1. Introduction

1.1 overview:

Objective of this project is to understand the flavor of machine learning and artificial intelligence to solve the real world problem of recommending similar product on e-commerce web-site like Amazon

For a given product, it try to recommend similar product on real world data. Learning of linear algebra, text processing and computer vision

It is simplified but similar to real world situation.

1.2 Problem statement:

To recommend similar apparel item/product in e-commerce, we have downloaded 27K data from Amazon.com API. So we will be dealing all with Amazon product where if a query is given as title that have many features like brand, price, color, size and other product description, we have recommend similar product/item based on title that is also known as content based recommendation where if you are looking for a round neck shirt, it may recommend another round neck shirt in different – different color. It will also use an image based recommendation.

Another way that is used by Amazon is to use collaboration filtering. Suppose a user U1 checked for product p1, p2, p3 and User U2 checked for product p1, p2, p3, now if a user Un checked product p1 then there is a high chances to recommend a product p2, p3 to user Un when it is on page product p1.

But we have a only part of data, so we are using content based recommendation that involve text and image where we use brand, color, price and title

1.3 plan of action:

Set-up python

Data acquisition

Data cleaning

Text processing

Linear algebra

Text based product recommendation

Bag of words

TF-IDF

Word2vec

Image based

A/B testing

2. System set-up

2.1 system requirement:

Desktop with RAM: >= 4GB

Disk space : >=1GB

Processor : AMD or Intel

Operating system : Window, MAC, Linux

2.2 Python, Anaconda and relevant packages:

Go to the official website

Download Python

To check it, type Window+R, type cmd and run. It opens the command prompt.

Type python. If it is installed, you will be in python cell

Go to official website and download Anaconda

Launch the editor by typing “jupyter notebook” in your command prompt.

3. Data acquisition & understanding

3.1 Amazon product advertising & API:

Initially I mentioned that we will work on real world data. We used Amazon product advertising API. This page explain how to get the Amazon data using any language in a policy complain manner which is very important because we do not want to break the Amazon policy. So by using this, we get data for women’s tops and we acquire data roughly around 183K product which have so many feature/column but I primarily focus on women’s top

3.2 Overview of the data and terminology:

Since till now we have installed all the packages that we need. Let’s go with some overview and description of the data. My next job is to lead the data set. In my data terminology is as –

Dataset- dataset is a set of point

Row - row is data point or a product

Columns- column is a feature or variable for a data point

Data point- a particular product details

Hare we have 183138 data point and 19 columns or feature but we will be using only 7 important features like ASIN(Amazon standard identification number ), Brand, Color, Price, Image-url , product type name (type of the apparel, Ex: shirt, t-shirt) and title.

4. data- cleaning

4.1 missing data in various feature:

In machine learning, it is very important to cleaning the data which have some unwanted field.

Without data cleaning it’s generally overloaded. So data cleaning sometimes take hours to days to clean the data for better performance.

More understand to data

Better to design models

In our dataset we found that there are some product which do not have color, price and title. We can know it by describe function for each feature

Ex: print(data[‘brand’].describe())

Output: Count- 182987

Unique- 10577

Top- Zago

Freq- 223

Name – Brand

Here we can see that how many product do not have brand name feature

183138-182787=151 missing value.

Similarly we can do the same for other features also to know missing value.

4.2 understand and remove duplicate row:

So till now we got 28K data by cleaning our data. Because this is product recommendation system and works based on the title, so we emphasis more on title.

First we try to know the duplicate title by using Pandas dataframe.duplicated().

Now we try to print the total duplicate title-

Print(sum(data.duplicated(‘title’)))

Output: 2325

Here data.duplicated() return a Boolean variable like true or false. If title is duplicated it return true otherwise false. Finally we are summing all the true and false (0, 1).

4.3 removing duplicates: case -1

Since we have mentioned that we will be using ‘title’ very extensively throughout this and in our data, there are so many title which is very short and this short title does not play very important role. So we remove those product which have title length less than 4.

Data\_sorted=data[data[‘title’]].apply(lambda x: len(x.split())>4)]

Output: after removal of product with short description: 27949

Now we will try to remove those title which are exactly same except 2 or 3 word at the end. To do that first we have to sort the title in alphabetical order. So let’s sort the title –

Data.sorted.sort\_values(‘title’, inplace=True, ascending=False)

Till now we have sorted out our product title. Now by using nested for loop and a counter variable, perform a set difference. If difference is less than or equal to 2, remove that title.

