Research on Natural Language Processing Algorithms on Review Data

LO Tsz Cheung[*] WONG Kin Fat[†] SALIM Hansel Narkian[‡]

December 6, 2017

Abstract

With the recent surge of review writing, review mining has become one of the most researched topics in computer science. There are two major tasks in review mining: keyword extraction and sentiment analysis. Various algorithms have been developed dealing with these two problems. In this paper, we evaluate the effectiveness of 2 widely used packages, Jieba[1] for keyword extraction and Stanford CoreNLP[2] for sentiment analysis with several state-of-art algorithms, on two different Chinese and English datasets.

1 Introduction

With the rapid growth of Internet, user reviews have increased significantly in term of size. Because user reviews have significant influence on potential customers, they are an important source of information for businesses and potential customers. However, a large number of user reviews make it harder for individuals to evaluate all the user reviews. Most of the time, potential customers make a decision only based on several user reviews. As such, an incomplete information leads to a biased decision.

NLP or Natural Language Processing is the ability of computer program to understand human language as its spoken. With NLP, a large number of user reviews can be evaluated with ease. Our research project focused on two major tasks in NLP, keyword extraction and sentiment analysis. Keyword extraction is a method that automatically identifies terms that best describe the subject of a text. $tf-idf$ and $TextRank$ are 2 most widely-used algorithms for keyword extraction. On the other hand, sentiment analysis is a method to extract and identify the emotions or feelings in a text, where **Sentiment Treebank** and **Recursive Neural Tensor Network** are 2 most widely-used models to achieve this task.

In this paper, we tested both Jieba and Stanford CoreNLP using two different datasets. The goal of this project is to evaluate the performance of each algorithm. The two datasets were WhaleCollege university review and Twitters tweets. University reviews were used for Jieba, while Twitters tweets dataset was used for Stanford CoreNLP.

[*]Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **tcloaa@ust.hk**

[†]Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **kfwongao@connect.ust.hk**

[‡]Department of Mathematics, the Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong. Email: **hnsalim@connect.ust.hk**

2 Methodology

2.1 Keyword Extraction on Chinese Review Data

The data is generously offered by WhaleCollege, an education start-up providing university information for fresh high-school graduates. The dataset contains 200,000 data points, each consisting of *university id, comment, upvote, downvote.* At the first step of the study, we are interested in extracting keywords of these comments, which will give us brief but concise ideas of describing the universities in certain words.

2.1.1 TF-IDF

In information retrieval, *tf-idf*, short for **term frequency-inverse document frequency**, is a numerical statistic that is intended to reflect how important a word is to a document in a collection.[3] It is often used as a weighting factor in searches of information retrieval, text mining, and user modeling. The *tf-idf* value increases proportionally to the number of times a word appears in the document, but is often offset by the frequency of the word in the corpus, which helps to adjust for the fact that some words appear more frequently in general.

$$tf_{w,d} = \frac{\text{\# of occurrence that } w \text{ appears in } d}{\text{max \# of occurrence in } d}$$
$$idf_w = \frac{\text{\# of total documents}}{\text{\# of documents that } w \text{ appears in}}$$
$$(tf - idf)_{w,d} = tf_{w,d} \times idf_w$$

Nowadays, *tf-idf* is one of the most popular term-weighting schemes, with 83% of text-based recommender systems in the domain of digital libraries using *tf-idf*. And here is a simple example to illustrate the idea:

Example 1. *Given a document d containing terms with frequencies: A(3), B(2), C(1). Assume the whole collection contains 10,000 documents, and document frequencies of these terms are A(50), B(1300), C(250). Then:*

$$A : tf = \frac{3}{3}, idf = log_2(\frac{10000}{50}) = 7.6, \therefore (tf - idf)_{A,d} = 7.6$$
$$B : tf = \frac{2}{3}, idf = log_2(\frac{10000}{1300}) = 2.9, \therefore (tf - idf)_{B,d} = 2.0$$
$$C : tf = \frac{1}{3}, idf = log_2(\frac{10000}{250}) = 5.3, \therefore (tf - idf)_{C,d} = 1.8$$

From Example 1, we can see that A becomes the most important word to a specific document in this collection, B follows next, with C the least important one.

