Introduction

Sentiment Analysis

“Sentiment Analysis is the interpretation and classification of emotions (positive, negative, or neutral) within data using text analysis techniques. It allows business to identify customer sentiment towards products or services in online conversations and feedback.” (Tran, et al. 2022).

The entertainment and multimedia sector occupies a large share of the market today, and an important big part of this sector goes to movie production.

There are a group of websites that offer platforms dedicated to review the movies, such as IMDB and Rotten Tomatoes, and these websites also provide a list of preferences for films in addition to many other services.

The task

In this research, I will do simple partial sentiment analysis to evaluate a sample of movies that share the same genre using the positive and negative reviews dataset of Rotten Tomatoes. The study could give a glance at the reaction of the population towards a specific type of film and that could be used to achieve two main goals: the first goal is to nominate some other movie names from the same genre, and the other goal is to help the people in the movie production industry to understand the reasons of success or failure of a movie and encourage them to focus on the formula of the success. I applied the unsupervised learning method (word2vec) to analyze 3 movies from the (organized crime) movies genre and those are (The Godfather, The Goodfellas, and Scarface).

Dataset and preprocessing

I chose to study the comments of the reviewers from the Rotten Tomato website, and I used the data set provided at Github.com (https://github.com/nicolas-gervais/rotten-tomatoes-dataset) published by the author Nicolas Gervais.

I started with loading the number of required packages such as nltk, genism, and panda, then added a function to recognize the punctuation in the text, after that used the panda package to call, read and process the dataset file. I examined the file reading function by obtaining the first and last 10 records of the dataset file.

After that, I converted the text to lower case so I can get rid of all the capitalized or upper case text, and get the source text unified. Then did the tokenization to split the text into pieces of words (Unigram), ignore too short or too long tokens, and create unique words dictionary.

The method

The next function was the creation of an embedding model from text using word2vec method in genism. The word2vec method algorithm is a technique used to process the natural languages, and it can learn the relationships of the words of a large text source, then generate synonyms and vocab suggestions or solutions as needed.

Then tried to create a bag of words to measure the times a specific word was repeated, but Jupyter was either collapsing or generating the message of (IOPub data rate exceeded) **(Fig.08, Appendix 01)**, after a long time of trying to solve this issue, I decided to go with another solution which is writing a function to search within the original file and get the times of a specific word occurrence (in both upper and lower cases) and this function worked perfectly. After that, I tried to explore the text and find the similarity between the negative and positive words.

I searched and brought some of the most common negative and positive adjectives and added them to a function that measures the similarity of a movie name alongside the adjective and provides a value of similarity. Here is a sample of the list of adjectives that I used within the function ('weak', 'wacky','stupid','silly' ,'average', 'brilliant','charismatic','charming','clever').

First, I used the words similarity values for each of the selected movies by calculating the sum of the values of the positive adjective then extracting the average value, and did the same with the negative values.

Second, I extracted the minimum and maximum of each of the positive and negative values. Third I tried to find the times that each of the movie names mentioned in the dataset. Finally, I generated some visualizations for the results.

Findings

The similarity of the adjectives towards each of the movies, after calculating the average of the positive and negative similarity in addition to the calculation of the minimum and maximum values, and the movie name repetition times within the reviewer comments are as follows:

The Godfather

	Average	Min	Max
Positive	0.111024035	-0.014134126	0.47434053
Negative	0.01386544	-0.089685224	0.11394882

Times of Repetition 142

The Goodfellas

	Average	Min	Max
Positive	0.125275256	-0.000528622	0.34355572
Negative	0.098974584	-0.033701137	0.20421866

Times of Repetition 70

Scarface

	Average	Min	Max
Positive	0.117413777	-0.004182309	0.36156192
Negative	0.0722755	-0.008026212	0.23615609

Times of Repetition 47

Results

It is noticeable from the up-mentioned result that the chosen movies achieved a high value of similarity regarding positive reviews compared to the negative reviews. The movie name repetition in the review comments shows that The Godfather movie was mentioned (147 times), the Goodfellas (70 times) while Scarface movie achieved the least value of mentioning (47 times).

