Item Recommender

Recommender systems are that offer suggestions to users based on many features and factors. Systems such as these predict the most product users are most likely to be interested in. Recommenders are one of the most common engines Data Scientist work with. There is a huge demand for these tools that most it is crucial that Data Scientist have a comprehensive knowledge of them.

Recommender system are considered invaluable to many online companies. Companies such as Netflix, use these tools to offer movie/tv suggestions that the customer might want to watch next. These suggestions promote increase viewership, which retains customers. Another giant, Amazon, uses recommender system in all facets of operation. Amazon uses recommender systems to help their users identify the correct product for them.

1. Introduction

1.1 Types of Recommender Systems

There are two separate recommender system types: Content & Collaborative Filtering. More modern-day systems use a composite of both types called Hybrid Recommenders.

1.2 Content

A content recommendation engine is based on the data collected from user's behaviors. The data is collected and analyzed using machine learning. Similarities are found between different products and is calculated based on attributes of the products. In content-based movie recommender systems, similarities between movies are calculated based on the actors in the movie, genre of the film, etc.

Some benefits to this type of filtering is that the model does not need any information from other users. Additionally, since these models work better on an individual level recommendation are not influenced by external factors, such as popularity.

Content filtering lacks in the ability to expand users' interest into unknown areas. Another disadvantage is that content filtering engines require are more labour intensive.

1.3 Collaborative Filtering

Collaborative filtering engines utilize public insight, by finding similarities between users and products to provide recommendations. The theme is as follows, if individual A buys product X then Y, and if individual B likes product X, then the chances of the individual B buying product Y is high. With this knowledge the engine recommends product Y for individual B.

As an example, if Sally buys a kitchen knife-set and buys a matching knife block, it is likely when Gary buys the knife-set he will buy the matching knife block. For movies recommender systems, collaborative filtering works on movie ratings. If users score similar rating to movie S, then the engine will recommend movie T based on the data from previous users.

A benefit to such engines is that no knowledge is needed, because all data is automatically obtained. These models can introduce new interest to users. It is always learning and improving.

Disadvantages to collaborative filtering are:

- Hard to start fresh, without any knowledge computer is guessing radically.
- Hard to change features once the learning has started, ie., movies by country, age.

Collaborative filtering is predominately the more powerful of the two and is used more often. Hybrid systems are now implemented to leverage both content and collaborative filtering, a best of both worlds situation.

2. Exercise Description

Using a sample dataset of 20,000 products from an e-commerce store, create a system that, when given a specific product, will recommend similar products for the user.

2.1 Tasks

Complete the following tasks:

1. Analyze the above dataset and come up with a strategy for recommending products similar to a specific product.
2. Create a demoable proof-of-concept of your recommendation engine. Be prepared to share and discuss your demo in a future interview.
3. Write and share a brief description of how your solution works and what data science principles are involved.
4. Write and share a brief description of how you could improve your solution if you had more time to work on it.

3. Report

3.1 Analyze the dataset

The dataset contains 20,000 products from an e-commerce store. The data is in a structured csv format, with 15 columns of various types. Initial profiling indicated a few null integer values. Column product specialization contains data of nested dictionary, future work is needed here to extract meaningful information. No user data was found with the sample dataset.

3.1.1 Strategy
With the initial quick profiling as seen in the previous section, a Content-filtering model was created. Since there is no user data, we are not able to perform collaborative filtering since a catalogue of historical data is needed to construct a decent filter. The model implemented was a simple K-Nearest Neighbor model (KNN). KNN are supervised models, which are used for classification data (predominately), essentially looking for

pattern recognition. The algorithm specifically used brute-force computation, calculating the distances between all pairs of points in the dataset. Finally, the default metric was used. Term Frequency-Inverse Document Frequency (TFIDF) a tool heavily used in Natural Language Processing (NLP) was also implemented. TFIDF gives a statistical values that is intended to reflect the importance of a word in a document.

3.2 Demo-able proof-of-concept

Excerpts are found within the markdown file, but for a full comprehensive script please refer to the Jupyter Notebook labelled *ItemRecommnder.ipynb*. Python 3.0+ is needed to run.

3.3 How your solution works

The goal of the model is to take a random item from the store database and recommend similar products for that user. To complete this model data first has to be cleansed. No duplicates were found in the sample dataset; however, in few columns, nulls were discovered. In-order to create a larger population for the model no null where removed.

