Project Report on

Classification Of Similar Users On Twitter

Rishab Ketan Doshi (12IT59)

Rohit John Joseph (12IT61)

Shravan Karthik (12IT77)

Siddharth P Ramakrishnan (12IT79)

Under the Guidance of,

Mr. Ram Shastry

Department of Information Technology, NITK Surathkal

Date of Submission: 7th April,2015

in partial fulfillment for the award of the degree

of

Bachelor of Technology

In

Information Technology

At

Department of Information Technology

National Institute of Technology Karnataka, Surathkal

April 2015

Department of Information Technology, NITK Surathkal
Minor Project
End Semester Evaluation Report (April 2015)

Course Code : IT 399

Course Title: Minor Project

Project Title: *Finding Similar Users On Twitter*

Project Group:

Name of the Student	Register No.	Signature with Date
Rishab Ketan Doshi	12IT59	
Rohit John Joseph	12IT61	
Shravan Karthik	12IT77	
Siddharth P Ramakrishnan	12IT79	

Place:

Date: *(Name and Signature of Minor Project Guide)*

Abstract

SimTwit is a product that identifies similar sets of users on Twitter based on the content of users tweets. The product scans through the list of all tweets associated with a user, builds a feature list and semantically compares each users feature list to compute a score vector associated with K sets of parameters, K is the number of distinct groups a user can be classified into. The score vector is then fed to advanced learning algorithms, both unsupervised and supervised to compute K distinct sets of users. The elements of each set, are sets of similar user

More formally the problem is defined as below Given a large dataset having N tweets from U distinct users distributed across the network, identify K groups of similar users.

Keywords: *Social Network Analytics,Machine Learning,Semantic Modelling*

Contents

List of Figures

List of Tables

1 Introduction

1.1 *Scope of the Work*

Twitter is an online social networking service which enables users to send and read short 140-character messages called tweets. Each user can follow other users on Twitter. As a user tweets, the tweet appears on the feed of his / her followers. Users also have a further option to re-tweet tweets that are present on their feed. This option endorses the tweet to his / her followers. A tweet can contain, images, links in addition to text.

While recommendation systems are commonplace in social networks, most of these recommendation systems are based on the graph structure of existing social networks. A user is categorized into various categories based on the sets of user followed. This is extensive work done on this model, with similarity of users computed using cosine similarity functions and Euclidean distance mechanisms. However identifying similar sets of users based on the content tweeted is far more challenging. There exists various sources for information in the tweet, the URL or link can be analysed to predict the nature of the article shared and thus the category to which the tweet belongs. The nature of tweets can also be studied by analysing the text present in the tweet. Sophisticated computer vision algorithms can also be used to categorize the image shared by the user.

Semantically analysing tweets, involves identifying the context of every word in the tweet. This is a far more complex method to analyse a user than based on the graph structure of the user. A semantic approach involves building a feature list for every user, building a taxonomy for every word in a users feature list, analysing the context of every word in the feature list, assign a score to k parameters, where k is the generic categories under which all words can be accommodated and finally feed this score vector to a learning algorithms to group similar users.

1.2 *Product Scenario*

Grouping similar users based on their tweets has a wide range of applications. Firstly it helps improve the accuracy of the recommendation systems. Since an in-depth analysis is performed on the content of the users tweets the nature of the user can be better analysed and has a greater accuracy than computing similarity using other methods such as cosine similarity of edges in a graph. This leads to recommending to the user, other users who

are similar to the user. Another application is that on identifying similar users, targeted advertising can be achieved. This involves generating customised advertisement for a user, based on their interest. A large portion of the revenue for major social networking sites is through advertisements and implementing this model, would considerably increase accuracy, thereby delivering relevant advertisements to a user.

Another feature of the application is its nature to adapt, i.e for example if initially the system categorizes user X based on the tweets under the broad category of sport, but if for some reason the user stops tweeting sport related topics and instead tweets topics related to politics then the system over a period of time changes the category to which user X belongs to from sports to politics. Thus the system can store the history of the interests of the user, which can be further used for analytics.

This software in particular analyses tweets to categorize users into a particular cluster, however the same principles can be applied to cluster users in any social network. This software is run constantly over the entire network and depending on the size of the network can take a few minutes to a few hours, to reorganize the clusters of users.