The results show that the godfather movie could be used as a major guide or measuring reference for movie recommendations from the same genre so whoever shows some positive review or reaction

towards the Godfather movie could be provided with a list of movies from the same classification such as The Goodfellas, etc.

The results also reflect the success of the Godfather movie formula, which could be followed as a guide in the production of any future movies.

This was a quick study based on limited knowledge of the tools, however, it could be applied to wider genres or classifications in the future and get more accurate to build a robust recommendation system based on sentiment analysis.

Reference:

Tran, D.D., Nguyen, T.T., Dao, T. (2022). *Sentiment Analysis of Movie Reviews Using Machine Learning Techniques. Proceedings of Sixth International Congress on Information and Communication Technology*. Lecture Notes in Networks and Systems, vol 235. Springer, Singapore. https://doi.org/10.1007/978-981-16-2377-6_34

Appendix 01

Create a word embedding model and explore patterns in the source dataset text- with PANDAS

In [1]:
```
# Import packages
```

In [2]:
```
import nltk
import re
import gensim
import csv
import nltk.tokenize
import pandas as pd
nltk.download('punkt', quiet=True)
from sklearn.manifold import TSNE
import numpy as np
import pandas as pd
```

In [3]:
```
#FUNCTION TO RECOGNISE PUNCTUATION
PUNCT_RE = re.compile(r'[^\w\s]+$')
def is_punct(string):
    return PUNCT_RE.match(string) is not None
```

USING PANDAS TO READ AND PROCESS TEXT

```
In [4]: df = pd.read_csv("F:\RottenTomato.csv", encoding='utf-8')
```

```
In [5]: #Returns the first N rows

        df.head(10)
```

Out[5]:

	Freshness	Review
0	fresh	Manakamana doesn't answer any questions, yet ...
1	fresh	Wilfully offensive and powered by a chest-thu...
2	rotten	It would be difficult to imagine material mor...
3	rotten	Despite the gusto its star brings to the role...
4	rotten	If there was a good idea at the core of this ...
5	rotten	Gleeson goes the Hallmark Channel route, dama...
6	fresh	It was the height of satire in 1976: dark as ...
7	rotten	Everyone in "The Comedian" deserves a better ...
8	rotten	Actor encourages grumpy Christians to embrace...
9	fresh	Slight, contained, but ineffably soulful.

Fig.01

```
In [6]: #Returns the last N rows
        df.tail(10)
```

Out[6]:

	Freshness	Review
479990	fresh	Ryan Coogler's sequel follows a "Rocky" road
479991	fresh	Disney brilliantly executed another film in t...
479992	rotten	Director John Crowley overestimates the comed...
479993	rotten	Here's a sobering thought: If every war gets ...
479994	rotten	Roland Joffe's deeply ridiculous movie is cau...
479995	rotten	Zemeckis seems unable to admit that the motio...
479996	fresh	Movies like The Kids Are All Right -- beautif...
479997	rotten	Film-savvy audiences soon will catch onto Win...
479998	fresh	An odd yet enjoyable film.
479999	fresh	No other animation studio, even our beloved P...

Fig.02

```
In [7]: #CONVERT STRINGS TO LOWERCASE

        reviews = df['Review'].str.lower()
```

```
In [8]: reviews

Out[8]: 0          manakamana doesn't answer any questions, yet ...
        1          wilfully offensive and powered by a chest-thu...
        2          it would be difficult to imagine material mor...
        3          despite the gusto its star brings to the role...
        4          if there was a good idea at the core of this ...
                                       ...
        479995     zemeckis seems unable to admit that the motio...
        479996     movies like the kids are all right -- beautif...
        479997     film-savvy audiences soon will catch onto win...
        479998                          an odd yet enjoyable film.
        479999     no other animation studio, even our beloved p...
        Name: Review, Length: 480000, dtype: object
```

