Twist:

User Timeline Tweets Classifier

By

Sonali Sharma

Priya Iyer

Vaidyanath Venkat

Project Report

Report presented towards the completion of the class project for INFO 290 Analyzing Big Data with Twitter

Date: 12/10/2012.

Mentor:

Andy Schlaikjer

Instructor

Marti Hearst

1. Abstract

Social media plays a big role in our day-to day lives. Over the last few years, micro-blogging services like Twitter have become a great source of information from friends, celebrities, organizations and a means for building social networks. These services are being increasingly used for real-time information sharing, news and recommendations.

This report describes the creation, implementation and evaluation of a classifier, trained using supervised machine learning techniques, which takes tweets as input, and classifies them into a set of predefined categories. These categories are Sports, Finance, Technology and Entertainment. In this report, we discuss the goals of this project, the necessary data collected to train and test the classifier. We further evaluate the options used; provide details of implementation and evaluation techniques used for this classifier. The output of the classifier is shown with the help of a wen interface.

2. Project Goals

Currently twitter allows users to create custom lists based on user profiles. This is a personalized list where the users classify Twitter profiles in one or more categories. E.g. @newyorktimes, @cnn, @guardian etc. are usually classified under "News" List. Similarly, one would classify @SportsCenter, @TwitterSports, @NBCSports, @SkySportsNews under "Sports" list. Generally, these lists are based on the twitter handles and not on the text of the tweet per se. The goal of our application is to:

a) Analyze the tweets, process them to remove redundant information and then employ machine learning techniques to automatically classify the tweets under one or more predefined categories based on one or more features of the tweets.
b) Create a web interface for the users to login using their twitter account and view the tweets from their home timeline classified under on these categories of Sports, Technology, Finance, Entertainment and Others.

3. Project Strategy

We divided the project into phases. The tasks under each of the phases were divided amongst the team members based on preferences and skill sets. The next step was to do a literature review to analyze previous work and to identify the best practices. We then divided our project into logical phases.

Phase 1:

- Create an outline of the application, identify software needs, plan the application architecture

- Data Collection, data consisted of tweets from handles under Sports, Technology, Finance and Entertainment categories on the "Who to Follow" page: *https://twitter.com/i/#!/who_to_follow/interests*
- Text Mining: Creation of custom algorithms to mine the text from the tweets and remove noise
- Creation of training and test files in the classifier library specific format
- Creation of algorithms to train and test the classifier with metrics for evaluation

Phase 2:

- Measure effectiveness of the classifier using precision, recall and cross validation.
- Refine the classifier using more training set and features.
- Create the web interface running on a Flask server using Python script for users to view classified tweets
- Code documentation and review
- Preparation of final project report and presentation.

4. Implementation

4.1 Overview

To create a classifier we collected over 100,000 tweets belonging to sports, technology, finance and entertainment. The implementation steps were divided into backend (data collection, processing of tweets and classifier module) and frontend (to show classified tweets).

4.2 Technologies Used

Tweepy

Tweepy is a Python library for accessing the Twitter API. We implemented the authorization and streaming modules of the live tweets using this package.

Twitter 4j API V.1.0

The Twitter REST API methods allow developers to access core Twitter data. This includes update timelines, status data, and user information. For the purpose of data collection we used Twitter 4j API version 1.0, implemented in Java, to collect tweets from a list of users.

Python 2.7.3

Preprocessing of the tweets, which consisted of cleaning the tweets by removal of special characters, spell checks, stemming, removal of stop words, tokenization, bigrams was done in python 2.7.3. Spell checking and stop-words removal was implemented using the NLTK toolkit.

LIBLINEAR

The library used for creating different classifier models was LIBLINEAR for Python.

4.3 Data Collection

In order to train the classifier we collected over 25000 tweets from each of the four categories using Twitter4j REST API V.1.1. To ensure that relevant tweets are collected, we got a list of the top 15-20 most influential users for each category from Twitter's recommended "Who to Follow" list (https://twitter.com/i/#!/who_to_follow/interests) as shown in Figure 1.

