Recommender Systems

By: Sarah Nooravi

People who liked this meetup, also liked...

Ticketmaster is HIRING

Check out: https://jobs.ticketmaster.com/

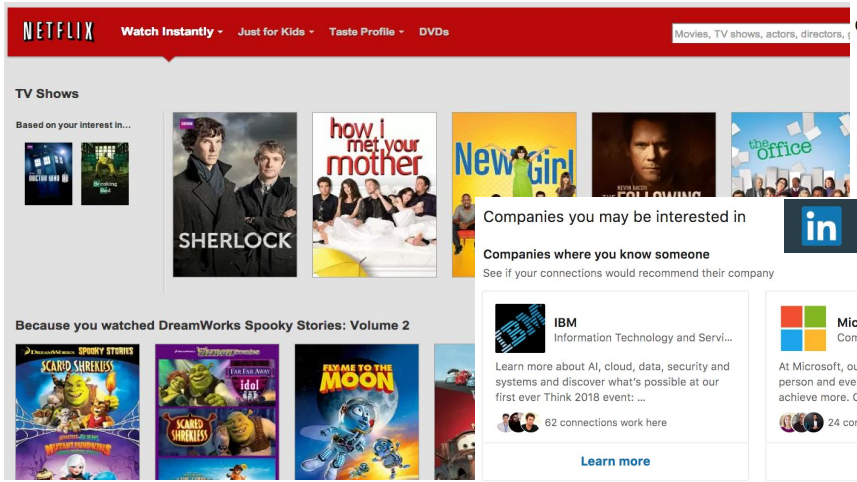
To get the slides and learn more..

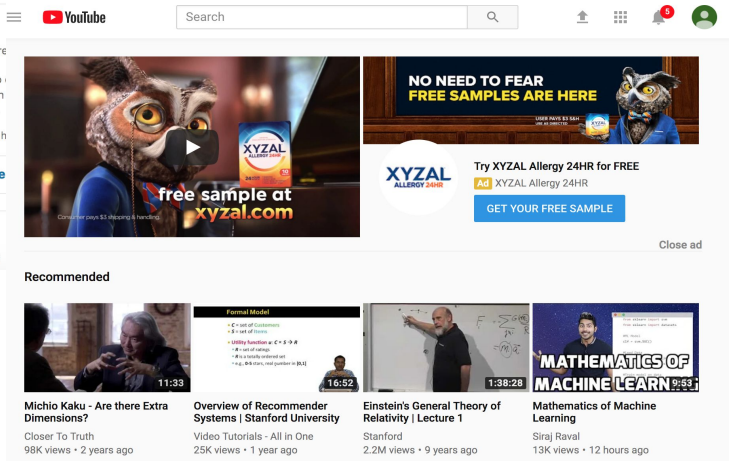
Linkedin: https://www.linkedin.com/in/snooravi/

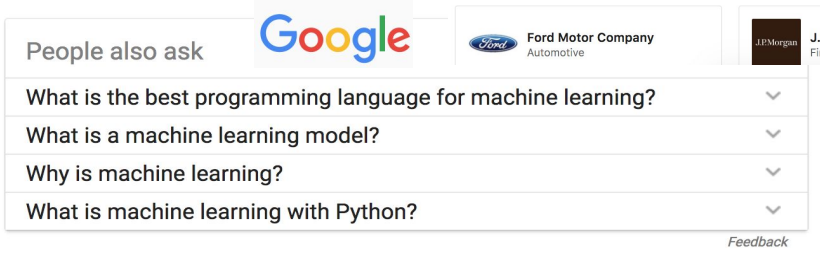
Github: https://github.com/snooravi/meetups

How many of you have ever felt like this?