2 Requirement Analysis

2.1 *Functional Requirements*

1. **Gathering tweets :** To perform analytics on tweets, a set of tweets is to be collected.This is achieved using the standard Twitter API, which allows access to users tweets once the user gives the necessary permissions.

2. **Cleaning tweets :** The tweets obtained from the standard Twitter API, returns tweets in a JSON notation. The tweets are now cleansed, where-in only the text of tweet is extracted an stored onto a repository.

3. **Building a feature list :** A feature list is a bag of words associated with every user.This feature list is built by removing stop words, shortened URL, mentions and other key twitter related terms (Ex:- RT for re-tweet, MT for mention tweet) from the users collection of tweets.

4. **Word Taxonomy:**Associated with every word in the feature list a taxonomy for the word is built.

5. **Semantic Modelling :**The feature list of the tweet, is semantically modelled and is mapped to a category using the taxonomy of all the words in the users feature list. The output is a normalized score for each user in the K groups of users.

6. **Clustering :**The vector score associated with each user is fed to a learning algorithm, which categorizes similar users.

2.2 *Non-functional Requirements*

1. **Taxonomy Storage :** Permanent storage of taxonomy, of every word on a NoSQL database.

2. **Cache :** Cache frequently used words along with the respective taxonomy.

3. **Training Data :**Age collection of training data to better predict results for supervised learning.

2.3 *Use-Case Scenarios*

SimTwit has a single use case that of given a collection of tweets,being able to cluster similar users. The system accepts as input, tweets from a set of users and outputs a well defined cluster of the various users. While there are many stand-alone modules the input to each module is the output from a preceding module, automating the entire process. Hence the only use case is clustering of similar users given a set of tweets.

2.4 *Software Engineering Methodologies Used*

Incremental Model: Different similarity models were created and tested to get the progressively more accurate results, in effect using an incremental approach.

Initially we developed a TFIDF model for computing the similarity of the users, but we realized that TFIDF model didnt work well, so we went back and re-analyzed the similarity model, and the TFIDF model was replaced with a semantic model.

Also the 8000 tweet dataset that was collected were all based on the keyword 'Modi' and hence the results obtained were skewed.So tweets from varied topics like the Cricket World Cup,The political elections at Delhi,Football based tweets were retrieved to have a diverse dataset.

All these are examples of an iterative/incremental approach

3 System Design

3.1 *Design Goals*

Build a software system that can retrieve, process and semantically treat the tweets in order to satisfy the functional requirement of finding similar users on twitter. Desirable Characteristics

1. Accurate: The clusters of users created should actually be similar, i.e., false grouping should be as low as possible

2. Scalable: The system should be able to scale, i.e., the system should be able to process large number of tweets.

3.2 *System Architecture*

The software system consists of a basic front end that displays the list of tweets of a user, along with the list of users that are similar to the given user. This front end is populated by reading from plain text files generated by running the python scripts on the tweets that have been stored on a NoSQL Database.So the system architecture is basic, and can be easily used as an independent module in a more complex software application.

1. **Operating System:** Ubuntu 14.04

2. **Prototyping Tool:** R/Octave: Proto-typing machine learning algorithms, data visualization

3. **Database: MongoDB:**Avoding in-memory processing of raw, unstructured tweets, by providing a buffer space.

4. **Programming Language: Python:**Python is used for cleansing the raw tweets into a well defined structure and moving the tweets to MongoDB. During this phase,Feature Selection is also performed.

5. **HTML/CSS/PHP:**Scripting on the Web Environment is used for a demo model of the social networking site.

3.3 *Key Concepts:*

Principal Component Analysis : Principal Component Analysis (PCA) is a standard method to reduce the dimension of a matrix. To perform analysis quickly, the dimensions of the matrices to be multiplied should be less. Matrix multiplication takes $O(n^3)$ complexity and thus greater the dimension of the matrix longer the time to arrive at a relevant result. PCA reduces dimension of matrix without leading to the loss of data,or at least the loss is minimal. PCA consists of the following steps for a matrix D of dimension n * n:

1. Compute the mean vector for every row m
2. Compute the scatter matrix or covariance matrix for matrix D
3. Compute eigenvectors and corresponding eigen values let the eigenvectors matrix be E
4. Sort eigenvectors by decreasing values.
5. Choose 'k' eigen vectors corresponding to 'k' largest eigen values. Let this matrix be W.
6. W has dimensions n * k
7. Transform original matrix D onto new subspace by multiplying with W. Let this be T
8. T is the reduced dimension matrix.