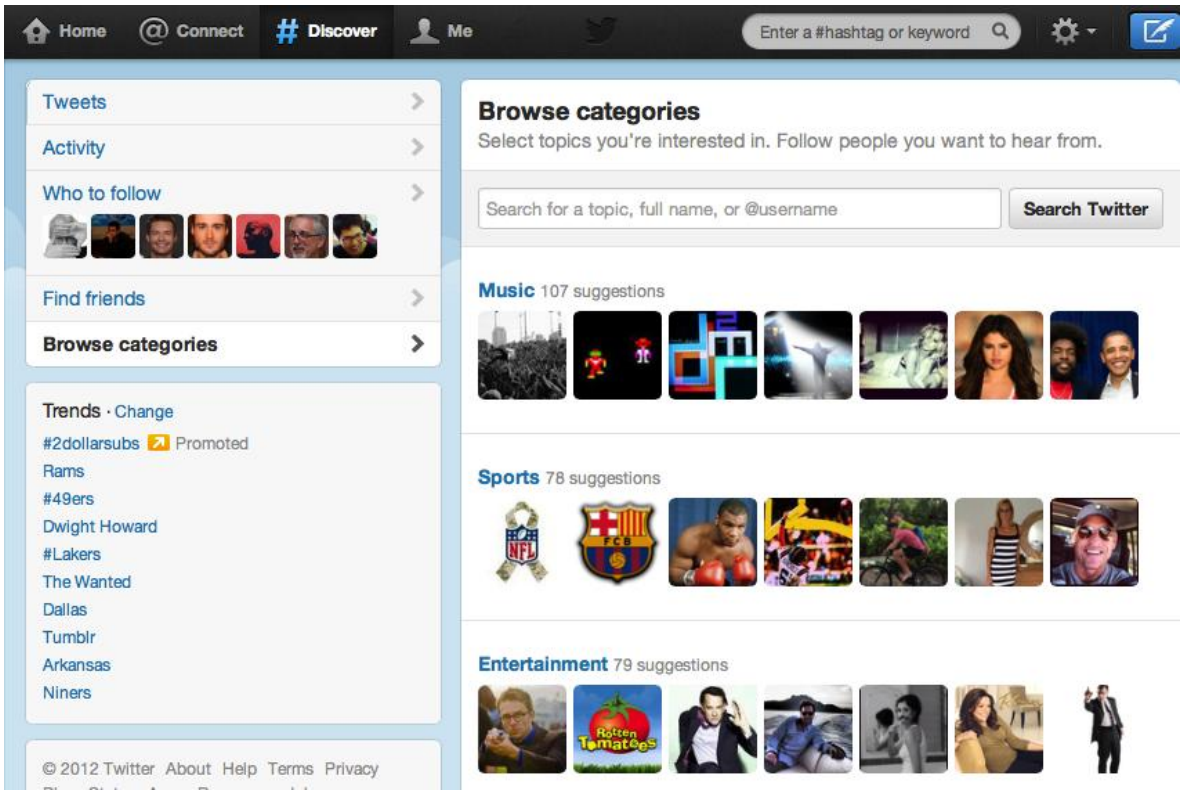

Figure 1. 'Who To Follow' suggestion lists on Twitter

We collected over 2000 tweets per user under each category into four separate text files. List of selected users for each category is described in Appendix A.

Note: In order to address the problem of rate limiting, tweets were collected in regular intervals as opposed to a one-time run. Paging was used which allows 200 tweets to be collected per page from each user.

4.4 Text Mining

Before using the Twitter data as training set, it was very important to pre-process the data to extract meaningful information. Twitter data consists of lots noise, which must be removed. Tweets are in an inconsistent format, they contain a lot of special characters, abbreviations, @ for mentions, # for tags, misspelt words, URLs and exclamatory words. We identified the following rules and built algorithms for each of these to clean and preprocess data in order to create a relevant training set.

Rule#1: Take only English language words.

Rule#2: Remove all special characters except hashtags and URLs

Rule#3: Correct words containing repeated letters e.g. "*sooooo good*", changed to "*so good*".

Rule#4: Apply spell check on words using Norwick's spell check algorithm

Rule#5: Remove stop words

Rule#6: Apply stemming to change the word to its root. Porter's stemming algorithm was use for this purpose.

Rule#7: Convert words to lower case.

Rule#8: Tokenize words using the whitespace and create bigrams.

We wrote methods in Python to execute the rules defined above and create a clean dataset for training and testing purposes.

This is illustrated in Figure 2.