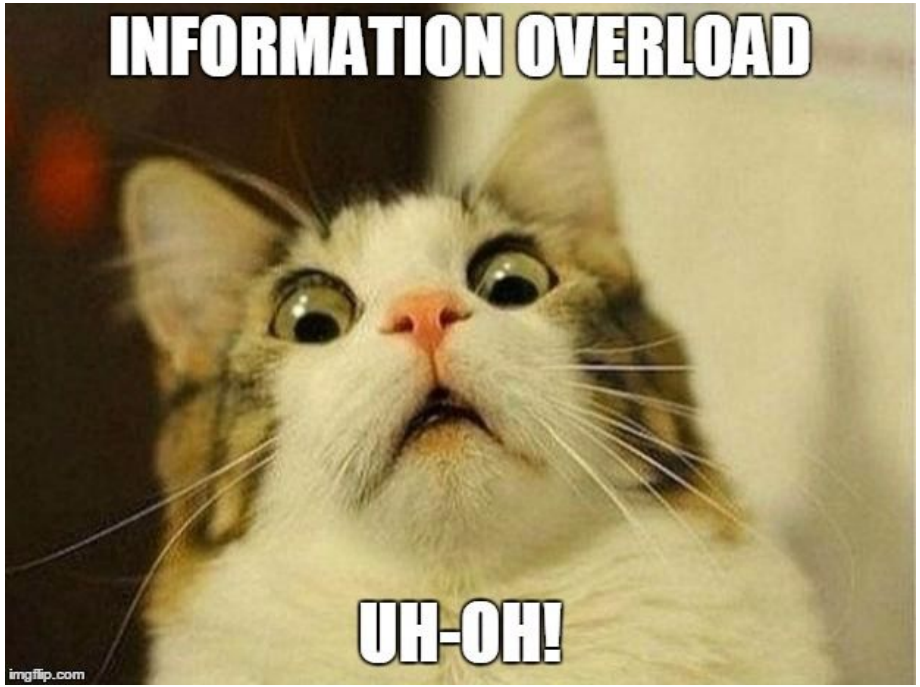
Recommendations are everywhere…

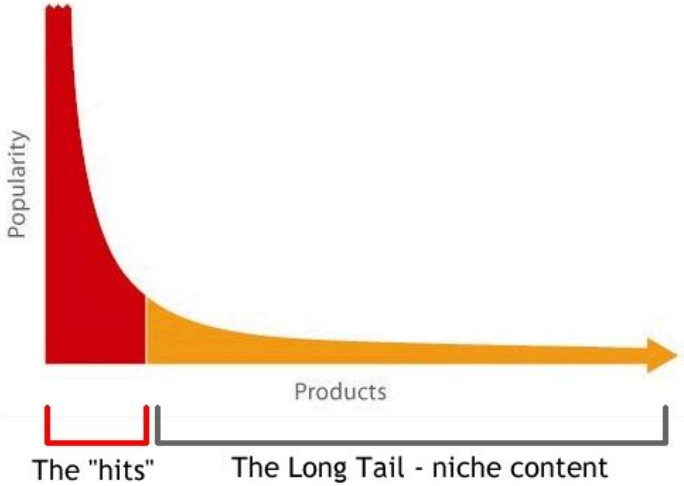






Why it’s necessary





Near zero cost dissemination

Types of Recommendations

Editorial and hand curated

- List of favorites

Simple aggregates

- Top 10, Most Popular, etc.

Tailored to users

- Amazon, Netflix, Pandora..

Reference: https://www.youtube.com/watch?v=1JRrCEgiyHM

Two Ways to Approach This Problem...

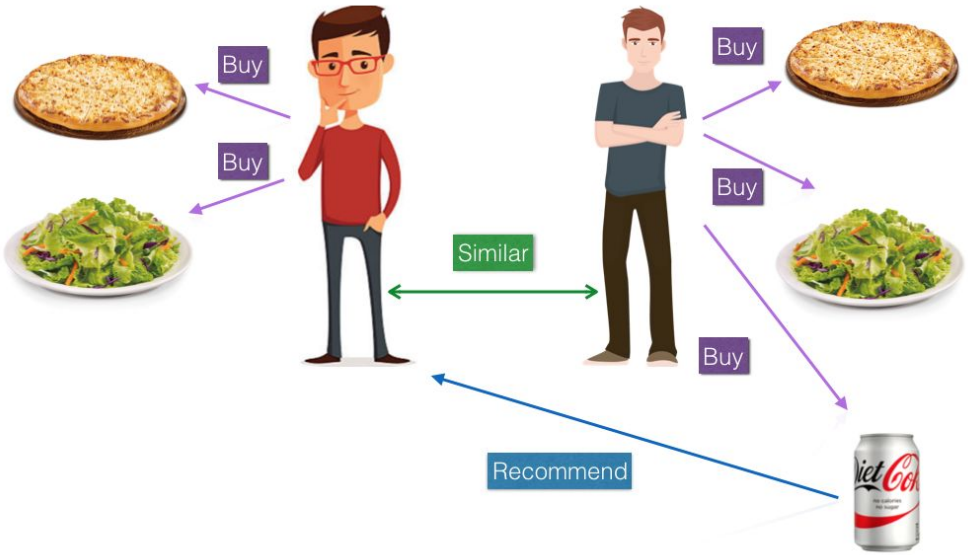
1. Collaborative Filtering
2. Content-Based Filtering

Collaborative Filtering

THOUGHTS?

