Content-Based Recommendation System

Description and implementation with Python

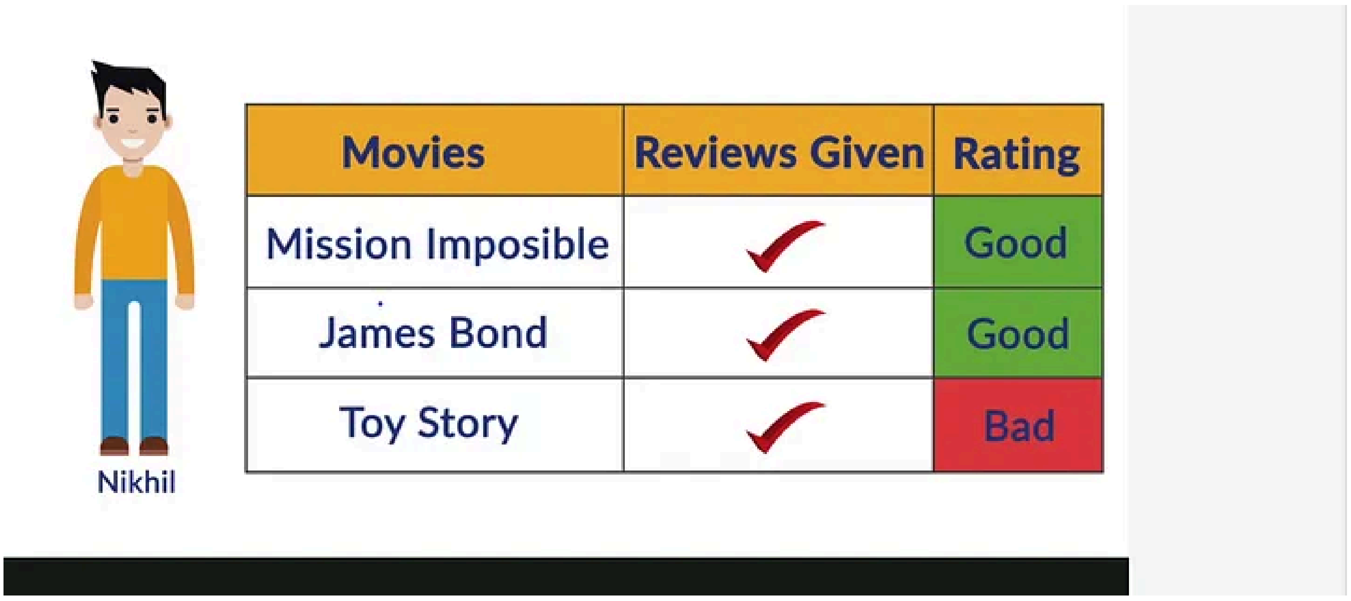
Bindhu Balu · Follow

6 min read · Oct 16, 2019

Listen Share

One popular technique of recommendation/recommender systems is **content-based filtering.** Content here refers to the content or attributes of the products you like. So, the idea in content-based filtering is to tag products using certain keywords, understand what the user likes, look up those keywords in the database and recommend different products with the same attributes.

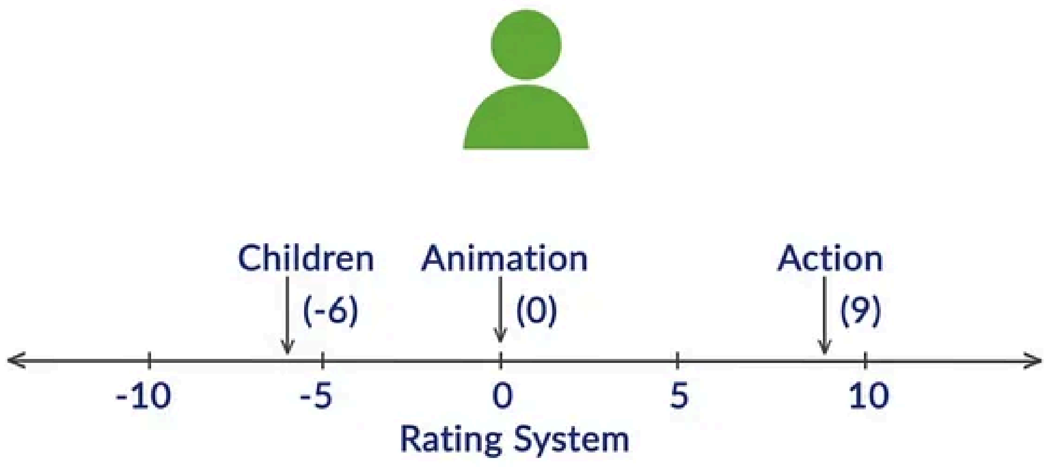
Example: Movie recommendation to a Netflix User by profile name Nikhil