K-Means Clustering : K-means clustering is an unsupervised algorithm, which successfully clusters similar entities, using the euclidean distance from 'K' distinct centroids. The K-means clustering algorithm has the following steps

1. Initialize randomly 'k' distinct centroids.
2. Compare distance of each point in the dataset, with the 'k' centroids. Associate this point with the centroid with least euclidean distance.
3. Update the centroid position as mean of all points associated with the centroid.

4. Repeat steps 2,3 till change is centroids position less than threshold t. (0¡ t ¡¡ 0.1)

K Nearest neighbours : K Nearest neighbours is a supervised learning algorithm which categorizes a point, based on minimum euclidean distance of 'K' neighbours. The mode of the labels of the 'K' neighbours is the neighbour of the point under consideration.

Random forest algorithm : Random forests are 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 class that is the mode of the classes (classification) or mean prediction (regression) of the individual tree. The steps involved are :

1. Bootstrap a sample of size N where N is the size of our training set. Use this bootstrapped sample as the training set for this tree.

2. At each node of the tree randomly select m of our M features. Select the best of these m features to split on. (where m is a parameter of our Random Forest)

3. Grow each tree to largest extent possible - i.e. no pruning.

Term Frequency Inverse Document Freequency (TF-IDF) : TFIDF, short for term frequencyinverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. In the case of the term frequency tf(t,d), the simplest choice is to use the raw frequency of a term in a document, i.e. the number of times that term t occurs in document d. If we denote the raw frequency of t by f(t,d), then the simple tf scheme is tf(t,d) = f(t,d).

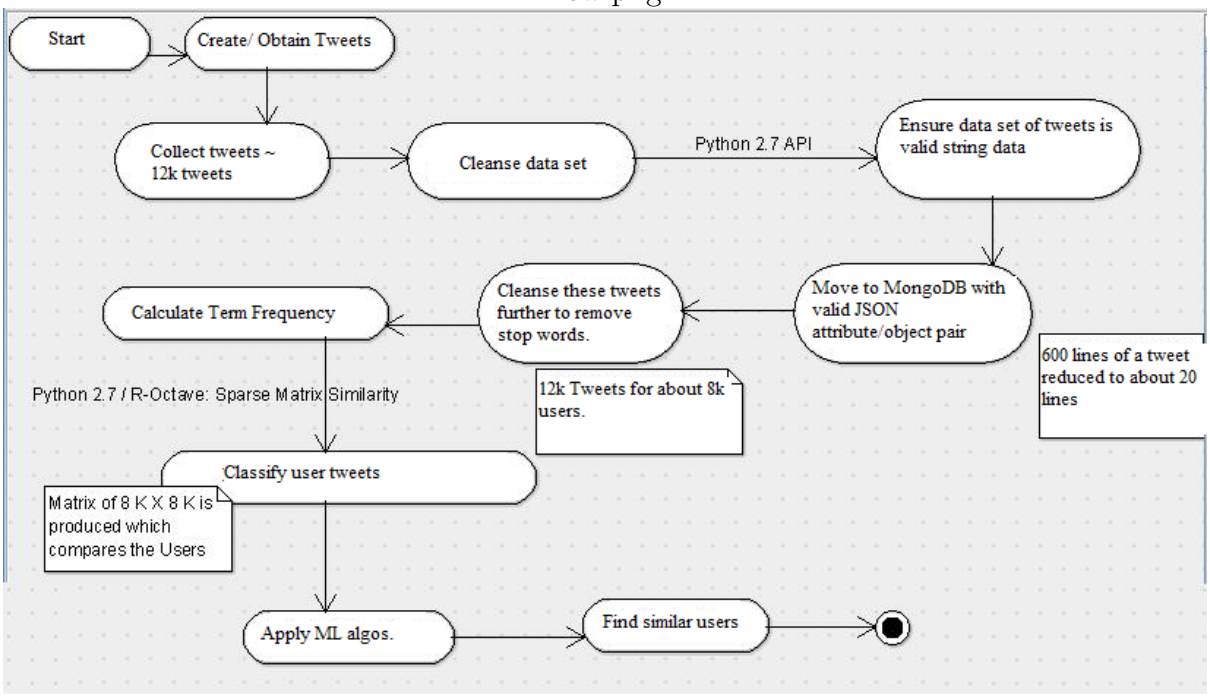
The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. It is the logarithmically scaled fraction of the documents that contain the word, obtained by dividing the total number of documents by the number of documents containing the term, and then taking the logarithm of that quotient.