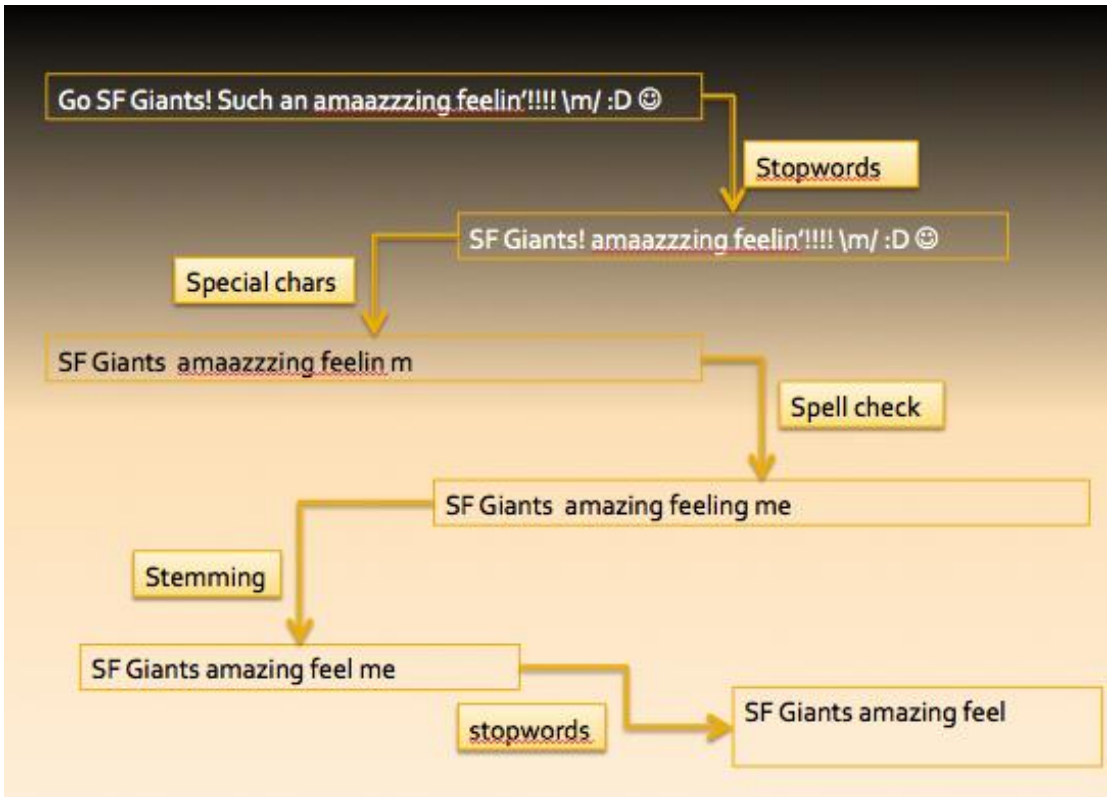

Figure 2: An illustration of the flow of Text Processing Module

4.5 Classifier

4.5.1 Rationale

For the purposes of text classification, we made use of multiclass linear classifiers. From the literature review, it was clear that in particular, the most common technique in practice has been to build one-versus-rest classifiers (commonly referred to as "one-versus-all'' or OVA classification), and to choose the class which classifies the test datum with greatest margin. Thus, our classifier had to classify into: Sports, Finance, Entertainment, Technology and Others. For this purpose, we implemented four *One vs. Rest* classifiers (commonly referred to as "one-versus-all" or OVA classification), one for each category.

Each tweet is passed through each of these classifiers. The output of each of these binary classifiers would be whether the tweet belongs to the category that classifier is trained for. Thus, after the tweet has passed through all the four classifiers, we will know the categories tweet belongs to. If it doesn't belong to any of the categories, it will be classified under 'Others'.

We used the LIBLINEAR library for python to implement linear classifiers.

LIBLINEAR is a linear classifier for data with millions of instances and features.

It supports

- L2-regularized classifier
- L2-loss linear SVM, L1-loss linear SVM, and logistic regression (LR)
- L1-regularized classifiers
- L2-loss linear SVM and logistic regression (LR)
- L2-regularized support vector regression
- L2-loss linear SVR and L1-loss linear SVR.

4.5.2 Choice of the Classifier

The choice of the classifier was crucial. From the literature review, it was clear to us that the two best machine learning techniques for text classification would be Logistic Regression and SVM (Support Vector Machines). Once we narrowed down to the use of the library, the next challenge for us was to determine the exact classifier that would be the most effective in classifying tweets. We conducted an experiment to choose the best classifier type which is explained in detail in the section 4.5.3.

4.5.3 Preparing the training set

From the text processing module as discussed in Section 4.4, we created four different training sets for each of the four classifiers. We created a custom algorithm to convert each of the tweets from text into the format required by the LIBLINEAR library. Also, the training data was taken in a ".txt" file as described in the section 4.3

After pre-processing of data, it was stored into four separate text files.

Database vs. Text Files: After obtaining the training data from Twitter and pre-processing it, there was a need to store the data in such a form that words could be assigned to a unique index. We first used a lightweight database package called 'sqlite3' but soon found that storing of words and tweets in a database was adding an extra overhead in retrieval of information and was slowing down the process of creation of feature vectors and input file. For 10,000 tweets containing on an average of 12 words it took almost 2 hours to read the data from the database, create a term-document dictionary and then convert it into the format required by the classifier.

Thus, for faster processing of tweets, we then switched to text files for storage of data and created the input to the classifier by directly reading data from the text files. This eliminated the need to maintain a separate database and reduced the processing time significantly. The following files were used to create feature vector and input to the classifier:

Word Dictionary – Containing a list of unique unigrams and bigrams along with their index. This is a global list of words for the training dataset. This is illustrated in Figure 3 below. Column 1 represents the unique index of the word and column 2 represents the token.

