Detecting Real-Time Messages of Public Interest in Tweets

Divij Gupta, Chanh Nguyen {divijg, chanhn}@cs.stanford.edu
4th June, 2010

1 Introduction

A user base of over 80 million has positioned Twitter as one of the largest micro-blogging platforms in the world, providing a wealth of user generated content in the form of short 140 character messages known as tweets. With users providing information on almost every aspect of their lives and that of their surroundings through these location-tagged tweets, an interesting proposition is to separate the information that might be useful for an individual or enterprise. To do so, we want to be able to detect whether a tweet was intended as a public announcement, rather than just a status update or an opinion. For example given the tweet, "Live in Birmingham, this Friday 4th June with The Destroyers - tickets here:http://lnk.ms/9NWV9 http://lnk.ms/9NWV9", we would like to be able to separate this tweet from a barrage of others which are of the form "I can now add 'freddy' to my extensive list of wrong names i've been called", and are essentially useless for our purposes.

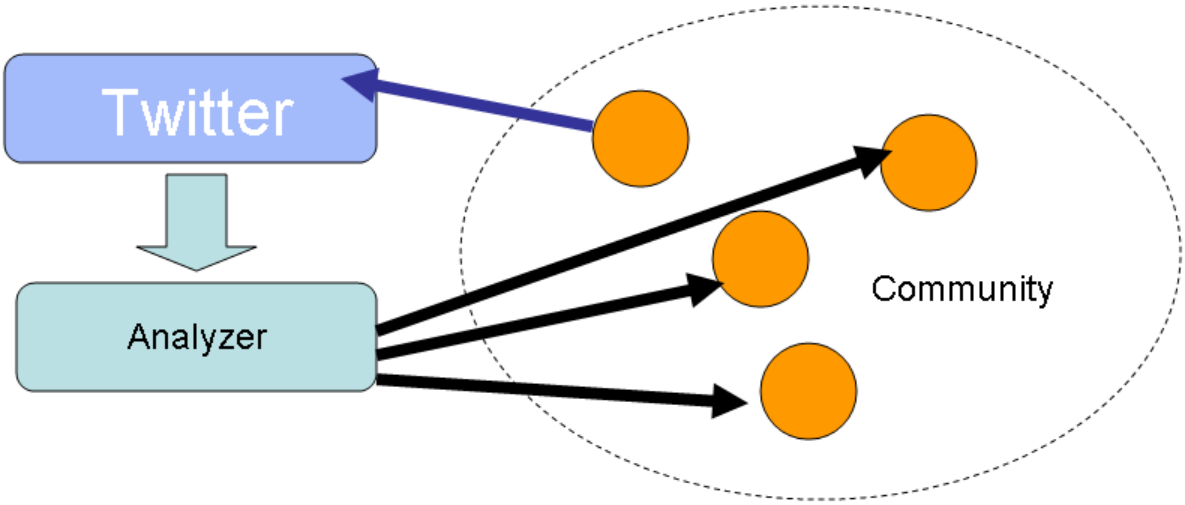
The inspiration for the project is the thought that what better place to look for event promotions or updates than on a site in which users are communicating in real-time. Consider a potential application: a user in a new city, say Berlin, looking for information about the best deals on sightseeing tours during that part of the year. Instead of using manually searching multiple sites via Google, which could possibly have outdated information, our application could instead leverage Twitter's public API to search for tweets about deals on sightseeing tours in Berlin, providing the user with a lot more real-time information and a better experience by automating the process. Hence one could extend this idea to any type of event one would like to search for anywhere in the world whether it has to do with food, music, movies, health, spirituality etc., just as long as people are tweeting about it.

A potential application would then need to constantly analyze a stream of incoming tweets and separate the one which appear to be event broadcasts from those that are not. Keeping in mind the nature of the problem we employ a number of machine-learning techniques to identify such tweets based on features extracted via natural language processing principles as well as the characteristics of publicly-aimed tweets. With the number of positive instances being rare relative to the number of negative instances we were aiming for a high negative accuracy rate so that most of the negatives would be rejected as well as a respectable positive accuracy rate given that some tweets turn out to be ambiguous in terms of them being intended as an announcement or not which might

lead to them being rejected by the system. For example "Back from study!!! Later I want my Macgriddle with Egg meal... Anyone want to join me?" might be an invitation to only the user's group of friends, but could also be an open invitation to anybody who happens to be in the same area.