3.4 *Detailed Design Methodologies*

1. Collecting and cleaning tweets.

Figure 3.1: Work Flow

flow.png



(a) Collecting tweets : Tweets are initially collected using the standard twitter API to fetch a live stream of tweets. The tweets can be collected based on location, topic and various other parameters. The set of tweets collected is stored on a file.

(b) Extracting text from tweets : The tweets obtained from the standard API are cleansed, extracting from each JSON object the relevant information, i.e the attribute text of the tweet is extracted, from the JSON file which contains other information about the tweet. The extracted text of the tweet is stored on MongoDB, a NoSQL database.

(c) Aggregating tweets based on users : This step involves aggregating all tweets which have the same user. This simplifies the task of building a feature list for a particular user.

(d) Further cleansing of individual tweets: The text associated with each tweet, is cleansed by removing stop words. Next mentions in the tweets are removed, finally the URL is also removed. The remaining words form the feature list associated with the user

2. Model 1 : Similarity using TFIDF model:

   (a) Comparison of users based of TFIDF: feature list of each user is compared with the feature list of all the other users based on the TFIDF model. Thus an similarity vector is associated with a vector. Thus if there are N users the after comparison of users, a symmetric matrix of dimension N * N is obtained with diagonal elements equal to unity

   (b) Dimension Reduction : The similarity matrix of dimension N * N matrix is large to perform any analytics. The dimension of the matrix is reduced using principal component analysis PCA. The matrix is reduced to dimensions N *K, where K is the parameter supplied to PCA. Ideally K lies in the region of 10 -15

   (c) Learning Algorithms: The reduced dimension matrix is fed to learning algorithms both supervised and unsupervised. Supervised learning algorithms used are K Nearest Neighbours (KNN) and random forest algorithm. Unsupervised learning algorithm used is K-Means clustering.

   (d) Grouping individual users : The clusters of similar users are grouped, and are pushed on the list of similar users associated with every user.

3. Model 2 : Computing similarity using Semantic modelling

Figure 3.2: Flow Diagram depicting the building of DMOZ

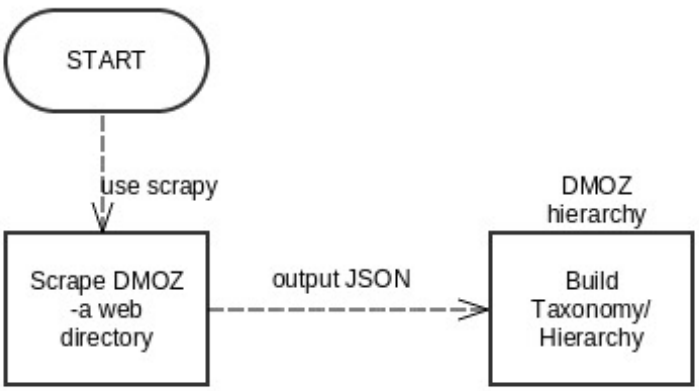
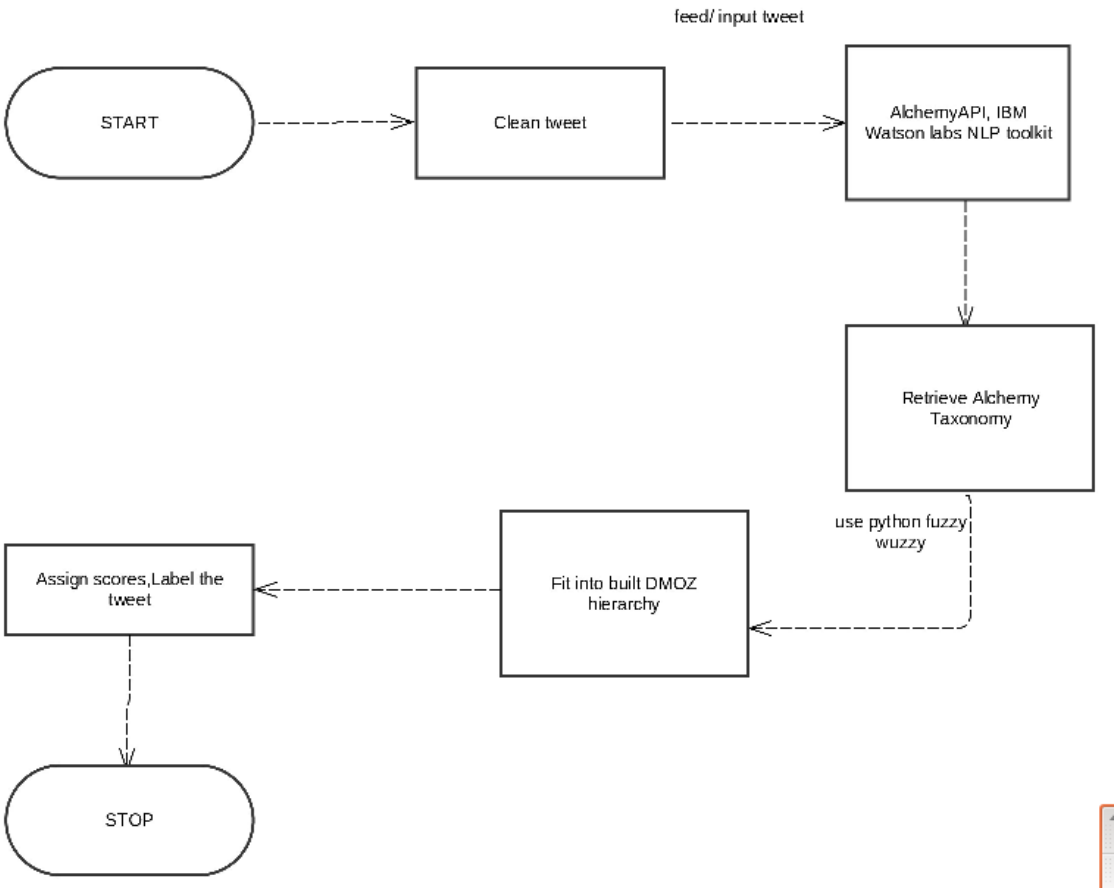


