# New Recommendation System Approach by using Yelp Challenge Data

Salih Zeki Irkdaş Neslican Yüzak Engin Cabar

zirkdas,neslicanyuzak,engincabar@sabanciuniv.edu.tr

***Sabancı University Faculty of Engineering and Natural Science***

***Data Analytics' Masters' Degree Program Term Project***

***August 2015***

**ABSTRACT**

Recommendation systems are valuable to understand people's preferences. The most common problems of recommendation systems are data sparsity and cold start. In this project, we present a new recommendation system approach which may overcome these problems by taking Yelp Academic Challenge Dataset as a reference. In our approach, users' preferences are determined by TF-IDF scores with respect to business categories that are reviewed or scored by clients and similarity between two people is calculated by these TF-IDF scores. Two well-known rating prediction algorithms -Alternating Least Squares- (ALS) and -K Nearest Neighbor- are applied to predict the client's ratings. Prediction accuracies are measured by mean square error metrics. The local hub users who could be considered like "popular users" of each city are identified by using network analysis techniques and average ratings of local hub users are used to recommend a business for the users' that has no historical preferences. With the help of this approach data sparsity problem is solved by involving TF-IDF scores to rating prediction algorithms, additionally, we are able to give recommendations to cold starters (new users or new items) at Yelp.

**Keywords**

Recommendation Systems, Data Mining, Machine Learning, Yelp Challenge, Hadoop, Spark, Python, Network Analysis

## 1. INTRODUCTION

In today's rapidly changing world, understanding users' expectations and preferences are crucial issue for sustainability and profitability of businesses. Recommendation systems provide customized recommendation for each user according to his/her previous ratings. Many companies such as Amazon and Netflix integrate recommendation systems to have better customer experience and improve their sales. The most commonly used recommendation system methodologies are content based filtering and collaborative filtering. Basically, content based filtering takes into consideration of properties of items and collaborative filtering takes into consideration the similarity of users and items. However the traditional recommendation algorithms cannot satisfy while dealing with the big data. The domain knowledge on the specified area and developing specific algorithms that cover the dynamics of problematic environment become vital.

Most of the real-world datasets are suffer from data sparsity and cold start. They have a negative effect on performance and accuracy of recommender systems. Lacks of user-item rating existence, recommendation systems do not have adequate information to extract users' preferences. Moreover, recommendation systems cannot find any historical record to predict ratings for cold starts.

In this project, we focus on developing alternative solution to improve recommendation system

accuracy by handling data sparsity and cold start. We try to minimize the effects of data sparsity problem by using TF-IDF scores of business categories and convert the dataset in a denser form. Furthermore, we try to find-out the local hub users to give suggestions to cold start users.

## 2. RELATED WORKS

In this section, we briefly present some of the researches related with collaborative filtering, content-based recommender systems, data mining, graph-analysis, sparsity and cold-start.

One of the base recommender systems is collaborative filtering algorithm which focuses on to find similar users with using their ratings. This kind of systems are collecting users rating with user interface based programs and recommend some items such as music, movie and etc. For example, the GroupLens research system presents some movies to collect ratings from you and finally recommends some items. [1]

Other approach is content based filtering which uses items preferences to identify similar users taste. Content-based recommendation systems analyze item descriptions to identify items that are of particular interest to the user. [2]

Other technologies have also been applied to recommender systems, including Bayesian networks and clustering. Bayesian network creates a model based on a training set with a decision tree at each node and edges representing user information. [1]

In graph-based approaches, the data is represented in the form of a graph where nodes are users, items or both, and edges encode the interactions or similarities between the users and items. [3] . A weight can also be given to this edge, such as the value of its corresponding rating. In another model, the nodes can represent either users or items, and an edge connects two nodes if the ratings corresponding to these nodes are sufficiently correlated. The weight of this edge can be the corresponding correlation value.

Chen et al. [4] presents the association retrieval technology to alleviate the sparsity problem and proposed a new collaborative filtering algorithm to increase the recommendation precision. Cold start problem affects collaborative filtering suffers from what is called the cold-start problem, due to its inability to address the systems new products and users. Although alternative ways are tried to overcome cold start problem, most of them use hybrid model that combines content based and collaborative based information to recommend. [7] [8] However, finding content based information is not easy in some cases. Instead of content based information, some researches work on network relationships. Article of Sahebi and Cohen [9] uses external social network connections of a new user to extract preferences of him/her. Although this approach looks like meaningful, it is not applicable for our project. Yelp Academic Challenge dataset provides user id information as encrypted format and shares only name of the user. Moreover, there is no Facebook ID / Twitter ID information available in our dataset.