Collaborative Filtering

Idea: Recommend items to customer x rated highly by **similar customers** to customer x.



Steps: Collaborative Filtering

Step 1: *Find the neighborhood of users that "similar" (i.e. share similar likes and dislikes).*

Step 2: *Make a prediction*

Step 3: *Make a business decision*

Example: Let’s go to the movies!

Example: Let’s go to the movies!

Example: Let's go to the movies!

	Greatest Showman	Jumanji	Love, Simon	Black Panther	Coco	The Circle	The Big Sick	Similarity
John		5	1	5			3	**?**
Eric	5	3	2			4		**?**
Matt		4		1	3	1		**?**
Dylan	5	2	5		5			**?**

Example: Let's go to the movies!

	Greatest Showman	Jumanji	Love, Simon	Black Panther	Coco	The Circle	The Big Sick	Similarity
John		5	1	5			3	**?**
Eric	5	3	2			4		**?**
Matt		4		1	3	1		**?**
Dylan	5	2	5		5			**?**
David		???	2	4		3	4	**?**

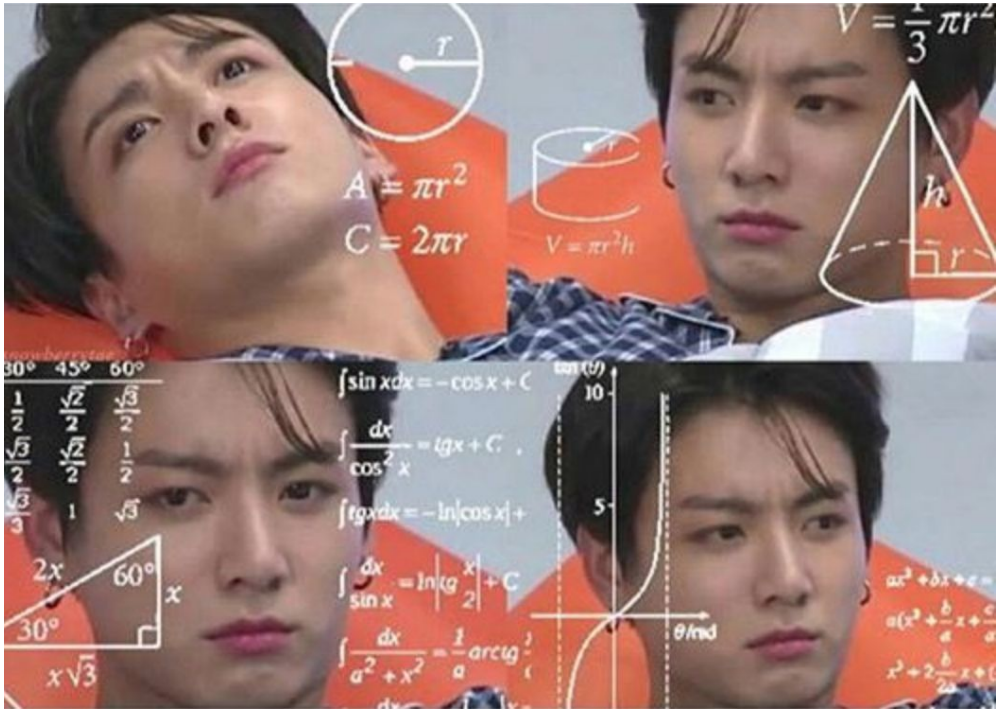
Step 1: Find Similar Users

- **Step 1a**: Define measure of similarity.
- **Step 1b:** Calculate similarity measure between user and all other users.
- **Step 1c**: Rank and choose top N.

Step 1a: Define Measure of Similarity

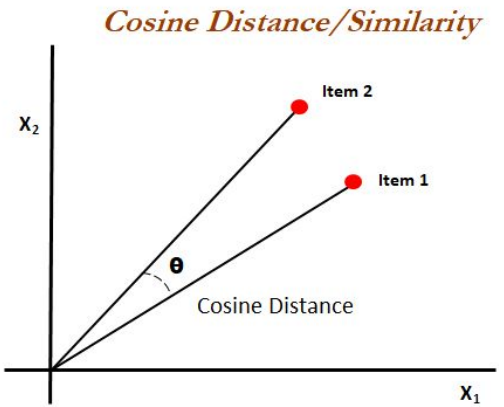
THOUGHTS?