Movies	Reviews Given	Rating
Mission Imposible	✓	Good
James Bond	✓	Good
Toy Story	✓	Bad

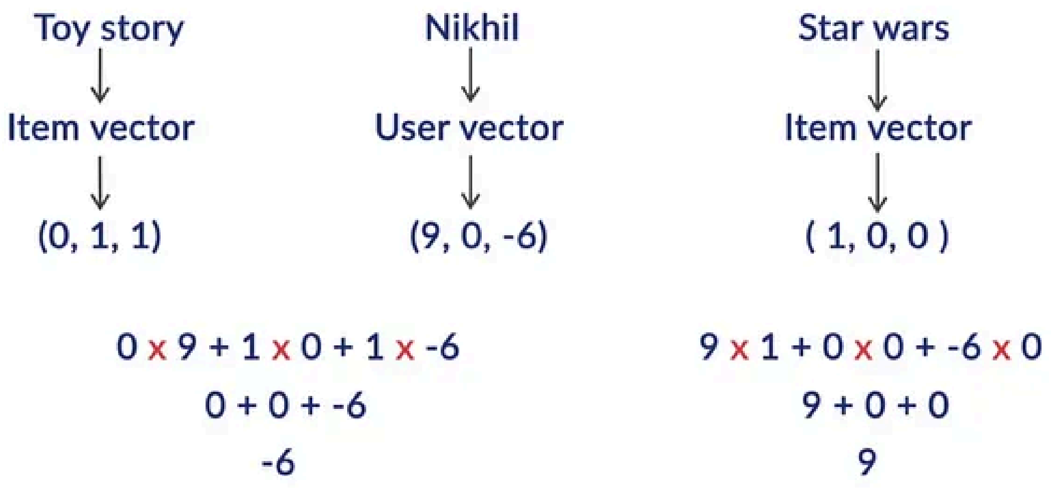
Assume Nikil has given Good ratings to movies like Mission Impossible and James Bond which are tagged as “Action” Genre and gave a bad rating to the movie “Toy Story” which is tagged as “Children” Genre.

Now we will create a User Vector for Nikhil based on his 3 ratings :



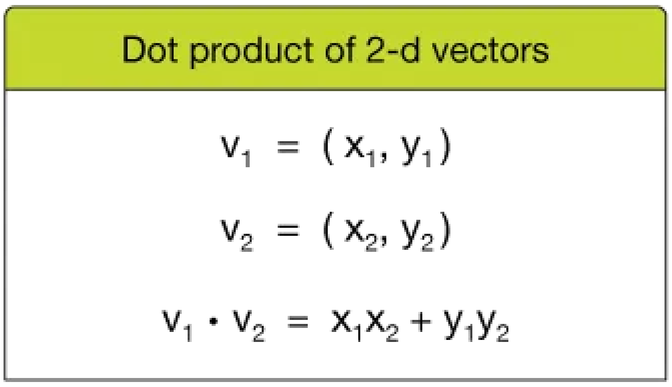
On a rating scale of -10 to 10, since Nikhil loves Action movies, we assign value of 9 to “Action”, Nikhil hasn't watched any Animation movies, we assign 0 to “Animation” and since Nikhil has given bad reviews for movies with Children genre — we assign ‘-6 ‘ to “Children”.