Figure 3.3: Mapping Tweet to Feature List



(a) Build Scrappy : Scrappy is a tool used to extract taxonomy associated with every word. DMOZ is a online web directory that organizes words into distinct categories. The scrapy tool built extracts words associated with every category upto two levels of hierarchy. Thus a taxonomy is established for a wide variety of topics.

(b) Finding context : Using the Alchemy API, the context of the feature list is determined. The feature list is fed to the alchemy API, which provides a context (category) to which the feature list belongs to.

(c) Mapping context to taxonomy : The result from the Alchemy API is mapped onto the existing taxonomy extracted using DMOZ. The match condition is performed using a standard Python API, fuzzy wuzzy.

(d) Assigning score to category : Once the feature list is mapped onto a category,a

score is assigned using the inverse of the distance of the node from the root. Thus each of the predetermined categories are assigned a score for each user. At the end of each step each user has a score assigned for each of the predetermined K attributes. Hence a N * K matrix is obtained at the end of this step.

(e) Learning Algorithms: The score matrix is fed to learning algorithms both supervised and unsupervised. Supervised learning algorithms used are K Nearest Neighbours (KNN) and random forest algorithm. Unsupervised learning algorithm used is K-Means clustering.

(f) Grouping individual users : The clusters of similar users are grouped, and are pushed on the list of similar users associated with every user.

4 Work Done :

4.1 *Development Environment*

1. Learning Algorithm Prototyping : Rstudio
2. Storage of tweets : MongoDB
3. MongoDB viewer : Robomongo
4. Dimensionality reduction and PCA : Octave and Rstudio
5. Scrapy DMOZ taxonomy : Python, Spyder
6. TFIDF modelling Gensim python module
7. Semantic Modelling Alchemy python API
8. Version Control System - Git Repository on bitbucket