Fig.03

```
In [9]: #LIST THE VALUES IN COLUMN

        reviews = df['Review'].str.lower().values
        reviews

Out[9]: array([" manakamana doesn't answer any questions, yet makes its point: nepal, like the rest of our planet, is a picturesque but
        far from peaceable kingdom.",
               " wilfully offensive and powered by a chest-thumping machismo, but it's good clean fun.",
               ' it would be difficult to imagine material more wrong for spade than lost & found.',
               ...,
               " film-savvy audiences soon will catch onto winterbottom's attempts to earn neo-noir credentials as patel and apte's cha
        racters go on the run. patel overdoes his character's pragmatic stoicism, and i eventually tired of the waiting game...",
               ' an odd yet enjoyable film.',
               ' no other animation studio, even our beloved pixar, can quite replicate the magic, wonder and weirdness [of studio ghib
        li].'],
              dtype=object)
```

Fig.04

```
In [11]: # Tokenise text ()
         reviews = [nltk.word_tokenize(review) for review in reviews]

In [12]: reviews

Out[12]: [['manakamana',
           'does',
           "n't",
           'answer',
           'any',
           'questions',
           ',',
           'yet',
           'makes',
           'its',
           'point',
           ':',
           'nepal',
           ',',
           'like',
           'the',
           'rest',
           'of',
           'our',
```

Fig.05

```
In [13]: #Import gensim's packages

         from gensim.utils import simple_preprocess
         from gensim import corpora
         #convert document into list of lowercase tokens, ignore too short or too long tokens, create unique words dictionary
         dictionary =  corpora.Dictionary(simple_preprocess(line, deacc = True) for line in open ('F:\RottenTomato.csv'))

         print (dictionary.token2id)

         {'freshness': 0, 'review': 1, 'answer': 2, 'any': 3, 'but': 4, 'doesn': 5, 'far': 6, 'fresh': 7, 'from': 8, 'is': 9, 'its':
         10, 'kingdom': 11, 'like': 12, 'makes': 13, 'manakamana': 14, 'nepal': 15, 'of': 16, 'our': 17, 'peaceable': 18, 'picturesqu
         e': 19, 'planet': 20, 'point': 21, 'questions': 22, 'rest': 23, 'the': 24, 'yet': 25, 'and': 26, 'by': 27, 'chest': 28, 'cle
         an': 29, 'fun': 30, 'good': 31, 'it': 32, 'machismo': 33, 'offensive': 34, 'powered': 35, 'thumping': 36, 'wilfully': 37, 'b
         e': 38, 'difficult': 39, 'for': 40, 'found': 41, 'imagine': 42, 'lost': 43, 'material': 44, 'more': 45, 'rotten': 46, 'spad
         e': 47, 'than': 48, 'to': 49, 'would': 50, 'wrong': 51, 'brings': 52, 'despite': 53, 'discovery': 54, 'gusto': 55, 'hard': 5
         6, 'hector': 57, 'on': 58, 'ride': 59, 'role': 60, 'shotgun': 61, 'star': 62, 'voyage': 63, 'an': 64, 'arson': 65, 'at': 66,
         'bad': 67, 'been': 68, 'buried': 69, 'core': 70, 'dog': 71, 'film': 72, 'flatulence': 73, 'idea': 74, 'if': 75, 'in': 76, 'j
         okes': 77, 'pile': 78, 'plot': 79, 'puns': 80, 'related': 81, 'ridiculous': 82, 'serial': 83, 'there': 84, 'this': 85, 'unsi
         ghtly': 86, 'was': 87, 'channel': 88, 'curious': 89, 'damaging': 90, 'entry': 91, 'gleeson': 92, 'goes': 93, 'hallmark': 94,
         'intermittently': 95, 'route': 96, 'subgenre': 97, 'time': 98, 'travel': 99, 'absurd': 100, 'as': 101, 'close': 102, 'dark':
         103, 'era': 104, 'height': 105, 'hell': 106, 'jerry': 107, 'nowhere': 108, 'objective': 109, 'patently': 110, 'reality': 11
         1, 'satire': 112, 'somewhere': 113, 'springer': 114, 'surely': 115, 'surpassed': 116, 'better': 117, 'comedian': 118, 'deser
         ves': 119, 'everyone': 120, 'movie': 121, 'actor': 122, 'christians': 123, 'embrace': 124, 'encourages': 125, 'grumpy': 126,
         'season': 127, 'contained': 128, 'ineffably': 129, 'slight': 130, 'soulful': 131, 'addresses': 132, 'angle': 133, 'approachi
         ng': 134, 'bell': 135, 'both': 136, 'conceivable': 137, 'debut': 138, 'enhancement': 139, 'every': 140, 'feature': 141, 'hum
         or': 142, 'intelligence': 143, 'issue': 144, 'performance': 145, 'subject': 146, 'with': 147, 'achieves': 148, 'almost': 14
         9, 'considerable': 150, 'director': 151, 'entirely': 152, 'except': 153, 'halloween': 154, 'irresistibly': 155, 'means': 15
         6, 'music': 157, 'power': 158, 'score': 159, 'shamelessly': 160, 'through': 161, 'visual': 162, 'zingy': 163, 'action': 164,
```