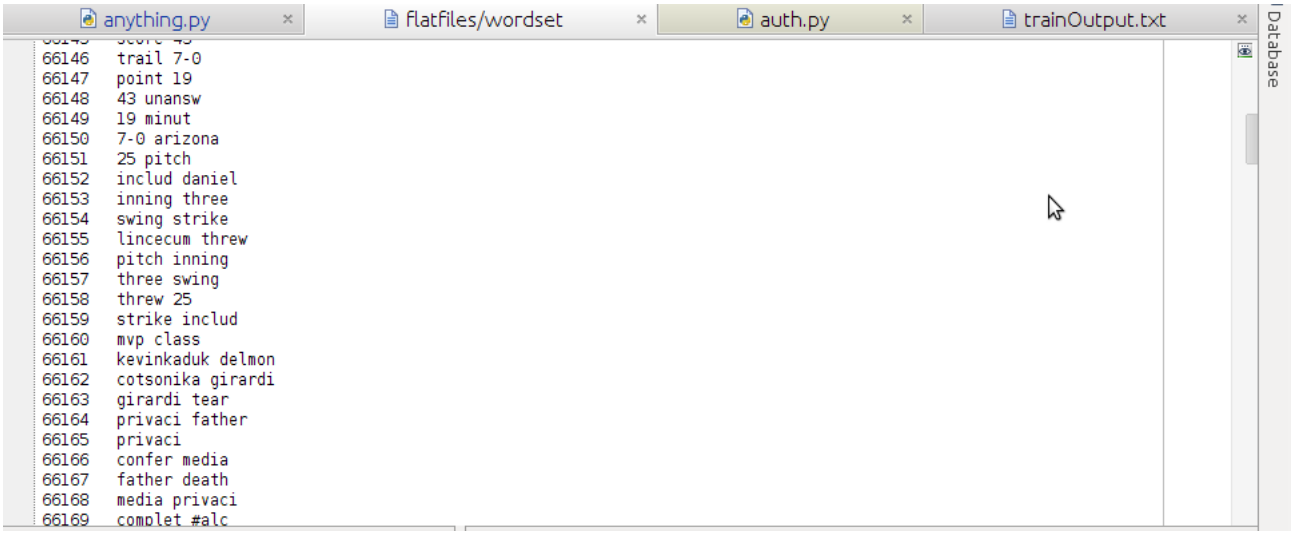

Figure 3. Word dictionary containing a unique index for each of the words occurring in the tweets present in the training data set

Tweet-Terms Dictionary – This contains category and tweet wise list of tokenized words and bigrams. The file is divided into 4 columns.

Column1 - Category type (1- Sports, 2- Finance, 3- Entertainment, 4- Technology)

Column2 - Tweet ID; follows the same order as the occurrence of the tweets in the training set

Column3 - Word ID, mapped to the Word Dictionary shown in Figure 3.

Column4 – 1 depicting presence of the word in the tweet; used later for classifier input file creation

This is illustrated in Figure 4 shown below.

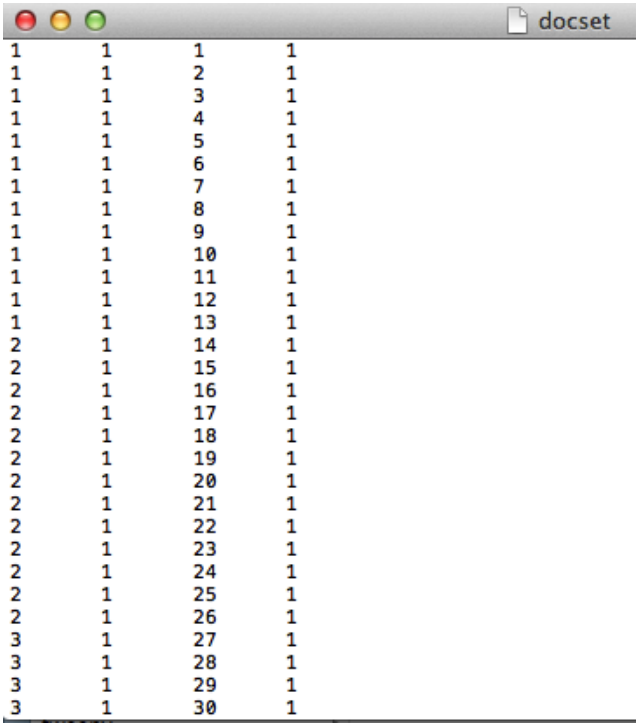

Figure 4: Tweet-Terms dictionary; maps the Tweet IDs to the index of the terms or words occurring in any given tweet.

Features – Unigrams and bigrams were used as features. TF-IDF (term frequency- inverse document frequency) was initially chosen as the weight of each of the features.

$$\mathrm{tf}(t,d) = \frac{\mathrm{f}(t,d)}{\max\{\mathrm{f}(w,d) : w \in d\}}$$

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

$|D|$: Cardinality of D, or the total number of documents in the corpus

$|\{d \in D : t \in d\}|$: Number of documents where the term t appears

Then, TF-IDF is calculated as

$$\mathrm{tfidf}(t,d,D) = \mathrm{tf}(t,d) \times \mathrm{idf}(t,D)$$

Twitter data set is not a traditional dataset as used in text classification. Since tweets are limited to 140 characters, words that were present in tweets had the same TF-IDF value. Moreover, the calculation of TF-IDF for 100,000 tweets was also a tedious process.

Hence, for faster and more efficient processing, we chose Boolean as the weight of the feature where 1 represents the presence of the word in the tweet and 0 represents absence of the word.

LIBLINEAR Input - Input to LIBLINEAR consisted of the feature representation of the selected unigrams and bigrams. The training files were created in the following format:

<label> <index1>:<value1> <index2>:<value2> ...
<label> <index1>:<value1> <index2>:<value2> ...
.
.

