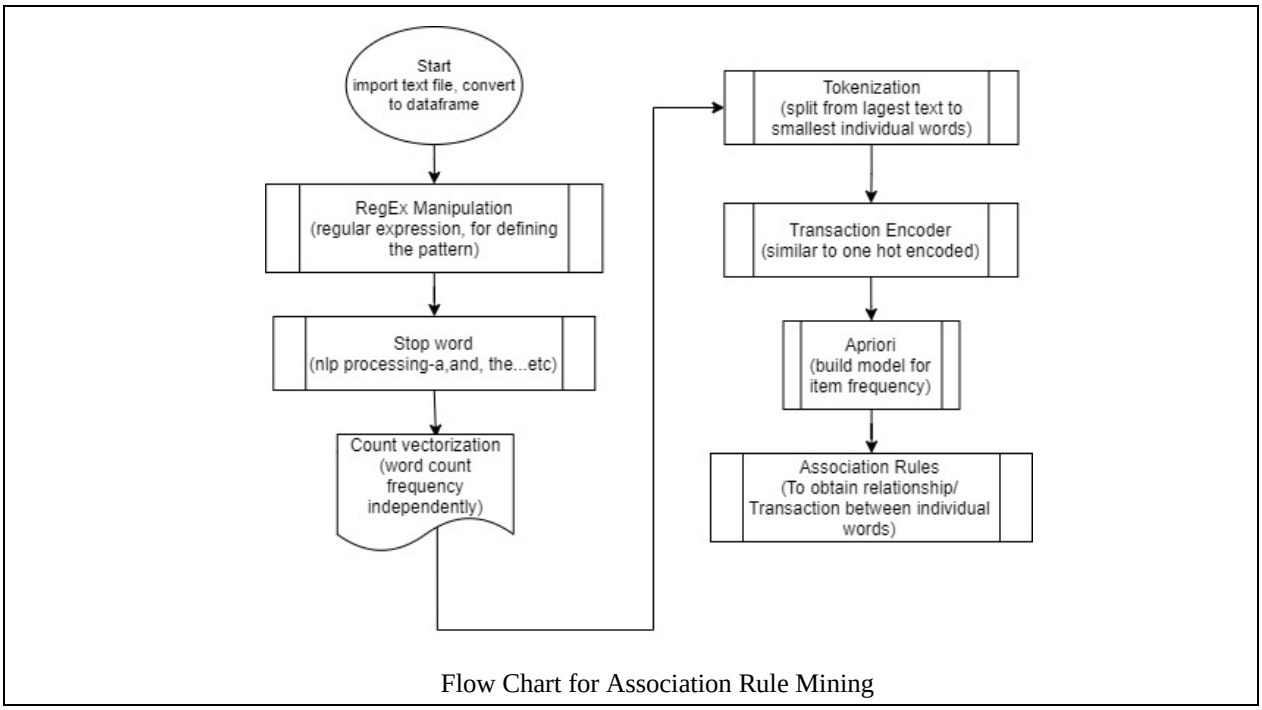
Task : Association Rules Mining

Problem Statement

Association Rules Mining is a machine learning model which observe the frequent occurring patterns that can be found in the database transaction or any relational data set. Association rule can be conceptually divided into two part such that antecedent(if) and consequent(then). A consequent is an item result that combine all the antecedent patterns. While antecedent means that something can be found in the data uniquely. Association rule mining is depending on the three parameters relationship. The relationships are support, confidence, and lift. Support represents how frequently or how much historical data support the rule such that if/then or (A→ B) appear in data set and the sequent not count. Confidence shows the how many times or how strong the relationship in both items can be found in the data set such that number of appearing B transaction that contain A. Next, lift is ratio of confidence and support. It is dividing the confidence by support of the probability. Hence, the relationship and item frequent are involved in the Apriori machine algorithm. It is commonly used to recommend products based on the customer previous or current selection.