Fig.06

```
In [26]: #get file object reference to the file
         file = open("F:\RottenTomato.csv", "r")

         #read content of file to string
         data = file.read()

         #get number of occurrences of the substring in the string
         occurrences1 = data.count("scarface")
         occurrences2 = data.count("Scarface")
         print('Number of occurrences of the word :', occurrences1)
         print('Number of occurrences of the word :', occurrences2)

         Number of occurrences of the word : 0
         Number of occurrences of the word : 47
```

Fig.07

In [14]:
```
#CREATE BAG OF WORDS FROM THE TOKENS FILE

tokens = [simple_preprocess(sentance, deacc = True) for sentance in open('F:\RottenTomato.csv') ]

gensim_dictionary = corpora.Dictionary ()
gensim_corpus = [gensim_dictionary.doc2bow(token, allow_update=True) for token in tokens]

word_frequencies = [[(gensim_dictionary [id], frequence)for id, frequence in couple ] for couple in gensim_corpus]

print (word_frequencies)
```

```
IOPub data rate exceeded.
The notebook server will temporarily stop sending output
to the client in order to avoid crashing it.
To change this limit, set the config variable
`--NotebookApp.iopub_data_rate_limit`.

Current values:
NotebookApp.iopub_data_rate_limit=10000000.0 (bytes/sec)
NotebookApp.rate_limit_window=3.0 (secs)
```

Fig.08

In [30]:
```
# Create embedding model from text

review_model = gensim.models.Word2Vec(reviews, min_count=1)
```

Fig.09

In [31]:
```
#Explore NEGATIVE patterns in text model - (most similar words)
#Source of adjectives is (https://descriptivewords.org/wp-content/uploads/descriptivewords-for-movies.pdf)

review_model.wv.most_similar(['wacky','stupid','silly','average','boring','bland', 'disappointing',
                              'distasteful', 'tiresome', 'moronic', 'predictable', 'predictable', 'silly', 'stupid', 'uninteresti
```

Out[31]:
```
[('ridiculous', 0.8540648818016052),
 ('implausible', 0.8319385051727295),
 ('tasteless', 0.8295560479164124),
 ('preposterous', 0.8243200778961182),
 ('unoriginal', 0.8195405602455139),
 ('dull', 0.8086024522781372),
 ('corny', 0.8064192533493042),
 ('sappy', 0.8007218837738037),
 ('inane', 0.8003897070884705),
 ('unimaginative', 0.7996317744255066)]
```

