Natarajan Chakrapani
Karthik Reddy Vadde
Kuldeep Kapade

Project Report

Tweetstrap: Apriori prediction of retweet counts

Objective

Given a test tweet, predict retweet count (apriori) based on the source tweeter tweet topic model and his effective local_retweet count (from his immediate followers). Bootstrap or "Tweetstrap" a marketer to assess which of a given set of celebrities will spread out his test tweet the most on the twitter network.

Approach

We initially approached building a model of predicting retweet counts from a topic model of tweet corpus from various celebrities (over a recent time period) along with the user attributes (like friends/follower ratios etc) . Approach 1 entailed trying to build a multi linear regression model, over the entire set of tweets from all users. This did not give satisfactory results (some predictive value in terms of relative order of users, but no statistical significance). Then we created a support vector regression model for each user to get better prediction results (relative ranking of users improved with a reduction in error margins)

Code and data attached
Instructions for getting Sorted Prediction results: python predict_cli.py "test tweet string"

Team Task Breakdown

Task	Natarajan	Karthik	Kuldeep
Research	**35%**	**35%**	**30%**
Data Collection	**40%**	**30%**	**30%**
Algorithm design	**33%**	**33%**	**33%**
Application interface design	**25%**	**25%**	**50%**
Documentation	**40%**	**40%**	**20%**

Natarajan Chakrapani
Karthik Reddy Vadde
Kuldeep Kapade

Approach 1:

Data collection:

Since we tried to build a topic model of tweets, our initial intuition revolved around gathering tweets using streaming API over topical hashtags like (#music, #art etc)

The data gathering was done over a 3 day period (from 10/22/2012 to 10/24/2012. The subsequent tweets were converted to lower case, and removal of various stop words. Spam was removed by deleting all tweets comprising only of hashtags or tweets having multiple occurrences of words with 3 or more consecutive characters. (this was using the observation that the English language has no word with 3 consecutive same character sequence)

Building the Topic model

After cleaning the data, we were left with 24902 tweets across topical hashtags via the streaming API, whose word cloud is seen above. This was to be fed as an input to create a LDA model for tweets topic distributions. Using the python package *gensim* for LDA, we arrived at top 10 topic distributions.

Sample result snippet:

2012-11-25 11:32:24,113 : INFO : topic #2: 0.016*out + 0.012*check + 0.008*help + 0.006*@ + 0.005*some + 0.005*celebrity + 0.005*one + 0.004*need + 0.004*me + 0.004*can + 0.004*love + 0.004*style + 0.004*$ + 0.004*news + 0.004*womens + 0.003*years + 0.003*th + 0.003*these + 0.003*do + 0.003*our

Each of the ten topics were formed from a cluster of words which occurred frequently in context, with the word distribution as seen above (for topic #2). However, due to the limited content of tweets, topics comprised words that were not semantically related and it was not intuitive to figure out what the topic was about by looking at just the word distributions.

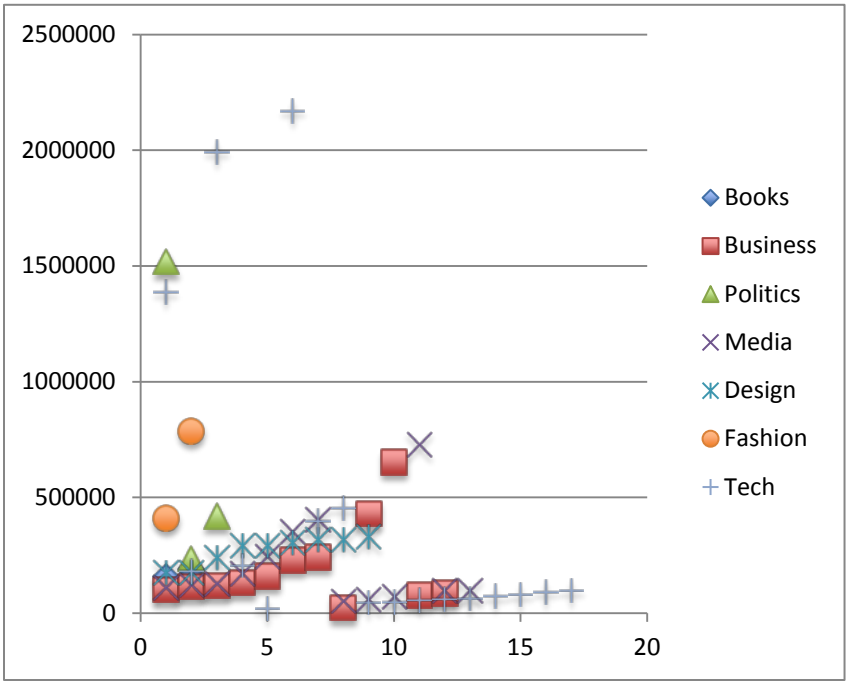
The idea was to use this topic model, to assign topic scores to celebrity tweets in order to co-relate it with the retweet count prediction. However, we discontinued with this approach as the topic model was not deemed satisfactory based on LDA.

