# Similarity Based Recommender System

Dataset Used - Amazon US Gift Card Reviews \n Link - https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Gift_Card_v1_00.tsv.gz (https://s3.amazonaws.com/amazon-reviews-pds/tsv/amazon_reviews_us_Gift_Card_v1_00.tsv.gz)

The dataset contains several parameters:

1. Marketplace - 2 letter country code of the marketplace where the review was written.
2. Customer id - Random identifier that can be used to aggregate reviews written by a single author.
3. Review id - The unique ID of the review.
4. Product id - The unique Product ID the review pertains to. In the multilingual dataset the reviews for the same product in different countries can be grouped by the same product_id.
5. Product parent - Random identifier that can be used to aggregate reviews for the same product.
6. Product title - Title of the product.
7. Product category - Broad product category that can be used to group reviews (also used to group the dataset into coherent parts).
8. Star rating - The 1-5 star rating of the review.
9. Helpful votes - Number of helpful votes.
10. Total votes - Number of total votes the review received.
11. Vine - Review was written as part of the Vine program.
12. Verified Purchase - The review is on a verified purchase.
13. Review headline - The title of the review.
14. Review body - The review text.
15. Review Date - The date the review was written.

In [1]:
```
import gzip
import csv
from collections import defaultdict
```

In [3]:
```
path = 'amazon_reviews_us_Gift_Card_v1_00.tsv.gz'
file = gzip.open(path,"rt")
reader = csv.reader(file, delimiter="\t")
headers = next(reader)
print("features of data - ",headers)
```

```
features of data -  ['marketplace', 'customer_id', 'review_id', 'produc
t_id', 'product_parent', 'product_title', 'product_category', 'star_rat
ing', 'helpful_votes', 'total_votes', 'vine', 'verified_purchase', 'rev
iew_headline', 'review_body', 'review_date']
```

In [4]:
```
# Cleaning the dataset
dataset = []
for obj in reader:
    ordered_obj = dict(zip(headers,obj))
    fields1 = ["helpful_votes","star_rating","total_votes"]
    for field in fields1:
        ordered_obj[field] = int(ordered_obj[field])
        fields2 = ["vine","verified_purchase"]
    for field in fields2:
        if ordered_obj[field] == "Y":
            ordered_obj[field] = True
        else:
            ordered_obj[field] = False
    dataset.append(ordered_obj)
```

In [5]:
```
dataset[0]
```

Out[5]:
```
{'marketplace': 'US',
 'customer_id': '24371595',
 'review_id': 'R27ZP1F1CD0C3Y',
 'product_id': 'B004LLIL5A',
 'product_parent': '346014806',
 'product_title': 'Amazon eGift Card - Celebrate',
 'product_category': 'Gift Card',
 'star_rating': 5,
 'helpful_votes': 0,
 'total_votes': 0,
 'vine': False,
 'verified_purchase': True,
 'review_headline': 'Five Stars',
 'review_body': 'Great birthday gift for a young adult.',
 'review_date': '2015-08-31'}
```

In [7]:
```
# Mapping users to items that users have purchased and mapping items to
 users who purchased said items
usersPerItem = defaultdict(set)
itemsPerUser = defaultdict(set)
itemNames = {}
itemCategory = {}

for obj in dataset:
    item = obj['product_id']
    user = obj['customer_id']
    usersPerItem[item].add(user)
    itemsPerUser[user].add(item)
    itemNames[item] = obj['product_title']
    itemCategory[item] = obj['product_category']
```

In [8]:
```
# Implementing Jaccard Similarity Function
def jacard(set1,set2):
    numerator = len(set1.intersection(set2))
    denominator = len(set2.union(set2))
    return numerator/denominator
```

In [11]:
```
# similarity function for finding similarity dataset for a purchased ite
m
def similarity(item):
    similarities = []
    itemUsers = usersPerItem[item]
    comparedItems = set()
    for user in itemUsers:
        comparedItems = comparedItems.union(itemsPerUser[user])
    for comparedItem in comparedItems:
        if comparedItem == item:
            continue
        comparedItemUsers = usersPerItem[comparedItem]
        similarity = jacard(itemUsers,comparedItemUsers)
        similarities.append((similarity,comparedItem))
    similarities.sort(reverse=True)
    return similarities[:10]
```

In [12]:
```
query = dataset[1]
item = query['product_id']
similar = similarity(item)
similar_names = [itemNames[x[1]] for x in similar]
similar_categories = [itemCategory[x[1]] for x in similar]
print("Names of similar items - ",similar_names)
print("Categories of similar items - ",similar_categories)
```

```
Names of similar items -  ['Amazon Gift Card - Facebook - Four Seasons
(Animated) [American Greetings]', 'Amazon Video Gift Card - Facebook -
Crazy Congrats Song', 'Amazon.com Gift Cards, Pack of 20 (Various Card
Designs)', 'Amazon.com Gift Card in a Greeting Card (Various Designs)',
'Amazon Gift Card - Print - Feliz Dia del Padre (Argyle)', 'Amazon.com
Gift Card in a Birthday Cupcake Tin (Birthday Cupcake Card Design)', 'A
mazon Gift Card - Facebook - Smile!', 'Amazon Gift Card - Facebook - Ha
ppy Chanukah', 'Amazon Gift Card - Print - Birthday (Libra: Sep. 23-Oc
t. 22)', 'Amazon Gift Card - Print - Amazon Sports and Outdoor']
Categories of similar items -  ['Gift Card', 'Gift Card', 'Gift Card',
'Gift Card', 'Gift Card', 'Gift Card', 'Gift Card', 'Gift Card', 'Gift
Card', 'Gift Card']
```

## Analysing Jaccard similarity output

1. Output shows 10 most similar items in decreasing order of jaccard index
2. Product names show pattern in similarity in terms of gift card usage and product category
3. No MSE as there isnt any expected output, we are identifying patterns and these items are similar because many users are in common

In [ ]:
```
```