Step 1a: Define Measure of Similarity



Step 1a: Define Measure of Similarity

Cosine Similarity	$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2}\sqrt{\sum_{i=1}^{n} B_i^2}}$
Pearson Correlation	$r = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \bar{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \bar{y})^2}}$



Step 1b: Calculate Similarity

	Greatest Showman	Jumanji	Love, Simon	Black Panther	Coco	The Circle	The Big Sick	Similarity
John		5	1	5			3	**0.70**
Eric	5	3	2			4		**0.47**
Matt		4		1	3	1		-0.14
Dylan	5	2	5		5			-0.21
David		???	2	4		3	4	1

Step 1c: Rank and Chose Top N (N=2)

	Greatest Showman	Jumanji	Love, Simon	Black Panther	Coco	The Circle	The Big Sick	Similarity
John		5	1	5			3	**0.70**
Eric	5	3	2			4		**0.47**
David		???	2	4		3	4	1

Step 2: Make a Prediction

How can we use the neighborhood of similar users to make a prediction?

	Greatest Showman	Jumanji	Love, Simon	Black Panther	Coco	The Circle	The Big Sick	Similarity
John		5	1	5			3	**0.70**
Eric	5	3	2			4		**0.47**
David		???	2	4		3	4	1

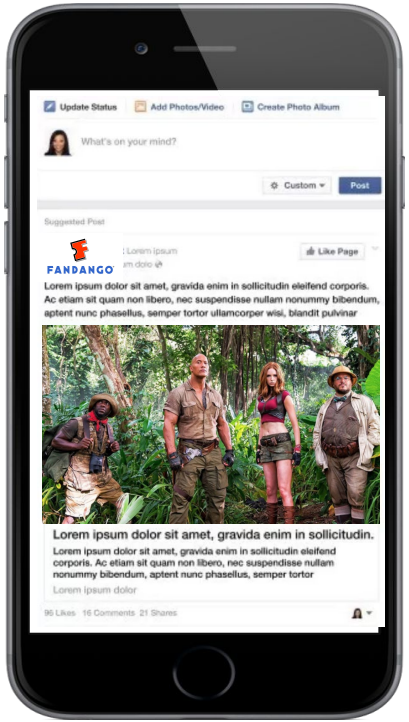
Step 2: Make a Prediction

Take an average or a weighted average:

$$average = \frac{5+3}{2} = \mathbf{4}$$

$$weighted\ average = \frac{0.7 * 5 + 0.47 * 3}{0.7 + 0.47} = \mathbf{4.19}$$

Step 3: Make a Business Decision



Advantages

1. Does not require context analysis and extraction
2. More diversified recommendations

Disadvantages

1. Bias towards popular products
2. Need information on other users
3. Work best when there is a large user space
4. Cold-start for new item

Content-Based Filtering

THOUGHTS?

Content-Based Filtering

Idea: Recommend items to customer **x** similar to **items** rated highly by customer **x**