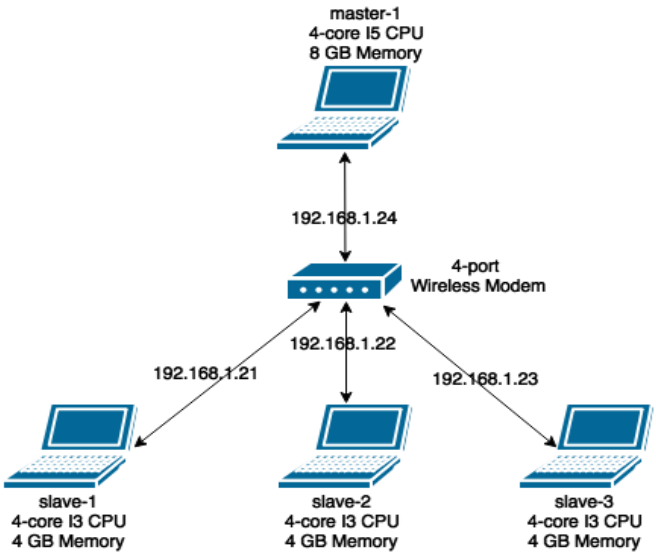
Our work explores the extent to which sparsity and cold-start problems with aid of graph based modelling and TF-IDF.

## 3. BIG DATA PROCESSING ENVIRONMENT

Data is getting bigger day by day and it requires set of techniques and technologies to uncover hidden values that lies beyond the massive datasets. In our case, we had a text-based dataset that have a size with more than 1.6 gigabytes. It is so easy to face with so-called -out of memory error, when trying to run machine learning algorithms on this amount of data with traditional data mining tools like Rapid Miner, Weka or machine learning libraries like Python's scikit-learn. Even those traditional software tools help to reduce the learning curves of the implementation of the algorithms with their user-friendly graphical interfaces; they all suffer from lack of computation power on commodity hardware and lower the performance metrics.

Apache Hadoop framework which brings distributed storage and distributed processing capability on large datasets stands as the state-of-the-art technology among other data processing environments. The framework can build a cluster of computers which act like a single machine. Besides its core elements HDFS and Map Reduce, it includes many useful tools to help discover,

prepare, transform, model and visualize the big data in its ecosystem. The strength of Apache Hadoop framework leads us to build or own Hadoop Cluster to process our Yelp Dataset in an efficient way.




**Graph 1. Big Data Processing Environment of the Project**

In order to get high performance from the computers we decided to run Apache Hadoop natively on 4-machine cluster instead of using pre-build virtual machines. The cluster provides us approximately 20 GB of memory and 16-core CPU (~2 GHz per core). The steps that we followed during creation of the cluster;

1. Download Ubuntu 12.04-LTS.
2. Install Ubuntu 12.04-LTS.
3. Set static IP addresses.
4. Configure FQDN hostnames (master-1, slave-1, slave-2, slave-3).
5. Configure known-hosts to introduce computers each other.
6. Disable firewalls and IPv6.
7. Ping Hosts to verify connection.
8. Install ssh-server and ssh-client.
9. Configure ssh-keygen certificates to allow passwordless access.
10. Install JDK 1.7
11. Install Maven 3.0.4

After creation of the local area network we analyzed the popular Linux-based Apache Hadoop distributions and decided to build a Hadoop Ecosystem with Cloudera CDH version 5.4 that includes many Hadoop related data processing software applications like Hive, Oozie, Hue or Sqoop and additionally machine learning frameworks Spark and Mahout.

## 4. DATASET

In this section, dataset that is used for generating a recommendation system is presented with details. Yelp is a popular business and restaurant review web-site that is founded in 2004 headquartered in San Francisco, California which allows people to review businesses and share information about them. People can write their experiences and make connections by firstly creating an account on yelp.com. After signing up, users can search for businesses, write reviews and give the business a star rating from 1 to 5. Yelp.com uses automated software to make recommendations looking at various signals such as measures of quality and reliability and activity on Yelp. As of Q1 2015, Yelp.com had an average of approximately 142 million monthly unique visitors and 77 million local reviews in the website.

The dataset that is used in our project is available at http://www.yelp.com/dataset_challenge. Dataset includes 61K business, 366K users, 1.6M reviews, 500K tips and 45K check-in information in separate 5 json files. In our recommendation project, business, users and reviews files are used.