<label> <index1>:<value1> <index2>:<value2> ...

Where,

<label> corresponds to category. For the tweets belonging to "Sports" present in the training file for the Sports classifier, the <label> values were 1 and the rest of the tweets had a label value of '-1'. Similarly, in each of the files for "Finance", "Entertainment" and "Technology", the <label> values for tweet were 2, 3 and 4 respectively while the rest of the tweets in those files had <label> values of -1.

<index$_i$> corresponds to the index of the i^{th} word in the tweet. This index is retrieved from the Tweet-Terms dictionary described in Figure 4. <value> is the Boolean value which corresponds to 1 signifying that the word exists in the tweet.

Below is the illustration of the training file for the category, "Technology".

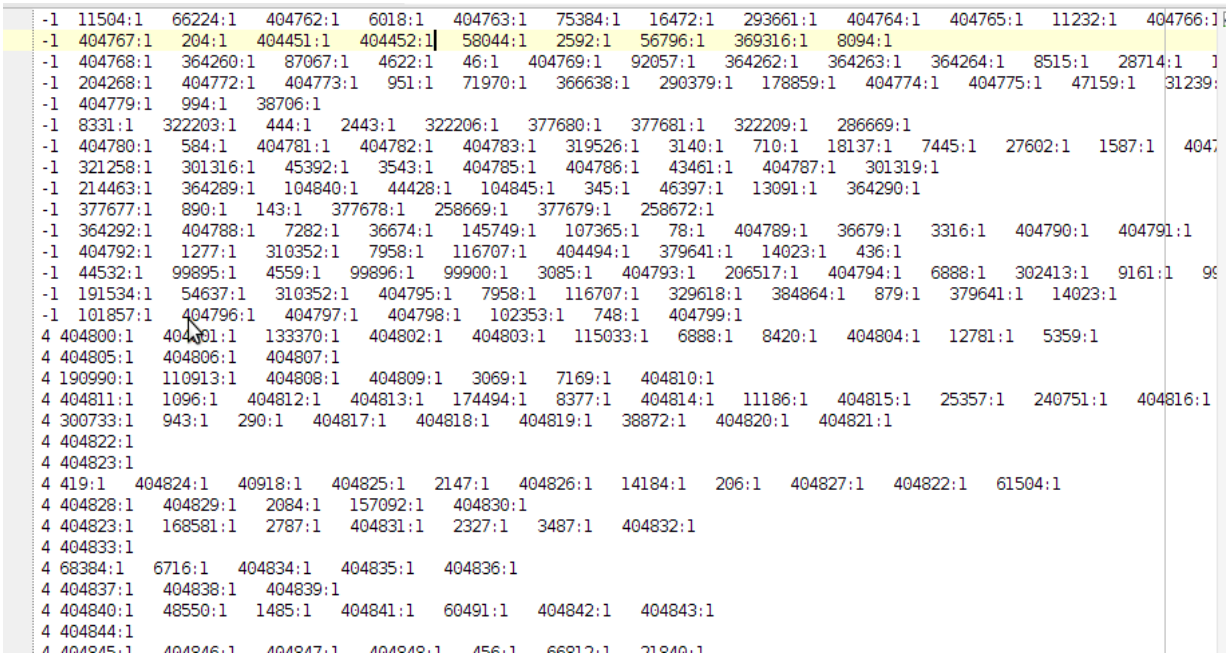

Figure 5. Training File for the category "Technology". Tweets belonging to this category have a label of 4 before them; all other tweets have a label of -1.

4.5.4 Creating the model

In order to choose the best model out of the models listed above, we had to find out the type of classifier that returned the best C-value for each of the classes. This understanding was gained from the paper [FC] (see Bibliography).

LIBLINEAR supports the option of "-c i" in its "train(..)" method call. Thus, our objective was to find the best value of "i". We approached this problem by iterating each of the 8 types of classifiers (Section 4.5.1) over 7 c-values ranging from 1000 to 0.001 and trained 4 separate classifiers on 100,000 tweets from all four categories.

We ran 10-fold cross validation on each classifier and received precision and recall values.

Precision: is the fraction of retrieved instances that are relevant

Recall: is the fraction of relevant instances that are retrieved.

Based on the different values of precision and recall for different classifier types, the model with type "**L1- logistic regression**" with a c-value of 5 exhibited the best performance since it had the overall highest average precision and recall values of 0.89 and 0.88 respectively for each of the categories.

4.6 Evaluation of the Classifier

In order to evaluate the classifier, we first created a new Twitter account called 'TwistClassifier' which follows a diverse set of Twitter users. Then, we streamed the home timeline data of this user from Twitter. The data was stored in text file and preprocessed (refer section 4.2 for pre-processing steps).

Word Dictionary - The unique words (tokens) and bigrams obtained from the test dataset were stored in global word dictionary (4.5.2). Words were only added to the dictionary if they were not already present.