Approach 2:

Data collection and topic model creation

To improve classification of tweet text we wanted it to be compared with larger corpus of text. To this end we used a Bayesian multi-class classifier service called **uclassify** which exposes a web service API which has been trained over 10 million documents. It classifies and identifies topic scores over 10 topics for any given text. The 10 topics are Arts,Business,Computers,Games,Health,Home,Recreation,Science,Society,Sports.

Using the twitter user recommendations categorized by topic (http://twitter.com/invitations/suggestions), we identified a set of users from various domains (a sample is shown below of follower count(y) vs. user ids from a domain(x)).

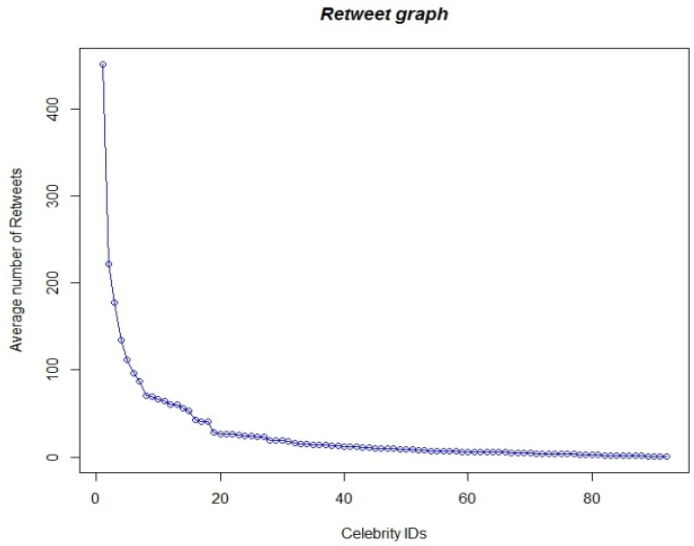


Total : 84 users
2 Books
12 Business
3 Comedy
4 Cook
9 Design
2 Fashion
13 Media
1 Movies
4 Music
3 Politics
7 Science
5 Sports
17 Tech
2 Travel

Natarajan Chakrapani
Karthik Reddy Vadde
Kuldeep Kapade

We intentionally targeted mid-level celebrities, as the major twitter celebrities like @barackobama or @justinebieber are retweeted irrespective of the content of their tweets, thereby adding a lot of noise (potentially) to our model.

Using the twitter search API, we scraped all tweets and retweets done by these users extending back over the 9-day period (search api limitation) ending on 10/23/2012.



Following is the power law graph indicating the average retweet counts by celebrity, confirming the intuition that some twitter celebrities are retweeted way more than the others.

The data collection from the search API was massaged into a JSON object , comprising raw fields from the api results and some computed fields

- Tweet text – Raw field from API results
- Local_count - Computed field indicating count of immediate followers who have retweeted.
- Retweet : Computed Boolean field indicating whether this was a RT(“true”) or created by the user(“false”) .
- Cascade_depth : Computed field to find hop length of tweet from given tweeter to the last person who has retweeted this message – to estimate spread of the message.
- Hashtags - Raw field from API result.
- Screenname – Raw field indicating source tweeter handle.
- Cascade_size – Computed field indicating count of all nodes in the source tweeter’s network who retweeted a given tweet.

- created_at – Raw field indicating tweet creation timestamp.