4.4 removing duplicates: case -2

Till now we got 17593 data point. By looking on my data we found another type of duplicate title (text) in our data set.

Since we are using “title” very expensively and found that some of title are exactly same except few (2 or 3) in middle of the title.

Ex:

TITLE: 1-

86222 ultarclub women classic long sleeve oxford pink xx-large

TITLE: 2-

75002 ultarclub ladies classic long sleeve oxford pink xx-L

So at the end of this we got 16042 products

5. Text pre-processing

5.1 tokenization and stop – word removal:

This is the first text pre-processing step. Here we remove all the stop word from our title because stop word is not meaningful in our title. We can know all the stop word by importing NLTK and its useful classes and function.

Ex: 86222 ultarclub women cool of classic long sleeve oxford pink xx-large

Here “of” have no meaning.

Note that stop-word removal is not important everywhere, but in my case it adds value.

5.2 stemming:

Another text processing is stemming. In this type of text processing we try to find out root word for any word.

Let me take an example like fisher, fishing, fished but the root word of all these 3 words is fish.

6. Text based product similarity

6.1 converting text to an N-d vector (bag of words):

We will use title which is simply text. Let’s say we have T1, T2, T3 ……… Tn title which is roughly 16K product. In linear algebra, if we can represent a point as a N- dimensional point or an array of size D then I can use Euclidean distance to find similar product. For each point we will use title. If I can represent a title as a vector (N-d point) then I can apply Euclidean distance to find similar product.

There are various way to convert our title into N- dimensional point. One of the basic way to represent title (text) into a N-d vector or point is Bag-of-word technique.

Imagine we have T1, T2, T3, T4…….Tn title (text)

Step 1- from each of the word, I will create a set of words which are presented in any of the title i.e. take all the words from all title and make a set S= set of all words present in T1, T2, T3, T4…….Tn. suppose the size of set is D i.e. D unique words. In our data there are 12609 unique words.

Step 2- I will take any title Ti from our titles. For each of our title we will make a vector of d-dimensional.

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | ………. | ……… | ………. | d |

Ti=

**D – Dimensional vector**

Now for Ti, if W1 is present in Ti then we put 1 in vector, if w2 is not present, we put 0 in vector. If w3 is presented 3 times then we put 3 in vector. I will do that for every word of my set.

Ex: TITLE- zago ladies t-shirt long sleeve t-shirt XL

L t-shirt XL sort zago sleeve women ladies …………………… D word

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 2 | 1 | 0 | 1 | 0 | 0 | 1 | ……….. | ……. |

**Converting a title Ti (text) into N- d vector as bag-of-word**

In bag of word, we are putting all word in a bag by counting how many times a word is presented in a title Ti. So it completely discard the sequence and its simply counting for word.

6.2 Tf-idf based text vectorization:

TF- term frequency

IDF- inverse document frequency

Assume – T1, T2, T3, ……………………………..Tn.

Here each title is known as document and the collection of all document for T1, T2, T3………. Tn is known as corpus (relation) or data corpus.

So TF for a particular title or document in title Ti (wi, Ti) = (# Wj occurs in Ti)/ (#words in Ti).

TF is large if Wj occurs multiple times in Ti. I can say then TF is measure for a given word in a given title with set of words.

For IDF, it apply for whole data corpus. So for any word Wj in data corpus D (where D= T1, T2, T3…….Tn)-

IDF (Wj, D) =log ((#document in D) / (#document in D that contain Wj)).

So basically IDF measure how rare a word is present in my data corpus. I can say that IDF is high if Wj is rare(less) in D.

Finally put both concept TF and IDF together for each of word in our set S:

W1=TF (W1, Ti)\*IDF (W1, D)

W2=TF (W2, Ti)\*IDF (W2, D)

W3=TF (W3, Ti)\*IDF (W3, D)

………

……..

……..

W1=TF (Wn, Ti)\*IDF (Wn, D)

So TF-IDF is much more powerful than bag-of –word because it gives larger value for less frequency words in D. It returns a sparse matrix which contain 0 or 1.

