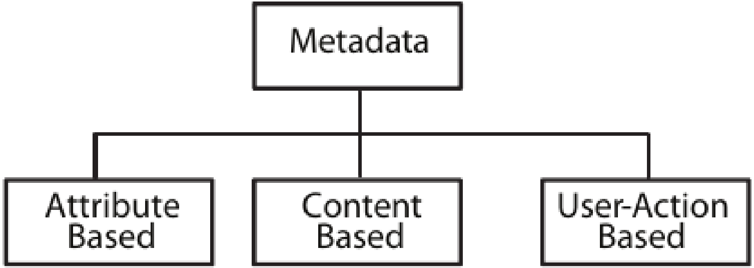
Recommender Systems and Collaborative Filtering



Collecting User Interactions

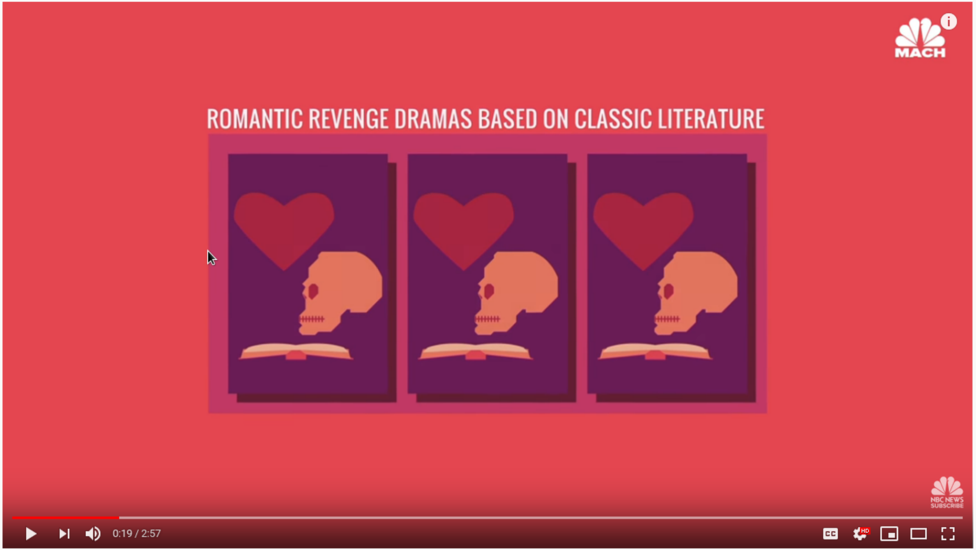


Item Metadata



Netflix Recommendations

René Witte



Why Netflix's Algorithm Is So Binge-Worthy | Mach | NBC News

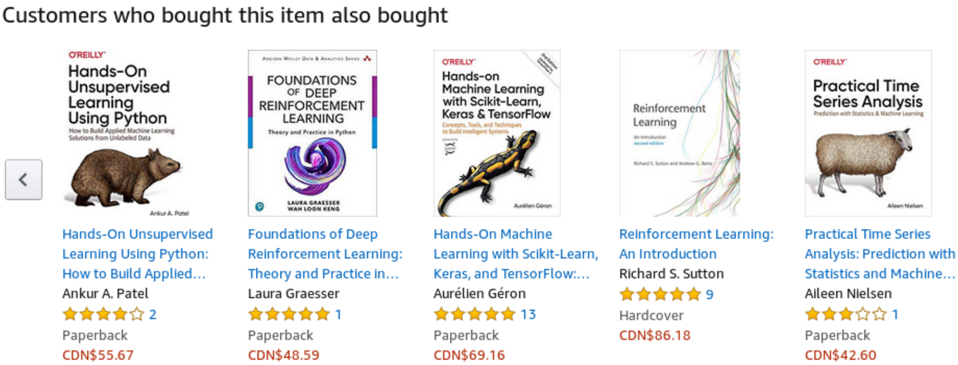
https://www.youtube.com/watch?v=nq2QtatuF7U