### 4.1. Descriptive Statistics & Exploratory Data Analysis

In our project, Python Pandas and Apache Solr / Hive are used for understanding data. Dynamic dashboards for dataset are designed in Apache Solr for visualization. Solr is the popular, fast, open source NoSQL search platform from the Apache Lucene project. Main features of Solr are text, timeline, line, bar, map, filters, grid, pie widgets, drag & drop dashboard builder and dynamic updated interface.

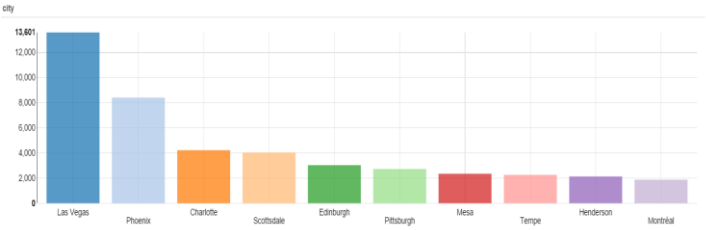
#### 4.1.1. Business Dataset

Business dataset includes 61,184 businesses information with below details:

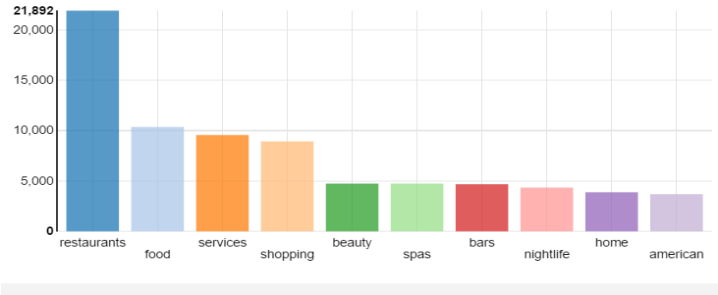
- business_id: vcNAWiLM4dR7D2nwwJ7nCA
- categories: [Doctors, Health & Medical]
- city: Phoenix
- full address: 4840 E Indian School Rd…
- hours: {Tuesday: {close: 17:00, open: 0...}

- latitude: 33.499313
- longitude: -111.983758
- name: Eric Goldberg, MD
- neighborhoods: []
- open: True
- review_count: 9
- stars: 3.5
- state: AZ
- attributes: {By Appointment Only: True}
- type: business

Business dataset contains 378 cities, 26 states and 783 different categories.

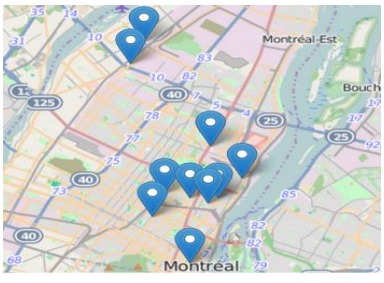



**Graph 2. City Based Business Distribution top 10**




**Graph 3. Category Based Business Distribution top 10**

Latitude and longitude information allow us to visualize businesses on map by using Solr. Graph 4 shows the restaurants whose stars in range 4-5 on map.




**Graph 4.Montreal Restaurants Star 4-5**

Business dataset includes 2 numeric features. When we look at the descriptive statistics of these features, both of them are found as a right skewed distribution (mean>median). Furthermore, Pearson correlation is found as significant for these variables with %5 significance level (p value=1.29e-08)

| | review_count | stars |
|---|---|---|
| count | 61184.000000 | 61184.000000 |
| mean | 28.272506 | 3.673305 |
| std | 88.652050 | 0.891207 |
| min | 3.000000 | 1.000000 |
| 25% | 4.000000 | 3.000000 |
| 50% | 8.000000 | 3.500000 |
| 75% | 21.000000 | 4.500000 |
| max | 4578.000000 | 5.000000 |

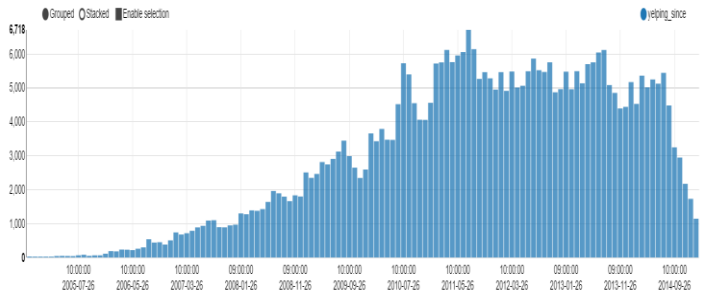
**Table 1 Descriptive Statistics of Stars and Review Count**

### 4.1.2. User Dataset

User dataset includes 366,715 users information with below details:

- type: user
- user_id: 18kPq7Gpye-YQ3LyKyAZPw
- name: Russel
- review_count: 108
- average_stars: 4.14
- votes: {funny: 166, useful': 278, cool: 245}
- friends: [(friend user_ids)],
- elite: [2005, 2006]
- yelping_since: 2004-10,
- compliments: { (compliment_type): (num_compliments_of_this_type)}
- fans: 69

Number of yelping users shows rapidly increase between 2005 and 2011, stability between 2011 and 2014, decrease since 2014-Q2.