2.1.2 TextRank

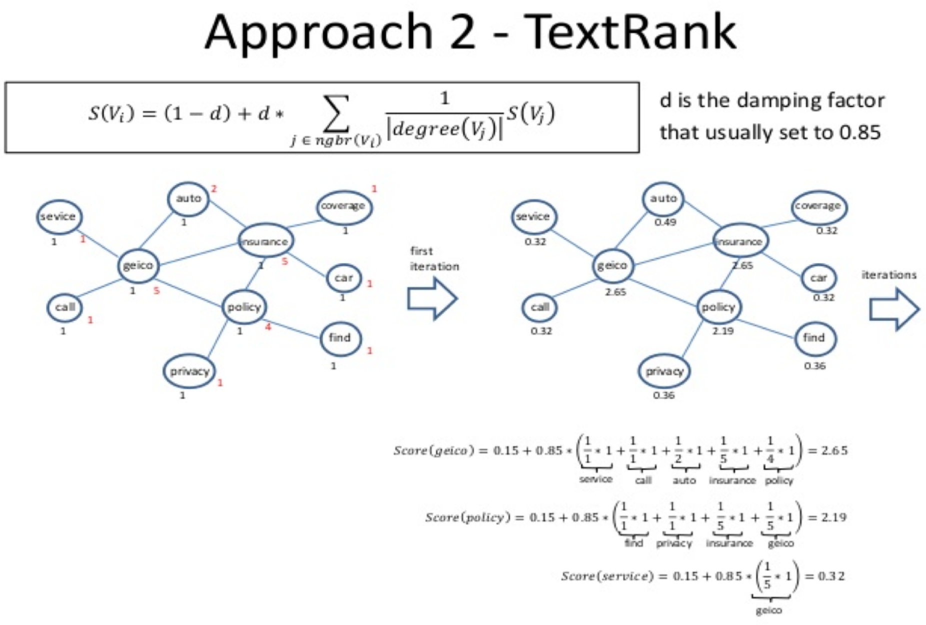
TextRank is a graph-based ranking model for text processing, inspired by Google's *PageRank*. The basic idea of TextRank is to provide a score (not naively frequency) for each word in the text, then you can take the top-n words and sort them as they appear in the text to build an automatic summary.

In details, *TextRank* first adds each word to the graph and relationships are added between the word and others in a sliding window around the word. Then, an iterative ranking algorithm is run on each vertex:

$$S(V_i) = (1-d) + d * \sum_{j \in ngbr(V_i)} \frac{1}{|degree(V_j)|} S(V_j)$$

where d is damping factor that usually set to 0.85. It will update all of the word scores based on the related word scores, until all scores stabilize. This typically runs for 20-30 iterations.

Here is an graph illustration of the idea of $TextRank$:



Approach 2 - TextRank

$$\boxed{S(V_i) = (1-d) + d * \sum_{j \in ngbr(V_i)} \frac{1}{|degree(V_j)|} S(V_j)}$$