Making Recommendations

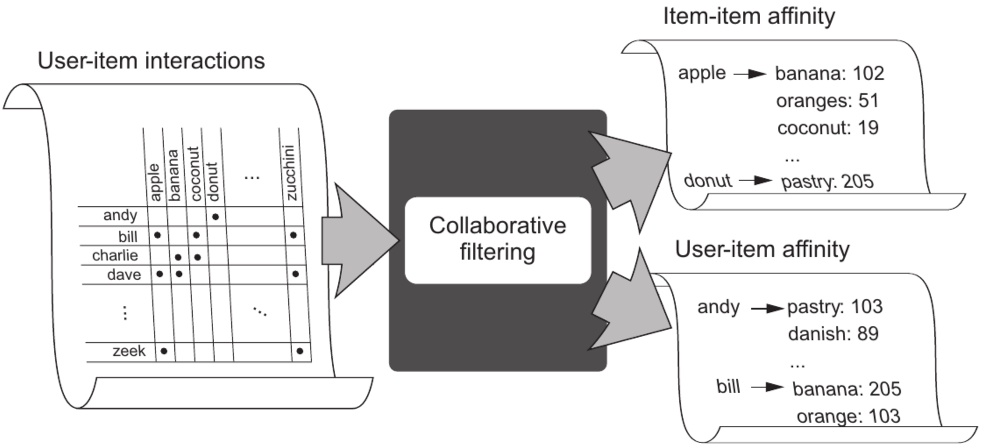
Given Information about a User...

...we want to be able to have a system

- recommending items (books, movies, music, photos, videos, etc.)
- find users interested in a new item
- find similar items, based on interests of *other* users



Collaborative Filtering



René Witte
Concordia University

Data Collection

René Witte

Date	User	Item
2015-01-24 15:01:29	Allison	Tunisia Sadie dress
2015-01-26 05:13:58	Christina	Gordon Monk stiletto
2015-02-18 10:28:37	David	Ravelli aluminum tripod
2015-03-17 14:29:23	Frank	Nikon digital camera
2015-03-26 18:11:01	Christina	Georgette blouse
2015-04-06 21:50:18	David	Canon 24 mm lens
2015-04-15 10:21:44	Frank	Canon 24 mm lens
2015-04-15 21:53:25	Brenda	Tunisia Sadie dress
2015-07-26 08:08:25	Elise	Nikon digital camera

Fun with ~~Flags~~ Vectors

Vectors

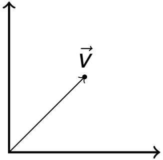
A vector $\vec{v}$ is an element of a vector space.

- For example, $\vec{v} \in \mathbb{R}^n$ with

$$\vec{v} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \in \mathbb{R}^n$$

Visualization

We can visualize vectors, e.g., in 2D:

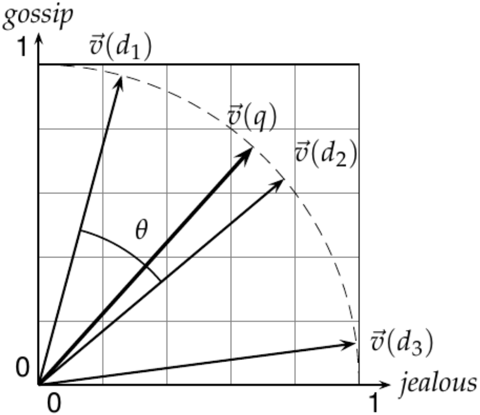


So what?

Vectors of words, users, products, ...

We can represent (users, documents, products) as vectors, e.g., using the count of tags or the weight of words. This is called a vector space model.

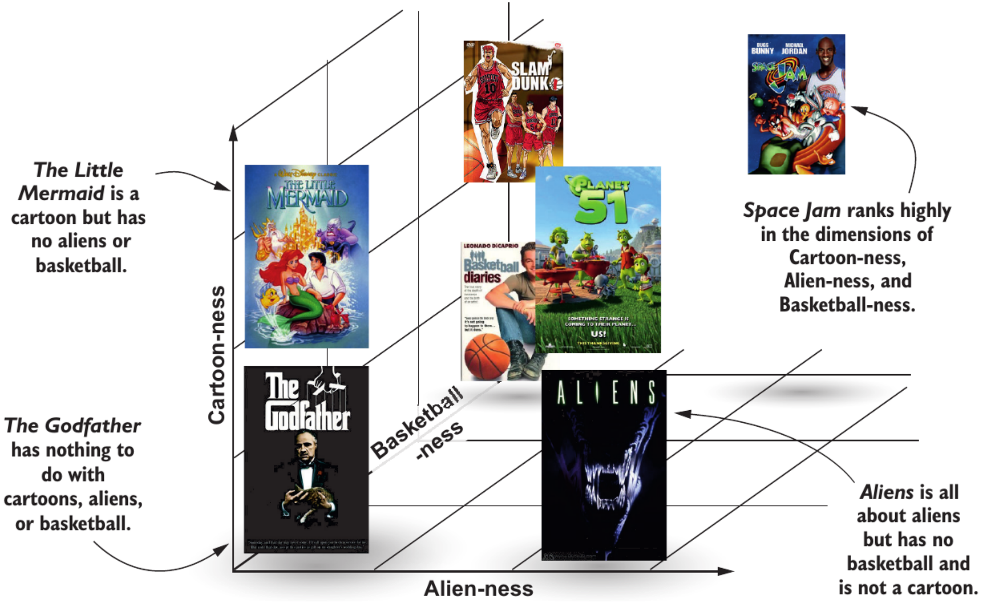
- Vector operations on entities, e.g., to compute their similarity