Fig.10

In [42]:
```
# Create positive embedding model from text
review_model = gensim.models.Word2Vec(reviews, min_count=10)
```

In [43]:
```
#Explore POSITIVE patterns in text model - ( most similar words)
#Source of adjectives is: (https://descriptivewords.org/wp-content/uploads/descriptivewords-for-movies.pdf)

review_model.wv.most_similar(['brilliant','charismatic','charming','clever','dazzling','excellent', 'good',
                              'exciting', 'imaginative', 'insightful', 'inspirational', 'intriguing', 'legendary', 'original'])
```

Out[43]:
```
[('engaging', 0.8168711066246033),
 ('inventive', 0.7930113673210144),
 ('captivating', 0.7663605809211731),
 ('ingenious', 0.7576407194137573),
 ('innovative', 0.7524563074111938),
 ('arresting', 0.7445882558822632),
 ('enchanting', 0.7367279529571533),
 ('endearing', 0.7363201975822449),
 ('enthralling', 0.7299625277519226),
 ('entertaining', 0.7290042638778687)]
```

Fig.11

In [40]:
```
#Source of adjectives is: (https://descriptivewords.org/wp-content/uploads/descriptivewords-for-movies.pdf)

adjectives = ['weak','wacky','stupid','silly','average','boring','bland', 'disappointing',
              'distasteful', 'tiresome', 'moronic', 'predictable', 'predictable', 'silly', 'stupid', 'uninteresting',
              'brilliant','charismatic','charming','clever','dazzling','excellent', 'good','amazing',
              'exciting', 'imaginative', 'insightful', 'inspirational', 'intriguing', 'legendary', 'original']

#Sorting alphabetically
#adjectives.sort()

#for loop reflecting the movie genre, the adjective and the frequency

for a in adjectives:
    print("godfather", a, review_model.wv.similarity('godfather', a))

for a in adjectives:
    print('goodfellas', a, review_model.wv.similarity('goodfellas', a))

for a in adjectives:
    print('scarface', a, review_model.wv.similarity('scarface', a))
```

```
godfather weak -0.09054291
godfather wacky 0.058230504
godfather stupid 0.018192202
godfather silly -0.017721623
godfather average 0.13879102
godfather boring 0.073217824
godfather bland 0.07282665
godfather disappointing 0.16790351
godfather distasteful -0.028919348
godfather tiresome -0.06346463
godfather moronic -0.0016719922
godfather predictable -0.012748338
godfather predictable -0.012748338
godfather silly -0.017721623
godfather stupid 0.018192202
godfather uninteresting -0.039034475
```

Fig.12

In [ ]:
```
#Save the new  model

review_model2 = gensim.models.Word2Vec(reviews, min_count=1)

review_model2.wv.save_word2vec_format('F:/RottenTomato02.csv')
```