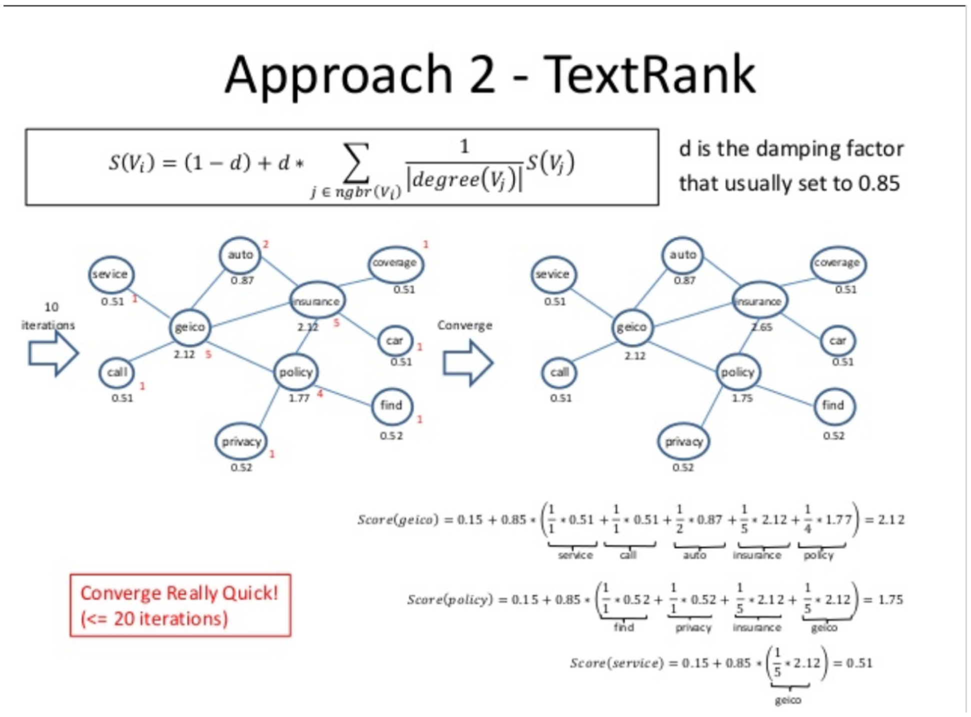
d is the damping factor that usually set to 0.85

$$Score(geico) = 0.15 + 0.85 * \left(\underbrace{\frac{1}{1} * 1}_{service} + \underbrace{\frac{1}{1} * 1}_{call} + \underbrace{\frac{1}{2} * 1}_{auto} + \underbrace{\frac{1}{5} * 1}_{insurance} + \underbrace{\frac{1}{4} * 1}_{policy}\right) = 2.65$$

$$Score(policy) = 0.15 + 0.85 * \left(\underbrace{\frac{1}{1} * 1}_{find} + \underbrace{\frac{1}{1} * 1}_{privacy} + \underbrace{\frac{1}{5} * 1}_{insurance} + \underbrace{\frac{1}{5} * 1}_{geico}\right) = 2.19$$

$$Score(service) = 0.15 + 0.85 * \left(\underbrace{\frac{1}{5} * 1}_{geico}\right) = 0.32$$

Figure 1: $TextRank$: 1st Iteration



Approach 2 - TextRank

$$\boxed{S(V_i) = (1-d) + d * \sum_{j \in ngbr(V_i)} \frac{1}{|degree(V_j)|} S(V_j)}$$

d is the damping factor that usually set to 0.85

$$Score(geico) = 0.15 + 0.85 * \left(\underbrace{\frac{1}{1} * 0.51}_{service} + \underbrace{\frac{1}{1} * 0.51}_{call} + \underbrace{\frac{1}{2} * 0.87}_{auto} + \underbrace{\frac{1}{5} * 2.12}_{insurance} + \underbrace{\frac{1}{4} * 1.77}_{policy}\right) = 2.12$$

$$Score(policy) = 0.15 + 0.85 * \left(\underbrace{\frac{1}{1} * 0.52}_{find} + \underbrace{\frac{1}{1} * 0.52}_{privacy} + \underbrace{\frac{1}{5} * 2.12}_{insurance} + \underbrace{\frac{1}{5} * 2.12}_{geico}\right) = 1.75$$

$$Score(service) = 0.15 + 0.85 * \left(\underbrace{\frac{1}{5} * 2.12}_{geico}\right) = 0.51$$

Converge Really Quick!
(<= 20 iterations)

Figure 2: $TextRank$: Itration stops when all scores stablize

These are the two algorithms ($td - idf$ and $TextRank$) that we will use to conduct keyword extraction task.