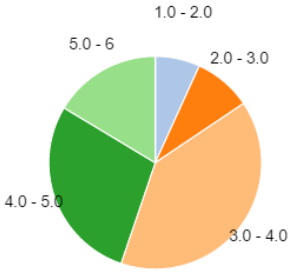



**Graph 5. Yelping Since**

User dataset includes 2 numeric features. When we look at the descriptive statistics of these features, distribution of "review count" is found as right skewed distribution (mean>median) however distribution of average stars has got a left skewed distribution (mean<median).

| | review_count | average_stars |
|---|---|---|
| count | 366715.000000 | 366715.000000 |
| mean | 32.214810 | 3.718782 |
| std | 94.837065 | 1.029877 |
| min | 0.000000 | 0.000000 |
| 25% | 2.000000 | 3.270000 |
| 50% | 6.000000 | 3.860000 |
| 75% | 21.000000 | 4.430000 |
| max | 8843.000000 | 5.000000 |

**Table 2 Descriptive Statistics of Average Stars and Review Count**




**Graph 6. Average Star**

As you see, some users do not have any review and they look like a cold start.

### 4.1.3. Reviews Dataset

Reviews dataset includes 1,569,264 reviews information with below details:

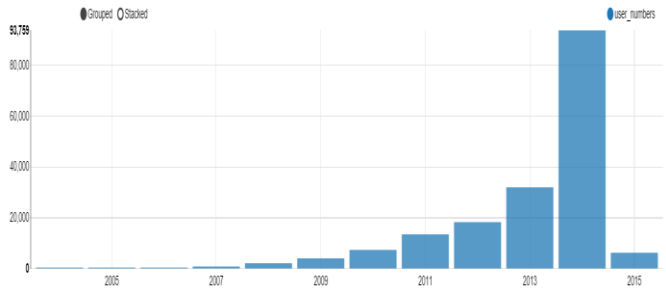
- 'type': 'review',
- 'business_id': 0vcNAWiLM4dR7D2nwwJ7nCA
- 'user_id': Xqd0DzHaiyRqVH3WRG7hzg
- review_id: 15SdjuK7DmYqUAj6rjGowg
- 'stars': 5
- 'text': "dr. oldberg offers everything i look for in.."
- 'date': 2007-05-17
- 'votes': {funny: 0, useful: 2, cool: 1}

Review dataset includes only one numeric feature. When we look at descriptive statistics of it, distribution of "star" is found as left skewed distribution (mean<median)

| | stars |
|---|---|
| count | 1569264.000000 |
| mean | 3.742656 |
| std | 1.311468 |
| min | 1.000000 |
| 25% | 3.000000 |
| 50% | 4.000000 |
| 75% | 5.000000 |
| max | 5.000000 |

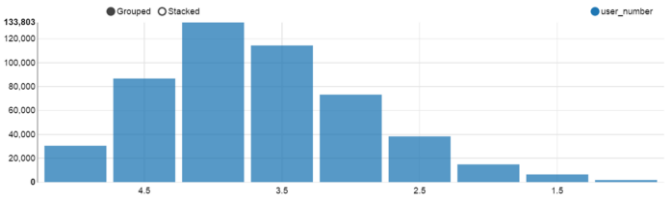
**Table 3. Descriptive Statistics of Stars**

Review dataset will be used for generating user-item matrix for extracting users' preferences in our further analysis. To have more accurate results, up to date review dataset is necessary. Because of that reason, we control the last review year of each user in Graph7 and the most of the users' last review year are found in 2014.



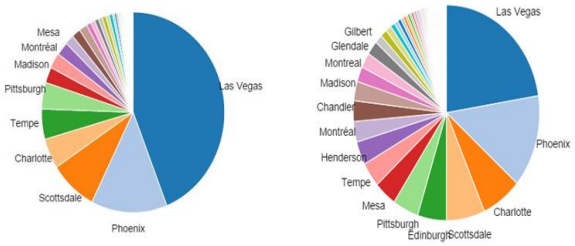

**Graph 7. Last Review Year vs Number of Users**

Moreover, most of the users tend to give reviews for businesses whose ratings are higher than 3.