Similar to the training file as shown in Figure 5., the test data tweets were all stored in another text file. The text file input for the LIBLINEAR testing method was as shown below:

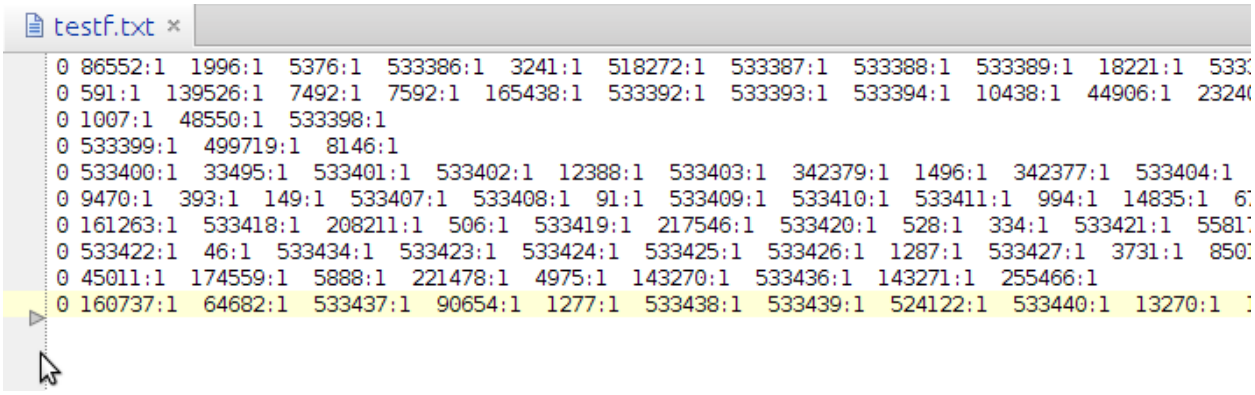

Figure 6. Illustration of the test input for the classifier.

Note that the category labels for the test data are all 0s. Each of the numbers before the ":" are the indices of the word in the global word dictionary. 1 after the ":" indicated the presence of that word in the tweet.

4.7 Output of the classifier

The tweets are passed through all these four classifiers are the output of the classifiers is illustrated as shown below:

Sports: [-1.0, 1.0, -1.0, -1.0, -1.0, -1.0, 1.0, 1.0, -1.0, -1.0, -1.0, -1.0, -1.0]

Finance: [-1.0, -1.0, -1.0, -1.0, -1.0, 2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 2.0, -1.0, -1.0, -1.0, -1.0, -1.0, 2.0, 2.0, -1.0, -1.0, -1.0, -1.0, -1.0, 2.0, -1.0, -1.0, -1.0, -1.0, -1.0, 2.0, -1.0, -1.0, -1.0, 2.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 2.0, -1.0, -1.0, -1.0, 2.0, 2.0]

Entertainment: [3.0, -1.0, -1.0, -1.0, -1.0, 3.0, -1.0, -1.0, -1.0, -1.0, 3.0, -1.0, -1.0, -1.0, -1.0, -1.0, 3.0, -1.0, -1.0, -1.0, -1.0, -1.0, 3.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 3.0, 3.0, -1.0, -1.0, -1.0, -1.0, 3.0, -1.0, -1.0, 3.0, -1.0, -1.0, -1.0, -1.0, -1.0, -1.0, 3.0, -1.0, -1.0, -1.0, -1.0]

Technology: [-1.0, 4.0, -1.0, 4.0, 4.0, -1.0, 4.0, 4.0, 4.0, -1.0, -1.0, 4.0, -1.0, 4.0, 4.0, 4.0, -1.0, -1.0, -1.0, -1.0, 4.0, 4.0, -1.0, 4.0, -1.0, -1.0, 4.0, -1.0, 4.0, -1.0, -1.0, -1.0, 4.0, 4.0, -1.0, 4.0, 4.0, -1.0, 4.0, 4.0, 4.0, -1.0, -1.0, 4.0, -1.0, -1.0, 4.0, 4.0, -1.0, -1.0]

Figure 7. Output of each classifier for 50 tweets

This illustrates the outputs of four different classifiers for a set of 50 test tweets. Each of the labels in the output list belongs to the tweets appearing in that order. It represents whether or not a particular tweet belongs to that category. The lists above contain 50 values each for 50 tweets. Thus, if the first tweet is related to Sports as well as Finance, then the first values in the respective lists would be 1 and 2 respectively.

We then interpret these lists for each of the tweets and come up with the category of that tweet.

4.8 Summary of the flows

In summary, there are two basic flows in our application: to train and to classify the tweets. The two figures below summarize, in an abstract manner, these two different flows.