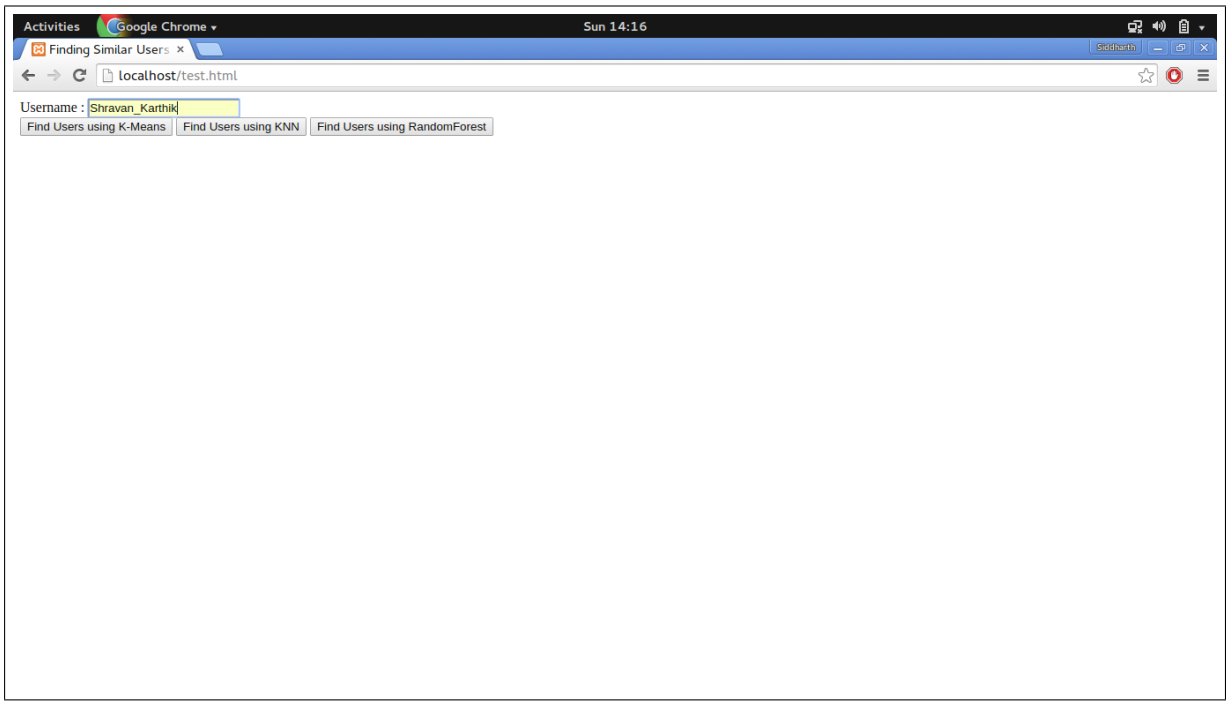
Table 1: Sample with 5 features

Username	Arts	Business	computers	games	health
spazkapur	0.0843574208	0.048545283	0.0131897441	0.0234265616	0.0516331394
NitinKapoor2020	0	0	0	0	0
suitsandcloves	0.1972437554	0	0.0422049957	0.0378983635	0
deepakshenoy	0.0756097939	0.1034609977	0.0591475812	0.0205559791	0.0504716723
DrGarekar	0.0888363716	0.0608878655	0.0135373943	0.0292666342	0.0623864241

4.2 *Results and Discussion*

SimTwit successfully identifies similar users based on their tweets. The TFIDF model is rather unsuccessful in clustering similar users. An inherent flaw in the TFIDF model is that for two users to be similar there has to be an exact match in the text prevalent in the feature list of both documents. Thus using this model doesnt take into account synonyms and general context to which the users feature list belongs to. For Ex: 44th President of the United States and Barrack Obama essentially refer to the same thing,however