Although some work has been done in this area (University of Tokyo - Semantic Twitter: Analyzing Tweets for Real-Time Event Notification by Makoto Okazaki and Yutaka Matsuo, the diagram shown below is based off one of their slides), their application queries twitter for certain search terms which makes it easier to detect events, ours is based more on a general streaming system such that in the optimal case we can detect any possible public announcement.

Figure 1. Possible architecture for a future application



2 Resources

Our primary data source was a set of tweets taken from the website of the Stanford class on data mining (cs345a). The dataset consists of about 25GB of tweets from June to December of the year 2009. Since no tagged dataset was available we had to manually construct out training and test sets which took a long time especially since the number of positive instances we encountered were very few. We specifically restrained ourselves from manually searching twitter for positive instances in order to simulate the streaming of data and to not bias our training or test sets.

For every tweet we read, we took one of three possible actions. If the tweet looks like it was meant to reach as many people as possible, such as the description of an event or a

question asking for public opinion, we marked it as positive. If not, we marked it negative. If the tweet was written in a foreign language, was unreadable, or was too ambiguous in terms of intended audience, then we would exclude it from our training or test sets.

We constructed a training set with 545 positive instances and 1457 negative instances. We went through the first 2700 tweets of the December 2009 data set and classified each tweet in one of the above three ways. The total does not add up to 2700 because at one point we began to skip everything but the positive examples because they were so rare. In order to find more publicly-aimed tweets, we connected to Twitter with the JTwitter API and searched for tweets using terms such as "sale", "today", "tomorrow", "free", and "anyone." Performing such searches did not return only positive results. Only about 40% we classified as positive, 40% we classified as negative, and the rest we discarded. Adding an equal number of positive and negative examples to the training set helps to prevent bias on the particular search term.

For the test set we did not include any results obtained by performing a specific search through the Twitter API because we wanted to test the system's ability to classify every message coming through Twitter. Thus, we went through a different data set, from June 2009, and started from the beginning and classified the first 1400 tweets, resulting in a test set of 112 positive and 983 negative tweets.

We employed four classification techniques, the first being Support Vector Machines (SVM's) for which we used the opensource LIBSVM package provided by Chih-Chung Chang and Chih-Jen Lin from the National Taiwan University, the second being Logistic Regression, the third being Naive Bayes (in two variations), and the last being Boosted Decision Trees. The code for Logisitic regression and Naive Bayes was taken from the ones written for CS109 and adapted to the current problem. In order to construct the feature vector we made use of WordNet from Princeton University's site as well as the Stanford Part of Speech Tagger provided by the Stanford NLP group. We used an open-source Java implementation of Boosted Decision Trees called JBoost which contains AdaBoost. We also used Unigram and Bigram language models that we previously built as part of CS224n.

3 Code Structure

The following is a description of the layout of the code in the files submitted. SVMTrain.java contains the code for training and testing with the aid of a jar for the LIBSVM package and also code for Part of Speech tagging the train and test sets via a jar for the Stanford POS tagger. The file LogisticRegression.java contains implementation of both the Logistic regression and Feature-Based Naive Bayes algorithms. Tweet.java, WordStore.java and EventDescription.java are used to store certain state associated with each tweet. Tweet.java also contains the function construct_feature_vector(), which processes the tweet

and constructs the feature vector for use in testing and training.

TwitterHarvester.java launches a UI that pulls tweets from either a file or from Twitter through a specified search query, and allows the user to classify and write each tweet to disk with just one keystroke. TweetProcessor.java is used to read in tweets from file and annotate them specifically for Unigram and Bigram based Naive Bayes as well as extracting features for Boosted Decision Trees. NaiveBayes.java contains the Unigram and Bigram based Naive Bayes algorithm.