Movies as Vectors

René Witte

Concordia University



Length normalization

René Witte

How do we compute the length of a vector?

- A vector can be (length-) normalized by dividing each of its components by its length – here we use the L_2 norm: $||x||_2 = \sqrt{\sum_i x_i^2}$
- This maps vectors onto the unit sphere ...
- ...since after normalization: $||x||_2 = \sqrt{\sum_i x_i^2} = 1.0$
- As a result, longer and shorter vectors (more/fewer tags) have weights of the same order of magnitude.

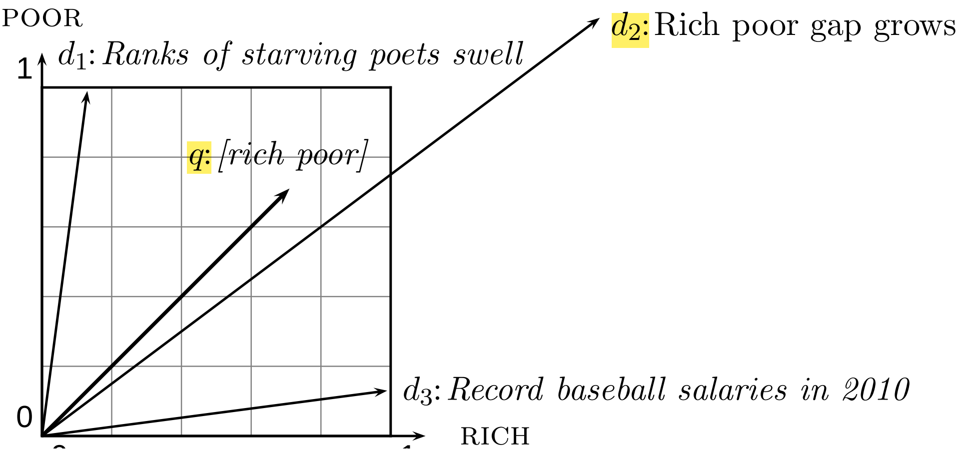
$\rightarrow$ Worksheet #4: Tasks 1, 2

How do we formalize vector space similarity?

Computing the similarity

- First cut: (negative) distance between two points
- (= distance between the end points of the two vectors)
- Euclidean distance?
- Euclidean distance is a bad idea . . .
- . . . because Euclidean distance is large for vectors of different lengths.

Why Euclidian distance is a bad idea



The Euclidean distance of $\vec{q}$ and $\vec{d_2}$ is large although the distribution of terms in the query q and the distribution of terms in the document d_2 are very similar.

From angles to cosines

René Witte

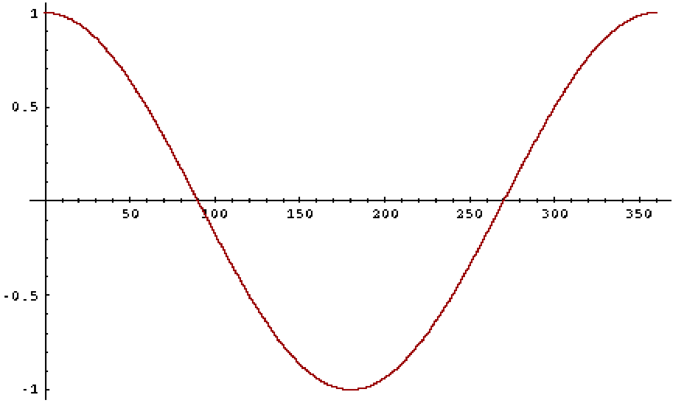
Comparing vectors

The following two notions are equivalent:

- Compare item vectors according to the angle between them, in decreasing order
- Rank item vectors according to cosine$(\text{item}_1, \text{item}_2)$ in increasing order

Cosine is a monotonically decreasing function of the angle for the interval $[0^\circ, 180^\circ]$