Sample object
{"text":"Maine co-anchors resign on air, say Fox management denies Climate Change. http://t.co/df7fae8O","local_count":271,"retweet":false,"cascade_depth":0,"created_at":"Sat Nov 24 18:01:07 PST 2012", "retweet_count":377, "hashtags":[], "screen_name":"ebertchicago", "cascade_size":0}

Note that, the default value of 0 for both fields as we could not compute the cascade depth and cascade size owing to the API quota limit in fetching the list of follower list of followers as we proceed through the tweet chain.

Limitation: The retweet_count returned by the API is a final aggregated count and is not a point-in-time retweet count. Hence, the impact of the user who retweets an already popular tweet cannot be ascertained easily. In other words, one cannot compute the difference in the retweet count of a tweet between two timestamps (say before a given celebrity user retweets to after his retweet). So, by focusing only on the Tweets authored by the user, the retweet_count is a measure of the influence of his own work across his follower networks. So all tweet objects with retweet='false' were chosen for building our model.

We also used the Twitter api to get user attributes for the source tweeters (84 users). Fetched fields included:

- Friend_count – number of people followed by this user
- Follower_count - number of people following this user
- Screen_name – user twitter handle
- Listed_count – number of twitter lists the user is a part of.

For example, sample user record:
{friend_count":175,"follower_count":21180,"screen_name":"btaylor","listed_count":904}

Initial hypothesis:
We tried to use a multiple linear regression model with dependent variable as retweet_count and then use 13 features of a given tweet as our independent variables.

The 13 features comprised:

[x1..x10] - 10 features corresponding the topic scores over the 10 topics
[x11] friend/follower ratio – base 10 log value

[x12] listed_count/follower_ratio – base 10 log value
[x13] percentage value of local_retweet_count/follower_count

Transformations in x1[1-3] were needed to normalize the distributions over all tweets across users. In order to see the significance of these variables, we used the OLS regression component in *numpy* module in python.

OLS regression output

Dependent Variable: y
Method: Least Squares
Date: Sun, 02 Dec 2012
Time: 12:33:14
obs: 17709
variables: 14

variable	coefficient	std. Error	t-statistic	prob.
const.	-94.865389	7.073409	-13.411551	0.000000
x1	4.146027	1.635802	2.534552	0.011268
x2	-2.448941	2.300227	-1.064652	0.287048
x3	4.624666	3.341381	1.384058	0.166358
x4	-3.006861	2.025955	-1.484170	0.137782
x5	-0.361226	2.147220	-0.168230	0.866404
x6	4.213738	2.499364	1.685924	0.091828
x7	5.952432	2.195103	2.711687	0.006701
x8	-0.423856	2.032856	-0.208503	0.834839
x9	-1.953506	1.775609	-1.100189	0.271265
x10	2.658242	1.836085	1.447777	0.147697
x11	-13.863916	1.218058	-11.381983	0.000000
x12	-38.302003	3.416484	-11.210941	0.000000
x13	2630.752394	57.954870	45.393120	0.000000

Models stats		Residual stats	
R-squared	0.124282	Durbin-Watson stat	1.674476
Adjusted R-squared	0.123639	Omnibus stat	49139.665925
F-statistic	193.175372	Prob(Omnibus stat)	0.000000
Prob (F-statistic)	0.000000	JB stat	2763530381.142520
Log likelihood	-113966.527747	Prob(JB)	0.000000
AIC criterion	12.872610	Skew	35.447064
BIC criterion	12.878762	Kurtosis	1936.967294

Interpretation: The R-square value suggests there is just a 12% explanation of the variance of the retweet_count by this model. Also, none of the topic score features are statistically significant, suggesting 17k tweets across the users are possibly less for achieving any statistical significance. To experiment in order to achieve higher significance, we removed feature listed_count/followers [x12] from the previous model, intending it would help the retweet_count get impacted more by the topic scores of tweets.

