Web Analytics Final Project Report

BIA 660 (Spring 2018)

Social Network Movie Response

<u>Guided by</u>
Prof. Zachary Wentzell

<u>Author</u>
Rui Song

Table of Contents

Introduction

Social network websites such as Facebook, Twitter, Quora has evolved into a modern decentralized media channel. Where people receive information about topics that interest them, share ideas/contents that they would other people be aware of. As a result, companies are starting to view such websites a critical data source and derive value out of analyzing public stance over certain topic by analyzing it.

Problem is, opposed to websites like Yelp and IMDB where people were asked to rate restaurant, movies quantitatively, general purpose social network website, does not require users to explicitly evaluate their post with a sentiment scoring system. Take the movie Pacific Rim Uprising for example, a typical comment could be "I watched pacific rim uprising with my friends tonight and enjoyed a lot". The absence of none numeric value adds a layer of complexity for those who wish to interpret such data with software. In summary, the data mining problem lies upon the challenge to extract information from social website comments and present them to business stakeholders in an informative way so that they can draw better business decisions.

In this project, I aim at utilizing leading natural language service provider – IBM Watson – to perform the task of processing data, in order to resolve the problem of unlabeled text. Which then were transformed into tangible dataset that ML algorithms can consume, specifically, I applied the following embedding method to convert text into digit vectors. Doc2Vec and Stanford Glove pre-trained word to vectors mean, normalized word count. Based on such vectors I applied two popular ML algorithms to predict the sentiment score Watson NLP system would give to a particular tweet.

In addition, I built a prototype HTML based dashboard, to demonstrate that stakeholders from all sides can easily consume analytic result without installing or even buying the license of proprietary software like Tableau, Power BI, etc.

1 Data

For this project, I am only interested in tweets that are relevant to pacific rim uprising. As I am targeting this movie as an example to analyze Twitter data.

My exploration and analysis were performed against 26354 tweets that was posted through the period between Mar. 31st to April 10th. Due to Watson API quota limitation and other technical issues, the twitter listening system was down for two short period of time but was resumed afterward. As a result, an estimate of 12 hours of tweets was lost.

Despite the original tweets that were gathered through Twitter real-time listening API. Simultaneously, tweet text was sent to Watson NLP API to perform sentiment analysis.

2 Prepare the data

2.1 Raw Data Understanding

The dataset I connected can be decomposed into two major sources. the first one is the returned twitter JSON data. The second one is the Watson NLP feedback, which is also in the form of JSON format.

```
{
  "created_at": "Fri Apr 01 20:20:07 +0000 2016",
  "id": 715997192373288962,
  "id_str": "715997192373288962",
  "text": "It's Friday!! \ud83d\udc79 #AprilFools - funny pranks https:\/\/t.co\/b9ZdzRxzFK",
  "source": "\u003ca href=\"http:\/\/twitter.com\" rel=\"nofollow\"\u003eTwitter Web
Client\u003c\/a\u003e",
  "truncated": false,
  "in_reply_to_status_id": null,
  "in_reply_to_status_id_str": null,
  "in_reply_to_user_id": null,
  "in_reply_to_user_id_str": null,
  "in_reply_to_screen_name": null,
  "user": {
    "id": 3001969357,
    "id_str": "3001969357",
    "name": "Jordan Brinks",
    "screen_name": "furiouscamper",
    "location": "Birmingham",
    "url": "http:\/\/indigofiddle.com",
    "description": "Alter Ego - Twitter PE",
    "derived": {
      "locations": [
        {
          "country": "United States",
          "country_code": "US",
          "locality": "Birmingham",
          "region": "Alabama",
          "full_name": "Birmingham, Alabama, United States",
          "geo": {
            "coordinates": [
              -86.80249,
              33.52066
            ],
            "type": "point"
          }
        }
      ]
    },
    "protected": false,
    "verified": false,
    "followers_count": 15,
    "friends_count": 24,
    "listed_count": 1,
    "favourites_count": 14,
    "statuses_count": 221,
    "created_at": "Thu Jan 29 18:27:49 +0000 2015",
    "utc_offset": -21600,
    "time_zone": "Mountain Time (US & Canada)",
    "geo_enabled": true,
    "lang": "en",
    "contributors_enabled": false,
    "is_translator": false,
    "profile_background_color": "000000",
    "profile_background_image_url": "http:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png",
    "profile_background_image_url_https": "https:\/\/abs.twimg.com\/images\/themes\/theme1\/bg.png",
    "profile_background_tile": false,
    "profile_link_color": "FF691F",
    "profile_sidebar_border_color": "000000",
    "profile_sidebar_fill_color": "000000",
    "profile_text_color": "000000",
    "profile_use_background_image": false,
    "profile_image_url":
"http:\/\/pbs.twimg.com\/profile_images\/601155672395227136\/qakfE9EU_normal.jpg",
    "profile_image_url_https":
"https:\/\/pbs.twimg.com\/profile_images\/601155672395227136\/qakfE9EU_normal.jpg",
    "profile_banner_url": "https:\/\/pbs.twimg.com\/profile_banners\/3001969357\/1432161817",
    "default_profile": false,
    "default_profile_image": false,
    "following": null,
    "follow_request_sent": null,
    "notifications": null
  },
  "geo": null,
  "coordinates": null,
  "place": {
    "id": "fd70c22040963ac7",
    "url": "https:\/\/api.twitter.com\/1.1\/geo\/id\/fd70c22040963ac7.json",
    "place_type": "city",
    "name": "Boulder",
    "full_name": "Boulder, CO",
    "country_code": "US",
    "country": "United States",
    "bounding_box": {
      "type": "Polygon",
      "coordinates": [
    },
    "attributes": {
    }
  },
  "contributors": null,
  "is_quote_status": false,
  "retweet_count": 0,
  "favorite_count": 0,
  "entities": {
    "hashtags": [
      {
        "text": "AprilFools",
        "indices": [
          16,
          27
        ]
      }
    ],
    "urls": [
      {
        "url": "https:\/\/t.co\/b9ZdzRxzFK",
        "expanded_url": "http:\/\/www.today.com\/parents\/joke-s-you-kid-11-family-friendly-april-fools-
pranks-t83276",
        "display_url": "today.com\/parents\/joke-s\u2026",
        "unwound": {
          "url": "http:\/\/www.today.com\/parents\/joke-s-you-kid-11-family-friendly-april-fools-pranks-
t83276",
          "status": 200,
          "title": "The joke's on you, kid: 11 family-friendly April Fools pranks",
          "description": "If your kids are practical jokers, turn this April Fools' Day into a family
affair."
        },
        "indices": [
          43,
          66
        ]
      }
    ],
    "user_mentions": [
    ],
    "symbols": [
    ]
  },
  "favorited": false,
  "retweeted": false,
  "possibly_sensitive": false,
  "filter_level": "low",
  "lang": "en",
  "timestamp_ms": "1459542007903",
  "matching_rules": [
    {
      "tag": null,
      "id": 715947673707089920
}
```