Cosine



Cosine for normalized vectors

René Witte

Computing similarity

For normalized vectors, the cosine is equivalent to the dot product or scalar product:

$$\cos(\vec{q}, \vec{d}) = \vec{q} \cdot \vec{d} = \sum_i q_i \cdot d_i$$

(if both $\vec{q}$ and $\vec{d}$ are length-normalized)

$\rightarrow$ Worksheet #4: Task 3

Item Recommendation

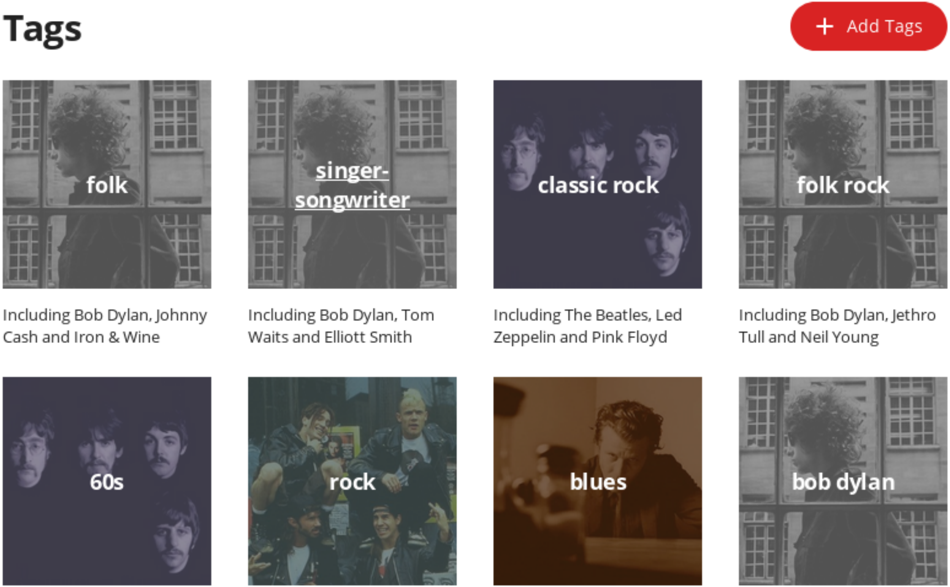
Simple Tag-based Recommendation

Collaborative tagging gives rise to simple recommender approaches:

- show other items (products, photos, videos, music) that were tagged similar by other users
- exploited in many e-commerce/social networking web sites

Finding related content

When multiple users tag the same resource, content can be discovered based on the most frequent tags (example: Last.fm).



René Witte
Concordia University

Recommendations

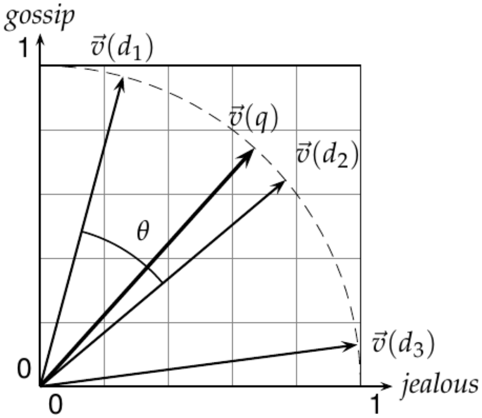
Recommendations based on tags

We can now exploit tags for a number of use cases:

- Recommend items related to other items
- Recommend items based on user's interest
- Find users interested in a new item

General Approach

- Represent users/items as (normalized) term vectors
- Compute cosine similarity between vectors; i.e., the angle between them (for normalized vectors, this is simply their dot product)



René Witte

Concordia University

Items related to other items

Simple point-to-point recommendation engine

- Create item vectors using raw count
- Normalize vectors
- Compute cosine similarity

Result is a similarity matrix

Items of interest to a user

Personalization

- Item-to-item is the same for all users
- How can we recommend items for a particular user?

Solution: build user-specific similarity matrix

- Computation of vectors, normalization as before
- This time, we calculate the cosine similarity between a user vector and an article vector

→ Worksheet #4: Tasks 4, 5

Finding relevant users for an item

René Witte

Recommending items to users

- New item comes in (blog post, photo, article, product, . . .)
- Which users would be interested in it?

Similar to before, compute similarity matrix between metadata of new item and metadata of users.

Cold-Start Problem

General issue in recommender system deployment

- New user ⇒ no user profile for recommendations
- New item ⇒ no user interactions for this item

No general solution...