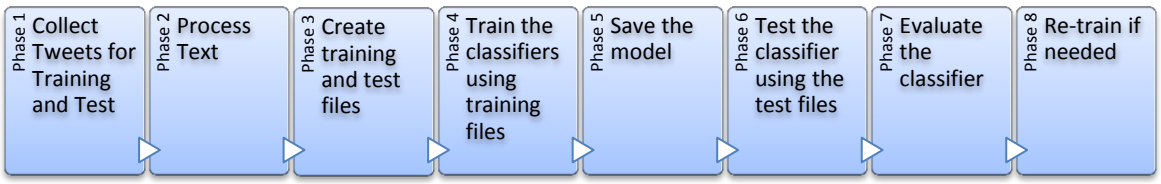

Figure 8. Training Flow of the Application

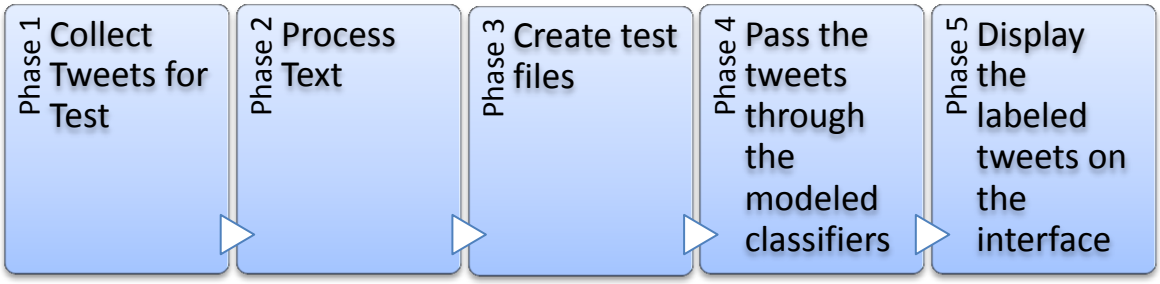

Figure 9. Live Flow of the Application

4.9 Front End

Front end offers a convenient way to the users to look at the predictions made by the classifier. In order to show the result of the classifier we created a web page which takes the user handle as the input and outputs the classified tweets on the user's home timeline. Figure 10. shows the screenshot of the running application.

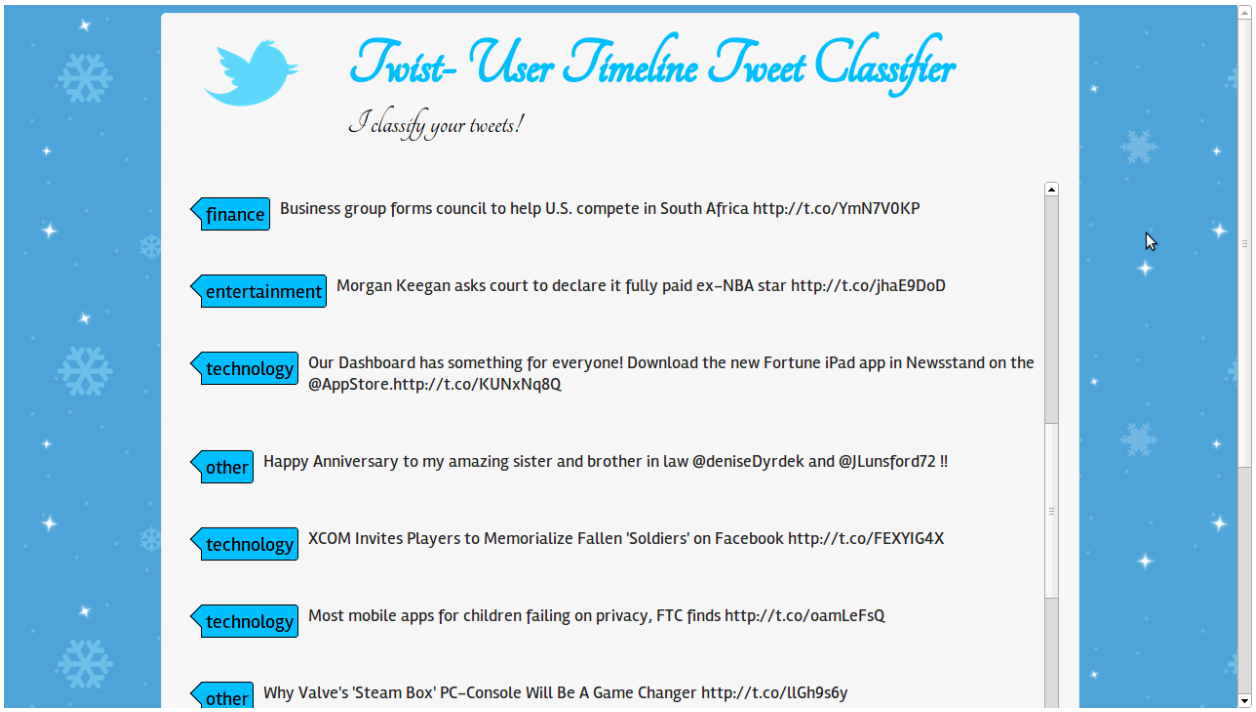

Figure 10. Illustration of the running application; each of the tweets are labeled

5. Coding & Documentation

Used Python for developing algorithms specified above.

Scripts are available at: **https://github.com/priya-I/Twist**

All the data collected for the purpose of training is available in the form of text files at: **https://github.com/priya-I/Twist/tree/master/flatfiles.** These files containing the training data set have been highlighted in the software documentation.