Fig.13

Appendix 02

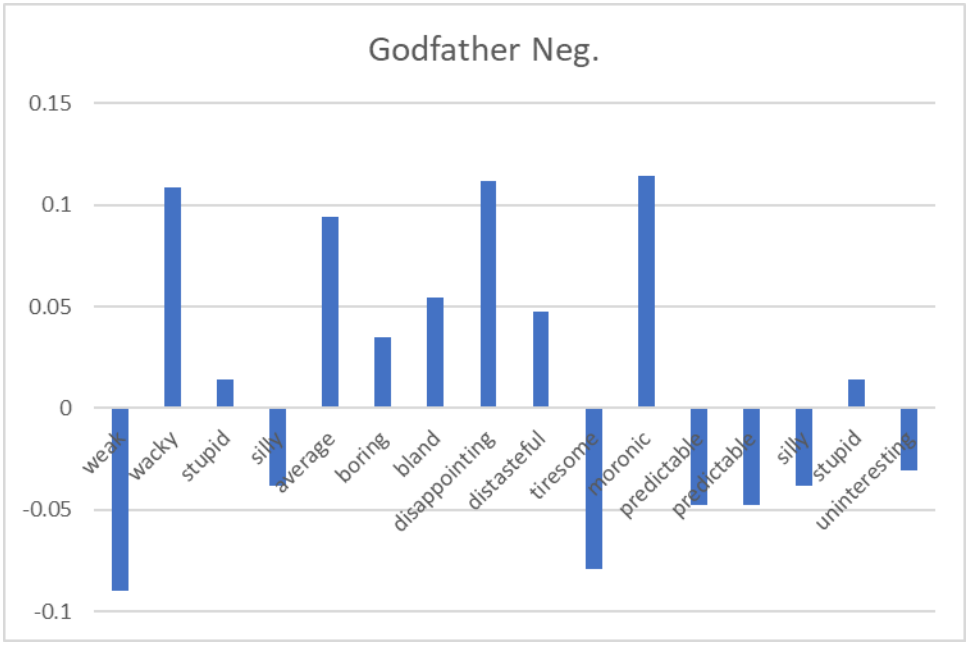

Fig.01 (Godfather movie negative adjective similarity)

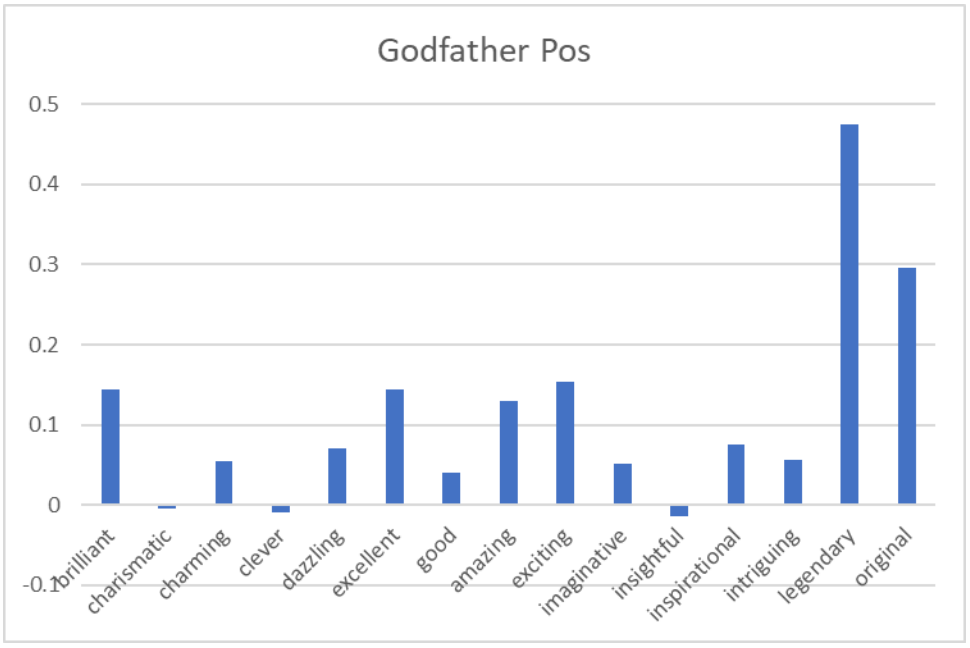

Fig.02 (Godfather movie positive adjective similarity)