Some strategies:

- Ask user for preferences during sign-up
- Recommend top-*n* items (e.g., currently most popular movies/songs/products)

Semantic Vocabularies for User Modeling

Semantic User Profiles

Idea: Use vocabularies instead of keywords in the vector representation of a user profile

Motivation

- Semantic recommendations (remember the “tree” example from Lecture #01)
- Open knowledge bases:
  - interoperable between applications
  - controlled by users, not corporations

Vocabularies

Generic user modeling vocabularies

FOAF

- The most popular generic user model offering descriptions for basic user information
- No comprehensive classes for describing preferences or interests

GUMO

- A generic user model that offers several classes for users' characteristics
- Basic user dimensions like Emotional States, Characteristics and Personality

IntelLEO

- Several ontologies strongly focused on personalization
- Enables describing user and team modelling, preferences, tasks and interests

The $1m Netflix Prize Competition (2009)

NETFLIX

Netflix Prize

COMPLETED

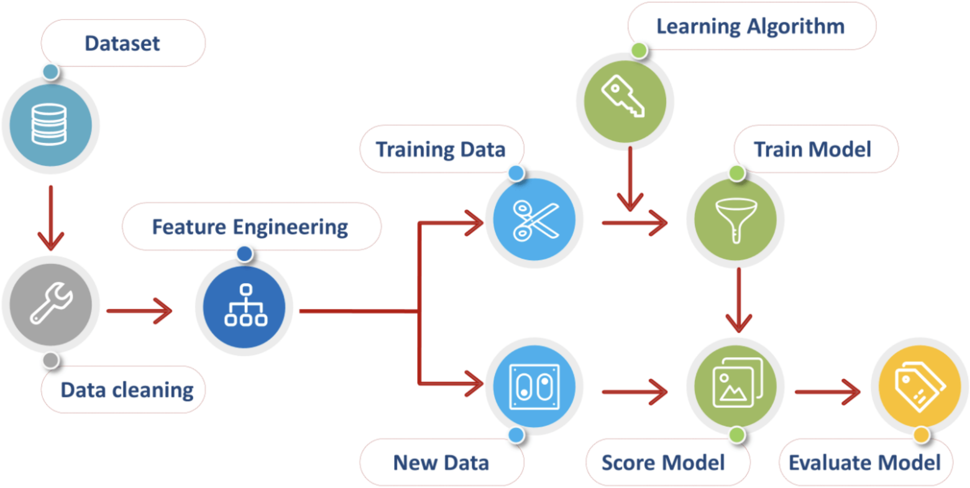
Home | Rules | Leaderboard | Update

Leaderboard

Showing Test Score. Click here to show quiz score

Rank	Team Name	Best Test Score	% Improvement	Best Submit Time
Grand Prize - RMSE = 0.8567 - Winning Team: BellKor's Pragmatic Chaos				
1	BellKor's Pragmatic Chaos	0.8567	10.06	2009-07-26 18:18:28
2	The Ensemble	0.8567	10.06	2009-07-26 18:38:22
3	Grand Prize Team	0.8582	9.90	2009-07-10 21:24:40
4	Opera Solutions and Vandelay United	0.8588	9.84	2009-07-10 01:12:31
5	Vandelay Industries !	0.8591	9.81	2009-07-10 00:32:20
6	PragmaticTheory	0.8594	9.77	2009-06-24 12:06:56
7	BellKor in BigChaos	0.8601	9.70	2009-05-13 08:14:09
8	Dace_	0.8612	9.59	2009-07-24 17:18:43
9	Feeds2	0.8622	9.48	2009-07-12 13:11:51
10	BigChaos	0.8623	9.47	2009-04-07 12:33:59
11	Opera Solutions	0.8623	9.47	2009-07-24 00:34:07

General machine learning process



Performance Evaluation

Measuring performance

- Is our fancy model better than giving out random recommendations?
- We need metrics to evaluate and compare the performance of different approaches against a ground truth (a.k.a. gold standard)

Precision and Recall

The precision provides a measure of the quality of the generated recommendations:

$$\text{precision} = \frac{\text{\#correct system recommendations}}{\text{\#all system recommendations}}$$

The recall indicates how many relevant recommendations were found by a system:

$$\text{recall} = \frac{\text{\#correct system recommendations}}{\text{\#all correct recommendations}}$$

Generally, there is a trade-off between precision and recall.

→ Worksheet #4: Task 6

Precision@k

Precision at *cutoff* k

- Return a ranked list of recommendations (e.g., based on cosine similarity)
- Evaluate only top-k recommendations (e.g., top-10)

$$\text{precision@k} = \frac{1}{k} \cdot \sum_{c=1}^{k} \text{rel}(c),$$

where rel(c) tells us if item at rank c was relevant (1) or not (0).