4 Feature Extraction

One of the difficulties in working with data from twitter from a natural language perspective is the highly informal and seemingly incoherent nature of tweets. Hence, in order to get the best results we needed to understand the characteristics of twitter jargon in order to transform it into a structured format such that it would be easier to operate on. The pre-processing steps are as follows:
a) Rectify words in tweets that have a stream of the same consecutive characters, e.g. users are likely to write "TICKETSSS AVAILABLE", which when we search for "TICKETSSS" in WordNet, dosen't show up, hence it needs to be corrected to become "TICKETS AVAILABLE"
b) Remove unnecessary punctuation that comes up in tweets such as in "Snowboarding: I love snowboarding esp with fresh powder all around me: (from the list "Sports and Outdoor Adventure... http://bit.ly/4ZOQed" , removing the punctuation marks like ":" and "..." which distort the contents of the tweet.
c) We also tried to use an open-source spell checker to rectify certain words which might have been misspelled by users, however the open-source spelling checker (Jazzy) that we were using took too much time to process each tweet and in most cases it was infeasible to pick the correct suggestion for the misspelled word given that only the user is aware of the context in which he meant to use it. A case being "Nooo! I was looking forward to this! Bummer. RT @WaltDisneyAnim The premier of Prep and Landing has moved to Tues, 12/8. ..." , where the user meant to use the word premiere which means movie opening instead of premier which means chancellor.

Once the tweet was pre-processed we then part-of-speech tagged it using the Stanford POS tagger and saved it for later use. We then moved onto the process of feature extraction. One of the key ideas that we had was to use the descriptions of a word and its synonyms provided by wordnet, to aid in constructing the feature vector. For example most of the positive instances that we encountered had some defining features such as an event location whether it be a city, country, public space etc. Another being an event purpose such as entertainment, performance, game, sale, activity etc. Hence, we used WordNet to check whether the description of a word contained one of the above characteristics and

accordingly constructed a vector of binary features signifying whether the feature was present or absent. Eg. Two tweets in the training set being "Rock concert tonight in Birmingham" and "Film festival in Berlin", when we search for all the words of each tweet in WordNet, Berlin and Birmingham will both be described as cities hence the feature for existence of place in the tweet will be 1. Similarly festival and concert will both have the word "entertainment" in their description and hence the feature for the presence of a event purpose will be 1. The approach is almost as if we were replacing the word in the tweet by a simpler constant version, with the optimal case being any place being replaced by PLACE, any activity being replaced by ACTIVITY and any date/time by DATE_TIME.

Apart from using Wordnet to get a simplified representation of a word, for other features we also heavily relied upon regular expressions. The case being the detection of date/time in which users represent them in multiple ways such as "26/7", "on the 26th", "26 May", "26-09-2010" etc. Hence we had to come up with regular expressions which took into account almost every possible representation of dates and times in tweets. The following two are for dates and times:

```
"((19|20)\\d\\d)|
(\\d{1,2}\\s*(th)+)|
(\\d+[/]\\d+)|
([3]\\s*(rd)+)|
([2]\\s*(nd)+)|
(\\d{1,2}[-/ ]{1}\\d{1,2}[-/]{1}\\d{1,4})"

"([1-2]?[0-9](:)+[0-5][0-9]\\s*(am|pm)?)|
([1-2]?[0-9][0-5][0-9]\\s*(am|pm)+)|
((([1][012])|([1-9]))\\s*(am|pm))"
```

We also took into account the description of conventional events which usually have to deal with the the sale or offer of an item, hence checking whether the tweet contains an indication of a price or percentage, indicative of an offer. Another feature that stood out was users preference of reaching out to certain groups when broadcasting events. Hence such tweets have the terms "anyone", "everyone" or nouns in the plural form such as "shoppers" or "music listeners"; this is contrast to negative instances of tweets in which people are more expressive about the happenings in their own lives which are characterized by the use of terms like "i", "my", "im", "ur" etc. We also looked for questions bydetecting words such as "who's" and "where" and the presence of a question mark. Other features include the presence of links, hashtags(#) and RT (re-tweets) which in a lot of relate to the promotion of an event while features such as presence of the @tag (refers to another twitter user) or emoticons (:) or :P) usually are present in more personal tweets. Hence, we ended up with a vector of 17 binary features for each tweet which were then used for training and testing with the classifiers.

5 Classifiers

5.1 Naive Bayes

Naive Bayes is a generative model characterized by a strong assumption of the independence of the features/words it operates on. The probability of a feature/word given the class is based on the joint probability of the feature/word and the class. The aim is to choose the class which maximizes the following:
$\prod_1^n P(f_i|c)$, where f_i represents the current feature/word and c is the current class being evaluated.