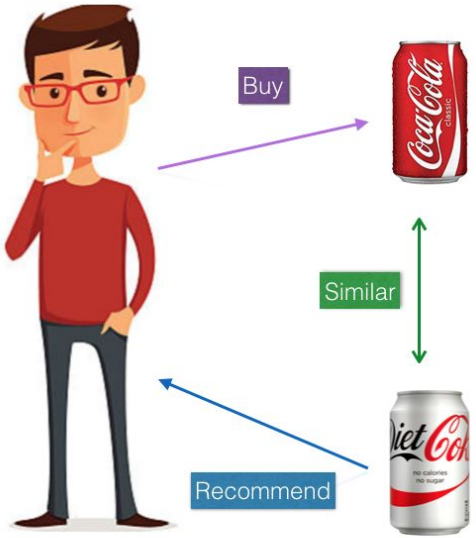
Steps: Content-Based Filtering

Step 1: Define Item Profiles

Step 2: Define User Profiles

Step 3: Calculate Similarity

Step 4: Rank and Recommend



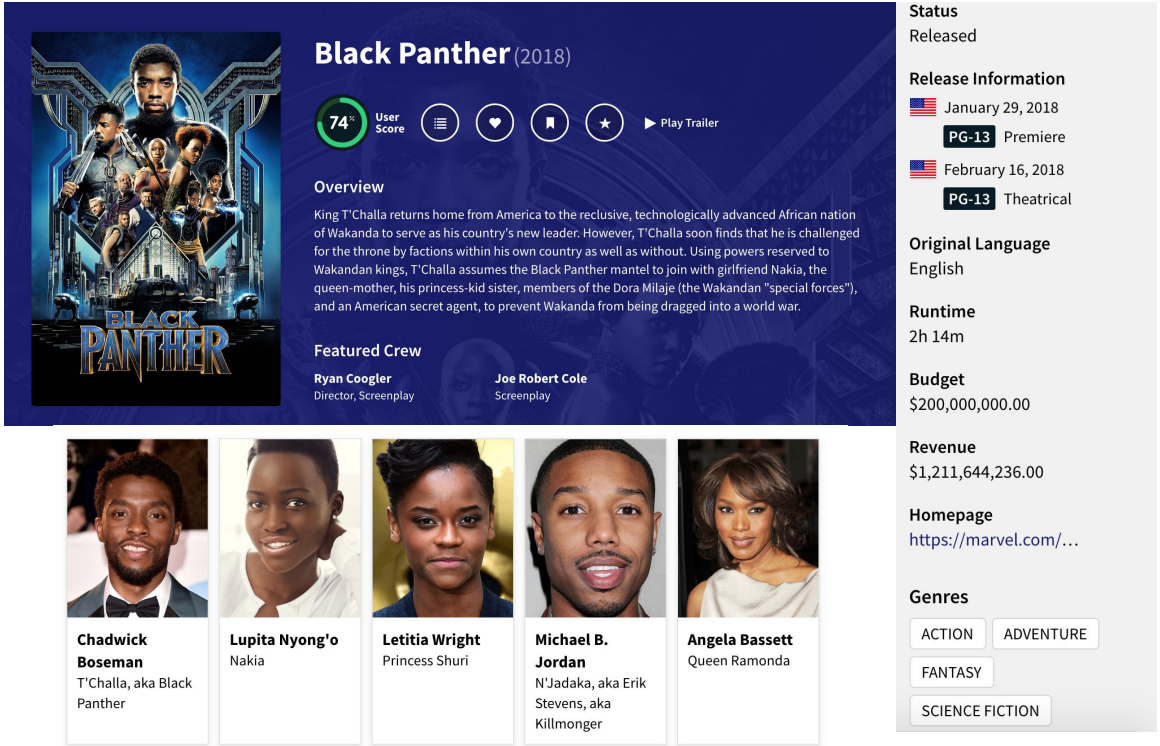
Step 1: Define Item Profiles

A profile is just a set of features. For movies:

- Actors
- Director
- Genre
- Overview
- Etc.

For articles:

- Topics or words in the articles

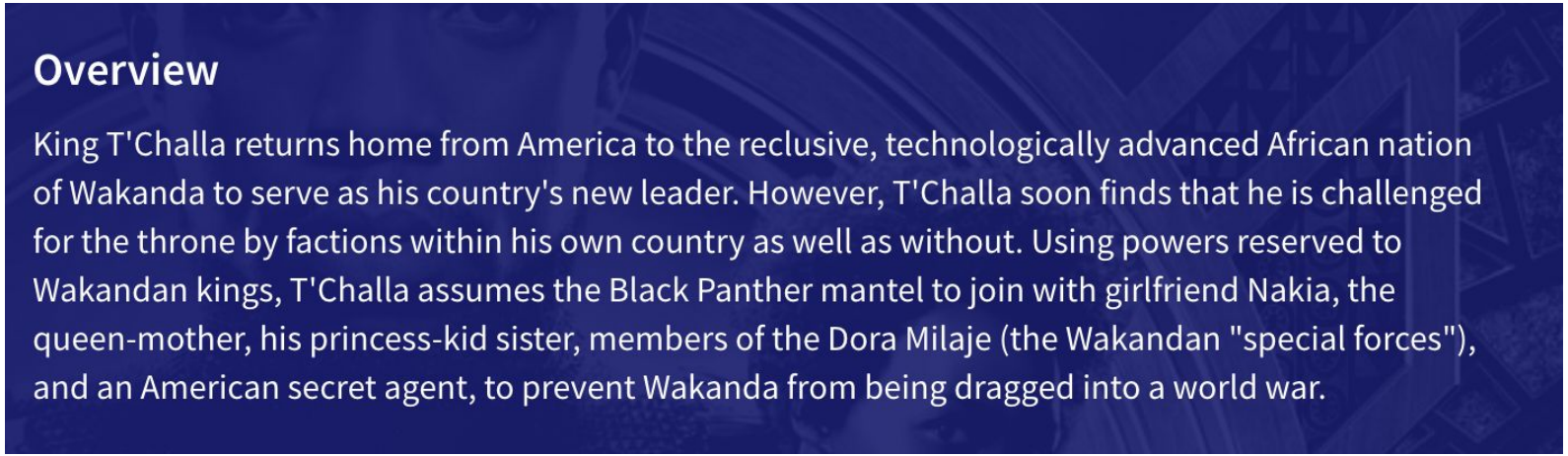


Step 1: Define Item Profiles

How does all of the text get translated into recommendations??