, as the upper sample shows, it contains a large amount of fields, many of which do not strictly associate with the purpose of Text analysis. By doing so, we were able to reduce the size of the dataset dramatically and hence able to store the dataset in a table.

Yelp Business dataset attributes:

Tweet ID	id
Tweet Time	Timestamp_ms
Tweet Text	text
Time Zone	User.time_zone
Follower Count	User.followers_count

The service of IBM Watson natural language processing service analyze text to extract meta-data from content such as concepts, entities, keywords, categories, relations and semantic roles. Following is a sample record which contains the TweetID that can be used to join back with raw twitter data.

```
{
  "TweetID": "979572206341623809",
  "_id": {
    "$oid": "5abdb953649faa4044711944"
  },
  "entities": [
    {
      "count": 2,
      "disambiguation": {
        "dbpedia_resource": "http://dbpedia.org/resource/Guillermo_del_Toro",
        "name": "Guillermo del Toro",
        "subtype": [
          "Actor",
          "AwardNominee",
          "AwardWinner",
          "FilmActor",
          "FilmDirector",
          "FilmProducer",
          "TVActor",
          "FilmWriter"
        ]
      },
      "emotion": {
        "anger": 0.175714,
        "disgust": 0.203247,
        "fear": 0.188037,
        "joy": 0.141949,
        "sadness": 0.372286
      },
      "relevance": 0.33,
      "sentiment": {
        "label": "positive",
        "score": 0.828112
      },
      "text": "Guillermo del Toro",
      "type": "Person"
    },
    {
      "count": 1,
      "emotion": {
        "anger": 0.159886,
        "disgust": 0.192974,
        "fear": 0.206119,
        "joy": 0.125385,
        "sadness": 0.390025
      },
      "relevance": 0.33,
      "sentiment": {
        "label": "positive",
        "score": 0.828112
      },
      "text": "Pacific Rim",
      "type": "GeographicFeature"
    }
  ],
  "keywords": [
    {
      "emotion": {
        "anger": 0.159886,
        "disgust": 0.192974,
        "fear": 0.206119,
        "joy": 0.125385,
        "sadness": 0.390025
      },
      "relevance": 0.934667,
      "sentiment": {
        "label": "positive",
        "score": 0.828112
      },
      "text": "Guillermo del Toro"
    },
    {
      "emotion": {
        "anger": 0.159886,
        "disgust": 0.192974,
        "fear": 0.206119,
        "joy": 0.125385,
        "sadness": 0.390025
      },
      "relevance": 0.522711,
      "sentiment": {
        "label": "positive",
        "score": 0.828112
      },
      "text": "Pacific Rim"
    }
  ],
  "language": "en",
  "usage": {
    "features": 2,
    "text_characters": 95,
    "text_units": 1
  }
}
```

Following is the parsed table

Tweet ID	id
Entity	List of entities
Anger	Range from 0 to 1
Disgust	Range from 0 to 1
Fear	Range from 0 to 1
Joy	Range from 0 to 1
Sadness	Range from 0 to 1
Sentiment Score	Range from -1 to 1
Sentiment	‘positive’, ‘neutral’, ‘negative’

2.2 Data Preparation

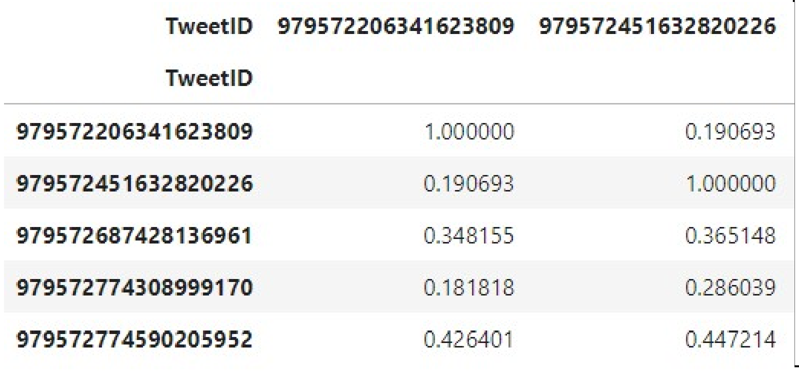
There are certain issues with the raw tweet text, the following part describes the steps that I take so that the dataset can be consumed by Machine Learning algorithms.

2.2.1 Drop duplications

In order to secure the performance of preceding predictive model, a challenge that I have to deal with was to identify and remove retweets. This problem can be viewed into two-fold, the simpler one is to remove identical retweet, which can be directly resolved by drop_duplicates function from pandas data frame. After which the sample size went down from 26354 to 15297 The much harder one is to identify almost identical one, meaning the tweet is slightly modified, but the tweet contains no new information. Following is an example

```
john boyega mo abudu rmd and more attend private screening of pacific rim uprising photos

john boyega mo abudu rmd and more attend private screening of pacific rim yungtosa

mo abudu rmd and others attend private screening of john boyega s pacific rim uprising

john boyega mo abudu rmd attend private screening of pacific rim uprising

photos john boyega mo abudu rmd attend private screening of pacific rim uprising
```

Initially, I applied word count algorithm to vectorize tweets and then calculated the item-item similarity between each other. Then designed a n squared algorithm to filter almost identical retweet.