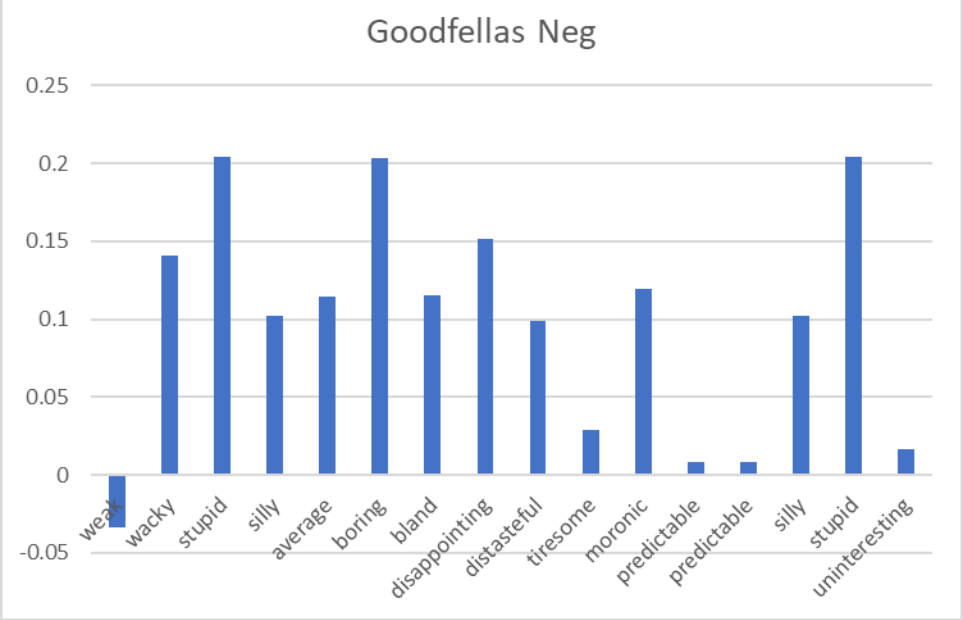

Fig.03 (Goodfellas movie negative adjective similarity)

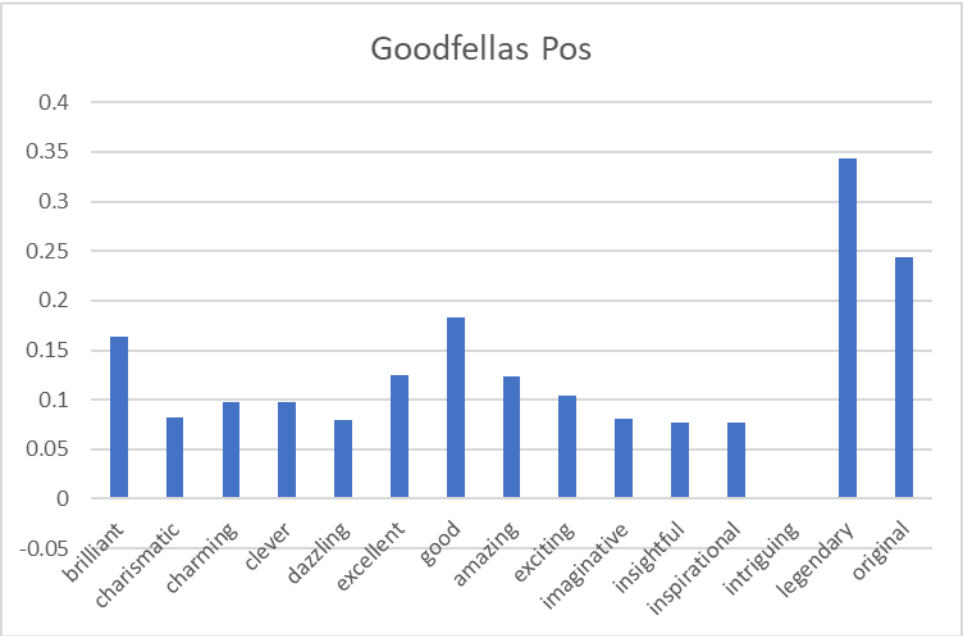

Fig.04 (Goodfellas movie positive adjective similarity)