2.2 Opinion Mining on English Twitter Tweet Data

Users opinions are crucial for innovation and improvement on clients' related products. In order to ensure the development or idea is useful, it is important to do different researches and tests. Creating a user-friendly computer application and demonstration of using internet, software and machine learning algorithms will be our methods to achieve optimal opinion mining on these Twitter data.

2.2.1 Twitter and Its API

Twitter is an online English social media platform. Some US and overseas universities are using it as a channel to announce universities polices and activities. Twitter has also been used to receive students or clients feedback on certain topics. This is one of the reasons to use tweets in Twitter as our English dataset.

The data is crawled by using **Twitter API** developed by Twitter Developers. [**GET**] and [**POST**] request are used for crawling. Different Twitter user accounts might collect different dataset based on the query. The upper limit of tweets that each time can be requested for searching is fixed. If the search hits the upper limit, the search will be locked for a while. The data is collected based on user specified words or phrases. The collection time of 1000 data for keyword is about 20 to 30 seconds.

In the implementation of tweets crawling, dataset was obtained by using Twitter API (twitter.search(query) and result.getTweets()). Moreover, query.setLang("en") will be added into the algorithm in order to optimize the process of fetching data.

2.2.2 Sentiment Treebank

Sentiment Treebank is introduced in the paper written by a group of researcher in Stanford University. The idea of this algorithm is to break a sentence in several part and applying a linguistic mapping to fully label all the individual words(including punctuation) to a value ranged from 1 to 25. Value $\in [1, 12]$ represents negative connotation while value $\in [14, 25]$ represents positive connotation. The word that is classified as neutral or punctuation will be assigned a value = 13. The labels are initialized at the leaves, often appeared in the deepest nodes of the binary tree. We now have some useful labels and a well defined binary tree, it is sufficient to build a mathematical model or deep learning algorithm to interact and extract information based on this sentiment tree. Hence, it is ready to introduce another algorithm to traverse this sentiment tree.

2.2.3 Recursive Neural Tensor Network

Recursive Neural Tensor Network abbreviated as **RNTN** is an algorithm for addressing the complexity of semantic compositionality. Research shows that RNTN is better/superior compared to some of the previous algorithms such as Recursive Neural Network(RNN), Matrix-Vector RNN and some other graph-based neural network.[4] The RNTN can also capture the negation of the semantic. However, it is not easy to find an model that can deal with all situations. Many models or algorithms can have their own pitfall. RNTN is an improvement of the Matrix-Vector RNN algorithm. RNTN uses the weight and bilinear vector as usual as Matrix-Vector RNN or RNN, however, introducing an tensor matrix for the update of the previous model. The definition and error term are given in the following:

The RNTN uses this definition for computing p_1:

$$p_1 = f(\begin{bmatrix} b \\ c \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} b \\ c \end{bmatrix} + W \begin{bmatrix} b \\ c \end{bmatrix})$$

where W is as defined in the previous models. The next parent vector p_2 in the tri-gram will be computed with the same weights:

$$p_2 = f(\begin{bmatrix} a \\ p_1 \end{bmatrix}^T V^{[1:d]} \begin{bmatrix} a \\ p_1 \end{bmatrix} + W \begin{bmatrix} a \\ p_1 \end{bmatrix})$$

The errors as a function of the RNTN paramenters $\theta = (V, W, W_S, L)$ for a sentence is:

$$E(\theta) = \sum_i \sum_j t_j^i logy_j^i + \lambda||\theta||^2$$

Movies reviews are used in the Stanford CoreNLP training dataset. Our project use this training dataset as the starting point for the analysis of the sentiment scores for each tweets that collected.

Inside the implementation, each tweet will be analysed and quantified based on the polarity of the tweet (0 = very negative, 1 = negative, 2 = neutral, 3 = positive and 4 = very positive) by using Stanford CoreNLP and API (pipeline.process(tweet)). The returned value will be classified and counted using simple statistics. The analyzing time for 1000 tweets is about one to two minutes.