**Graph 8. Business Stars vs Number of Reviewed Users**

When we try to understand city preferences of users, the most frequent reviewed city is selected as a preferred city of a person and below city preferences distribution is generated. Although the Edinburg is the 5$^{th}$ city according to business numbers, Edinburg is not in the top 5 list for city preferences. It is a good indicator to take into consideration location based information for our further analysis.




**Graph9. City Preferences vs Business City**

Results that are found by using descriptive statistics and basic analysis are as below:

- Yelp Challenge Academic Dataset has got data sparsity and cold start problem. %99.99 of user-item matrix is empty. In our modelling approach we have to find alternative solutions to prevent data sparsity and cold start problem.
- City preferences percentage is not same with the business city percentage. In our modelling approach we have to take into consideration location based information.
- Category distribution of businesses is dominated by Restaurants. To extract users' preferences, we have to take into consideration category differentiation.

### 4.2. Preprocessing

Dataset taken from Yelp Academic Challenge for our project is in json format. Unfortunately Hadoop Hue does not support transferring json files to HDFS. To eliminate this problem, a code that is converting json files to csv format is designed by using Python. After the converting process is done, metadata tables are created for business, users and review datasets in HDFS.

While transferring dataset to Hadoop environment, some features such as review text information are not selected to optimize storage capacity of HDFS.

Recommendation algorithms that we plan to use require user and item pairs as numeric variables. Because of that reason, string variables for user id and business id are converted to numeric variables by using hash function of Hive.

According to descriptive statistics, we have already found that some users have got zero or one review. To build more accurate recommendation systems, we selected users who have got more than one review record in our database for modelling phase. After this selection, our final Hive tables include 178,455 users and 1,381,004 reviews. On the other hand, filtered users are decided to use as cold start.

## 5. MODEL

In this section, our modelling approach is presented with details.

### 5.1. Keyword-based Vector Space Model by using TF-IDF scores

User-item matrix of our dataset shows %99.99 sparsity. Lack of user-item pair existence, recommendation system that we try to generate does not have adequate information to extract users' preferences by using user-item matrix directly. Although dataset includes 61,184 businesses, total number of business categories that show us the category preferences of users is 783. If a user likes a category, we expect to have more reviews in this category for him/her. Because of that reason, we decided to extract users' preferences by using the category of their previous reviews to overcome data sparsity problem. However, number of reviews for restaurant category is generally higher than other categories in our dataset. If we use directly number of reviews for each category, we would possible to find that restaurant is the first preference of all users at the end of the analysis but it is not a proper way to understand user's preferences correctly. If a user has got reviews of a category that is rarely reviewed by other users, it means that the user prefers this category more prominently. [5]

To take into consideration this kind of differentiation, we decided to calculate TF-IDF (term frequency- inverse document frequency) scores of each category for each user. To calculate TF-IDF scores, review history of a user is used as a document and categories are considered as terms in the document. In our dataset, review data does not contain business categories. To prepare suitable data format to implement TF-IDF, first of all, we use Hive to join tables. Dataset is converted to the below format after joining process.

| user_id | business_id | categories |
|---|---|---|
| 1111 | 25252 | Food, Nightlife |
| 1111 | 52468 | Restaurants, Kebab |
| 1111 | 7874 | Healt and Medical |

Table 3.Hive Joined Table Result

However, we need to create one document for a user that includes all categories that user has already reviewed. For this converting process, we write a User Categorization Matrix Map-Reduce code. In this code, mappers split the data according to delimiter of categories variable and reducers append categories. After this map-reduce process, data is converted to below format.

| user_id | categories |
|---|---|
| 1111 | Food, Nightlife, Restaurants, Kebab, Healt and Medical |

**Table 4. User Categorization Matrix Map Reduce Result**

In our approach, TF (Term frequency) which calculates how frequently a term occurred in the document is calculated by using below equation. Since each document has got a different length and a term occurs much more time in a long document, we prefer to use term frequency by dividing length of document to normalize TF values. [6]

$$\mathrm{TF}(t, d) = \frac{(\text{Number of times term t appears in a document})}{(\text{Total number of terms in the document})}$$

IDF (Inverse Document Frequency), which is used to measure importance of term is calculated by using below equation where N shows the total number of documents (in our case: total number of users), $\{d \in D : t \in d\}$ shows the number of documents that includes term t (in our case: number of users who has reviewed category i )

$$\mathrm{IDF}(t, D) = \log \frac{N}{|\{d \in D : t \in d\}|}$$

Finally, TF-IDF ((term frequency- inverse document frequency) scores are calculated by using below equation.