Featured selection was performed to keep all the meaningful information; in total five columns remained. The columns undergone further natural language cleaning processes to normalize and remove unwanted characters. Tags where defined as a dictionary for TFIDF

```
# Tags to be used a dictionary for TFIDF
tags_raw = [item.split() for item in clean_v3['product_category_tree']]
tags = list(set(flatten(tags_raw)))

print('Number of tags used in TFIDF:', len(tags))
```

Next the categorical tree found within the dataset needed to be converted into arrays for TFIDF. This array with a size of 20000 rows is used to give value to words found within the document.

```
# Product category array need to transform TFIDF
product_category = clean_v3['product_category_tree'].values
product_category[0:2]


Output** array([' clothing women clothing lingerie sleep swimwear shorts alisha
shorts alisha solid women cycling shorts ', ...)
```

With the defined vocabulary and a document TF-IDF was transformed and fitted to the dataset. This csr_matrix "*transform_data*" contains a numerical value of the sentences with regards to dictionary.

```
# Performing Term Frequency-Inverse Document Frequency
tfidf = TfidfVectorizer(stop_words='english',vocabulary=tags)
transform_data = tfidf.fit_transform(product_category)
```

Values from TFIDF are placed into the model mentioned above. The model fits K-NN to a feature plane, and pairs are compared. The model returns the top 10 nearest neighbors to the point in question.

```
# Model: KNearest Neighbours
# Calculating 10 nearest neighbours
model_Knn= NearestNeighbors(metric='cosine',algorithm= 'brute', n_neighbors=
10).fit(transform_data)
```

Here at random the model picks one item from the sample dataset and which it compares to the list of pairs completed by KNN. The pairs with the smallest distances are the pairs that are most similar.

```
# Randomly picks one item to put into the recommender model
# Determines the score of all selections.
rand_item =random_item(len(raw_data))
input_text = Convert(final_data.iloc[rand_item]['product_name'])
result_ratings = model_Knn.kneighbors(tfidf.transform(input_text))
```

Finally, the model displays the top 10 items that have similar traits. These top 10 items are what the model recommends to the user.

```
# Nearest Neighbour Results
print('Original Item: \n', raw_data.iloc[rand_item]['product_name'],'\n')

for item in result_ratings[1][0]:
    print('Recommended Item')
    print('Product Number:', item)
    print('Brand: ', raw_data.iloc[item]['brand'])
    print('Product Name:', raw_data.iloc[item]['product_name'])
    print('Product Retail Price:',raw_data.iloc[item]['retail_price'], ('(INR)'))
    print('Product URL:', raw_data.iloc[item]['product_url'])
    print('Tags associated with Brand: ', list(set(product_category[item].split('
'))) ,'\n')
```

```
Output**
Original Item:
 Favourite BikerZ FBZ 6LED 59 Cast Iron Front Fog Lamp Unit

Recommended Item
Product Number: 7644
Brand:  Favourite BikerZ
```

```
Product Name: Favourite BikerZ 3514 RAD air filter Ionic Air Filters For Hero HF
Dawn
Product Retail Price: 999.0 (INR)
Product URL: http://www.flipkart.com/favourite-bikerz-3514-rad-air-filter-ionic-
filters-hero-hf-dawn/p/itmehqzykrhuhnar?pid=VAFEHQZYGH5HRJJP
Tags associated with Brand:  ['', 'f', 'filter', 'rad', 'favourite', 'ionic',
'air', 'bikerz', '3514']

Recommended Item
Product Number: 15597
Brand:  Favourite BikerZ
Product Name: Favourite BikerZ FBZ WIPER BLADE 05 Windshield Wiper For Maruti Alto
Product Retail Price: 699.0 (INR)
Product URL: http://www.flipkart.com/favourite-bikerz-fbz-wiper-blade-05-
windshield-maruti-alto/p/itmegkpgymkzbv8t?pid=CWIEGKPGR69BBHZ8
Tags associated with Brand:  ['', '05', 'spares', 'performance', 'spare',
'favourite', 'fbz', 'automotive', 'wiper', 'wipers', 'w', 'bikerz', 'windshield',
'parts', 'accessories', 'car', 'blade']
```

3.4 Improvement & Future Steps

This model is a fairly simply in-terms of recommendation engines. If more time was permitted on this problem the following are suggestions to improve the solution.

Neural Networks are powerful tools Data Scientist use in the modern-world. A multi-layer word Embedding would be implemented on this model to help further accuracy. Although the computing power will greatly increase, the self-sufficient deep learning model will prove to become invaluable with an enormous amount of data.

Implementation of hierarchical classification would also be constructed. Such tools would help reject the truly negative results from the recommendation pool. An example would be, if user X was looking at plant/gardening products with traits that are also found with infant/maternity products this recommender system could offer the infant products. With hierarchical modelling, we can implement conditions that the two are not related at all, which prevents the recommender engine from recommending the infant products.

In the dataset given, column product_specification has dictionary tendencies filled with a lot of jargon. Implementing a proper feature extraction script could bring the more meaningful information to the top.

Image similarity would also be a feature that would improve the solution. Convolutional Neural Network can be implemented to qualify the degree of similarity between items. This would be identical to the similarity of the tags found on each item, just with images.

More historical data. Implementation of customer data would permit the model to become either a Collaborative or Hybrid filter. Affinity analysis can be added to further recommend items that have a history of being sold with other items, as with the knife-set example above. Connecting one purchase to the next is a crucial step to a comprehensive recommender system.

Lastly, pricing implementation would be an interesting improvement to the model. Placing similar pricings together with similar items could help the user pick future products they desire. This would also include discount/sales information. Usually products worth more are built better, respected more, or desired, so recommending more expensive products when they are near the same price of the cheaper product could incentivise purchases.