Social graph based algorithms to improve business ranking, compared with simple rating-based ranking

The goal of analysis is to explore alternative business ranking approaches to traditional sorting based on rating, a basic stars and number of rewiews ranking. The dataset examined is from Yelp Dataset Challenge.

Introduction

The primary question the analysis aims to answer was:

"Can we define a more expressive rank of businesses (e.g. restaurants in Las Vegas) based on social network, different from simple stars based ranking?"

The ultimate goal is to try to improve business recommendation systems for Yelp users, providing them a better and more relevant ranking.

I decided to reduce the size of dataset selecting the business category "Restaurants" and city "Las Vegas".

I used a **graph based algorithms** to define a different rank of restaurants that take account of more social behavior "hidden" into users and review dataset features (using both **rating** and **text** in the reviews).

The algorithms applied to the graph are **PageRank** and **In-degree**.

The **nodes of graph are business** (in this case restaurants in Las Vegas) and the **edges represent the user preferences**. The model is based entirely on the social network and tries to calculate the **strength of each node** in the network.

The output from the analysis are **three lists with different rankings**, one for each method adopted, and interpretation of results. The results show significant variations and performance among the different methods.

The final results answer the primary question.

Methods and Data

Data Exploration

The object data involved are **users**, **reviews** and **business**. I obtained the data subset by filtering the data relating to **restaurants in Las Vegas, NV**. The table below shows dataset numbers:

Restaurants	Users	Reviews
4120	123615	370194

The simple rating sort ranking has been produced sorting by stars together with by count of reviews. According to that ranking the top three restaurants are:

Name	Stars	Review Count
Snow Ono Shave Ice	5	139
Mariscos Playa Escondida	5	103
Poppa Naps BBQ	5	38

The stars and review_count summaries are:

Column	Min.	First.Qu.	Median	Mean	Third.Qu.	Max.
stars	1.000	3.000	3.500	3.426	4.000	5.000
review_count	3.000	9.000	28.000	99.150	91.000	4578.000

Model Description

This section describes the four steps I followed to build the model.

Step#1 - User Authority Score Calculation

First of all I calculated the **user authority** score. As described below it is used to boost the weight of nodes graph relationships. The main idea behind assumes the value and impact of review with stars = 5, for example, from an authoritative user is different from another review with starts = 5, coming from a less influential one.

I calculated the user authority score using four variables in a simple linear polynomial expression multiplied with a boost factor. The used formula is:

$$score = boost * (\beta_1 X_1 + \beta_2 X_2 + \beta_3 X_3)$$

```
where
boost= K * (number_of_elite_years / numer_of_years_in_Yepl);
X1= review_count / numer_of_years_in_Yepl;
X2= number_of_fan / number_of_friends;
X3= votes_useful / review_count
K= 10, and other constants = 1
```

I used *z-score normalization* for all variables. I decide to use the "*elite*" information as boost because I think that is the most important. I also tried several experiments changing constants and the degree of the polynomial expression, but the impact on final results was not so evident.

Step#2 - Sentiment Analisys

The second step consists in running sentiment analysis for each text in the review. The sentiment score represent how "positive" or "negative" is the content in the review. I figured **sentiment analysis** as a classification problem, so I decide to use Multinomial Naive Bayes classifier, suitable for classification with discrete features. The **outcome variable is the review stars**. Different model has been tested using cross validation. I obtained the best performance by integer feature counts (worlds frequency), better than tf-idf based features vectorization. I considered the whole review dataset and selected the 30k features with the highest values of chi-squared to training and test the classification model. The average accuracy of the model for 10-fold cross-validation used is greater than 60%. For our purposes it is not particularly important to have an high accuracy because I applied the sentiment analysis just to define the "best" business in the pair preference comparing, in case the user's stars are the same for them (see below for more detalis). For each review **the final sentiment is the classification prediction**.

Step#3 - Social Business Graph Building

The graph is the heart of the model. The nodes represents the business related to the review. The arrows connecting them conveys the weight, which expresses the "strength" of that relationship. I built the graph from the *pair preferences* of users of Yelp social network.

For each user, who wrote at least two reviews, I considered the whole possible 2-combinations to build *pair preferences*. For example if user U wrote 3 reviews, r1, r2 and r3, their 2-combinations are `{r1,r2}, {r1,r3}, {r2,r3}` (the order of selection does not matter). For each pair review I calculated the user preference. So if for example the pair review is related to *business#1* (*B1*) and *business#2* (*B2*), the "*pair preference*" is the difference of stars the user assigned them. At this point I can add two nodes (*B1*,*B2*) into the graph and connect them with an edge, weighted by user preference. The intuition behind the *pair preferences* is **to exploit the different opinions of two business from the same user**. At the end I have a full weighted graph.

The algorithms require Weighted Directed Graph. The following pseudo-code better describes the social business graph building.