So user Vector for Nikhil is (9, 0, -6) in order of (Action, Animation, Children).



The item vector for movie “Toy Story” is (0,1,1) and the movie “Star Wars” is (1,0,0) in order of (Action, Animation, Children).

We now need to make dot product of two 2-D vectors — Item vector and User Vector


Dot product of 2-d vectors

$$v_1 = (x_1, y_1)$$

$$v_2 = (x_2, y_2)$$

$$v_1 \cdot v_2 = x_1x_2 + y_1y_2$$

Accordingly, the dot product of "Toy Story" is -6 and that os "Star Wars" is 9.

Hence "Star Wars" will be recommended to Nikhil — which also matches our intuition that Nikhil likes Action movies and dislikes Children movies.

In a similar manner — we can calculate the dot products of all the item vectors of all the movies in-store and recommend top 10 movies to Nikhil

Python Implementation of Content-Based Recommendation:

Link to download Input data(CSV file) and python code :

BindhuVinodh/Content-based-recommendation

You can't perform that action at this time. You signed in with another tab or window. You signed out in another tab or...

github.com

Loading the data

```
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import linear_kernel
ds = pd.read_csv("/home/nikita/Downloads/sample-data.csv")
```

Creating a TF-IDF Vectorizer

The **TF*IDF algorithm** is used to weigh a keyword in any document and assign the importance to that keyword based on the number of times it appears in the document. Put simply, the higher the TF*IDF score (weight), the rarer and more important the term, and vice versa.

Mathematically [don't worry it's easy :)],

Each word or term has its respective TF and IDF score. The product of the TF and IDF scores of a term is called the TF*IDF weight of that term.

The **TF (term frequency)** of a word is the number of times it appears in a document. When you know it, you're able to see if you're using a term too often or too infrequently.

```
TF(t) = (Number of times term t appears in a document) / (Total
number of terms in the document).
```

The **IDF (inverse document frequency)** of a word is the measure of how significant that term is in the whole corpus.

```
IDF(t) = log_e(Total number of documents / Number of documents with
term t in it).
```

$$w_{x,y} = tf_{x,y} \times \log\left(\frac{N}{df_x}\right)$$

TF-IDF

Term x within document y

$tf_{x,y}$ = frequency of x in y

df_x = number of documents containing x

N = total number of documents

In Python, scikit-learn provides you a pre-built TF-IDF vectorizer that calculates the TF-IDF score for each document's description, word-by-word.

```
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3), min_df=0,
stop_words='english')
tfidf_matrix = tf.fit_transform(ds['description'])
```

Here, the `tfidf_matrix` is the matrix containing each word and its TF-IDF score with regard to each document, or item in this case. Also, stop words are simply words that add no significant value to our system, like 'an', 'is', 'the', and hence are ignored by the system.

Now, we have a representation of every item in terms of its description. Next, we need to calculate the relevance or similarity of one document to another.

Calculating Cosine Similarity

```
cosine_similarities = linear_kernel(tfidf_matrix, tfidf_matrix)
results = {}
for idx, row in ds.iterrows():
   similar_indices = cosine_similarities[idx].argsort()[:-100:-1]
   similar_items = [(cosine_similarities[idx][i], ds['id'][i]) for i
in similar_indices]
   results[row['id']] = similar_items[1:]
```

Here we've calculated the cosine similarity of each item with every other item in the dataset, and then arranged them according to their similarity with item `i` , and stored the values in `results` .

Making a recommendation

So here comes the part where we finally get to see our recommender system in action.