So the new features were:
[x1..x10] - 10 features corresponding the topic scores over the 10 topics
[x11] friend/follower ratio – base 10 log value
[x12] percentage value of local_retweet_count/follower_count

Also, we tried to add additional user features, relating to his overall topic distribution among his tweets. For instance, if a user had 100 tweets, we summed up the log transformed topic scores over each topic to get a aggregated/averaged topic score . This would give us the perspective of the tweet topic scores in relation with the overall topic scores for the user. Additional features employed were [x13…x22] – 10 features for each tweet incorporating aggregated user topic scores over his tweets.

OLS regression output
Dependent Variable: y
Method: Least Squares
Date: Sun, 02 Dec 2012 Time: 20:21:46
obs: 13229 # variables: 23
==========variable
coefficient std. Error t-statistic prob.
==========

const	-77.051543	8.535870	-9.026794	0.000000
x1	1.675754	2.078910	0.806073	0.420215
x2	-1.981195	3.013254	-0.657494	0.510875
x3	2.269374	4.277470	0.530541	0.595746
x4	0.805491	2.580542	0.312140	0.754939
x5	-0.678587	2.720730	-0.249414	0.803045
x6	5.297815	3.206816	1.652048	0.098548
x7	11.473859	2.835027	4.047178	0.000052
x8	-0.039205	2.570366	-0.015253	0.987831
x9	-1.738659	2.267910	-0.766635	0.443312
x10	2.384252	2.311474	1.031486	0.302332
x11	-19.399368	1.565339	-12.393078	0.000000

x12	2963.346052	81.330161	36.436004	0.000000
x13	0.328308	0.037087	8.852438	0.000000
x14	0.211944	0.069199	3.062808	0.002197
x15	0.458551	0.110740	4.140782	0.000035
x16	-0.328082	0.045902	-7.147501	0.000000
x17	-0.328024	0.086538	-3.790512	0.000151
x18	-0.043634	0.058132	-0.750600	0.452907
x19	-0.070149	0.072266	-0.970710	0.331711
x20	-0.498613	0.103230	-4.830100	0.000001
x21	0.178580	0.040539	4.405157	0.000011
x22	0.385545	0.087431	4.409682	0.000010

Models stats		Residual stats	
R-squared	0.124984	Durbin-Watson stat	1.852332
Adjusted R-squared	0.123526	Omnibus stat	36634.888704
F-statistic	85.740613	Prob(Omnibus stat)	0.000000
Prob (F-statistic)	0.000000	JB stat	1873952825.716981
Log likelihood	-86113.755072	Prob(JB)	0.000000
AIC criterion	13.022414	Skew	35.233211
BIC criterion	13.035436	Kurtosis	1845.485376

Interpretation: the R-squared value did not exhibit any improvement over the previous model, indicating the variance is not better explained by the addition of these new features. However, the addition of the aggregated user topic feature for each tweet added a statistically significant effect to the retweet_count prediction.

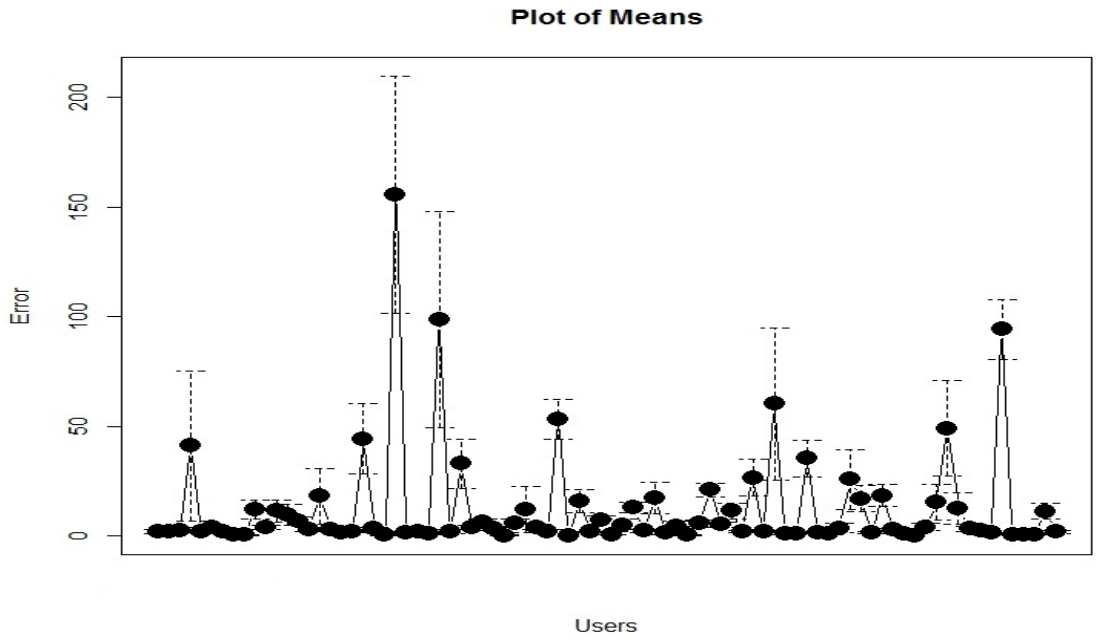
Owing to such insignificant effects of the topic scores on the retweet_count predictions, we tried to theorize if there was