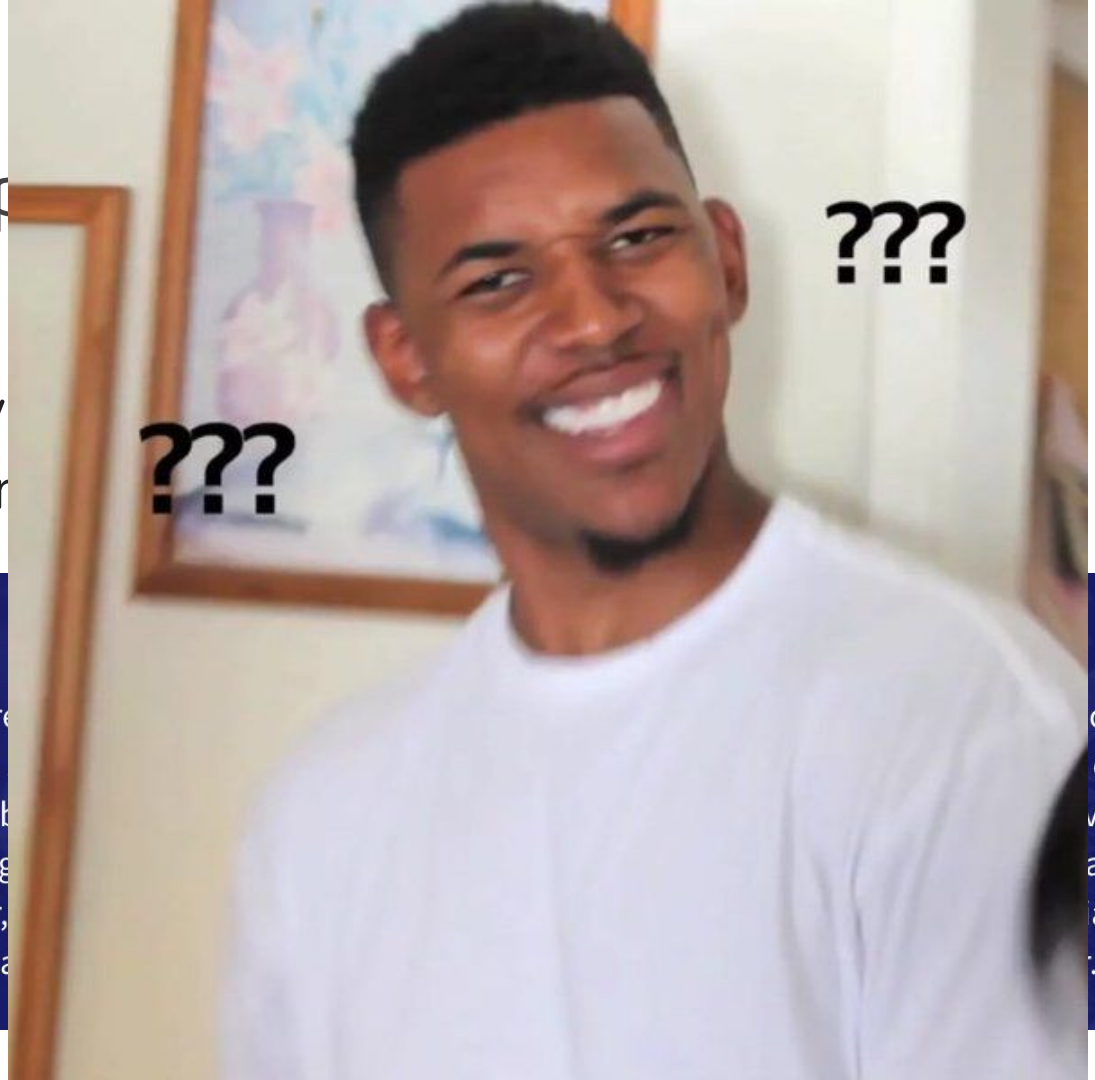
Step 1: Define Item Profiles

How does all of the text get translated into recommendations??



Step

How
recor



Overview

King T'Challa re
of Wakanda to
for the throne b
Wakandan king
queen-mother,
and an America

can nation
challenged
ved to
a, the
al forces"),
.

Understanding TF-IDF

What is TF-IDF?

TF-IDF stands for "Term Frequency, Inverse Document Frequency". It is a way to score the importance of words (or "terms") in a document based on how frequently they appear across multiple documents.

Intuitively...