3 Results

3.1 Keyword Extraction on Chinese WhaleCollege Data

3.1.1 Jieba: a Python Package

The tool we are using is called Jieba, built to be the best Python Chinese word segmentation module. For this keyword extraction NLP task, 2 common algorithms *tf-idf* and *TextRank* will be used.

Jieba Package provides some functions including:

1. word-splitting

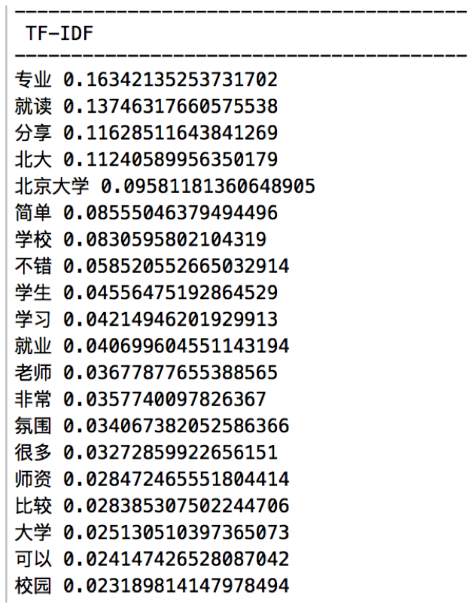
2. keyword extraction

3. Part-of-Speech Tagging

4. Tokenize: return words with position

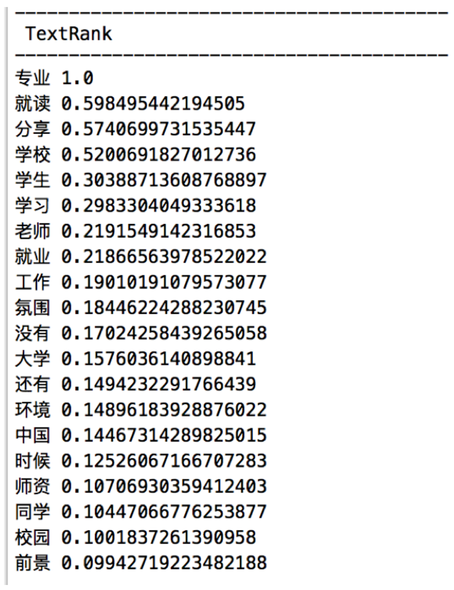
from which we will use keyword extraction based on both $tf-idf$ and $TextRank$ seperately.

3.1.2 Results on Peking University



```
-----------------------------------------
 TF-IDF
-----------------------------------------
专业 0.16342135253731702
就读 0.13746317660575538
分享 0.11628511643841269
北大 0.11240589956350179
北京大学 0.09581181360648905
简单 0.08555046379494496
学校 0.0830595802104319
不错 0.058520552665032914
学生 0.04556475192864529
学习 0.04214946201929913
就业 0.040699604551143194
老师 0.03677877655388565
非常 0.0357740097826367
氛围 0.034067382052586366
很多 0.03272859922656151
师资 0.028472465551804414
比较 0.028385307502244706
大学 0.025130510397365073
可以 0.024147426528087042
校园 0.023189814147978494
```