Figure 4.1: SimTweet Home Screen



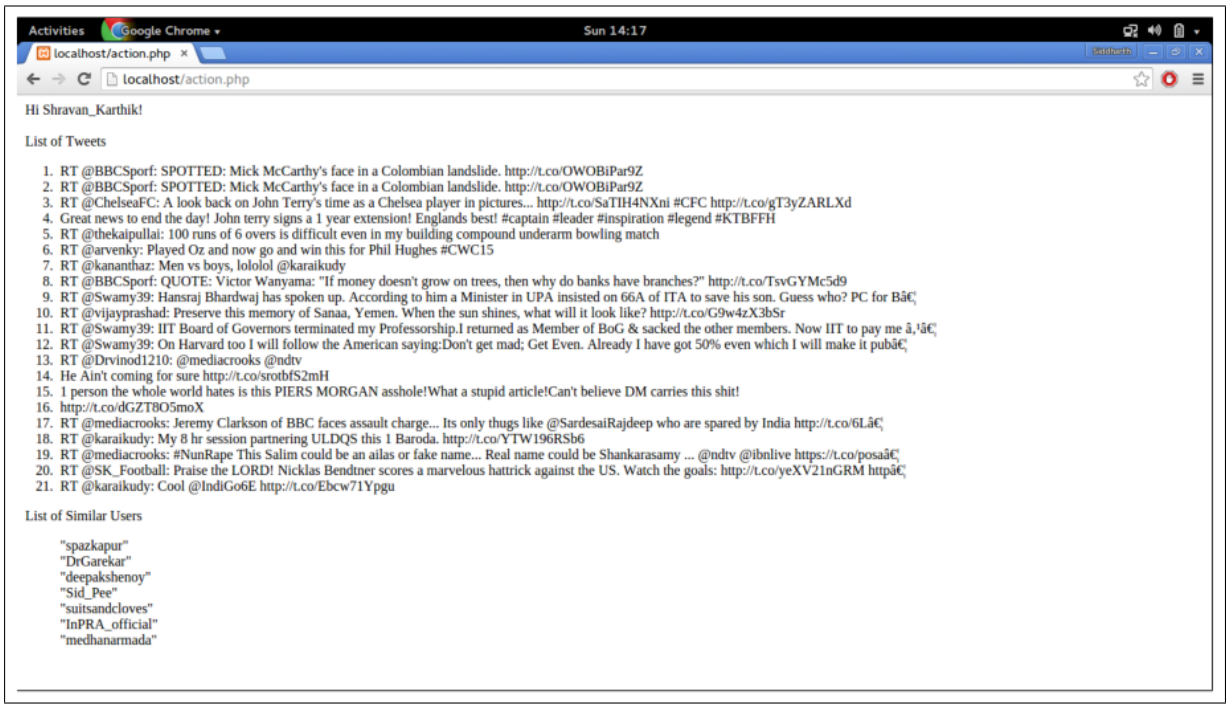
since the TFIDF model is based entirely only on matching strings, it fails to identify this similarity. Hence two feature lists that contain these above are not mapped similarly, infarct they are considered to be disparate with a significantly low similarity score.

The semantic model however overcomes the obvious error that is present in the TFIDF model. By building a taxonomy, and identifying a list of words that are associated with a particular topic, we can identify synonyms and can better understand the broad context of the tweet. Thus 44th POTUS and Barrack Obama are mapped to the same parent root, and share a common hierarchy uptil a particular level. Thus these two are classified broadly under the same category. Also using the Alchemy API provides an accurate description based not only on the taxonomy but takes the sense of the word into account.

For Ex: White House when scrutinised individually, white taxonomy is associated with that of a colour and house taxonomy is associated with that of the building, however the White House is actually the residence of the POTUS and thus Alchemy senses this and provides an output of the taxonomy of the sentence to be politics and building.

Another reason why the semantic model works well is that a word can belong tomany different categories. For Ex: the word Goal can fall under the category of the sport football. A goal is scored in football when the ball ends up in the back of the net. However

Figure 4.2: User Home Screen



goal can also refer to the aims and ambitions of an individual. Thus while assigning a score for the term goal, the inverse of the distance of the node from the root is used, i.e Goal with respect to the context of football has the taxonomy /Sport/Football/Goal thus the score assigned is $1/3 = 0.33$. However Goal in the context of setting targets is /Society/Individual/Planning/goal is $1/4 = 0.25$. Intuitively this makes sense as greater the distance from the root, there is a far less probability that it is this being referred to.

Various learning algorithms have also been used. We note that the supervised learning algorithms have greater accuracy while predicting the group to which a user belongs to. The supervised learning algorithm used were Random Forest Algorithm and the K nearest neighbours algorithm. The random forest algorithm is an ensemble method, i.e it computes various permutations of attributes before picking the set of attributes with the least entropy. Random forest algorithm also serves as a standard classification algorithm,thus similarity in this algorithm is list of all users that fall under the same category. The K nearest neighbours algorithm also produces significant results, it is a supervised clustering algorithm. The nature of similar users is not clearly defined, it rather forms boundaries for the datasets, to represent varied clusters. Thus in clustering algorithm, even though a user may belong multiple categories, he may be belong to the

same cluster. This ideally replicates the idea of a similar user. For example consider user X is interested in Politics and Sports and user Y is interested in Sports and Science, user X and user Y can still be considered to be similar.