However, the task given to find a text document for implementing the association rule mining. We obtain text file from *Kaggle website: https://www.kaggle.com/marklvl/sentiment-labelled-sentences-data-set.* The data is a collection of movie feedback from the reviewers. In the task 2, we try to build an association rules model to obtain the important transaction of each character in the text file. We analysis the transaction result to determine the movie feedback whether is good or bad review. First, we apply Natural Language Processing Tool to clean the data for reducing the data complexity. Apply Tokenization for splitting the large document into small independent words. We apply transaction encoder to transform the all the unique transaction words into True or False {0,1} value. The first model is Apriori algorithm for obtaining the combination of frequent itemset. Thus, we define matric either “support”, “confidence” or “lift” on the association rules to obtain best fit model. Below is the flow chart of Association Rules Mining.



Flow Chart for Association Rule Mining

Text Pre-Processing

Import Library

```
import pandas as pd
import numpy as np

# natural language toolkit
import nltk
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('averaged_perceptron_tagger')
nltk.download('punkt')

# common meaningless english word --> the, an, of
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop = stopwords.words('english')

from sklearn.feature_extraction.text import CountVectorizer

import re

import matplotlib.pyplot as plt
import matplotlib as mpl
import os
```

- Import nltk library. NLTK is natural language toolkit which is a set of human language of text processing library toolkit.
- Import stopwords for removing the common English word such that a, and the etc.
- Import word_tokenize for splitting larger document into small independently words
- Import CountVectorizer to convert all the text documents into vector form with counting.

- Import re is regular expression of natural language processing to replace special syntax.

Text to DataFrame

```
data = pd.read_csv('imdb_labelled.txt',delimiter = "\t",sep=" ", header=None)
data = data.rename(columns = {0:'review', 1:'event'})

# 1st step cleaning data, obtain text feature and id label
label = data.iloc[:,1]
feature = data.iloc[:,0]
data
```

	review	event
0	A very, very, very slow-moving, aimless movie ...	0
1	Not sure who was more lost - the flat characte...	0
2	Attempting artiness with black & white and cle...	0
3	Very little music or anything to speak of.	0
4	The best scene in the movie was when Gerardo i...	1
...	...	...
743	I just got bored watching Jessice Lange take h...	0
744	Unfortunately, any virtue in this film's produ...	0
745	In a word, it is embarrassing.	0
746	Exceptionally bad!	0
747	All in all its an insult to one's intelligence...	0

748 rows × 2 columns

- We import data and convert the text file to DataFrame in Jupyter noted book.
- Apply delimiter parameter to separate the tab “\t”.
- Assign the data column into target and feature separately.

RegEX Processing

```
def nlp_process(processed_feature):

    # symptom comes from RegEX study
    for sentence in range(0, len(feature)):

        # Remove all the special characters
        processed = re.sub(r'\W', ' ', str(feature[sentence]))

        # Converting to Lowercase
        processed = processed.lower()

        # Remove digits
        processed = re.sub('\d+',' ', processed)

        # remove all single characters
        processed= re.sub(r'\s+[a-zA-Z]\s+', ' ', processed)

        # Remove single characters start from 1st characters
        processed = re.sub(r'\^[a-zA-Z]\s+', ' ', processed)
```

- RegEx call as Regular Expression. It is a key concept of Natural Language Processing (NLP). NLP is an applications service that able to understand human language.

- RegEX is important part of pre-processing in text data. It is mainly used to defines the pattern of characters in complex string format
- Re.sub function use for replacing the abnormal string pattern to common string format such that replace the symbol with empty space or replace the number with empty space.

Stop Word

```
data['new_phrase'] = pd.DataFrame(pd.DataFrame(review_list)
                                  .apply(lambda x: [item for item in x if item not in stop]))
#restructure the table columns
data = data.reindex(columns = ['event','review', 'new_phrase'])

# replace empty '' to nan
data = data.replace(r'^\s*$', np.nan, regex=True)

# drop nan & drop review column
data = data.dropna()
data = data.drop(columns=['review'])

data
#data.to_csv('text_data.csv')
```

	event	new_phrase
0	0	slow moving aimless movie distressed drifting ...
1	0	sure lost flat characters audience nearly half...
2	0	attempting artiness black white clever camera ...