Language Model Approach

We began by using the words in each sentence as features themselves. The probability of a word comes from either a Unigram Language Model or Bigram Language Model, both of which keep count of how many times a word or pair of words appear given a particular label. We started with both models using Delta Smoothing, although we found that the Bigram model performed poorly due to the relatively small training size, so we switched to a Backoff model that returned the Unigram probability if an unknown bigram was encountered.

Figure 2. Training Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	499	495	99.1983
Negative tweets	503	503	100

(Testing Set Accuracies)

Figure 3. Unigram - No Features

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	52	64.19753
Negative tweets	833	686	82.35294

Figure 4. Unigram - All features

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	59	72.8395
Negative tweets	833	674	80.9123

For the language models, we represented features by replacing certain words with special tags. For example, we used regular expressions to detect times such as "10pm" and replaced it with the tag _TIME. This way, all occurrences of time in tweets would be understood by the language model as one word and it can make more powerful guesses. We also grouped together similar words such as "hire, hiring, job, career" into a single synonym so that we reduce the number of unknowns.

Figure 5. Bigram - Add Delta Smoothing

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	62	76.5432
Negative tweets	833	395	47.4189

Figure 6. Bigram - Unigram Backoff

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	50	61.7283
Negative tweets	833	689	82.7130

Feature Based Approach

We noted that the performance of our model improved as we added features, so we wanted to try and use Naive Bayes to operate only on a limited set of features, obtained by the described feature extraction algorithm in section 4. We took the features and used a binary feature vector where 1 meant the feature was present in the tweet and 0 meant it was absent. Hence given our training vectors we evaluated the probabilities of the binary values of the features vectors for each class, with Laplacian smoothing in order to take into account possible zero counts for a certain feature. The training set was made to consist of an equal number of positive and negative sample (500 each). The results are as follows:

Figure 7. Training Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	495	403	81.41414
Negative tweets	503	427	84.890656

Figure 8. Testing Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	59	72.83951
Negative tweets	833	646	77.55102

5.2 Logistic Regression

Logistic regression is a probabilistic model that fits data to the logisitic curve, the outcome z is confined between the values of 0 and 1 by operating the logisitic function $f(x) = 1/(1+e^{-z})$, on it giving $f(z)$, such that $z = \beta_1 * f_1 + \beta_2 * f_2 + \ldots + \beta_n * f_n$, where the betas represent the regression coefficients that are tuned via the stochastic gradient descent algorithm. The probability of the class being positive is set to be above a certain threshold, hence in testing after the outcome of our provided feature vector has been calculated, if the result is above the threshold it is classified as positive else negative.

Probability Threshold = 0.3, epochs = 5000, acceptable error = 0.0001

Figure 9. Training Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	495	403	92.72727
Negative tweets	503	427	80.5169

Figure 10. Testing Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	67	82.71605
Negative tweets	833	551	69.868996

5.3 Support Vector Machines

Support Vector Machines are a class of supervised learning algorithms that can be used for regression or classification (as is the case for our purpose). Intuitively, in the training phase it maps a given set of points to a higher dimension using a pre-defined kernel and then finds a set of hyperplanes such that it separates the points of the different categories by as large a margin as possible. The points closest to the hyperplane are known as the support vectors with the aim being that the ones from the different categories lie as far apart from each other. In our case we use the API provided by the LIBSVM package, the parameters

for the selected pair of svm type and kernel function are found by a parallel grid search with cross-fold validation on the training data, the python script for which is already provided as part of the package.

Svm Type = C_SVC (type-2 classification), Kernel type = RBF (Radial Basis Function), Gamma = 0.0078125, C = 2048 (optimized via grid search for the training set)

Figure 11. Training Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	497	442	88.9336
Negative tweets	503	457	90.85487

Figure 12. Testing Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	81	60	72.289154
Negative tweets	833	665	79.83193

5.4 Boosted Decision Trees

Decision trees are a classification mechanism which during the process of training finds the feature that is most important to split on based on the potential information gain. Our motivation for using this classifier was the fact that decision trees seem immune to the use of redundant features and hence this would select the best features to split on. Boosting is a optimization that works by assigning greater weights to stronger classifiers, which are derived from weak classifiers by retraining on samples that the weak classifiers misclassified.
Using the same binary feature vectors, we ran the test set on an AdaBoosted Decision Tree and achieved better classification for negative tweets, but much worse classification on positive tweets.