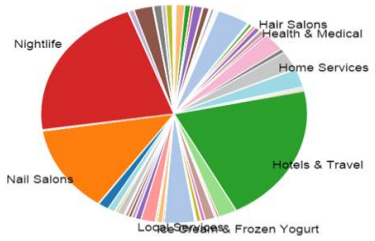
$$\mathrm{TFIDF}(t, d) = \mathrm{TF}(t, d) \times \mathrm{IDF}(t, D)$$

Due to the size of the dataset, we also implement 3 Map-Reduce codes for TF, IDF and TF-IDF calculation. Result of the TF-IDF Map-Reduce code creates a table as below data format.

| user_id | category | tf | idf | tfidf |
|---|---|---|---|---|
| 2222 | Restaurants | 0,5 | 0,01 | 0,005 |
| 2222 | Automotive | 0,3 | 0,5 | 0,15 |
| 2222 | Museums | 0,2 | 2 | 0,4 |

**Table 5. TF-IDF Map Reduce Table**

Graph 10 shows the max TF-IDF score distribution for users in our dataset.




**Graph 10. Max TF-IDF Score Distribution**

Our main goal is identification of similar users by using TF-IDF scores. We plan to use Spark Pyspark MLlib Library to create user- similarity matrix by using Pearson Correlation. Pearson correlation of MLlib library requires RDD vector for calculation and calculates correlation column based. For pivot operation we convert our RDD object to Pandas. After pivot function is applied, data frame is converted to RDD vector to calculate correlations. Due to the number of column limitation of Spark (65K), user-similarity matrix is not created. However, we plan to calculate similarity between two users iteratively in the running recommendation algorithm instead of directly using user-similarity matrix as an alternative solution for column limitation problem.

TF-IDF score similarity is valuable for overcoming the data sparseness problem and calculating similarity between users who do not have any review for the same business.

### 5.2. Running Algorithms

In our project, we prefer to use Spark capabilities for running algorithm part. Apache Spark is faster and cluster computing system. It uses HDFS as a file system like Hadoop. Due to the fact that, it works on in-memory, it is 100 times faster than Hadoop. Its infrastructure is based on Resilient Distributed Datasets (RDD) system. RDD is a distributed memory abstraction that lets programmers perform in-memory computations on large clusters in a fault-tolerant manner [10]

Every Spark application has a driver program which consists of main function and makes parallel processing on each node at cluster. Moreover, RDD can be persisted in the memory to allow it to be reused efficiently across parallel operations.

### 5.2.1. Alternating Least Squares (ALS)

Alternating Least Squares algorithm in Spark is the same as matrix factorization algorithm. This algorithm uses latent factors to extract hidden tastes of users Latent factor models are an alternative approach that tries to explain the ratings by characterizing both items and users on, say 20 to 100 factors inferred from the ratings patterns. Latent factor is used for decomposing NxM matrix into NxK and KxM matrices where K is the latent factor.

In our project, we have used matrix factorization methods from Apache Spark MLlib. These techniques aim to fill in the missing entries of a user-item association matrix. MLlib currently supports model-based collaborative filtering, in which users and products are described by a small set of latent factors that can be used to predict missing entries. MLlib uses the alternating least squares (ALS) algorithm to learn these latent factors. We have preferred to use this method as a baseline model to predict missing rates. Due to the data sparsity, prediction power is not found as adequate. Table 6 represents the performance of the model by changing lambda, rank and number of iterations. Lambda is used to specify regularization parameter and default value is 0.01, iterations is the number of iterations to run and default value is 10, rank is the number of latent factors in the model and default value is 10. Increasing rank parameter shows better results however increases the cost of computation. By using below parameters, we have selected lambda=0.5, rank=75 and number of iterations=30 as a baseline model with 1.82 MSE on test dataset.

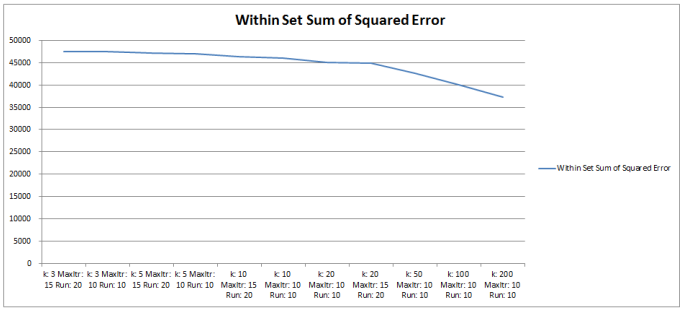
| Lambda | Rank | Number of Iterations | Mean Squared Error |
|---|---|---|---|
| 0,1 | 25 | 25 | 2,05 |
| 0,1 | 50 | 10 | 2,21 |
| 0,1 | 50 | 20 | 2,02 |
| 0,1 | 50 | 25 | 2 |
| 0,5 | 75 | 30 | 1,82 |