- a non-linear relationship among our dependent and independent variables.
- Creating a model for each user to predict retweet count for a test tweet, rather than having one giant model incorporating tweets from all users.

Guided by that vision, we used the scikit-learn package in python to *apply support vector regression* using a Gaussian kernel (rbf) with parameters (gamma = 0.2 and C (penalty parameter) = 1e3). Since we were now using the per user model, we used the following features

- [x1...x10] – topic scores for each tweet over the 10 topics . all topic scores < 0.01 were set as zero, to reduce noise in the model. (and compensating for classifier inaccuracy,since really low topic scores possibly indicated false positives for the presence of that topic in the tweet)
- [x11] – local_retweet count (i.e count of immediate followers who retweeted)

Our intuition seemed right, when we looked to evaluate the accuracy of our per user model. See the below box plots for mean errors (predicted vs. actual retweet counts) over a training set for each user. The dashed line indicates the standard error margin



Once a model was built for each user, we took a test tweet, got its topic scores across the 10 topics and also estimated its local retweet count.

<u>Estimation of local retweet count for a test tweet.</u>

Approach 1:

- Use median of local_retweet counts for each user to apply to a given test tweet.
- However, the predictions were highly inaccurate.

Approach 2

- For each user, gather the tweets *most similar* to this test tweet . By similarity, I mean a subset of the tweets that have non-zero topic scores for any non-zero topic score for the test tweet.

- For example: if test tweet was 20% about topic 1 and 40% about topic 2, then consider all tweets of the user which have non zero topic1 or topic2 scores..
- Generate their corresponding local_retweet counts as a weighted contribution of topic scores * local_retweet count. For the example above it will be 0.2*topic1_score + 0.4*topic2_score
- Aggregating these local_retweet counts , one can weigh it with the ratio of the (no. of such similar tweets identified/total tweets in model for that user)
- This gives a probable estimate of the local_retweet count for that test tweet, which we can use to predict the final global retweet_count

Using that estimate gave us better confidence governing our predictions for final retweet_count for a test tweet per user. Below is a sample of top 12 users who get final retweet count for the below test tweet

Test tweet: "great shoes for your marathon running health"

Result:

deadmau5,Music,249
RealMikeWilbon,Sports,182
nickkroll,Comedy,143
ErinAndrews,Sports,118
SteveLevitan,Media,102
JerrySeinfeld,Comedy,98
jack,Tech,87
cher,Music,82
tconrad,Music,77
billprady,Media,74
sparker,Tech,70

Interpretation: expecting sports personalities to have high global retweet count, we were surprised to see @deadmau5 and @nickkroll topping the charts. On digging deeper, we did notice that they benefited from a active immediate follower community, who comprised a sizeable fraction of the final retweet count. However, the presence of sports media personalities like @ErinAdrews and @RealMikeWilbon indicated our model's results were intuitive.

