photoboxgroup

Recommender Systems 101

Nathalie Skrzypek

Agenda

photoboxgroup

- Why Recommenders?
- Recommenders in a Nutshell
- Types of Recommenders & Evaluation Metrics

Why Recommenders?

photoboxgroup

- Personalise user experience

  Because you watched GoT you may like Vikings

  People who listen to 'Let It Go' also like 'All About that Bass'

- 30% of clicks on Amazon come from Recommendations
- More effective E-mail campaigns (e.g. Photobox)
- Cross-sell & Up-sell

Happier customers → more $$$

Recommender Systems in a Nutshell

Use historic website / purchase data to recommend relevant products to customers

Implicit data: From user actions, not active ratings

Explicit data: Users have consciously rated products

Difference between predictions and recommendations

Predictions: Estimate how much you will like a product

Recommendations: Give top-n suggestions for products you may like

Non-Personalised / Stereotyped

- Summary Statistics (e.g. most read)
- Popularity / external Community data
- Keyword / topic searches
- Group Preference (e.g. 18-25 year olds)

Most read

1. IS bride 'should live in Holland' - husband
2. Wind speeds pick up as Storm Freya nears
3. Woman tried to save stabbed girl, 17
4. Trump launches furious attack on Mueller
5. Brexiteers outline EU deal terms

Example: Reddit Algorithms

Reddit Stories Ranking

- Newer stories have higher scores
- Logarithmic scale weighing first votes higher
- Polarising stories get lower ratings

Reddit Comments Ranking for Sorting

- Gives comments provisional rankings
- Balances proportion of positive ratings with uncertainty of small number of observations

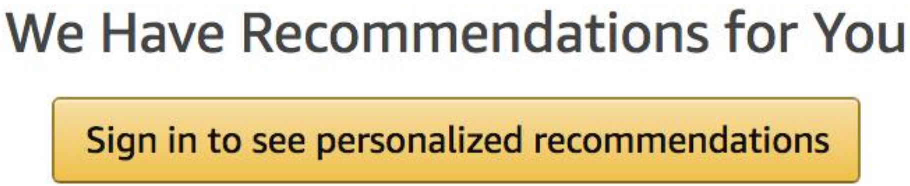
Content-Based Recommenders

Recommends items similar to those users liked in the past

\+ **Association Rules** (Market Basket Analysis), TF-IDF

\- **Won't find surprising connections**

\- Needs well-structured data



Example: Association Rules

photoboxgroup

$\{X\} \rightarrow \{Y\}$ (People who liked X also liked Y)

Support: Default popularity of an item

$$\text{Support (X)} = \frac{\text{Transactions containing (X)}}{\text{Total Transactions}}$$

Confidence: Likelihood that item Y is bought if X is bought

$$\text{Confidence (X}\rightarrow\text{Y)} = \frac{\text{Number of Transactions (X\&Y)}}{\text{Number of Transactions (X)}}$$

Lift: Increase in ratio of sales of Y when X is sold

$$\text{Lift (X}\rightarrow\text{Y)} = \frac{\text{Support (X \& Y)}}{\text{Support (X) * Support (Y)}}$$

Example: Apriori Algorithm in Python

photoboxgroup

	user_id	movie_id	movie title
0	196	242	Kolya (1996)
1	186	302	L.A. Confidential (1997)
2	22	377	Heavyweights (1994)
3	244	51	Legends of the Fall (1994)
4	166	346	Jackie Brown (1997)

	user_id	movie_views
0	1	[Three Colors: White (1994), Grand Day Out, A ...
1	2	[Rosewood (1997), Shall We Dance? (1996), Star...
2	3	[How to Be a Player (1997), Devil's Own, The (...
3	4	[Mimic (1997), Ulee's Gold (1997), Incognito (...
4	5	[GoldenEye (1995), From Dusk Till Dawn (1996),...

```python
from apyori import apriori

df = df.groupby(['user_id'])['movie title'].apply(
    lambda x: x.values.tolist()).reset_index(name='movie_views')

df_listoflists=[]
for row in df.movie_views:
    df_listoflists.append(list(row))

association_rules = apriori(df_listoflists,
                            min_support=0.2,
                            min_confidence=0.1,
                            min_lift=3,
                            max_length=2)
association_results = list(association_rules)

for item in association_results:

    pair = item[0]
    print(pair)
    items_list = [x for x in pair]
    print("Rule: " + items_list[0] + " -> " + items_list[1])
    print("Support: " + str(item[1]))
    print("Confidence: " + str(item[2][0][2]))
    print("Lift: " + str(item[2][0][3]))
    print("=====================================")
```

```
frozenset({'20,000 Leagues Under the Sea (1954)', '12 Angry Men (1957)'})
Rule: 20,000 Leagues Under the Sea (1954) -> 12 Angry Men (1957)
Support: 0.2
Confidence: 1.0
Lift: 5.0
=====================================
frozenset({'Abyss, The (1989)', '12 Angry Men (1957)'})
Rule: Abyss, The (1989) -> 12 Angry Men (1957)
Support: 0.2
Confidence: 1.0
Lift: 5.0
=====================================
```