- We assign new column without stop word component and name the column as new_phrase.
- Stop word is one of the NLP processing to remove most common English words such as want, to, the, this and so on.
- .apply function provides to pass all the value into DataFrame. Lambda function uses the argument with only one expression or condition.
- Reindex is used for restructuring the index of data column.

Tokenization

```
#list to list
y = list(data.apply(lambda x: nltk.word_tokenize(x['new_phrase']),axis=1))
y
```

```
[['slow',
  'moving',
  'aimless',
  'movie',
  'distressed',
  'drifting',
  'young',
  'man'],
 ['sure',
  'lost',
  'flat',
  'characters',
  'audience',
  'nearly',
  'half',
  'walked'],
```

- nltk.word_tokenize commonly used to determine or extract the single words from sentence. Word_tokenize is one of the text processing functions in natural language toolkit.
- Splits the largest text, phrase or sentence into smaller individual words.
- Apply all the words into list-to-list form.

```
from mlxtend.preprocessing import TransactionEncoder
from mlxtend.frequent_patterns import apriori, association_rules

te = TransactionEncoder()
te_array = te.fit(y).transform(y)
df = pd.DataFrame(te_array, columns=te.columns_)
df
```

	aailiyah	abandoned	ability	abroad	absolutely	abstruse	abysmal	academy	accents	accessible	...
0	False	False	False	False	False	False	False	False	False	False	...
1	False	False	False	False	False	False	False	False	False	False	...
2	False	False	False	False	False	False	False	False	False	False	...
3	False	False	False	False	False	False	False	False	False	False	...
4	False	False	False	False	False	False	False	False	False	False	...
...	...	...	...	...	...	...	...	...	...	...	...
741	False	False	False	False	False	False	False	False	False	False	...

- Mlxtend is machine learning extension library tools.
- Import Apriori function to extract the frequent item sets for association rule mining
- Import TransactionEncoder to labels the data from set of string characters into unique labels. It is used to keep track the transaction data in python list.
- We transform the transaction into a one-hot encoded and convert it into DataFrame.
- Each column will consist of each transaction with True or False value.

Build Apriori Model

```
# Build Model

frequent_item = apriori(df, min_support = 0.015, use_colnames = True,verbose = 1,low_memory=True)
frequent_item = frequent_item.sort_values(by='support', ascending=False)
frequent_item
```

```
Processing 36 combinations | Sampling itemset size 2
```

	support	itemsets
32	0.178284	(movie)
17	0.168901	(film)
37	0.075067	(one)
5	0.071046	(bad)
21	0.060322	(good)
...	...	...
45	0.016086	(right)
46	0.016086	(saw)