Instructions to install and run the code are available in README.txt

6. Conclusion

In this project, we attempted to present a way to create a classifier module using supervised machine learning techniques. Using Logistic Regression, we were able to predict the category(ies) of a given Tweet with an average precision-recall value of 0.89-0.88 and an accuracy of 92% on an average.

With respect to the short and sparse information transported with a single Tweet, we showed that it is best to collect tweets only from the influential people on Twitter and not use Twitter public streams for a positive training data set. Using pre-processing (stop word removal, spell check, stemming & lemmatization) and bi-gram representation has a positive impact on the level of accuracy, but reduces the processing time. Changing the cost parameter and type parameter for LIBLINEAR train module helped us choose the best type of classifier for our classification.

In order to create multiple labels for tweet and to categorize the tweet under 'Others' in case it does not belong to any of the other predefined categories, we created four separate one versus rest classifiers and for each classifier model we used the same training set but divided into positive and negative examples for that category, thus ending up with four different training files for the same data set. Increasing the size of the training dataset further added to improvement in the overall precision and recall.

7. Project Milestones and Timeline

Date	**Milestone**
Oct 28, 2012	Collect twitter dataset and install required APIs.
Nov 1, 2012	Given a set of sample tweets, algorithm should classify it into the first set of categories.
Nov 10, 2012	Refining the algorithm by assessing the initial classification and identifying the hidden topics.
Nov 13, 2012	First Deliverable
Nov 20, 2012	Creation of a high fidelity front end prototype supporting the application. Machine learning will be in progress.
Nov 27, 2012	Refining the algorithm and finalizing the front end.
Dec 4, 2012	Creation of front end.
Dec 6, 2012	Final Deliverable

8. Percentage Contribution of Each Team Member

Phase 1

Tasks	Vaidy	Priya	Sonali
Data Collection	35	30	35
Literature Review	33	33	33
Planning Applications Architecture	25	50	25
Data Base Design	20	40	40
Text mining	60	20	20
Insertion into database	30	35	35
Code Optimization	40	40	20
Documentation	20	20	60
Meetings	33	33	33

Phase 2

Tasks	Vaidy	Priya	Sonali
More Data Collection	20	20	60
Refinement to Text Mining	50	30	20
Transforming data into LIBLINEAR training and test set	20	60	20
Initial training and testing of the LIBLINEAR model	20	60	20
Tuning parameters for the model	20	45	35
Evaluation of the model	15	45	40
Twitter Website	30	20	50
Code Optimization	35	35	30
Code integration	40	40	20

Documentation	30	30	40
Meetings	33	33	33

9. Acknowledgment

We are extremely grateful to our mentor, Andy for saving our precious time with his extremely insightful inputs. We owe him the timely completion of this project.

Also, we would like to thank our Professor, Marti, who made sure we were always on track with the project and for setting us up with one of the most helpful mentors.

Appendix A : User accounts used for training

Sports	**Entertainment**	**Finance**	**Technology**
SInow	EW	WSJ	karaswisher
SportsCenter	Marvel	TheEconomist	cultofmac
TwitterSports	GoogleMandE	mutualofomaha	sacca
NBCSports	MSN_Entertain	YahooFinance	ForbesTech
SkySportsNews	tw_top_ent	Forbes	om
YahooSports	starz_channel	daily_finance	kevinrose
SkySports	SummitEnt	GoogleFinSvcs	guardiantech
FOXSports	msnents	FinancialTimes	ericschmidt
WarrenSapp	CNNshowbiz	jnovogratz	pierre
rudygay22	TODAY_ent	zappos	mikeyk
DaraTorres	ThinkFlash	brianmoran	woot
FCBarcelona	THR	jimcramer	LaughingSquid
nyjets	Variety	Reuters_Biz	FCC
AroundTheHorn	19News	CNBC	google
womensprosoccer	digg_entertain	BBCBusiness	davemorin
London2012	bbcentertain		waltmossberg
Shaun_White	CALEntertainmnt		cshirky
ChrisJohnson28	celebcircuit		gadgetlab
TroyAikman	eonline		arstechnica
JozyAltidore	pulse_entertain		TheNextWeb
paulpierce34	accesshollywood		TEDchris

MichelleDBeadle ErinAndrews NCAA SportsNation KingJames Olympics robdyrdek	kingsthings WarnerBrosEnt		wired_business TechCrunch RWW

Bibliography

[FC] Rong-En Fan Kai-Wei, Chang Cho-Jui Hsieh, Xiang-Rui Wang, Chih-Jen Li LIBLINEAR: A Library for Large Linear Classification http://jmlr.csail.mit.edu/papers/volume9/fan08a/fan08a.pdf

[H] Christopher Horn Analysis and Classification of Twitter messages http://know-center.tugraz.at/wp-content/uploads/2010/12/Master-Thesis-Christopher-Horn.pdf

[CL] Chih-Wei Hsu, Chih-Chung Chang, and Chih-Jen Lin A practical guide to libSVM http://www.csie.ntu.edu.tw/~cjlin/papers/guide/guide.pdf

Websites

www.wikipedia.org

www.stackoverflow.com

www.google.com