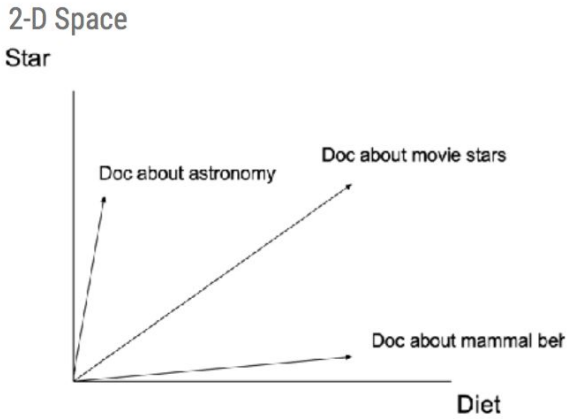
- If a word appears frequently in a document, it's important. Give the word a high score.
- But if a word appears in many documents, it's not a unique identifier. Give the word a low score.

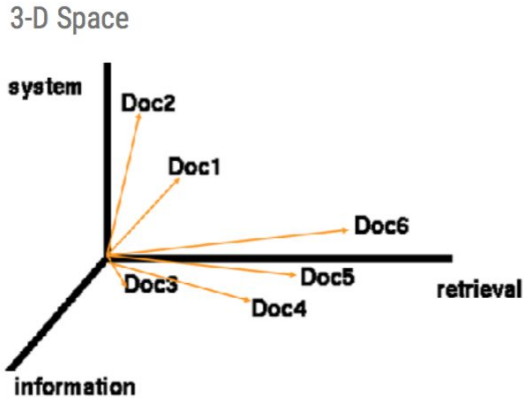
Therefore, common words like "the" and "for", which appear in many documents, will be scaled down. Words that appear frequently in a *single* document will be scaled up.

Understanding TF-IDF

How it works

- Documents are represented as vectors in term space.
- Primary assumption of the Vector Space Model: Documents that are close together in space are similar in meaning.



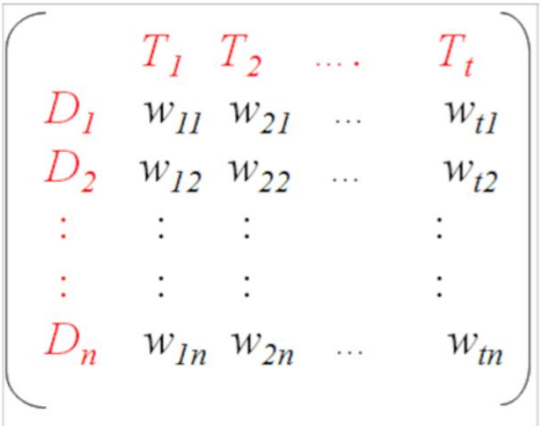


Understanding TF-IDF

How it works

- TF-IDF computes a weight which represents the importance of a term inside a document. It does this by comparing the frequency of usage inside an individual document as opposed to the entire dataset (a collection of documents).

Each row is a document



Each column is a term

Each term is a feature

https://blogs.msdn.microsoft.com/spt/2008/03/05/information-retrieval-amp-search-basic-ir-models/

Understanding TF-IDF

Step 1 Frequency Matrix

TF (Term Frequency, Part 1): measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear more often in a long document than a shorter one. Thus, the term frequency is often divided by the document length as a way of normalization.

Understanding TF-IDF

Step 1 Frequency Matrix

TF (Term Frequency, Part 1): measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear more often in a long document than a shorter one. Thus, the term frequency is often divided by the document length as a way of normalization.

Variants of term frequency (TF) weight

weighting scheme	TF weight
binary	$0, 1$
raw count	$f_{t,d}$
term frequency	$f_{t,d} / \sum_{t' \in d} f_{t',d}$
log normalization	$1 + \log(f_{t,d})$
double normalization 0.5	$0.5 + 0.5 \cdot \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$
double normalization K	$K + (1 - K) \frac{f_{t,d}}{\max_{\{t' \in d\}} f_{t',d}}$

Understanding TF-IDF

Step 2: How to calculate weights?

IDF (Inverse Document Frequency, Part 2): measure of how unique a word is i.e. how infrequently the word occurs across all documents.

Understanding TF-IDF

Step 2: How to calculate weights?

IDF (Inverse Document Frequency, Part 2): measure of how unique a word is i.e. how infrequently the word occurs across all documents.