Figure 3: $tf - idf$ Ranking of Results



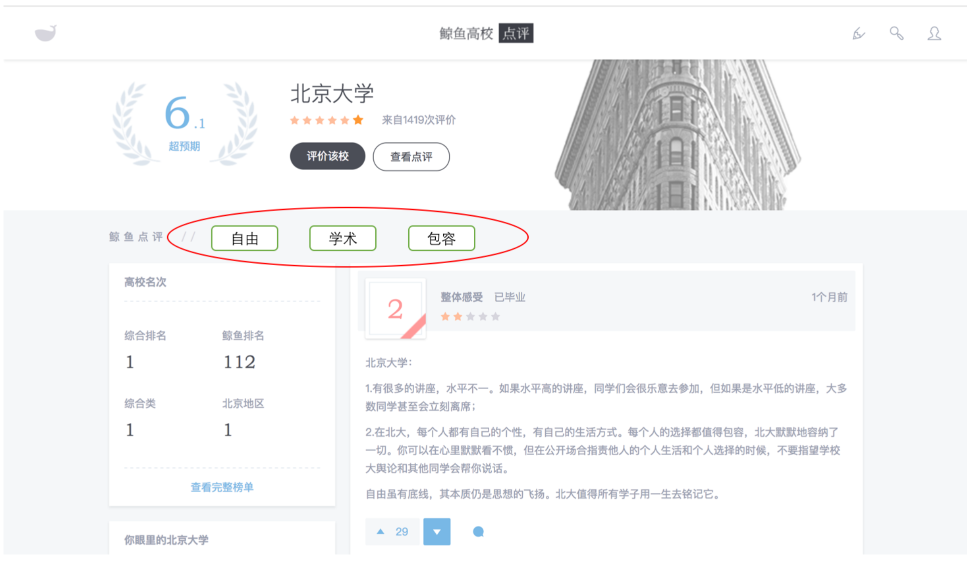
```
-----------------------------------------
 TextRank
-----------------------------------------
专业 1.0
就读 0.598495442194505
分享 0.5740699731535447
学校 0.5200691827012736
学生 0.30388713608768897
学习 0.2983304049333618
老师 0.2191549142316853
就业 0.21866563978522022
工作 0.19010191079573077
氛围 0.18446224288230745
没有 0.17024258439265058
大学 0.1576036140898841
还有 0.1494232291766439
环境 0.14896183928876022
中国 0.14467314289825015
时候 0.12526067166707283
师资 0.10706930359412403
同学 0.10447066776253877
校园 0.1001837261390958
前景 0.09942719223482188
```

Figure 4: $TextRank$ Ranking Results

Figure 4 and Figure 5 list the top-20 keywords generated by $tf - idf$ ranking and $TextRank$. The results look pretty similar generated by 2 different algorithms, from which we find that "nice uni", "nice prof", "good career outlook" can be used to describe Peking University.

3.1.3 Practicality

What is the practicality of conducting keyword extraction on these WhaleCollege university review data? Or, after extracting some keywords, how can we provide some suggestions for WhaleCollege to improve user friendliness for the user interface of their website? One flexible idea is that, we can add the keywords as tags on each own page of the university (Figure 6), so that when future visitors stop by the site, they can quickly have an idea about how this university is by reading these keywords.



北京大学

来自1419次评价

高校名次

综合排名	鲸鱼排名
1	112
综合类	北京地区
1	1

2 整体感受 已毕业 1个月前

北京大学：

1.有很多的讲座，水平不一。如果水平高的讲座，同学们会很乐意去参加，但如果是水平低的讲座，大多数同学甚至会立刻离席；

2.在北大，每个人都有自己的个性，有自己的生活方式。每个人的选择都值得包容，北大默默地容纳了一切。你可以在心里默默看不惯，但在公开场合指责他人的个人生活和个人选择的时候，不要指望学校大舆论和其他同学会帮你说话。

自由虽有底线，其本质仍是思想的飞扬。北大值得所有学子用一生去铭记它。

Figure 5: Proposed Tags Features on Websites

3.2 Sentimental Analysis on English Tweets Data

3.2.1 Stanford CoreNLP

The analysis tools chosen in this part is Stanford CoreNLP package for Java. A reasonable suggestion to use Java to analyze the sentiment of the data is easy to use and more robust for large project. The Stanford CoreNLP provides numerous methods for data analysis on multiple languages, however, the sentiment analysis version for Chinese is not yet supported. This is the reason the opining mining will only do English Twitter tweets. Another reason is that the pre-trained model by Stanford CoreNLP is based on English language and the data they trained although is hidden to user, they known to be employed the **Sentiment Treebank** and **Recursive Neural Tensor Network** machine learning algorithm.