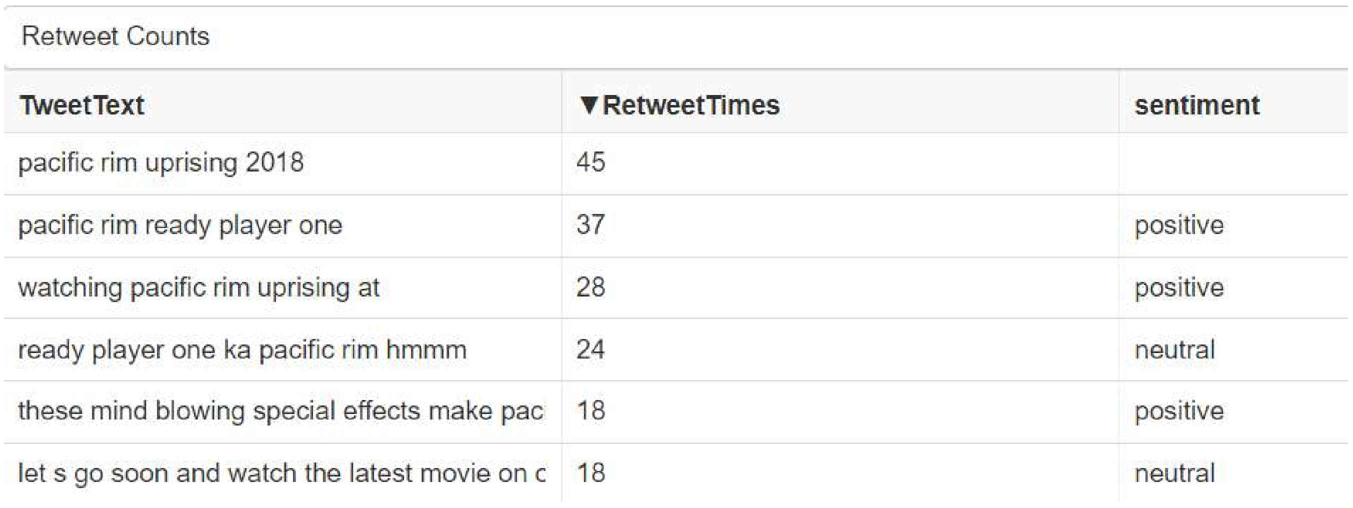


TweetID	979572206341623809	979572451632820226
TweetID		
979572206341623809	1.000000	0.190693
979572451632820226	0.190693	1.000000
979572687428136961	0.348155	0.365148
979572774308999170	0.181818	0.286039
979572774590205952	0.426401	0.447214

```
def filter_high_sim():
    remove_id = set()
    n = sim_df.shape[0]
    for i in range(n):
        if sim_df.index[i] not in remove_id:
            for j in range(i+1, n):
                if sim_df.index[j] not in remove_id:
                    if sim_df.iloc[i,j]>=0.8:
                        remove_id.add(sim_df.index[j])
    return remove_id
```

```
def filter_high_sim():
    remove_id = set()
    sim_dict = dict()
    n = sim_df.shape[0]
    for i in range(n):
        if Text_df.index[i] not in remove_id:
            i_index = sim_df.index[i]
            sim_dict[i_index] = set()
            s1 = Text_df.loc[i_index, 'TweetText']
            for j in range(i+1, n):
                j_index = sim_df.index[j]
                if j_index not in remove_id:
                    s2 = Text_df.loc[j_index, 'TweetText']
                    if SM(None, s1, s2).ratio()>=0.85:
                        remove_id.add(j_index)
                        sim_dict[i_index].add(j_index)
    return sim_dict, remove_id
```

The previous function relies on the similarity matrix which can be easily computed through matrix multiplication, but the performance of this approach is not as good as the left one, which uses character-by-character string comparison algorithm. However, This one takes ten times more computational power at the SM(None, s1, s2) step. Proceeding steps are based on the left one, which further removes 2027 slightly modified retweets.



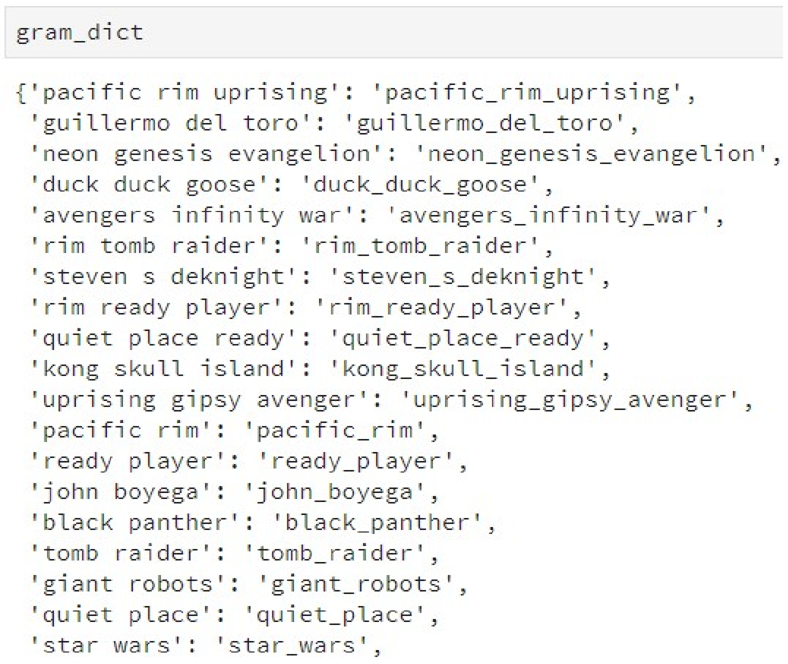
Retweet Counts

TweetText	▼RetweetTimes	sentiment
pacific rim uprising 2018	45	
pacific rim ready player one	37	positive
watching pacific rim uprising at	28	positive
ready player one ka pacific rim hmmm	24	neutral
these mind blowing special effects make pac	18	positive
let s go soon and watch the latest movie on c	18	neutral

Above is a sorted table to demonstrate the retweets that get slightly modified.