We build the Aprior model by initializing the transaction “df” DataFrame with min support threshold 0.015, and use_colnames is True for returning the data into DataFrame. We apply sort index from largest to smallest for the support probabilities. From the Aprior model result shows that there was 36 combination of group characters and the sampling itemset was maximun 2 or pair of itemsets. The highest probability of frequent item set was movie with 17.8%. The lowest support probability of frequent item set is (waste, time). Hence, we would process the Aprior model to association rule model.

```
rules = association_rules(frequent_item, metric="lift", min_threshold = 1)
rules.head(10)
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(like)	(film)	0.048257	0.168901	0.020107	0.416667	2.466931	0.011957	1.424741
1	(film)	(like)	0.168901	0.048257	0.020107	0.119048	2.466931	0.011957	1.080356
2	(bad)	(movie)	0.071046	0.178284	0.020107	0.283019	1.587459	0.007441	1.146077
3	(movie)	(bad)	0.178284	0.071046	0.020107	0.112782	1.587459	0.007441	1.047042
4	(one)	(movie)	0.075067	0.178284	0.018767	0.250000	1.402256	0.005383	1.095621
5	(movie)	(one)	0.178284	0.075067	0.018767	0.105263	1.402256	0.005383	1.033749
6	(seen)	(one)	0.029491	0.075067	0.018767	0.636364	8.477273	0.016553	2.543566
7	(one)	(seen)	0.075067	0.029491	0.018767	0.250000	8.477273	0.016553	1.294013
8	(one)	(movies)	0.075067	0.029491	0.018767	0.250000	8.477273	0.016553	1.294013
9	(movies)	(one)	0.029491	0.075067	0.018767	0.636364	8.477273	0.016553	2.543566

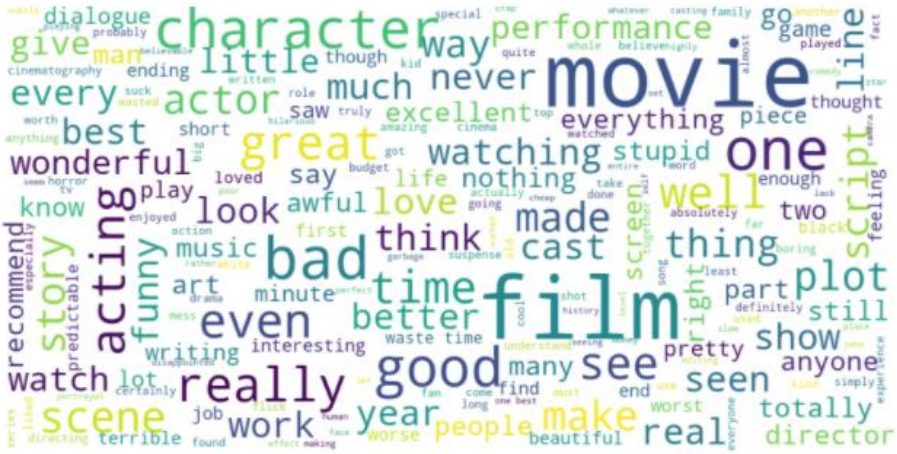
In the association rule model, we set the metric as lift method and minimum threshold with 1. Metric provides the evaluation the association rule by different parameter such that support, confidence and lift. The reason we choose lift because we wish to get how much confidence has increased from consequent result (B) will be rated by antecedent (A) reviewer. By looking at the output above, clearly the highest confidence is at 63.64% where both for rule {seen -> one} and {movies -> one}. However, they are appearing frequently across all the transactions, where the association could simply be a fluke. With the known lift value to be 8.47 means that they are positively correlated. On the other hand, the rule {like -> film} and {bad -> movie} are having the support higher than the rest in the table, where it indicates the rule are often occurs in the transaction.

```
rules[(rules['confidence']>= 0.4) & rules['support']>=0.015]
```

	antecedents	consequents	antecedent support	consequent support	support	confidence	lift	leverage	conviction
0	(like)	(film)	0.048257	0.168901	0.020107	0.416667	2.466931	0.011957	1.424741
6	(seen)	(one)	0.029491	0.075067	0.018767	0.636364	8.477273	0.016553	2.543566
9	(movies)	(one)	0.029491	0.075067	0.018767	0.636364	8.477273	0.016553	2.543566
16	(made)	(film)	0.029491	0.168901	0.016086	0.545455	3.229437	0.011105	1.828418
18	(great)	(film)	0.040214	0.168901	0.016086	0.400000	2.368254	0.009294	1.385165
22	(waste)	(time)	0.017426	0.050938	0.016086	0.923077	18.121457	0.015198	12.337802

Furthermore, by filtering out the confidence with percentage higher than 40% and we see that the highest consequent support is around 2.0 % which is the rule {like -> film}. The following transaction shows they are occuring frquently among all the groups. We may think of the meaning of like as it is a good movie or even a conjunction to give an example. Last but not least, we look at the case 18 and 22, we have rule {waste -> time} and {great film} with confidence of 92.3% and 40.0% simultaneously. In the fact that the lift of rule {waste -> time} to be 18.121, and we know that it is highly correlated. As a conclusion, we would know that it is an bad movies when we have the word transaction rule with {waste -> time}.

Example visualization



End