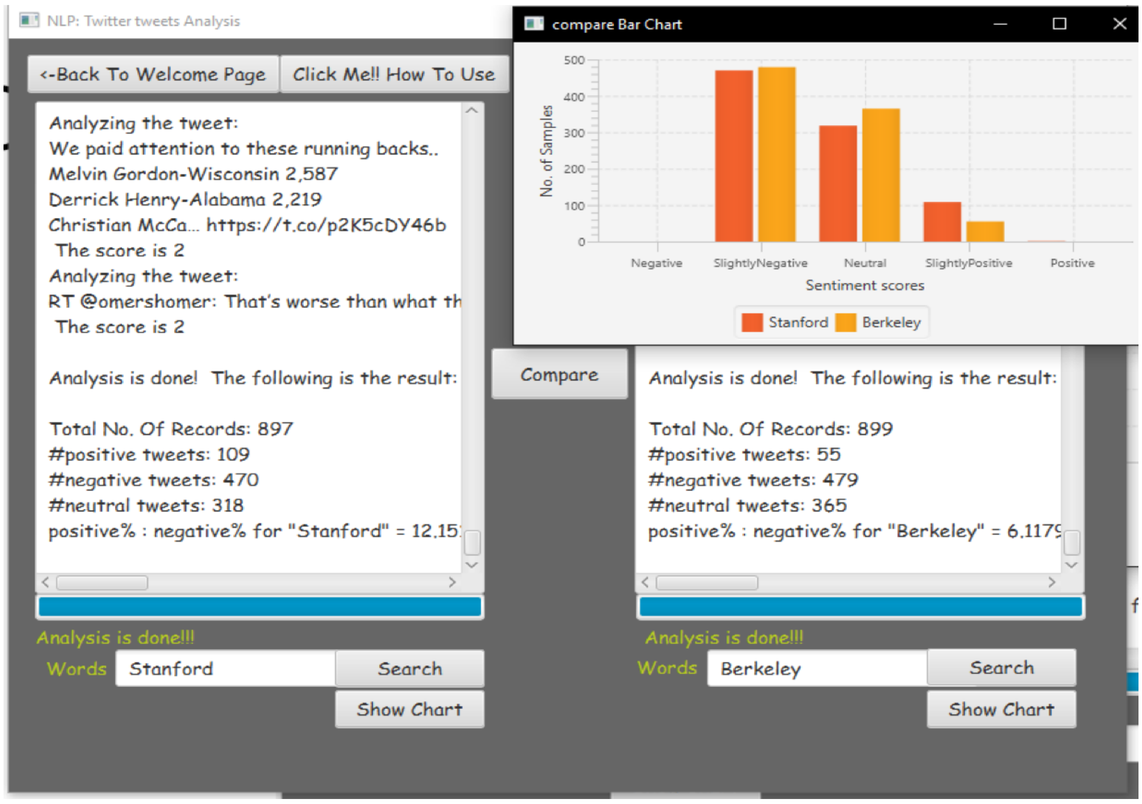
3.2.2 Graphical User Application

To be more user-friendly, our NLP program is coded together with Graphical User Interface (GUI) supported by JavaFx Application. You can find the open source project which we uploaded on this GitHub repo.

Since the priginal training data of Stanford CoreNLPis large, we are not supposed to upload it. If you would like to re-train the model, then you may need to download this library file from its website yourself. The other are zipped and attached in the GitHub project.