2.2.2 Phrase detection

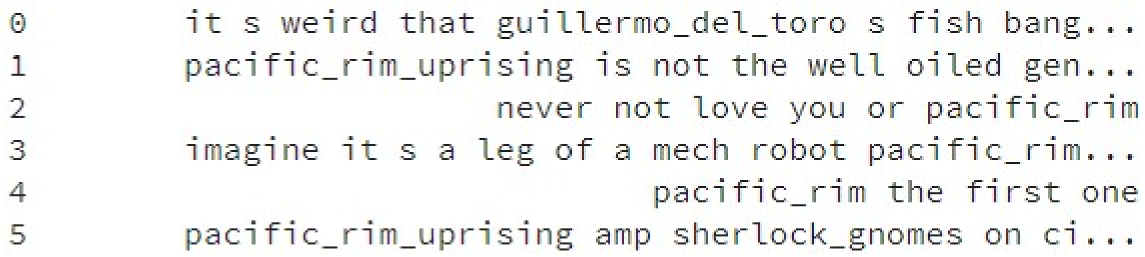
Noun phrase detection is a standard step to allow models to consider entities as a whole as opposed to separated none related word, a typical example in my project is the name of the movie director Guillermo del Toro, if divided, each of the words is not even English word and ideally should not contain any information. With phrase detection process, later models can avoid the risk of noise introduced by those single word. However, given that Watson NLP service return entities from the tweets, I was able to use a novel way to perform this task instead of none supervised learning method. Following are the steps that I used to find right combinations.



```
gram_dict
```

```
{'pacific rim uprising': 'pacific_rim_uprising',
 'guillermo del toro': 'guillermo_del_toro',
 'neon genesis evangelion': 'neon_genesis_evangelion',
 'duck duck goose': 'duck_duck_goose',
 'avengers infinity war': 'avengers_infinity_war',
 'rim tomb raider': 'rim_tomb_raider',
 'steven s deknight': 'steven_s_deknight',
 'rim ready player': 'rim_ready_player',
 'quiet place ready': 'quiet_place_ready',
 'kong skull island': 'kong_skull_island',
 'uprising gipsy avenger': 'uprising_gipsy_avenger',
 'pacific rim': 'pacific_rim',
 'ready player': 'ready_player',
 'john boyega': 'john_boyega',
 'black panther': 'black_panther',
 'tomb raider': 'tomb_raider',
 'giant robots': 'giant_robots',
 'quiet place': 'quiet_place',
 'star wars': 'star_wars',
```

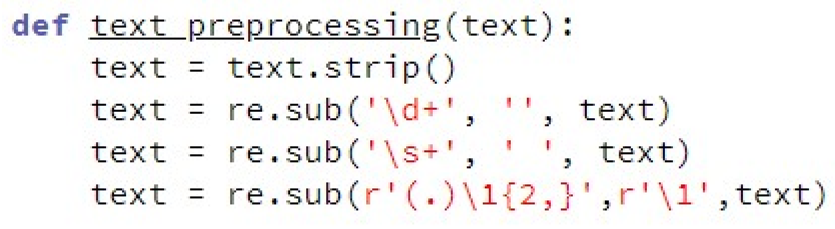
To begin with, the entity detection functionality from Watson does not perform perfectly, certain recognized entities also contain adjective word, if used blindly, semantic meaning may end up lost. Secondly, pluralism and other none lemma form of words cause unnecessary duplication, movie review and movie reviews are the same thing but will be considered differently, for this one, I have removed some but still not perfect. The filtered most frequent entities are stored in the above dictionary, the phrases are ready to be substituted by dash connected whole entities. one minor nuance is that three-word phrases have to be replaced first, then two-word phrase for otherwise pacific rim uprising would end up to be pacific_rim uprising.



```
0        it s weird that guillermo_del_toro s fish bang...
1        pacific_rim_uprising is not the well oiled gen...
2                        never not love you or pacific_rim
3        imagine it s a leg of a mech robot pacific_rim...
4                                pacific_rim the first one
5        pacific_rim_uprising amp sherlock_gnomes on ci...
```

2.2.3 Other issues

Given the non-formality of tweets, other issues also arouse comparing to an article like formal text data analysis. People put multiple spaces, the appearance of hashtags, repeating character to express feeling such as yyyyyeees also require ways to be remedied, at this stage, regular expression plays a critical role in handling such problem. here are some examples:



```
def text_preprocessing(text):
    text = text.strip()
    text = re.sub('\d+', '', text)
    text = re.sub('\s+', ' ', text)
    text = re.sub(r'(.)\1{2,}',r'\1',text)
```