7. Text semantics base product

7.1 average word2vec:

Till now we used bag-of-word of TF-IDF which are frequency based text processing but they do not account for semantic words.

Semantic mean here that Zebra is semantically related to stripes and tiger is related to leopard.

Now task is to convert our title into vector which is N-d array by a new technique as “word2vec”. It is invented by Google in 2013. It is very powerful and recent.

Suppose we have title Ti that have K words, then I will create a vector for every word of size 300(word2vec len). After this I will sum all vector index wise then divide each sum by K to find out average word2vec for a title Ti. In this way we can do for all title.

7.2 TF-IDF weighted word2vec:

We have D= (T1, T2, T3…………Tn). Suppose T1 have 5 word W1, W2, W3, W4, W5 i.e. K=5.

Step-1: compute TF-IDF for each word and multiply it by Wi vector.

Step-2: add all the vector for W1, W2, W3, W4 and W5, get a vector of sum of all.

Step-3: add up all the TF-IDF. Let’s say N.

Step -4: multiply 1/N with the vector which I get into step-2

So word word2vec have word importance as well as semantic similarity, not just only word frequency like bag of word.

7.3 weighted similarity using brand & color:

Till now we just use only title for product similarity. For title similarity we created a vector of size 300 by using word2vec for a particular product Pi. Similarly for each product Pi, we created a vector Bi for Brand and Ci for color.

Suppose we have brand B1,B2, B3,………Bm among all product and for product Pi, have brand Bi, then in brand vector of size M, I will put 1 for brand Bi and put 0 in M-1 places in brand vector using one hot encoder.

Similarly I will do for color vector too. Now we have 3 vector as title, brand and color. I will combine these three vectors into a single vector.

So for any product Pj, I will calculate Euclidean distance using the combined vector to find the similar product.

There in another case suppose a customer wants the similar product based on brand or color, then we have to modify our Euclidean distance into weighted Euclidean distance. Three different weighted are there as like Wt for title, Wc for color and Wb for brand. I will multiply each element from title to Wt, multiply each element from brand to Wb, and multiply each element from color to Wc. After multiplying, take the Euclidean distance that is equal to our weighted Euclidean distance.

7.4 building real world problem:

Till now we have done bag-of-word, TF-IDF, weighted word2vec and weighted but in reality most companies used multiple technique because there is no single technique that perform best.

So in real world, they apply all technique and combine the result of all technique to get better result. This is also known as business rule. For a single problem, there may be more than one solution. To find out which one is better we do A/B testing

8. Deep learning based visual product similarity

8.1 how to featurize an image:

Till now primarily we used text feature which are available in product like title, color and brand by using different method.

Now in image based, given an image URL, curtail part is how to featurize an image. So again we convert an image into n-d vector. In an image, various feature may present like edges, shape, shape color and pattern. For a given image, we will convert it into N-d vector in which some part of vector present pattern, some will present shape and edges and other feature into a single vector.

Once we converted all feature into a single vector for an image, I can apply Euclidean distance for similarity.

To get this vector till 2012, lot of researcher try to convert all the feature into a vector. But since 2012, deep learning comes to automatically convert this image into D-dim vector form such that it account all the feature like color, pattern and edges using convolutional neural network (CNN) technique.

8.2 using keras + tensorflow to extract feature:

Earlier we mentioned that we are using CNN to convert an image into N-d vector. There are various way to convert it. I am using VGG-16 algorithm which is design by Oxford University.

To compute the vector we use tensorflow and keras library. Tensorflow is a deep learning library where keras is a open source library.

So here VGG-16 take an image as input and return a N-d (25008) vector which is a dense vector.

9. A/B testing

Now days mostly all companies like Amazon, Flipchart do not use a single solution for better performance. Even if they have multiple solution, the question arise that which one solution would be best? To short such type of problem they use A/B testing

A/B testing is a technique to find out the best solution. In A/B testing, suppose we have a bunch of user. We will divide all those user into two different group without overlapping to each other. Now I will use solution one for first user group U1 and use solution second for second user group U2. By analyzing the sale of each group, we can decide the better solution. If user U1 have purchased much more than user U2 by using first solution, then we can say that our first solution is better than second. This is the fundamental technique that A/B testing uses.