Content and Popularity based RecoSys

Learning Objectives

- Introduction to recommendation system and its applications.
- Why recommendation system is important.
- Key approaches of Recommendation system.
- Market basket analysis.
- Content Based Recommendation system and its applications.
- Case studies for Content based and Popularity based RecoSys.

Introduction to recommendation system and its applications

- Recommender systems are one of the most successful and widespread application of machine learning technologies in business.
- You can apply recommender systems in scenarios where many users interact with many items.

Intro to RecoSys Contd.

- You can find large scale recommender systems in retail, video on demand, or music streaming.
- Recommendation of news or videos for media, product recommendation or personalization in travel and retail can be handled by similar machine learning algorithms.
- There are also recommender systems for experts, collaborators, jokes, restaurants, garments, financial services, life insurance, romantic partners (online dating), and Twitter pages.

Why recommendation system is important

Recommendation systems can help match users to items

They help item provider deliver their items to right user.

- Identify products most relevant to the user.
- Personalized content
- Eg. Top and Offers

Help website improve user engagement and increase online sales

Key approaches of Recommendation system

The key approaches to RecSys

- Collaborative filtering
- Content based
- Popularity based

Popularity based Recommendation system

- Recommended items viewed/purchased by most people.
- Recommendations : ranked list of items by their purchase count / viewed count “Popular news”.
- It uses:
  - Text only
  - Purchase history
  - User and item features
  - Scale

Note: The problem with popularity based recommendation system is that the personalization is not available with this method . For eg: for a song recommendation if the model is popularity based recommender, it is not personalised towards any user and will output the same list of recommended songs

Content Based Recommendation system and its applications

- A content based recommender works with data that the user provides, either explicitly (rating) or implicitly (clicking on a link).
- Based on that data, a user profile is generated, which is then used to make suggestions to the user.
- As the user provides more inputs or takes actions on the recommendations, the engine becomes more and more accurate.
- The information source that content-based filtering systems are mostly used with are text documents. A standard approach for term parsing selects single words from documents.

Content based RecoSys Contd.

Advantages

- Content-based recommender systems do not use a lot of user data.
- Only requires item data and you can start giving recommendations to users.
- It is viable to give recommendations to even your first customer as long as you have sufficient data to build his user profile

Challenges

- The item data needs to be well distributed
- Recommendations to the user will be mostly substitutes, and not complements.
- Less dynamic

Bag of Words (BOW) vectorization

- Text data cannot be directly used for modelling.

- One approach to extract features involves the BOW approach.

- The BOW approach generates a Document-Term Matrix (DTM) where columns represents all the unique words in the corpus and the cell value indicates the frequency of that word in a given document.

TF-IDF

TF-IDF is composed of 2 terms:

- Term Frequency (TF) measures how frequently a term appears in the document.
- Inverse Document Frequency (IDF) measures the relative importance of words in a corpus.

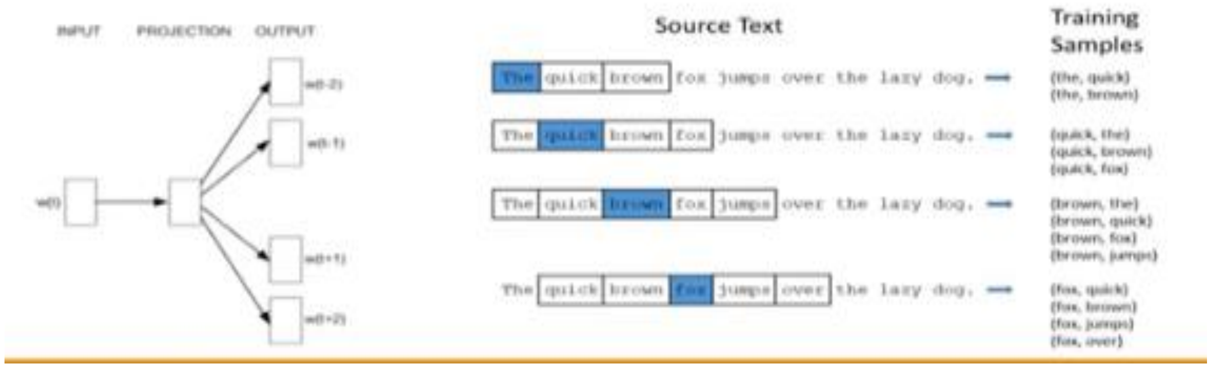
$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

$tf_{i,j}$ = number of occurrences of i in j
df_i = number of documents containing i
N = total number of documents

Word Embedding Models (Word2Vec)

- Word vectors are trained by setting up a fake training task.

- The weights of the hidden layer learnt during the training process provides the feature vector for each word – words that appear in similar context ends up having similar feature vectors.



Case Study

Anonymous Ratings from the Jester Online Joke Recommender System

Abstract

Ratings are real values ranging from -10.00 to +10.00 (the value "99" corresponds to "null" = "not rated").

One row per user

The first column gives the number of jokes rated by that user. The next 100 columns give the ratings for jokes 01 - 100.

Dataset

http://eigentaste.berkeley.edu/dataset/

Steps to follow

1. Read the dataset
2. Take care about the header as there are no column names given in the dataset.
3. Consider ratings named dataframe with only first 200 rows and all columns from 1(first column is 0) of dataset.
4. Change the column indices from 0 to 99
5. In the dataset, the null ratings are given as 99.00, so replace all 99.00s with 0
6. Normalize the ratings using StandardScaler and save them in ratings_diff variable
7. Find the mean for each column in ratings_diff i.e, for each joke
8. Consider all the mean ratings and find the jokes with highest mean value and display the top 10 joke IDs.

Case study on Content based RS

Context:

This dataset contains ratings for ten thousand popular books. As to the source, let's say that these ratings were found on the internet. Generally, there are 100 reviews for each book, although some have less - fewer - ratings. Ratings go from one to five

Both book IDs and user IDs are contiguous. For books, they are 1-10000, for users, 1-53424. All users have made at least two ratings. Median number of ratings per user is 8.

There are also books marked to read by the users, book metadata (author, year, etc.) and tags.

Here is to recommend good book

Attribute information

1. id
2. book_id
3. best_book_id
4. work_id
5. books_count
6. isbn
7. isbn13
8. authors
9. original_publication_year
10. original_title
11. title
12. language_code
13. average_rating
14. ratings_count
15. work_ratings_count
16. work_text_reviews_count
17. ratings_1
18. ratings_2
19. ratings_3
20. ratings_4
21. ratings_5
22. image_url
23. small_image_url

Steps to follow

1. Get the data.
2. Perform data pre-processing
3. Vectorize title
4. What happens when we change the value of min_df value? What is this argument controlling?
5. Find Similarity Between Items
6. What are the most similiar books? Why does "First to Kill", "Kill me if you can" etc all have same ratings?
7. Find out what features have been considered by the vectorizer for a given title?

greatlearning

HappyLearning

Questions

?