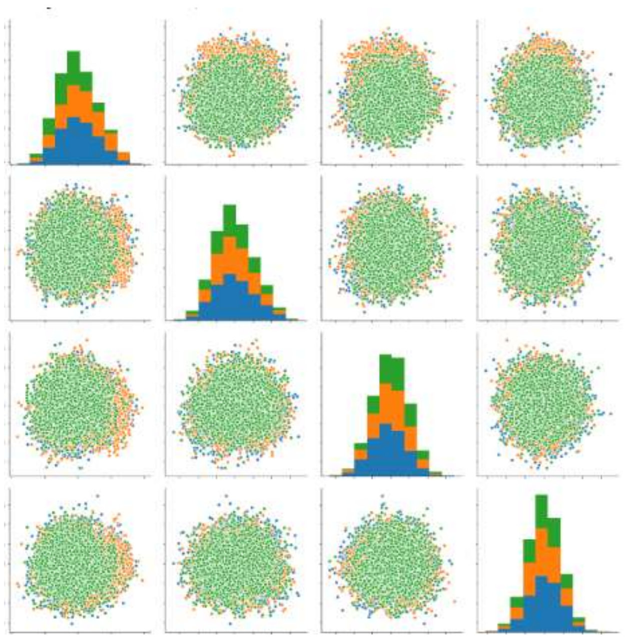
3 Modeling

For small companies, or for the purpose of experimental project, It is not always feasible for data teams to spend too much resource on getting the right format of data that a model can consume, in the case of my project, a tweet does not include user labeled semantic information. Therefore, the ability to combine the power of advanced analytic service and later build its own model can be of help in the prototyping stage.

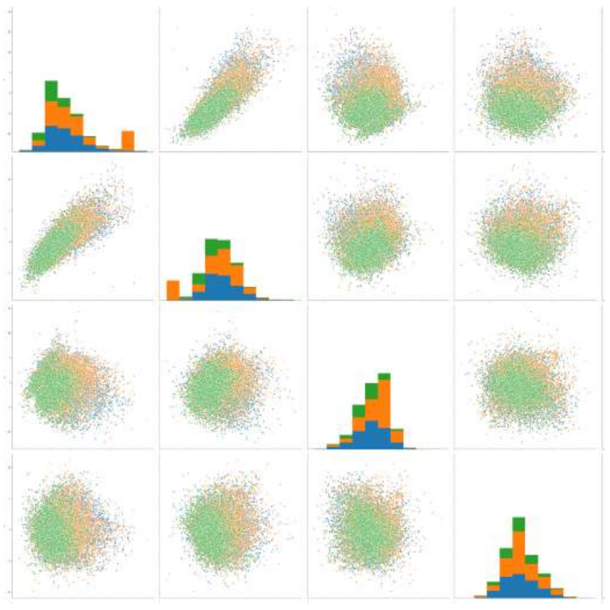
In my project, though from the screenshot above we know that it was far from perfect, none the less I still chose to call Watson NLP service to label my collected tweets about Pacific Rim Uprising. Which returns a sentiment score ranging from negative one to positive one. In the context of predictive modeling, these scores represent the objective values,

3.1 Text Embedding

For my project, I experimented two embedding technics, the first one being document to vector embedding, the second one being Stanford Glove word to vector embedding. Ideally, I expected that my customized doc2vec representation would achieve better performance in the later process but it turned out otherwise, further investigation through plotting PCA showed that the doc2vec embedding method does not differentiate data points very well, below are the plotting.



Doc2Vec Embedding

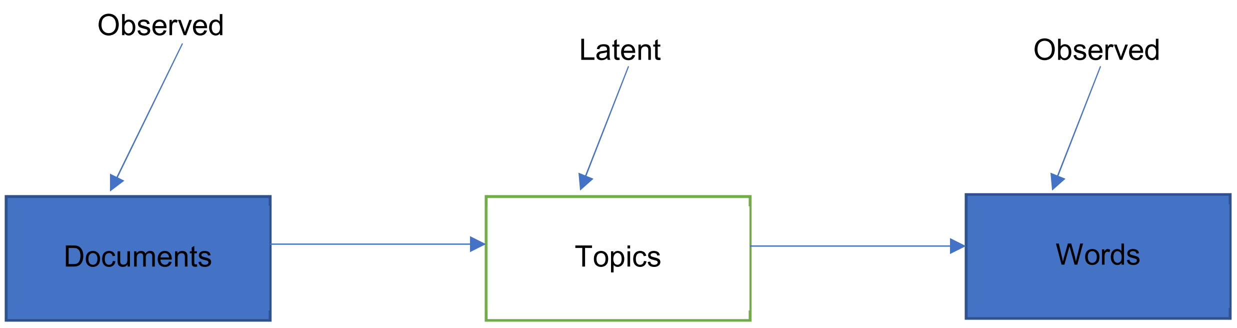


Glove Word2Vec Embedding

My assumption is that the result of my self-trained doc2vec model was limited by certain characteristics of my dataset, either the data are mostly short, mostly less then twenty words, which is way less than the 100-dimension embedding vector size that I chose. Or Homogeneous, meaning all tweets are basically a combination of words from a limited words space. While the Glove doc2vec model was trained upon hundreds of millions of tweets, a naive averaging approach can still achieve better performance.

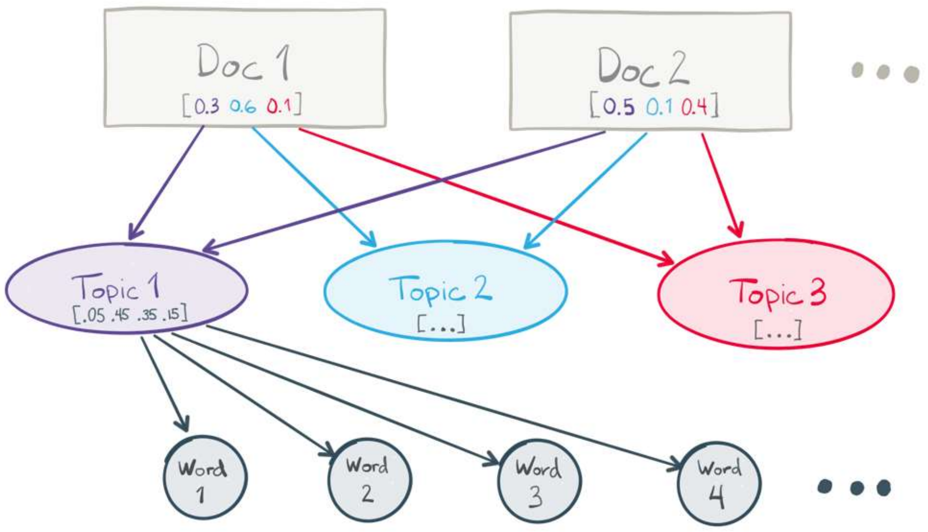
3.2 Latent Dirichlet Allocation (LDA)

LDA is a generative probabilistic model of a corpus. The basic idea is that the documents are represented as random mixtures over latent topics, where a topic is characterized by a probability distribution over words.