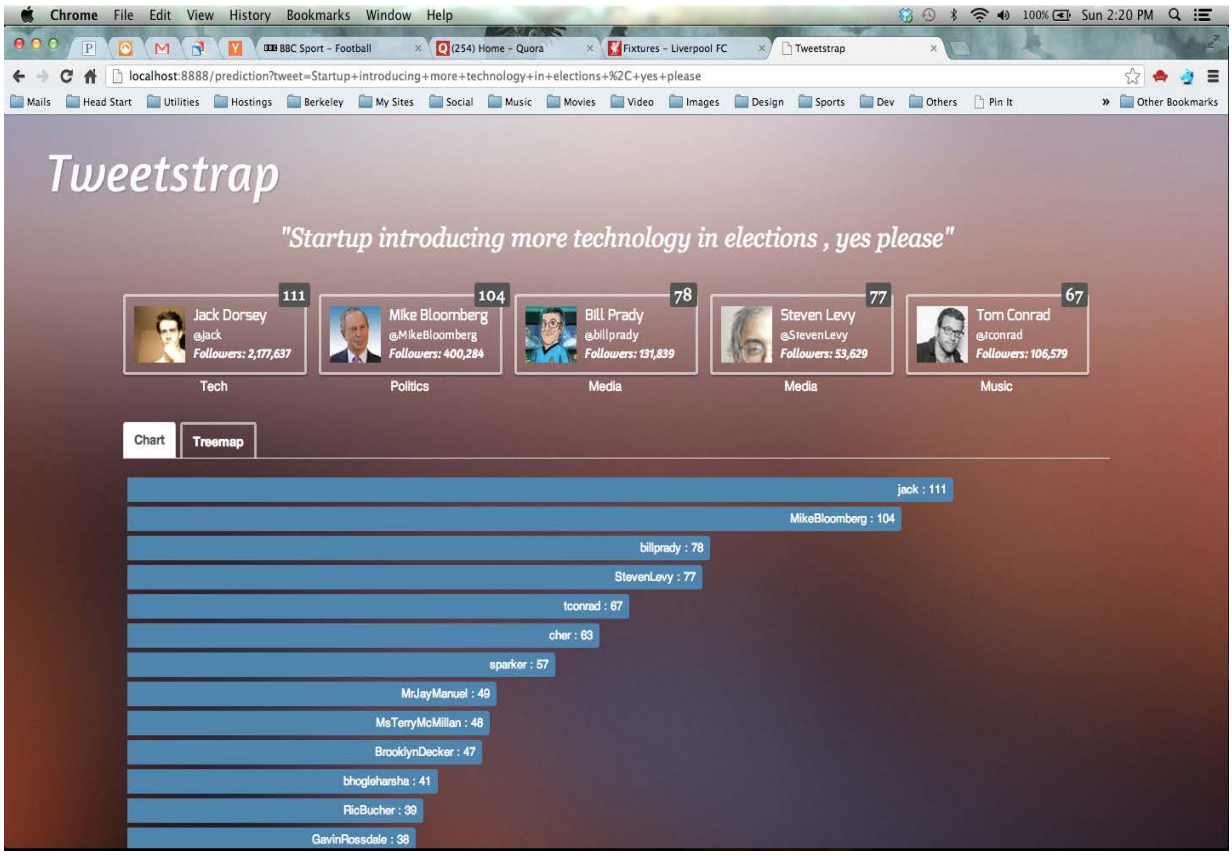
In order to interpret the output, we focused on a treemap (relative ranking) and histogram visualizations of the relative retweet counts obtained for test tweets, for our various celebrities
For instance,