**Table 6. MSE of ALS Algorithm - Baseline**

To improve accuracy of the model, we have decided to cluster dataset by using MLlib library K-Means algorithm. TF-IDF vectors are used to calculate similarity between two users as a similarity metric for clustering. Results of the clustering are showed as below. k is the number of desired clusters, max Iterations is the maximum number of iterations to run, initialization Mode specifies either random initialization or initialization via k-means, runs is the number of times to run the k-means algorithm. MLlib K-Means algorithm returns the within set sum of squared error (WSSSE)

| Number of Clusters | MaxIterations | InitializationMode | Runs | Within Set Sum of Squared Error |
|---|---|---|---|---|
| 3 | 15 | Random | 20 | 47462 |
| 3 | 10 | Random | 10 | 47459 |
| 5 | 15 | Random | 20 | 47169 |
| 5 | 10 | Random | 10 | 47002 |
| 10 | 15 | Random | 20 | 46298 |
| 10 | 10 | Random | 10 | 46056 |
| 20 | 10 | Random | 10 | 45049 |
| 20 | 15 | Random | 20 | 44846 |
| 50 | 10 | Random | 10 | 42635 |
| 100 | 10 | Random | 10 | 40113 |
| 200 | 10 | Random | 10 | 37314 |

**Table 7. WSSSE of K-Means**



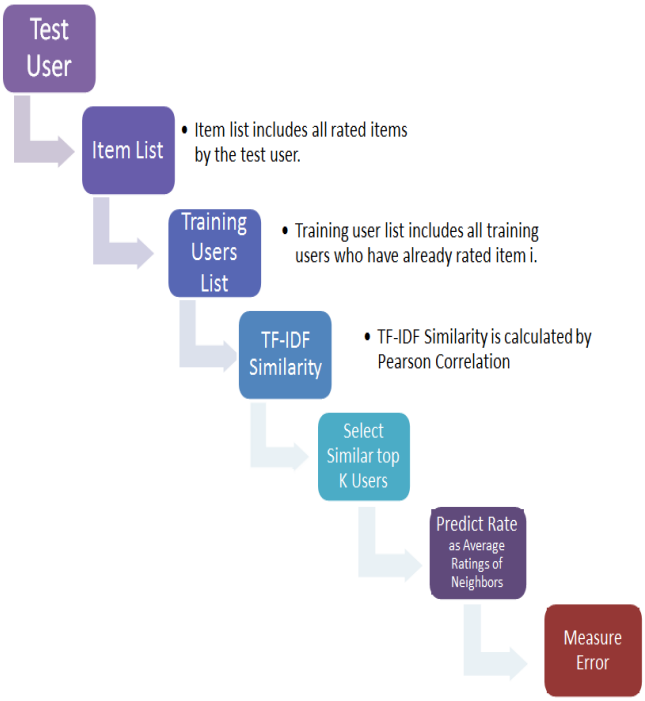

**Graph 11. WSSSE Graph**

Although we have expected to find elbow shape graph, clustering results does not show this kind of property. Because of that reason, we have preferred to select the point that shows rapidly decreasing k=20 as number of clusters with max iterations=15 and runs=20 parameters. . After the selection of number of clusters, ALS method is applied on each cluster and then weighted average of MSE is calculated as 1.50. It means that, TF-IDF similarity improved accuracy of ALS.

### 5.2.2. K-Nearest Neighbors (K-NN)

K-Nearest Neighbor Algorithm is one of collaborative filtering techniques for recommendation systems. K-NN algorithms compute similarity between users or items. According to similarity measure, K-NN finds the nearest K neighbors and uses ratings of neighbors to predict unknown rates. Although simplicity of implementation and explainable predictions are advantages of the algorithm, computing similarity for all dataset takes time by using K-NN algorithm.

In our modelling approach, we prefer to use TF-IDF vectors to calculate similarity between users. Moreover; dataset is divided to two sets as training and test to measure performance of recommendation model. Process flow is also represented as below.