3.2.3 Results

Here is the demo of comparing the analyzed result of 2 top universities in US.



```
Analyzing the tweet:
We paid attention to these running backs..
Melvin Gordon-Wisconsin 2,587
Derrick Henry-Alabama 2,219
Christian McCa... https://t.co/p2K5cDY46b
 The score is 2
Analyzing the tweet:
RT @omershomer: That's worse than what th
 The score is 2

Analysis is done! The following is the result:

Total No. Of Records: 897
#positive tweets: 109
#negative tweets: 470
#neutral tweets: 318
positive% : negative% for "Stanford" = 12.15
```

Words: Stanford

```
Analysis is done! The following is the result:

Total No. Of Records: 899
#positive tweets: 55
#negative tweets: 479
#neutral tweets: 365
positive% : negative% for "Berkeley" = 6.1179
```

Words: Berkeley

Figure 6: Demo of Comparisons between Stanford and UC Berkeley

In the result of Figure 7, we can see that about half of the tweets are negative but there are very little positive tweets. This is an unexpected result. Research shows that the errors may come from the technology limitation of the NLP library itself. The machine is incapable of learning with insufficient information and instruction client's gave, due to the following 4 cases:

1. Ambiguous sentiment words
2. Missed negations
3. Quoted/Indirect text
4. Comparisons and anything subtle

The solution we suggest is that, in the implementation of the algorithm, we shall use population mean and ceiling function to reduce the negative or ambiguous sentences to make it shift to the more positive side. Another approach is to further breaking sentences into sub parts and individually rate them. This can be done by using Regular Expression (*Regex*) or by using custom delimiter to further filter out unwanted quotes or splitting the tweets into more precise sentence.

4 Conclusion

In this paper, we have conducted two NLP tasks, keyword extraction and sentiment analysis on Chinese and English datasets seperately. Although both performed reasonably well, further development can be made. The first one is to use a mixed training set. Because of dataset limitation, right now we can only train our model using movie review. However, if the training set can be varied, like a combination of movie review, university review and restaurant review, it would improve the performance of the model.

The second one is to develop a ranking system (Figure 8) by combining both models, sentiment analysis and keyword extraction. Lets say we want to rank the university based on 4 categories. The quality of the professors, the price of the tuition fee, the quality of the housing and the facility offered from the university. When we run the sentiment analysis program, there are many tweets that do not fit into these 4 categories. By using keyword extraction model, we can remove all the irrelevant tweets from the analysis. After we have a score for each category, we can take a weighted average to get the final score. This score will be used for ranking the university.

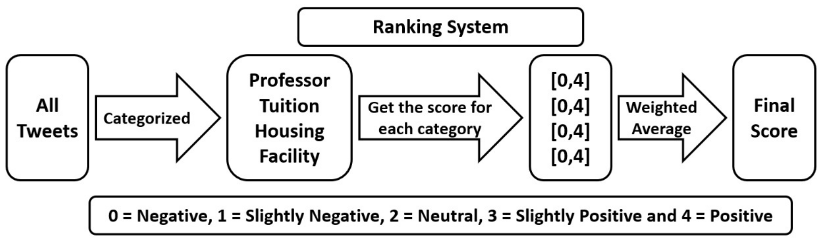

Ranking System
All Tweets
Categorized
Professor Tuition Housing Facility
Get the score for each category
[0,4] [0,4] [0,4] [0,4]
Weighted Average
Final Score
0 = Negative, 1 = Slightly Negative, 2 = Neutral, 3 = Slightly Positive and 4 = Positive

Figure 7: Proposed Ranking System

References

[1] FXSJY, *"Jieba" (Chinese for "to stutter") Chinese text segmentation*, URL: https://github.com/fxsjy/jieba

[2] C. D. MANNING, M. SURDEANU, J. BAUER, J. FINKEL, S. J. BETHARD, & D. MCCLOSKY, *The Stanford CoreNLP Natural Language Processing Toolkit In Proceedings*

of the 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations, pp. 55-60. [pdf] [bib], 2014.

[3] G. Salton, & M. J. McGill, *Introduction to modern information retrieval*, McGraw-Hill, 1983.

[4] R. Mihalcea, & P. Tarau, *TextRank: Bringing Order into Texts*, Proceedings of EMNLP, 2004.

[5] R. Socher, A. Perelygin, J. Y. Wu, J. Chuang, C. D. Manning, A. Y. Ng, & C. Potts, *Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank*, In Conference on Empirical Methods in Natural Language Processing, , 2013b.