Intuitively…

The percentage of correct recommendations in the top-k.

Wait, what happened to *Recall*?

Well…in this application scenario, we don't really care (there are millions of potentially relevant items on Amazon or movies on Netflix)

→ Worksheet #4: Task 7

Average Precision

René Witte

Average Precision at N

If we recommend N items to a user, where there are at most m relevant items in $1 \ldots N$,

$$\mathrm{AP@N} = \frac{1}{m} \sum_{k=1}^{N} \mathrm{precision@k} \cdot \mathrm{rel}(k)$$

again, rel(c) is 1 if the recommendation at rank c is relevant, 0 otherwise

Note

AP “rewards” (gives a higher score to) higher-ranked, correct recommendations

→ Worksheet #4: Task 8

Mean Average Precision (MAP)

MAP

- So far, everything was calculated for one user $u \in U$
- But we want to know how well the system works across all users
- Hence, average the AP for all users:

$$\text{MAP@N} = \frac{1}{|U|} \sum_{u=1}^{|U|} \text{AP@N}(u)$$

But wait, there's more. . .

- Accuracy, Sensitivity, F-measure, . . .
- Non-binary ranked results (i.e., not just correct or wrong, but a *Likert*-scale): Compute the *discounted cumulative gain* (DCG),

$$\text{DCG}_u = rel_1 + \sum_{c=2}^{|C|} \frac{rel_c}{\log_2 c}$$

Content-based Recommendations

Motivation

- So far, we build our model using vectors of concepts (e.g., tags, movie categories, etc.)
- What if we want to create recommendations based on the content
  - Movie description/summary
  - Blog post
  - News article
  - Research publication
  - . . .

Approach

Same idea, but now we have to build vectors out of whole documents

- Basic idea of information retrieval (IR)
- Used in Internet search engines

Binary incidence matrix

René Witte

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

[from *Introduction to Information Retrieval*]

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
...							

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Content Models

René Witte

Bag of words model

- We do not consider the order of words in a document.
- *John is quicker than Mary* and *Mary is quicker than John* are represented the same way.
- This is called a bag of words model.

Term frequency tf

The term frequency $\mathrm{tf}_{t,d}$ of term t in document d is defined as the number of times that t occurs in d.

Frequency in document vs. frequency in collection

- In addition, to term frequency (the frequency of the term in the document) ...
- ... we also want to use the frequency of the term in the collection for weighting and ranking.
- Rare terms are more informative than frequent terms.
  - Consider a term in the query that is rare in the collection (e.g., ARACHNOCENTRIC).
  - A document containing this term is very likely to be relevant.
  - $\rightarrow$ We want high weights for rare terms like ARACHNOCENTRIC.

Desired weight for frequent terms

René Witte

Weighting scheme

- Frequent terms are less informative than rare terms.
- Consider a term in the query that is frequent in the collection (e.g., GOOD, INCREASE, LINE).
- A document containing this term is more likely to be relevant than a document that doesn't ...
- ... but words like GOOD, INCREASE and LINE are not sure indicators of relevance.
- → For frequent terms like GOOD, INCREASE, and LINE, we want positive weights ...
- ... but lower weights than for rare terms.

Document Frequency

Document Frequency (df)

- We want high weights for rare terms like ARACHNOCENTRIC.
- We want low (positive) weights for frequent words like GOOD, INCREASE, and LINE.
- We will use document frequency to factor this into computing the matching score.
- The document frequency is the number of documents in the collection that the term occurs in.

René Witte

Concordia University

inverse document frequency (idf)

- df_t is the document frequency, the number of documents that t occurs in.
- df_t is an inverse measure of the informativeness of term t.
- We define the idf weight of a term t as follows:

$$\mathrm{idf}_t = \log_{10} \frac{N}{\mathrm{df}_t}$$

  (N is the number of documents in the collection.)
- idf_t is a measure of the informativeness of the term.
- $[\log N/\mathrm{df}_t]$ instead of $[N/\mathrm{df}_t]$ to “dampen” the effect of idf
- Note that we use the log transformation for both term frequency and document frequency.