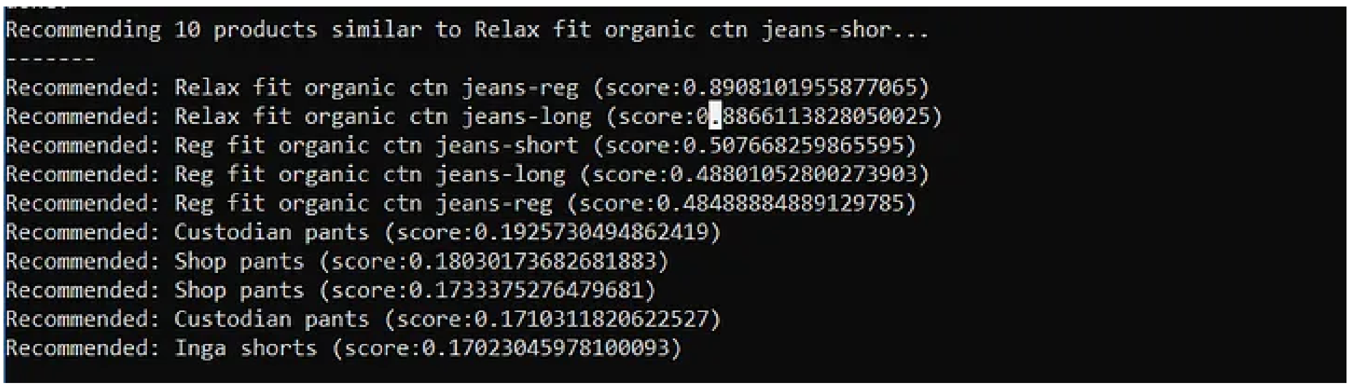
```
def item(id):
  return ds.loc[ds['id'] == id]['description'].tolist()[0].split(' -
')[0] # Just reads the results out of the dictionary.def
recommend(item_id, num):
    print("Recommending " + str(num) + " products similar to " +
item(item_id) + "...")
    print("-------")    recs = results[item_id][:num]
    for rec in recs:
       print("Recommended: " + item(rec[1]) + " (score:" +
str(rec[0]) + ")")
```

Here, we just input an `item_id` and the number of recommendations that we want, and voilà! Our function collects the `results[]` corresponding to that `item_id` , and we get our recommendations on the screen.

Results

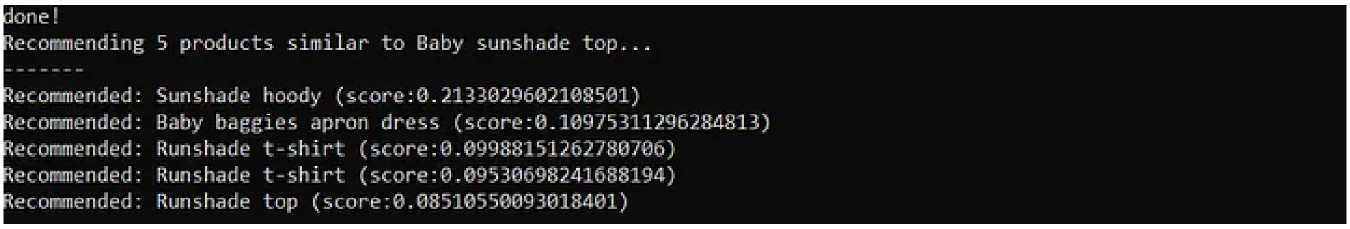
Here's a glimpse of what happens when you call the above function.

```
recommend(item_id=11, num=5)
```



```
Recommending 10 products similar to Relax fit organic ctn jeans-shor...
-------
Recommended: Relax fit organic ctn jeans-reg (score:0.8908101955877065)
Recommended: Relax fit organic ctn jeans-long (score:0.8866113828050025)
Recommended: Reg fit organic ctn jeans-short (score:0.507668259865595)
Recommended: Reg fit organic ctn jeans-long (score:0.48801052800273903)
Recommended: Reg fit organic ctn jeans-reg (score:0.48488884889129785)
Recommended: Custodian pants (score:0.1925730494862419)
Recommended: Shop pants (score:0.18030173682681883)
Recommended: Shop pants (score:0.1733375276479681)
Recommended: Custodian pants (score:0.1710311820622527)
Recommended: Inga shorts (score:0.17023045978100093)
```

Recommendations similar to organic cotton jeans-shorts



```
done!
Recommending 5 products similar to Baby sunshade top...
-------
Recommended: Sunshade hoody (score:0.2133029602108501)
Recommended: Baby baggies apron dress (score:0.10975311296284813)
Recommended: Runshade t-shirt (score:0.09988151262780706)
Recommended: Runshade t-shirt (score:0.09530698241688194)
Recommended: Runshade top (score:0.08510550093018401)
```

Recommendations similar to baby sunshade top

Advantages of Content-Based Filtering

- **User independence:** The content-based method only has to analyze the items and a single user's profile for the recommendation, which makes the process less cumbersome. Content-based filtering would thus produce more reliable results with fewer users in the system.