Unsupervised learning, such as k-means clustering also provides substantially relevant results. It successfully clusters users, based on the matrix provided. The matrix however should be normalized before feeding to the learning algorithm. As with any unsupervised learning algorithm the clusters are not labelled. This serves a great advantage, as the no of clusters into which users have to be grouped into are not fixed, and the definition of each cluster is varied. Similarity can thus formulated easily using unsupervised learning algorithms.

4.3 *Individual Contribution*

1. Rishab Ketan Doshi 12IT59 - Scrapy, Building DMOZ taxonomy, Semantic similarity model

2. Rohit John Joseph 12IT61 - Cleansing of tweets, building feature list, storage of tweets on MongoDB, interactive demo

3. Shravan Karthik 12IT77 PCA, Supervised learning - K-nearest neighbours, Random Forest Algorithm

4. Siddarth P Ramakrishnan 12IT79 TFIDF similarity modelling, Unsupervised learning algorithm K-means clustering

5 Conclusion and Future Work

SimTwit successfully identifies similar users based on their tweets.As expected the semantic model of similarity works much better than the TFID model.

As of now, users are broadly classified into 16 main categories, as the DMOZ root has in total 16 children. Since SimTwit only scrapes the directories upto two levels of hierarchy, the categories to which users can belong to are limited. However if the entire directory is scraped recursively, the category to which the users belong to is far more, thereby providing a greater accuracy at clustering groups of similar users.

Another step to increase accuracy would be parse the URL present in the tweets and identify what the URL refers to. Since the URL in a tweet can aptly summarize the tweets content this would also help enhance the accuracy of clustering of users. This would involve parsing through the URL, extracting meta tags and identifying the taxonomy of these tags.

Table 2: Comparison of learning algorithms

Algo	Type	Complexity	Info
K-Means Clustering	Unsupervised	O(n^2)	Classic method for clustering sets which cant be labeled. Minimization of Eucledian distance key to determining Centroid position. K indicates number of centroids
K-Nearest Neighbors	Supervised	O(n^2)	Another clustering model, but for labeled sets. Mode of neighbors is the label of the dataset. K indicates mode among K neighbors
Random Forest Algorithm	Supervised	O(nLg(n))	Ensemble method, provides great accuracy for categorization problems. Minimization of entropy is key to determing category of entity

5.1 *Performance Metric Of Learning Algorithms*

Goodness Of Clusters: To measure the goodness of clusters we compute two matrices:

1. Proximity Matrix
2. Incidence Matrix

One row and one column is associated for each data point.

1. An entry is 1 if the associated pair of points belong to the same cluster
2. An entry is 0 if the associated pair of points belongs to different clusters

Since the matrices are symmetric, only the correlation between n(n-1) / 2 matrix entries needs to be calculated. High correlation indicates that points that belong to the same cluster are close to each other.

Confusion Matrix: Used to measure accuracy of prediction of categorization of the user.Consists of a 2*2 matrix as indicated below

Table 3: Confusion Matrix

-	True	False
True	A	B
False	C	D

1. Precision = A/(A+B)
2. Recall = A/(A+C)

Higher precision indicates greater accuracy of the categorization algorithm.

Based on the above metrics we conclude that for a labelling/categorizing the dataset,the random forest algorithm predicts with the greatest precision.However in the case of identification of similar users clustering algorithms due to the ambiguous nature of similarity produce relevant clusters of users sharing commonalities.

References

1. Discovering Similar Users on Twitter: Ashish Goel, Aneesh Sharma, Dong Wang, Zhijun Yin
2. Content-Based Recommendation Systems : Michael J. Pazzani and Daniel Billsus
3. Probabilistic Matrix Factorization : Ruslan Salakhutdinov and Andriy Mnih
4. Classification and Regression by randomForest Andy Liaw and Matthew Wiener
5. An Efficient k-Means Clustering Algorithm: Analysis and Implementation : TKanungo, D.M.Mount, N.S.Netyanahu, C.Piatko, R.Silverman and A.Y.Whu
6. `.http://en.wikipedia.org/wiki/Tf-idf` - Term Frequency - Inverse Document Frequency
7. `.http://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm` - KNN Algorithm
8. `http://www.dmoz.org/` - DMOZ The Open Directory
9. `.http://www.alchemyapi.com/products/demo/alchemylanguage` - AlchemyLanguage Web Client
10. `.http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/` - For string matching
11. `http://en.wikipedia.org/wiki/Multinomial_logistic_regression` - Maximum Entropy Classifier
12. `http://radimrehurek.com/gensim/` - For TFIDF Implementation