Examples for idf

Calculation

Compute idf_t using the formula: $\text{idf}_t = \log_{10} \frac{1{,}000{,}000}{\text{df}_t}$ (example size $N = 1{,}000{,}000$)

term	df_t	idf_t
calpurnia	1	6
animal	100	4
sunday	1000	3
fly	10,000	2
under	100,000	1
the	1,000,000	0

Effect of idf on ranking

- idf affects the ranking of documents for queries with at least two terms.
- For example, in the query “arachnocentric line”, idf weighting increases the relative weight of ARACHNOCENTRIC and decreases the relative weight of LINE.
- idf has little effect on ranking for one-term queries.

René Witte

Computing tf-idf

The tf-idf weight of a term is the product of its tf weight and its idf weight:

$$w_{t,d} = (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}$$

- Set to 0 if $\mathrm{tf}_{t,d} = 0$
- Best known weighting scheme in information retrieval
- Note: the "-" in tf-idf is a hyphen, not a minus sign!
- Alternative names: tf.idf, tf x idf

Summary: tf-idf

René Witte

Computation

Assign a tf-idf weight for each term t in each document d:

$$w_{t,d} = \begin{cases} (1 + \log \mathrm{tf}_{t,d}) \cdot \log \frac{N}{\mathrm{df}_t}, & \text{if } \mathrm{tf}_{t,d} > 0 \\ 0, & \text{otherwise} \end{cases}$$

Effect

The tf-idf weight . . .

- . . . increases with the number of occurrences within a document (due to the term frequency)
- . . . increases with the rarity of the term in the collection (due to the inverse document frequency)

$\longrightarrow$ Worksheet #4: Task 9

Binary incidence matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	1	1	0	0	0	1	
BRUTUS	1	1	0	1	0	0	
CAESAR	1	1	0	1	1	1	
CALPURNIA	0	1	0	0	0	0	
CLEOPATRA	1	0	0	0	0	0	
MERCY	1	0	1	1	1	1	
WORSER	1	0	1	1	1	0	
...							

Each document is represented as a binary vector $\in \{0, 1\}^{|V|}$.

[from *Introduction to Information Retrieval*]

Count matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	157	73	0	0	0	1	
BRUTUS	4	157	0	2	0	0	
CAESAR	232	227	0	2	1	0	
CALPURNIA	0	10	0	0	0	0	
CLEOPATRA	57	0	0	0	0	0	
MERCY	2	0	3	8	5	8	
WORSER	2	0	1	1	1	5	
...							

Each document is now represented as a count vector $\in \mathbb{N}^{|V|}$.

Binary $\rightarrow$ count $\rightarrow$ weight matrix

	Anthony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
ANTHONY	5.25	3.18	0.0	0.0	0.0	0.35	
BRUTUS	1.21	6.10	0.0	1.0	0.0	0.0	
CAESAR	8.59	2.54	0.0	1.51	0.25	0.0	
CALPURNIA	0.0	1.54	0.0	0.0	0.0	0.0	
CLEOPATRA	2.85	0.0	0.0	0.0	0.0	0.0	
MERCY	1.51	0.0	1.90	0.12	5.25	0.88	
WORSER	1.37	0.0	0.11	4.15	0.25	1.95	
...							

Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$.

Vector Space Model

Documents as vectors

Each document is now represented as a real-valued vector of tf-idf weights $\in \mathbb{R}^{|V|}$

- So we have a $|V|$-dimensional real-valued vector space.
- Terms are axes of the space.
- Documents are points or vectors in this space.
- Very high-dimensional: tens of millions of dimensions when you apply this to web search engines
- Each vector is very sparse – most entries are zero.

Cosine similarity between query and document

René Witte

Goal: Query a dataset *d* to find similar items

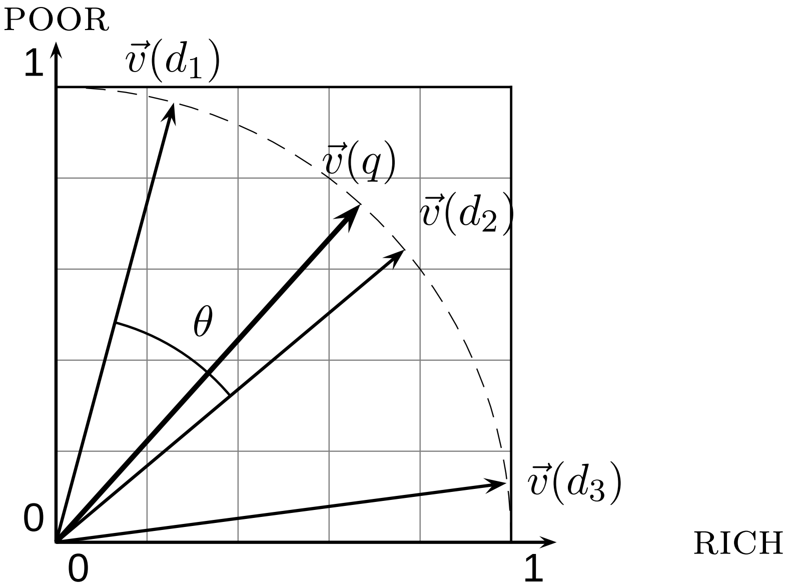
Similar products, photos, movies, songs, ... to a given item *q*