Example: TF-IDF for documents

E.g. Finding similar research papers based on your interests

Term frequency: # of occurrences of a term in the document

Inverse Document Frequency: How many documents contain term

TF-IDF ranks documents by term overlap and terms by frequency, demoting common terms

TF-IDF in Content-Based Recommendation systems

Create document profile as weighted vector of its tags

Combined those with ratings to create user profiles

photoboxgroup

```python
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.metrics.pairwise import cosine_similarity

def item(movie_id):
        return df_tfidf.loc[
            df_tfidf['movie id'] == movie_id]['movie title'].tolist()[0]

df_tfidf = items[['movie id', 'movie title']].head(100)

# get relevant word combinations
tf = TfidfVectorizer(analyzer='word', ngram_range=(1, 3),
                     min_df=0, stop_words='english')


# Learn vocabulary and idf, return term-document matrix.
tfidf_matrix = tf.fit_transform(df_tfidf['movie title'])

cosine_similarities = cosine_similarity(tfidf_matrix,
                                        Y=None,
                                        dense_output=False)

# dictionary created to store the result in a dictionary format (ID : (Score,item_id))#
for idx, row in df_tfidf.iterrows(): #iterates through all the rows
    results = {}
    # stores similar ids based on cosine similarity.
    # sorts them in ascending order and keeps top 5
    # 0 means no similarity and 1 means perfect similarity
    similar_indices = cosine_similarities[idx].data.argsort()[:-7:-1]

    similar_items = [(cosine_similarities[idx].data[i],
                      df_tfidf['movie id'][i])
                     for i in similar_indices]

    #stores 5 most similar books, you can change it as per your needs
    results[row['movie id']] = similar_items[1:]

    for key, value in results.items():
        print('The top 5 movies recommended for', item(key), 'are:')

        for val in range(0, len(value)):
            print(item(value[val][1]), '(score:', round(value[val][0], 5), ')')

        print("\n----------------------------------------------------------")
```

Collaborative Filtering: Memory Based

Matrix of users and products / movies

user-based: Similarities between users are used to make recommendations

- Computer intensive for a large number of users
- Doesn't deal well with changes in user preferences

item-based: Similarities between items calculated using people's ratings of those items

\+ Changes less frequently due to high number of ratings

\- Leads to fewer serendipity discoveries

Different Similarities

photoboxgroup

Jaccard Similarity:

→ Typically used where **products don't have numeric ratings**

$$J(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}$$

Cosine Similarity:

→ Use for **sparse data**

$$cos(\theta) = \frac{\sum_{i=1}^{n} X_i Y_i}{\sqrt{\sum_{i=1}^{n} X_i^2}\sqrt{\sum_{i=1}^{n} Y_i^2}}$$

Pearson Similarity:

→ Use when **data is subject to user-bias/different rating scales**

$$r_{xy} = \frac{\sum_{i=1}^{n}(x_i - \overline{x})(y_i - \overline{y})}{\sqrt{\sum_{i=1}^{n}(x_i - \overline{x})^2}\sqrt{\sum_{i=1}^{n}(y_i - \overline{y})^2}}$$

Evaluation Metrics for Recommenders I

photoboxgroup

For Top N-Recommendations: Decision support metrics

Recall: Ratio of items that a user liked that were recommended?

Precision: Out of recommended items, how many did the user like?

F1: (2 * recall * precision) / (recall + precision)

For recommenders making predictions: Prediction accuracy

E.g. Mean Absolute Error (MAE), Root Mean Square Error (RMSE)

Evaluation Metrics II - Rank Metrics

Fraction of Concordant Pairs: Fraction of all pairs in correct order

Coverage: % of products recommended

Diversity: How different are recommended items?

Mean reciprocal rank (MRR):

$$\boxed{MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}}$$

Discounted Cumulative Gain:

$$\boxed{DCG_n = \sum_{i=1}^{n} \frac{rel_i}{log_2(i+1)}}$$

In reality: Lift, cross-sales, up-sales, conversion

Conclusions

photoboxgroup

- Used in many e-commerce websites
- Types of Recommenders
  - Non-Personalised
  - Content-Based
  - Collaborative Filtering (Memory & Model based)
- Real applications are usually a mix of several types
- Improves User Experience / Financials

photoboxgroup

Thank you

Any Questions?