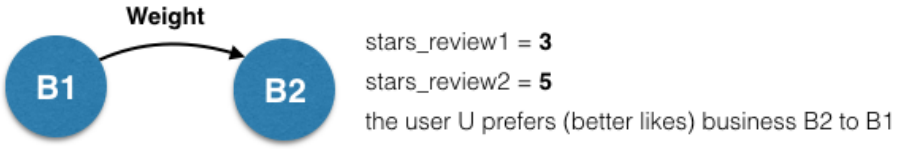
```
Intialize a weighted directed graph #G

FOR EACH user:
   get user's reviews list #reviews_list
   from reviews_list obtain 2-combinations list #2_combo_reviews_list
   FOR EACH pair in 2_combo_reviews_list:
      get the stars of 2 business          #(stars_B1, stars_B2)
      IF |stars_B1 - stars_B2| > 0:        #there is a winner, the user likes more
                                           #one of the two business
        insert into graph G two nodes B1 and B2 and
                                           #the directed edge connecting them with weight W
                                           #W is based on|stars_B1 - stars_B2| and user authority

      ELSE IF |stars_B1 - stars_B2| == 0:  #there is no a winner, the number of stars are the same
        calculate the sentiment analysis of the text in the two reviews
        #S1 = estimated_stars * estimated_probability_of_the_sentiment_classifier for review1
        #S2 = estimated_stars * estimated_probability_of_the_sentiment_classifier for review2
        IF |S1 - S2| > 0:                  #there is a winner
          insert to graph G two nodes B1 and B2 and the directed edge connecting them with weight W
                                           #W is based based on|S1 - S2| and user authority


# if the two nodes B1, B2 already exists into the graph G the weight W is updated with the new one.
```

The edge direction matters for the selected graph algorithms. If the user U prefers business B2 to business B1, the edge is from node B1 to B2. The picture shows an example where the incoming edge is to B2, the preferred business.



As shown in the pseudo-code, I applied the **sentiment analysis** only if the user assigned the same stars to both the business. In that case to find a "winner" between them I used the text inside each review to calculate an estimated sentiment. In both case **user authority** boosts the edge weight. The base idea

behind that score assumes that difference of preferences should take into account the authority of user, as author of review.

Step#4 - PageRank and In-degree Algorithm

Once the social graph has been built I can run the two algorithms on it to try different rankings: **PageRank** and **In-degree centrality**.

PageRank, originally designed as an algorithm to rank web pages by Google, computes a ranking of the nodes in the graph based on the structure of the incoming links. In other words, the importance of a node is determined mainly by its backlinks.

From the original paper:

> "We assume page A has pages T1...Tn which point to it (i.e., are citations). The parameter d is a damping factor which can be set between 0 and 1. We usually set d to 0.85. There are more details about d in the next section. Also C(A) is defined as the number of links going out of page A. The PageRank of a page A is given as follows:
>
> PR(A) = (1-d) + d (PR(T1)/C(T1) + ... + PR(Tn)/C(Tn))
>
> Note that the PageRanks form a probability distribution over web pages, so the sum of all web pages' PageRanks will be one. PageRank or PR(A) can be calculated using a simple iterative algorithm, and corresponds to the principal eigenvector of the normalized link matrix of the web."

The **In-degree centrality** for each node is the fraction of nodes its incoming edges are connected to it. From Wikipedia "In graph theory and network analysis, indicators of centrality identify the most important vertices within a graph."

Results

Model Comparison

The approach adopted to compare the three different ranking models (**PageRank**, **In-degree centrality** and **simple rating sort**) is based on splitting the dataset into training and testing data. Actually I sliced the dataset not with random shuffle, but cutting it in a specific day of the timeline, so that the data "before" the cut-day are 70% of the whole dataset, used for training, and the remaining data "after" are the 30% for testing. The cut-day was *2014-01-22*. The two algoritms are run on the training set. Their output are two ordered lists of business ranked according to the algorithm adopted. Essentially the task of comparing the three models is to find the list, among the three, with the "best" ranking.

To test the three models I used two metrics: **MAP (Mean Average Precision)** and the **number of wins**. The MAP (Mean Average Precision) is a very popular performance measure in information retrieval. It is a metric to interpret and compare multiple ordered correct answers in a list. In other words MAP compares two lists: the first one is considered the "*correct list*" (order does matter) and the second one is the *list of elements that are to be predicted* (order doesn't matter). I have just used the first K elements of the "*correct list*" to calculate the MAP. The following pseudo-code explains how I setup the testing:

```
Initialize a list of 10k users in a random way.         #random_10kusers
FOR EACH sample of 100 users from random_10kusers list: #sample_100users
  FOR EACH user in sample_100users
```

```
Get the business reviewed by he/she and by his/her friends from the test review dataset
                (after cut-day 2014-01-22) and remove duplicates  #sample_test_business

Calc MAP(pagerank_list, sample_test_business)
Calc MAP(indegree_list, sample_test_business)
Calc MAP(simple_rating_list,  sample_test_business)
Calc countWINS(sample_test_business,pagerank_list, indegree_list, simple_rating_list)
```

I ran the experiment with K = 10, 20, 30 The metric **number of wins** simply count how many times a business obtained by user/friends list is in a higher position than the three ranked lists. If for example business "l6QcUE8XXLrVH6Ydm4GSNw" is in position 89 in pagerank_list, position 53 in indegree_list and position 102 in simple_rating_list,I assign win to In-degree.

Discussion

The plots below show the perfomances of the three different ranking mechanisms per sample. A set of top businesses from this ranking could form a recommendation system. According to MAP metric **Pagerank** obtains the best performance. The second is In-Degree and the third is the simple rating. The main idea behind this approach is based on "*Homophily*". According to the principle of "*Homophily*" people are friends with people like them. User's context heavily influences their behavior. The `sample_test_business` represents the real users behavior after the cut-day and MAP measures how the three rankings would be "good" related to that users behavior. The second plot shows **In-degree** provides more wins than others.

Yes, it is possible to answer the primary question. We can have valid business rankings alternative to simple rating sorting ranking.

All the software I developed to implement the analysis is in R and Python, scikit-learn as machine learning library, NetworkX as graph library, numpy/scipy as math-stats library.