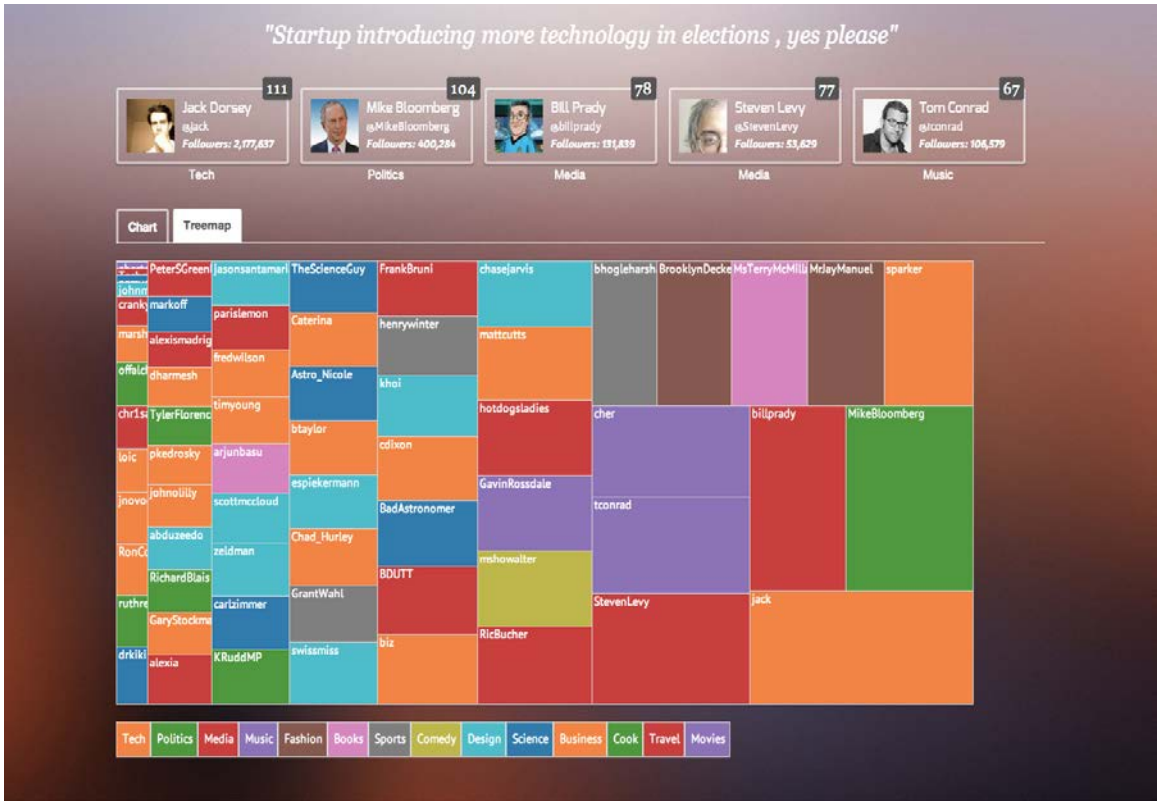
Natarajan Chakrapani
Karthik Reddy Vadde
Kuldeep Kapade

test tweet: "Startup introducing more technology in elections , yes please (top results)"

jack,Tech,111
MikeBloomberg,Politics,104
billprady,Media,78
StevenLevy,Media,77
tconrad,Music,67
cher,Music,63
sparker,Tech,57

Sample screenshots





The colors of the tiles are domains that various celebrities belong to, with the size of the tiles governed by the magnitude of the prediction count .

References

[1] *Meenakshi Nagarajan, Hemant Purohit, Amit Sheth.* ***2010.*** **A Qualitative Examination of Topical Tweet and Retweet Practices**. International Conference on Weblogs and Social Media (ICWSM).

[2] *Meeyoung Chay ,Hamed Haddadi, Fabrıcio Benevenuto, Krishna P. Gummadi*. 2010. **Measuring User Influence in Twitter: The Million Follower Fallacy.** Association for the Advancement of Artificial Intelligence.

[3] *Wojciech Galuba, Karl Aberer. 2010.* **Outtweeting the Twitterers - Predicting Information Cascades in Microblogs.** Workshop on Online Social Network.

[4] *Roja Bandari, Sitaram Asury and Bernardo Hubermanz.* **The Pulse of News in Social Media: Forecasting Popularity.**

[5] *Bakshy E, Hofman M J, Wason A. M, Watts J. D*. 2011. **Everyone's an Influencer: Quantifying Influence on Twitter.** WSDM

[6] *Jake Lussier and Jacob Bank.* 2011.**Final Report: Local Structure and Evolution for Cascade Prediction.**

[7] *Liangjie Hong Ovidiu Dan Brian D. Davison. 2011.* **Predicting Popular Messages in Twitter.** World Wide Web (WWW)

[8] *Sasa Petrovic , Miles Osborne , Victor Lavrenko*. 2011. **RT to Win! Predicting Message Propagation in Twitter.** International Conference on Weblogs and Social Media (ICWSM).

[9] *Xufei Wang, Huan Liu, Peng Zhang, Baoxin Li*. **Identifying Information Spreaders in Twitter Follower Networks**