- **Transparency:** Collaborative filtering gives recommendations based on other unknown users who have the same taste as a given user, but with content-based filtering, items are recommended on a feature-level basis.

- **No cold start:** As opposed to collaborative filtering, new items can be suggested before being rated by a substantial number of users.

Disadvantages of Content-Based Filtering

- **Limited content analysis**: If the content doesn't contain enough information to discriminate the items precisely, the recommendation itself risks being imprecise.

- **Over-specialization:** Content-based filtering provides a limited degree of novelty since it has to match up the features of a user's profile with available items. In the case of item-based filtering, only item profiles are created and users are suggested items similar to what they rate or search for, instead of their past history. A perfect content-based filtering system may suggest nothing unexpected or surprising.

Data Science Machine Learning Artificial Intelligence Recommender Systems Recommendation System

Follow

Written by Bindhu Balu

225 Followers

Amateur Writer , Mom , loves travelling and learning.

More from Bindhu Balu

Bindhu Balu in PM101

Spotify Case Study

Spotify's Current Position

4 min read · Jan 13, 2024

266 2

Bindhu Balu in PM101

Solving Growth Problems — A Case Study

Problem Statement:

4 min read · Jan 14, 2024

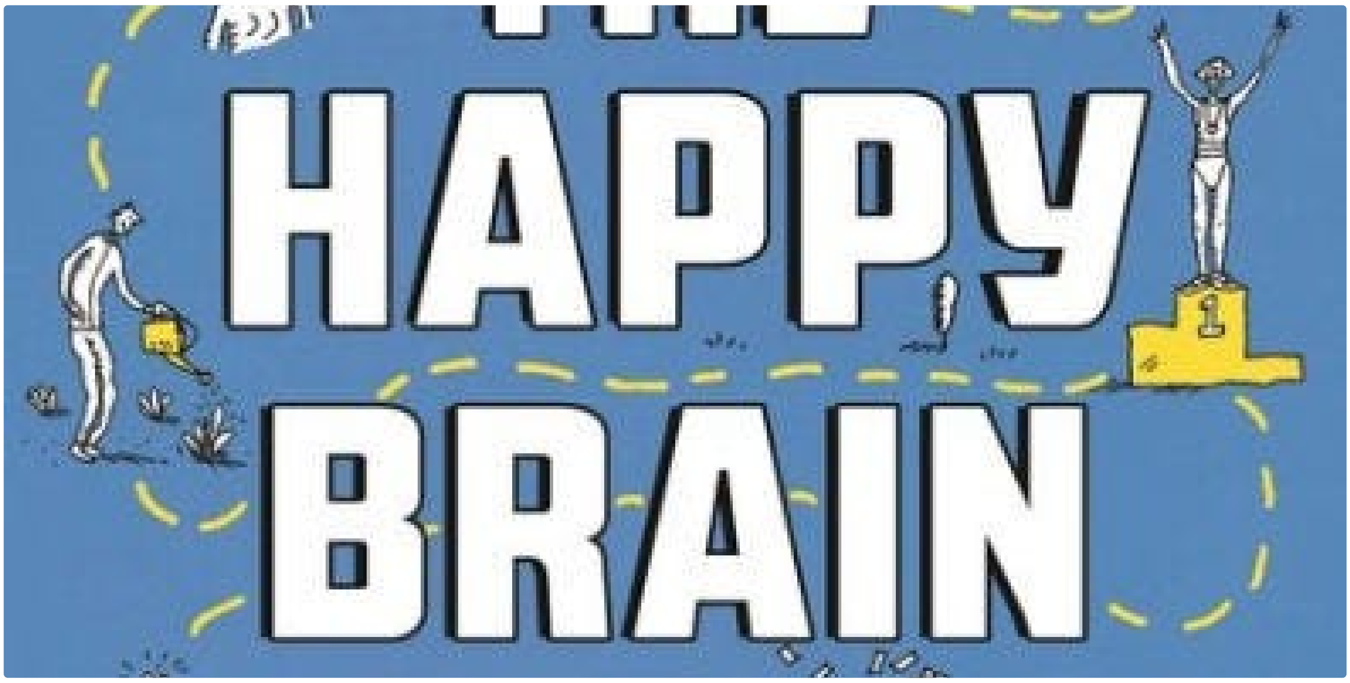
220 1

Bindhu Balu in PM101

Product Strategy — "Should Bing rebrand itself as a sports search engine?"

2 min read · Jan 22, 2024

66



Bindhu Balu

“The Happy Brain — The Science of where Happiness Comes from and why” — Book Review

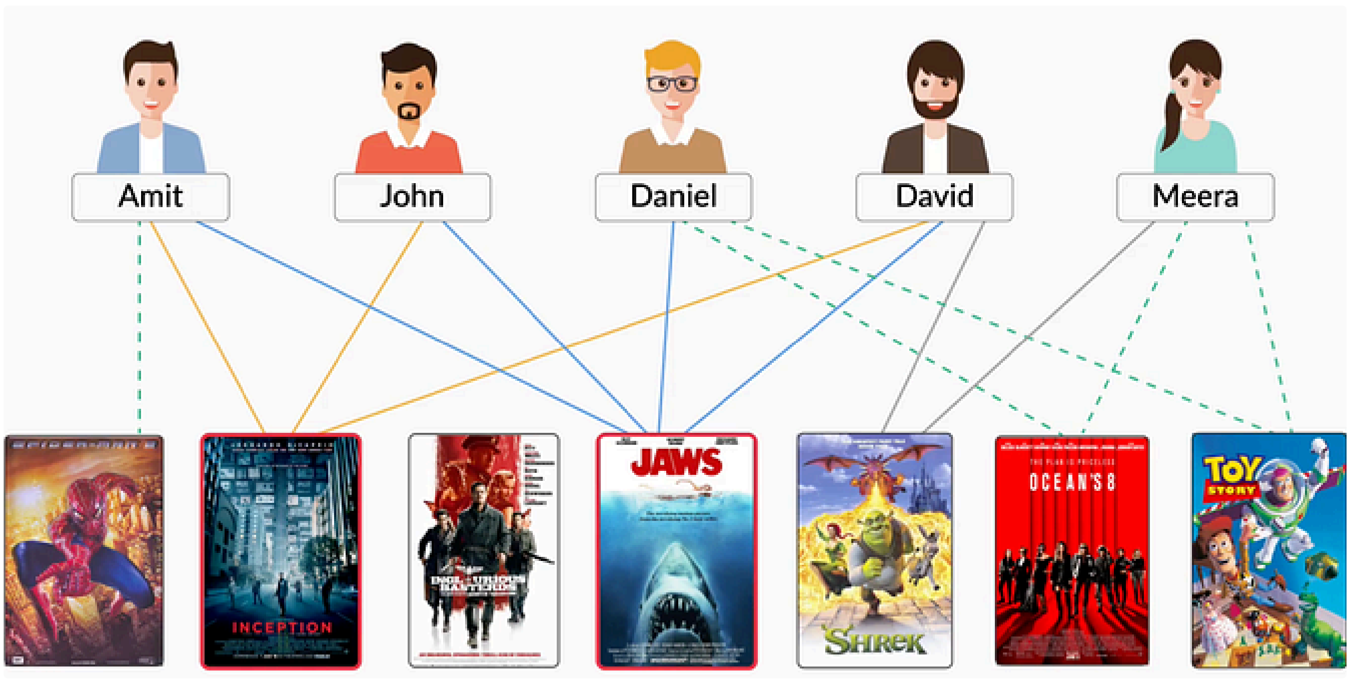
IT’S HUMAN NATURE TO want to be happy, but people know relatively little about the science behind the emotion.