Variants of inverse document frequency (IDF) weight

weighting scheme	IDF weight ($n_t = \|\{d \in D : t \in d\}\|$)
unary	1
inverse document frequency	$\log \frac{N}{n_t} = -\log \frac{n_t}{N}$
inverse document frequency smooth	$\log\left(1 + \frac{N}{n_t}\right)$
inverse document frequency max	$\log\left(\frac{\max_{\{t' \in d\}} n_{t'}}{1 + n_t}\right)$
probabilistic inverse document frequency	$\log \frac{N - n_t}{n_t}$

Understanding TF-IDF

Putting it all together

$$w_{ik} = tf_{ik} * \log(N / n_k)$$

T_k = term k in document D_i

tf_{ik} = frequency of term T_k in document D_i

idf_k = inverse document frequency of term T_k in C

N = total number of documents in the collection C

n_k = the number of documents in C that contain T_k

$$idf_k = \log\left(\frac{N}{n_k}\right)$$

Understanding TF-IDF

Example

Example: Tagging Blog Posts

Step 1: Generate Scores for Each Document

Let's say you have a 100 word blog post with the word "JavaScript" in it 5 times. The calculation for the Term Frequency would be:

Next, assume your entire collection of blog posts has 10,000 documents and the word "JavaScript" appears at least once in 100 of these. The Inverse Document Frequency calculation would look like this:

To calculate the TF-IDF, we multiply the previous two values. This gives us the final score:

Understanding TF-IDF

Example

Example: Tagging Blog Posts

Step 1: Generate Scores for Each Document

Let's say you have a 100 word blog post with the word "JavaScript" in it 5 times. The calculation for the Term Frequency would be:

```
TF = 5/100 = 0.05
```

Next, assume your entire collection of blog posts has 10,000 documents and the word "JavaScript" appears at least once in 100 of these. The Inverse Document Frequency calculation would look like this:

To calculate the TF-IDF, we multiply the previous two values. This gives us the final score:

Understanding TF-IDF

Example

Example: Tagging Blog Posts

Step 1: Generate Scores for Each Document

Let's say you have a 100 word blog post with the word "JavaScript" in it 5 times. The calculation for the Term Frequency would be:

```
TF = 5/100 = 0.05
```

Next, assume your entire collection of blog posts has 10,000 documents and the word "JavaScript" appears at least once in 100 of these. The Inverse Document Frequency calculation would look like this:

```
IDF = log(10,000/100) = 2
```

To calculate the TF-IDF, we multiply the previous two values. This gives us the final score:

Understanding TF-IDF

Example

Example: Tagging Blog Posts

Step 1: Generate Scores for Each Document

Let's say you have a 100 word blog post with the word "JavaScript" in it 5 times. The calculation for the Term Frequency would be:

```
TF = 5/100 = 0.05
```

Next, assume your entire collection of blog posts has 10,000 documents and the word "JavaScript" appears at least once in 100 of these. The Inverse Document Frequency calculation would look like this:

```
IDF = log(10,000/100) = 2
```

To calculate the TF-IDF, we multiply the previous two values. This gives us the final score:

```
TF-IDF = 0.05 * 2 = 0.1
```


WOO HOO
I'M ALL DONE!
makeameme.org


GOOD TRY, OLD BEAN,
BUT NOT QUITE GOOD ENOUGH
quickmeme.com

Step 2: Define User Profiles

Explicit: Ask users to rate items

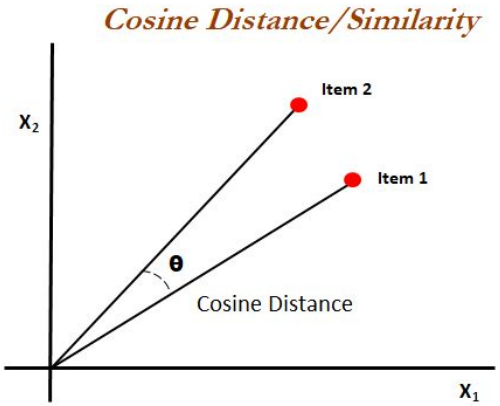
- More accurate
- Harder to scale: only a fraction of users leave ratings/reviews

Implicit: Observing a user's behavior (Look at clicks, purchases etc)

- Easy to scale
- Can be collected with little or no cost to user
- Hard to collect low-rated products

Step 3: Calculate Similarity

Cosine Similarity	$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$



Where: A is the Item Profile and B is the User Profile

Advantages

1. Don’t need data on other users
2. Able to recommend to users with unique tastes
3. Able to recommend new and unpopular items
4. Can give explanations

Disadvantages

1. Finding features can be hard (images, movies, music)
2. Overspecialization
3. Cold-start for new users

Let’s do an example..

Here