Solution: cosine similarity

$$\cos(\vec{q},\vec{d}) = \mathrm{SIM}(\vec{q},\vec{d}) = \frac{\vec{q}\cdot\vec{d}}{|\vec{q}||\vec{d}|} = \frac{\sum_{i=1}^{|V|} q_i d_i}{\sqrt{\sum_{i=1}^{|V|} q_i^2}\sqrt{\sum_{i=1}^{|V|} d_i^2}}$$

- q_i is the tf-idf weight of term i in the query.
- d_i is the tf-idf weight of term i in the document.
- $|\vec{q}|$ and $|\vec{d}|$ are the lengths of $\vec{q}$ and $\vec{d}$.
- This is the cosine similarity of $\vec{q}$ and $\vec{d}$ ......or, equivalently, the cosine of the angle between $\vec{q}$ and $\vec{d}$.

Cosine similarity illustrated



Basic Recommender Engine using Vector Space Model

Approach

- Represent all documents (movie descriptions, blog posts, research articles, ...) as a weighted tf-idf vector
- Compute the cosine similarity between the target vector and each document vector
- Rank documents with respect to the target
- Return the top k (e.g., $k = 10$) to the user

Summary

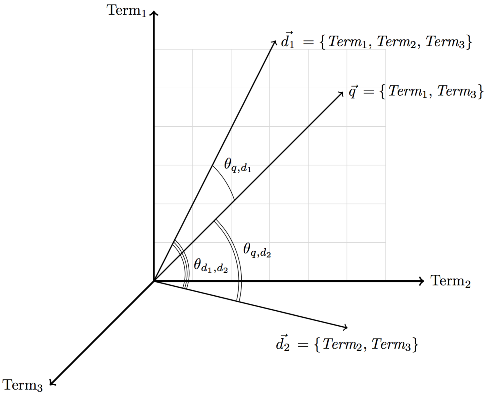
Vector Space Model

- A mathematical model to portray an *n*-dimensional space
- Entities are described by vectors with *n* coordinates in a real space $\mathbb{R}^n$
- Given two vectors, we can compute a similarity coefficient between them
- Cosine of the angle between two vectors reflects their degree of similarity

$$\text{tf} = 1 + \log(\text{tf}_{t,d}) \tag{1}$$

$$\text{idf} = \log \frac{N}{\text{df}_t} \tag{2}$$

$$cos(\vec{q}\,, \vec{d}\,) = \frac{\sum_{i=1}^{|v|} q_i \cdot d_i}{\sqrt{\sum_{i=1}^{|v|} {q_i}^2} \cdot \sqrt{\sum_{i=1}^{|v|} {d_i}^2}} \tag{3}$$



Lucene™

Search with Apache Solr... @ select provider

Core (Java) | Solr | PyLucene

Ultra-fast Search Library and Server

Lucene Solr

Apache Lucene and Solr set the standard for search and indexing performance

Welcome to Apache Lucene

DOWNLOAD

Apache Lucene 8.4.1

DOWNLOAD

The Apache Lucene™ project develops open-source search software, including:

- Lucene Core, our flagship sub-project, provides Java-based indexing and search technology, as well as spellchecking, hit highlighting and advanced analysis/tokenization capabilities.

Outline

1. Introduction
2. Collaborative Filtering
3. Content-based Recommendations
4. **Notes and Further Reading**

Reading Material

Required

- [Ala09, Chapters 2, 3] (Recommendations)
- [MRS08, Chapter 8] (Evaluation)

Supplemental

- [MRS08, Chapter 6] (Vector Space Model, tf-idf)

References

[Ala09] Satnam Alag.
Collective Intelligence in Action.
Manning, 2009.
https://concordiauniversity.on.worldcat.org/oclc/314121652.

[MRS08] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze.
Introduction to Information Retrieval.
Cambridge University Press, 2008.
http://informationretrieval.org.

[TB16] Doug Turnbull and John Berryman.
Relevant Search.
Manning, 2016.
https://concordiauniversity.on.worldcat.org/oclc/954339855.