4 min read · Mar 19, 2022

144 1

See all from Bindhu Balu

Recommended from Medium



Sharan Harsoor in AI Mind

Recommendation Systems: An Overview

A recommendation system, also known as a recommender system or engine, is a type of software application or algorithm designed to provide...

25 min read · Nov 13, 2023

118

Umair Iftikhar

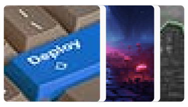
Part 3: Exploring Content-Based Filtering in Recommendation Systems

In the previous parts of our recommendation system series, we covered the fundamentals of item-item collaborative filtering and how to...

3 min read · Oct 26, 2023

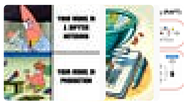
65

Lists



Predictive Modeling w/ Python

20 stories · 1057 saves

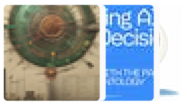


Natural Language Processing

1344 stories · 827 saves

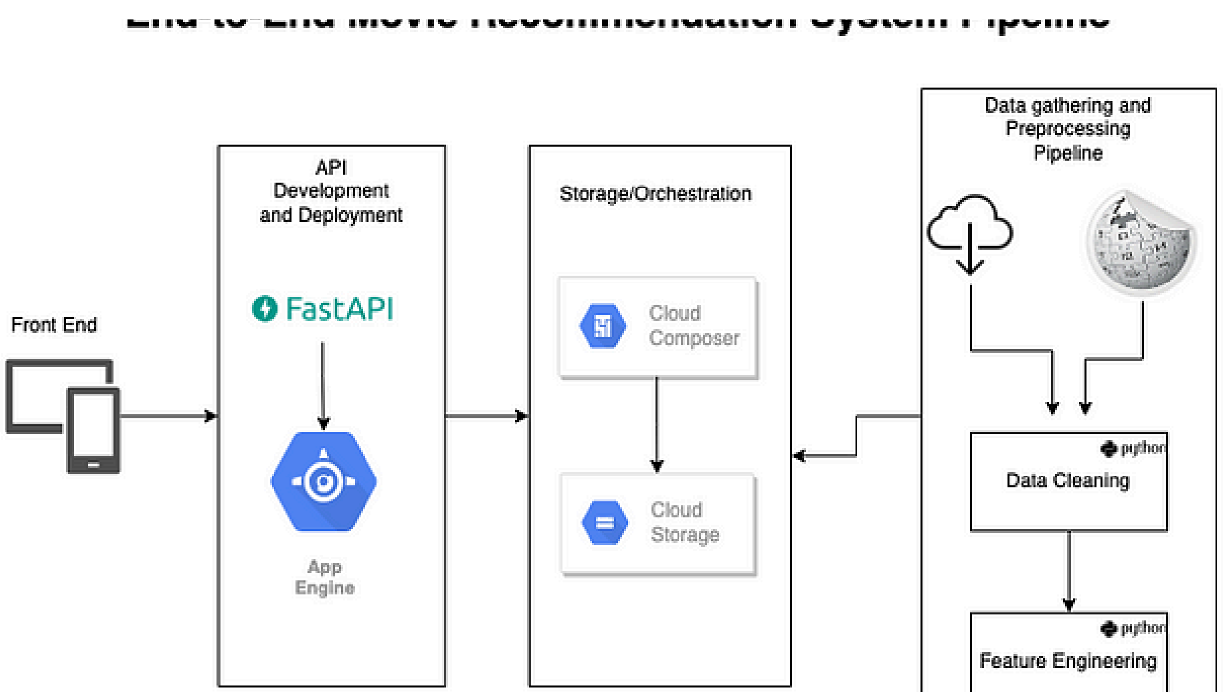
Practical Guides to Machine Learning

10 stories · 1264 saves



data science and AI

40 stories · 122 saves