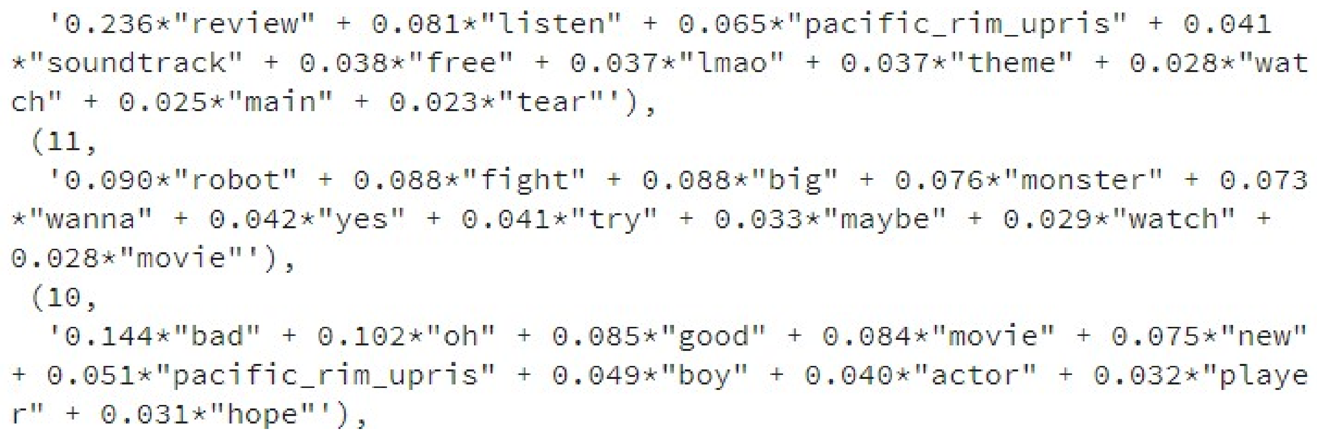


The intuition of LDA is as follows:

- Each document may be viewed as a mixture of various topics where each document is considered to have a set of topics that are assigned to it via LDA.
- Initially randomly assign each word in the document to one of the K topics. Each round, assuming that all topic assignments except for the current word in question are correct, then updating the assignment of the current word.



For this part, I used all unique tweets to train the LDA model, and present 30 topics. The below list illustrates a few topics, and words with corresponding probabilities associated with it.



```
  '0.236*"review" + 0.081*"listen" + 0.065*"pacific_rim_upris" + 0.041
*"soundtrack" + 0.038*"free" + 0.037*"lmao" + 0.037*"theme" + 0.028*"wat
ch" + 0.025*"main" + 0.023*"tear"'),
 (11,
  '0.090*"robot" + 0.088*"fight" + 0.088*"big" + 0.076*"monster" + 0.073
*"wanna" + 0.042*"yes" + 0.041*"try" + 0.033*"maybe" + 0.029*"watch" +
0.028*"movie"'),
 (10,
  '0.144*"bad" + 0.102*"oh" + 0.085*"good" + 0.084*"movie" + 0.075*"new"
+ 0.051*"pacific_rim_upris" + 0.049*"boy" + 0.040*"actor" + 0.032*"playe
r" + 0.031*"hope"'),
```

3.3 Regression to predict score

The steps above are in one way to prepare the text data to make predictions of the sentiment underneath. So that my system does not have to rely solely on Watson service to analyze the data. TO illustrate my work, I will list two machine learning regression models against the two embedding result I have adopted above, and compare them with each other using scatter plot and Mean Absolute Error.

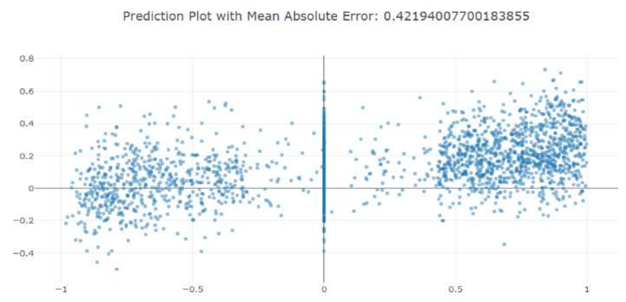
3.3.1 Baseline

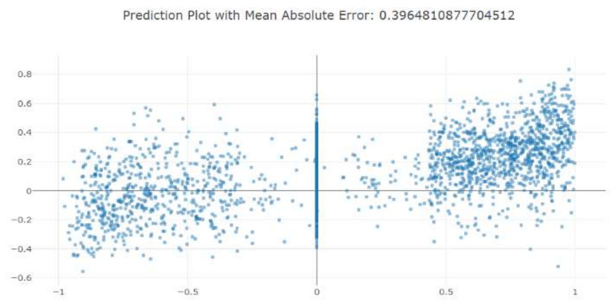
For our baseline, I simply predict 0 as it falls into the middle of rating ranges and means neutral. Which yield a mean absolute error of 0.45 and mean squared error of 0.33. Later I will compare models with this baseline.

3.3.2 Random Forest

Random forest or random decision forest is an ensemble learning method for classification, regression, and other tasks, that operate by constructing a multitude of decision trees at training time and outputting the mean prediction of individual trees. [2]

I set up my model at 500 trees with bootstrap, when train and test against doc2vec embedding, the model yield a mean absolute error of 0.42 and mean squared error of 0.25, which is better, but not in a significant way. This from another perspective proves that the doc2vec embedding method performs poorly.