**Graph 12. K-NN Process Flow**

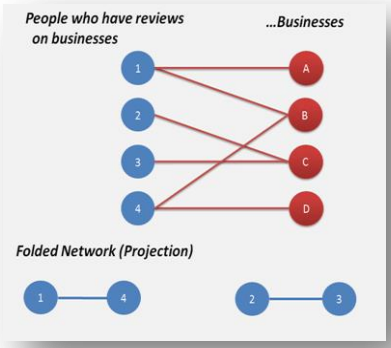
Due to the iterative process, running algorithm by using Spark takes much more time than expected. Although Spark code runs, results are not reachable in a day.

### 5.2.3. Cold Start Solution

Cold start problem is one of the typical problems of recommendation systems due to the lack of information on users or items. If a new user enters the system, recommendation engine cannot predict preferences of the new user and wait until collection of adequate number of historical records for him / her.

In our project, we prefer to identify local hub users for each city. Although we have done researches to find graph library in Hadoop or Spark, suitable library could not find for them and we decided to use Python NetworkX for creating a graph of each city. In this approach, network is designed by using review information of users. City of reviewed business is used for filtering data per city.

First of all, review data is transferred to Python and stored as pandas data frame. Secondly, bipartite graph whose nodes represent both users and businesses and edges represent the user-businesses relationship is generated for each city and projected to folded network. In the folded network, nodes are represented as users. Folded network has got an edge between two users if they have reviewed at least one common business.




**Graph 13. Sample Bipartite & Folded Network**

Thirdly, HITS algorithm is applied to find local hub users of each city. Hub scores are used to represent knowledge of the user for selected city. In our analysis, top 1K hub users are identified and stored in a csv file.

Instead of preferences of all users, average ratings of local hub users is used to recommend a business for cold start users in our modelling approach.

## 6. FUTURE WORKS

While processing our large dataset, we faced with many performance issues. To overcome this

problem, we will improve our machine learning algorithms to process our data in more efficient way. Additionally, we are planning to move our cluster to a cloud based web services like Amazon Web Services (AWS) in order to utilize our system with more computation power

## 7. CONCLUSION

In this project we have found a chance to apply many concepts that we have learned during the master program on a real dataset. We had got familiar with the life-cycle of a data analytics project and understood the importance of data preparation, transformation and data munging. The effects of the size of data on algorithm performances has been observed and experienced. The descriptive statistics results revealed the sparsity and cold start problems. In order to prevent the sparsity problem, users' preferences have been determined by TF-IDF scores of business categories. To overcome cold start problem, location-based network analysis techniques have been applied and preferences of local hub users have been used as a reference for the cold start users. Two widely-used algorithms, Alternating Least Squares and K-NN algorithms are used for modelling and rating prediction. The process of calculating the similarity between two users was too complex so that the K-NN algorithm has not returned with any result in 24 hours. For alternating least square algorithm, base model compared with K-Means clustering results. We observed that using TF-IDF similarities within the ALS recommender algorithm improved the accuracy of the base model 18%.

## REFERENCES


[1] B. Sarwar, G. Karypis, J. Konstan ve J. Riedl, «Item-Based Collaborative Filtering Recommendation».

[2] M. Pazzani ve D. Billsus, «Content-based Recommendation Systems».

[3] F. Ricci, L. Rokach, B. Shapira ve P. Kantor, «Recommender Systems Handbook».

[4] Y. Chen, C. Wu, M. Xie ve X. Guo, «Solving the Sparsity Problem in Recommender Systems Using Association Retrieval,» *Journal of Computers,* cilt 6, no. 9, pp. 1896-1902, 2011.

[5] E. Spyromitros-Xioufis, E. Stachtiari, I. Vlahavas ve G. Tsoumakas, «A Hybrid Approach for Cold-start Recommendations of Videolectures».

[6] A. Schein, A. Popescul, L. Ungar ve D. Pennock, «Methods and Metrics for Cold-Start Recommendations,» *University of Pennsylvania,* 2002.

[7] S. Sahebi ve W. Cohen, «Community-Based Recommendations: a Solution to the Cold Start Problem».

[8] J. Bao, Y. Zheng ve M. Mokbel, «Location-based and Preference-Aware Recommendation Using Sparse Geo-Social Networking Data».

[9] Jain, Behera, Mandal ve Mohapatra, «A Novel Modified Apriori Approach for Web Document Clustering,» *Computational Intelligence in Data Mining - Volume 3*, 2014, p. 163.

[10] M. Zaharia, M. Chowdhury, T. Das, A. Dave, J. Ma, M. McCauley, M. Franklin, S. Shenker ve I. Stoica, «Resilient Distributed Datasets: A Fault-Tolerant Abstraction for In-Memory Cluster Computing».