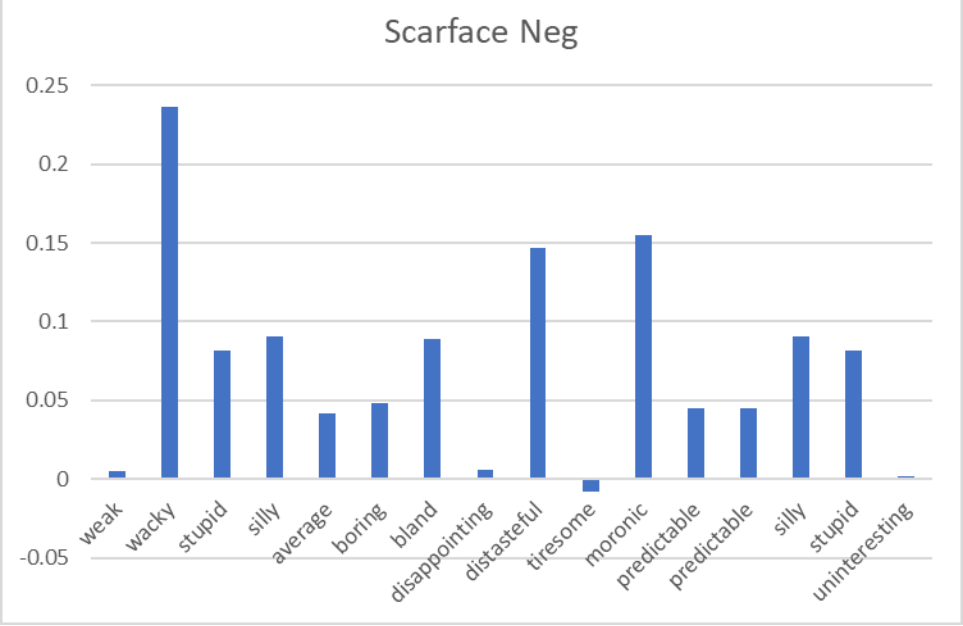

Fig.05 (Scarface movie Negative adjective similarity)

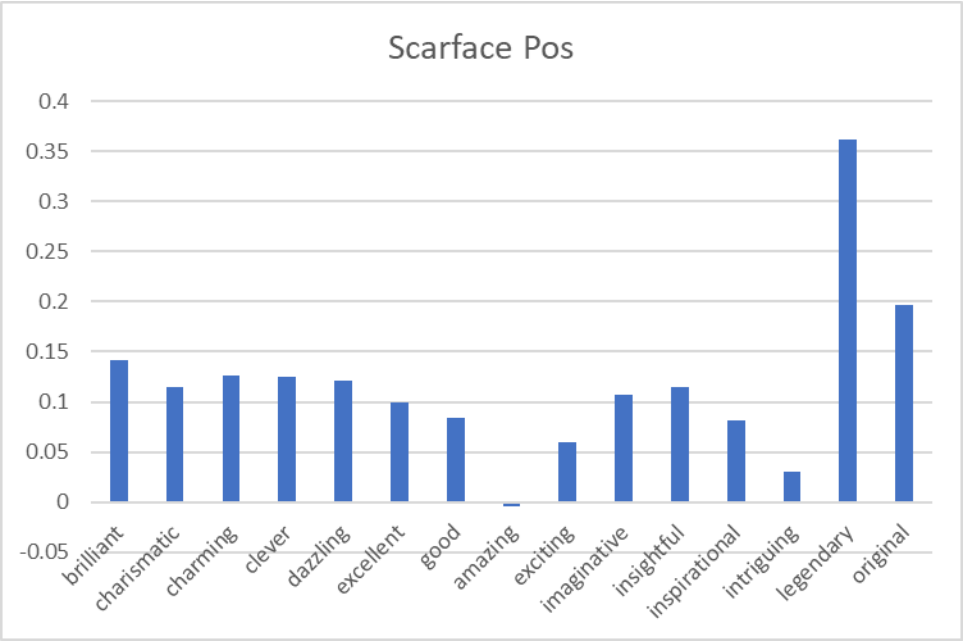

Fig.06 (Scarface movie Negative adjective similarity)