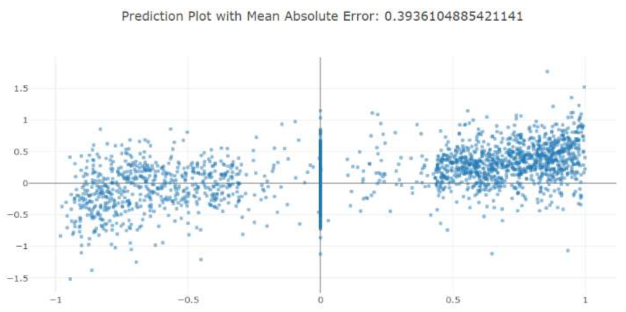


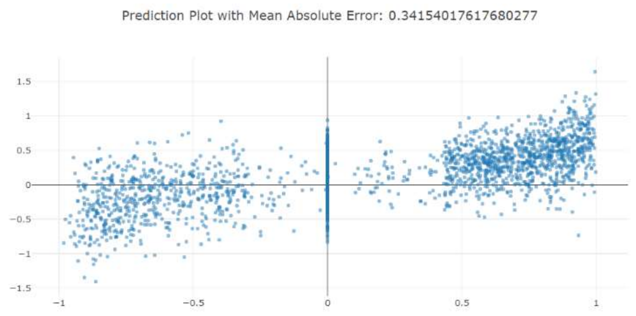


Similarly, I also use the random forest to train and test against Stanford Glove word to vector average, the model yield a mean absolute error of 0.396 and mean squared error of 0.226, which proves to perform better than the doc2vec random forest model.

3.3.3 Support Vector Machine

Support vector machine constructs a hyperplane or set of hyperplanes in a high- or infinite-dimensional space, which can be used for regression.[3] Unlike linear regression, it does not penalize observations that fall within a certain margin. And the projection from lower dimension to higher dimension makes it easier to detect possible regression.

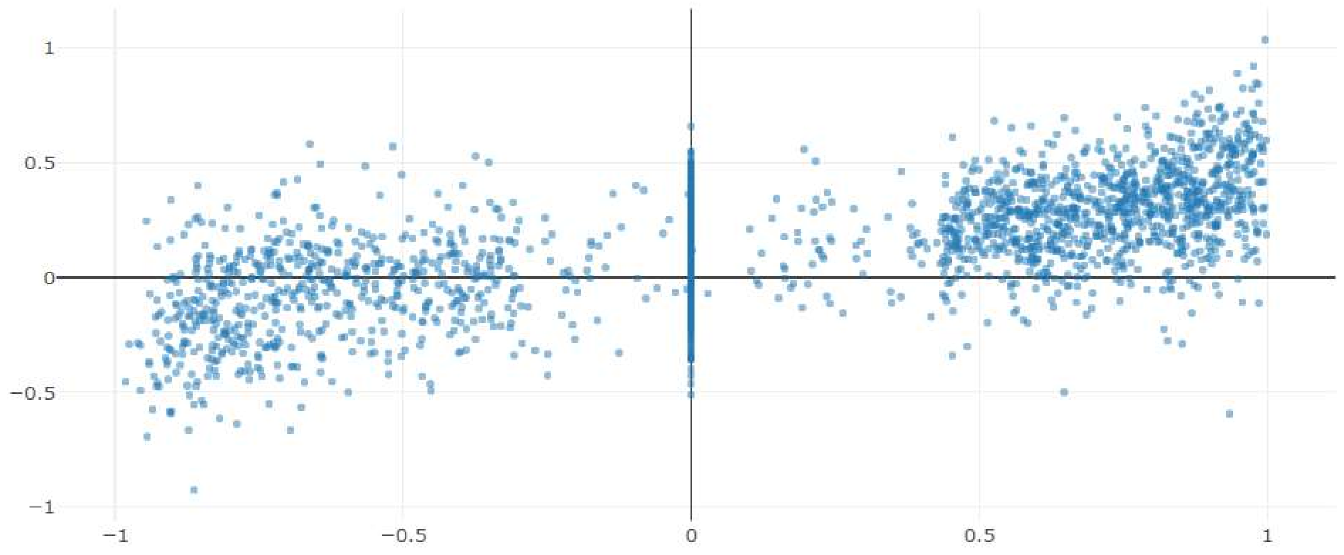




For the SVM model that was based on doc2vec, Due to the noisiness of vector result (from the above PCA plotting), I chose high penalty – 35 to train the model, while for glove word2vec, I chose low penalty parameter – 0.8 to train the model. The doc2vec yield a mean absolute error of 0.394 and mean squared error of 0.237. While glove word2vec yield a mean absolute error of 0.34 and mean squared error of 0.18.

3.3.4 Ensemble

Finally, I've ensembled all the prediction result from all described algorithms, specifically, I took the average of four individual algorithm prediction result. The model achieved slightly better RMSE of 0.1800 compared to SVM against glove word2vec 0.1809. While the mean absolute value gets slightly worse from 0.3415 to 0.3462.