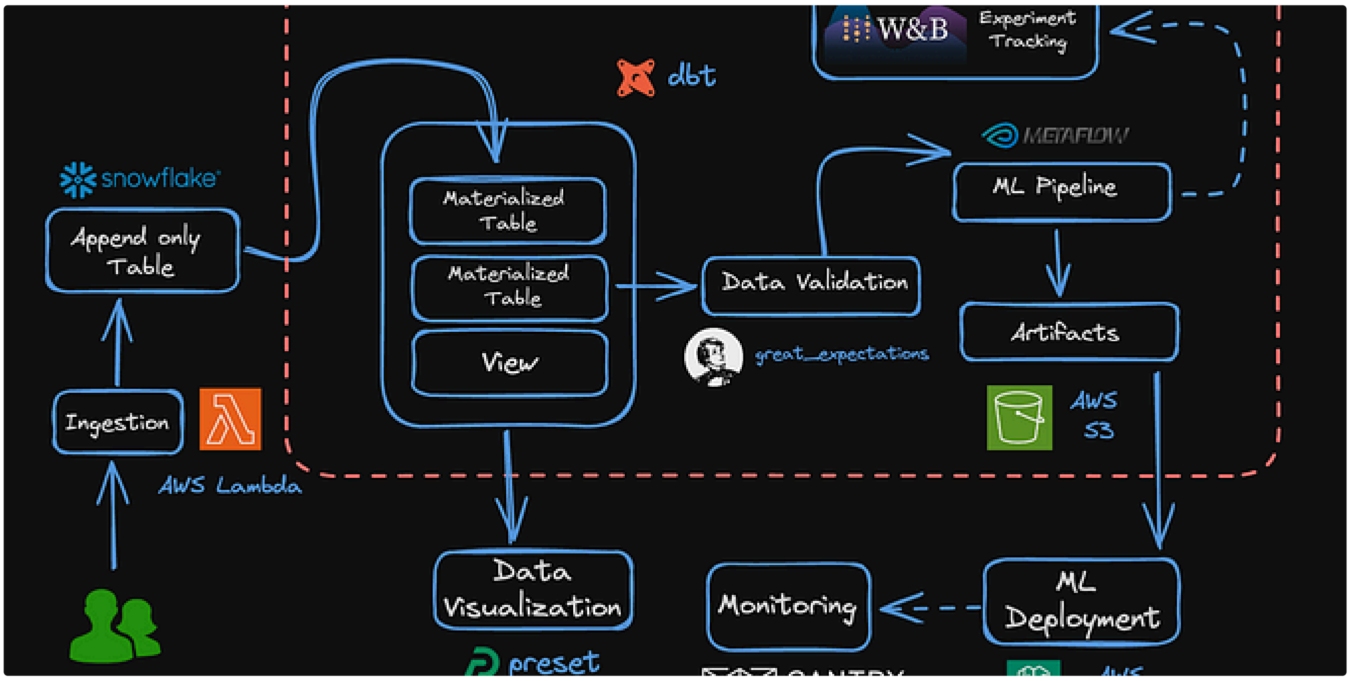
Oham Ugochukwu in Towards AI

Building an End-to-End Recommendation System

Leveraging transformer embeddings and a vector database to speed up inference

· 4 min read · Oct 24, 2023

6



Zain ul Abideen

One-Stop Guide for Production Recommendation Systems

Candidate Generation, SSR for CG, ANN Search, Ranking, Evaluation Metrics, and Architectural paradigm of Real-time Recsys

24 min read · Nov 21, 2023

271 2

Ndubisiprecious

Building an Enhanced Movie Recommendation System Using Machine Learning

Recommendation systems play a pivotal role in various industries, aiding users in discovering relevant content or products. However...

4 min read · Nov 24, 2023

Leonie Monigatti in Towards Data Science

Building a Recommender System using Machine Learning

“Candidate rerank” approach with co-visitation matrix and GBDT ranker model in Python

✦ · 6 min read · Mar 1, 2023

703 9

See more recommendations