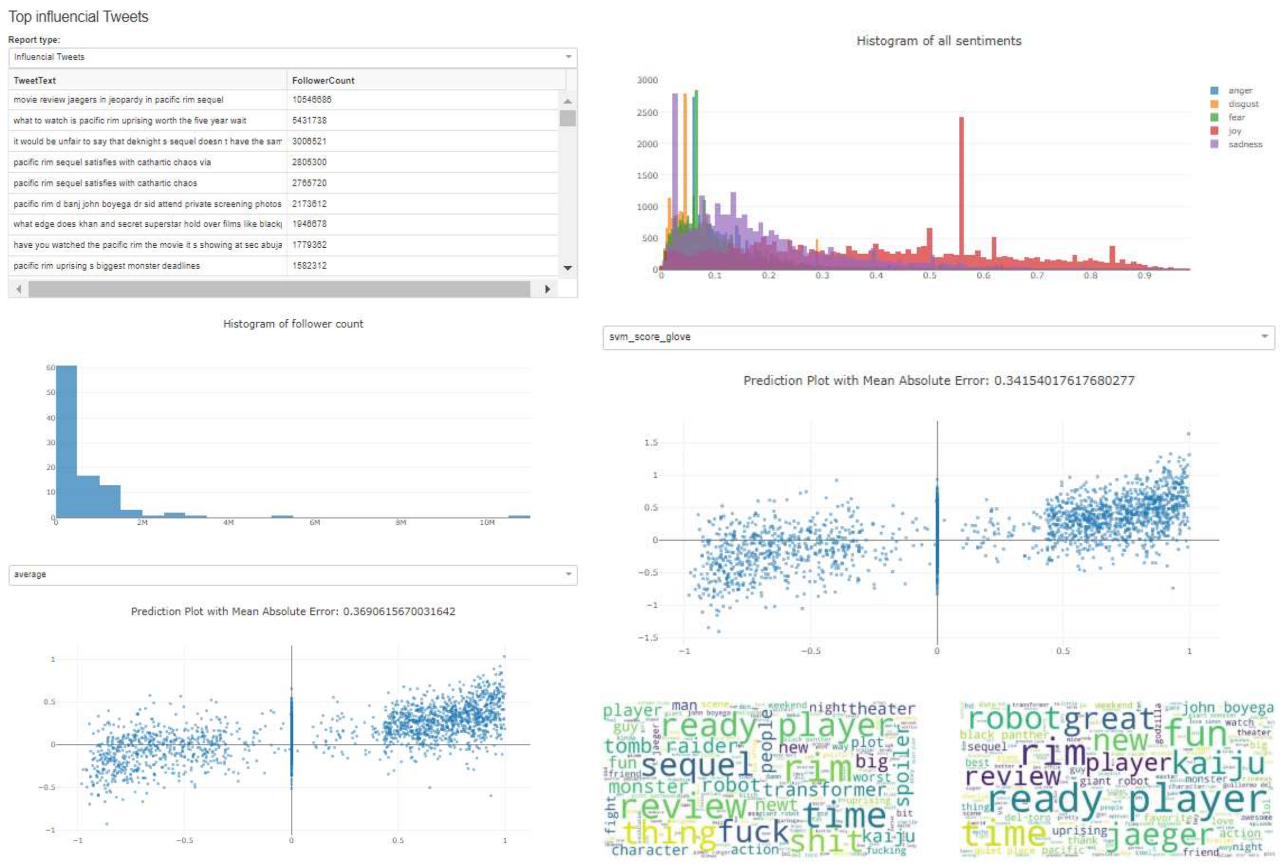
3.3.5 Conclusion

None the less, from the above graph it is clear to see that, the model performs relatively good on the positive side as majority positive tweet data points lie within the first quadrant. However, when it comes to the neutral and negative side, the model only returns a slightly better result comparing to a random guess.

4 Dashboard as a flexible visualization tool

Dashboards are amongst the most used end products of an entire process of presenting data analytical result to answer business questions and becoming a widely desired skillset data scientist have to acquire, while there are widely known products such as Tableau, PowerBI that enables people none-technicians to easily create dashboard using drag and drop, but the range of available presenting options are still limited, for example, HTML format is not supported by Tableau, word cloud on the other hand, can not be easily created by Tableau. [4]

Being a major 3rd party visualization package producer for python, Plotly announced Dash, an open source framework to create web applications with Python. Dash combines Flask with Plotly's python library to allow end user easily create web-based dashboard in an intuitive and rapid manner.



Top influencial Tweets

Report type: Influencial Tweets

TweetText	FollowerCount
movie review jaegers in jeopardy in pacific rim sequel	10546686
what to watch is pacific rim uprising worth the five year wait	5431738
it would be unfair to say that deknight s sequel doesn t have the sam	3006521
pacific rim sequel satisfies with cathartic chaos via	2805300
pacific rim sequel satisfies with cathartic chaos	2765720
pacific rim d banj john boyega dr sid attend private screening photos	2173612
what edge does khan and secret superstar hold over films like blackı	1946678
have you watched the pacific rim the movie it s showing at sec abuja	1779362
pacific rim uprising s biggest monster deadlines	1582312

svm_score_glove

average

Histogram of all sentiments
anger
disgust
fear
joy
sadness
Histogram of follower count
Prediction Plot with Mean Absolute Error: 0.34154017617680277
Prediction Plot with Mean Absolute Error: 0.3690615670031642

Due to time constraint, the dashboard that I built was not aiming to fully answer business or tech related questions, but more like a mixture of both that aims to reveal the potential of what such dashboard is capable of. First of all, anyone can get access to the dashboard with a simple browser if given permission, which is critical in a business setting. Meanwhile, researchers can also share their model result with colleges or tech community with simple plots that I showed above. Without sharing data and the extraneous effort that has to be put in setting up the environment. as well as their interactions.

5 Conclusion

In this project, I have exposed myself to a wide spectrum of data science project processes, from data collection to data warehousing, to using 3rd party service, to combine multiple data source, to perform ETL against raw data, to machine learning modeling, to deliver end-user consumable dashboard. Though for each of the step, I can further improve it. But the overall goal of my independent project was to summarize the various topics, tools, technics that I have exposed thus far. Moreover, through this project, I realized that NLP against short text data differentiates itself from long article like data.

6 Bibliography

[1] https://en.wikipedia.org/wiki/Twitter
[2] https://en.wikipedia.org/wiki/Random_forest
[3] https://en.wikipedia.org/wiki/Support_vector_machine
[4] https://medium.com/a-r-g-o/using-plotlys-dash-to-deliver-public-sector-decision-support-dashboards-ac863fa829fb