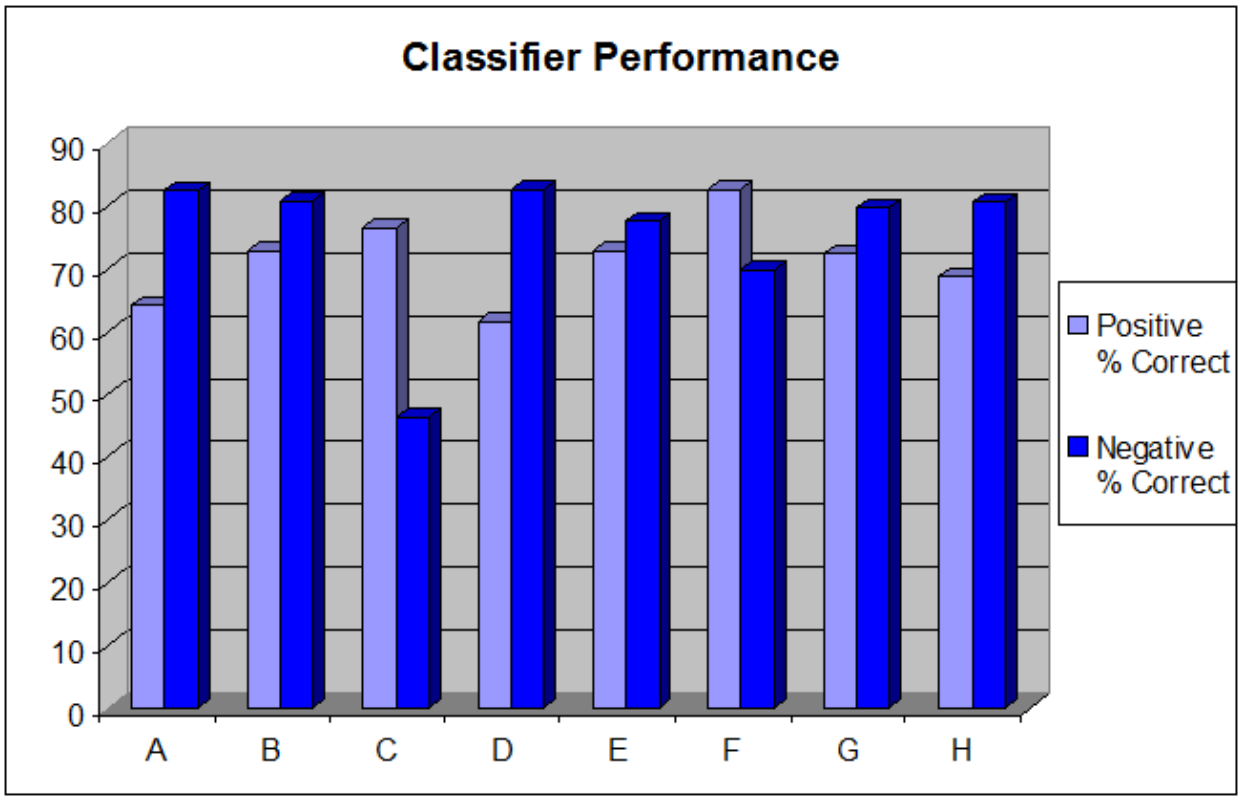
Figure 13. Training Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	497	442	89.13
Negative tweets	503	472	93.8

Figure 14. Test Set Accuracy

	Number Tested	Correctly Classified	Accuracy (%)
Positive tweets	83	57	68.7
Negative tweets	833	671	80.55

Figure 15. Comparison of All Methods (test set)



A. Naive Bayes (unigram no features)
B. Naive Bayes (Unigram - All features*)*
C. Naive Bayes (Bigram - Add Delta Smoothing*)*
D. Naive Bayes (Bigram - Unigram Backoff*)*
E. Naive Bayes (Feature-Based Approach)
F. Logistic Regression
G. Support Vector Machines
H. Boosted Decision Trees

6. Error Analysis

6.1 Language Model Error

The biggest problem with using the language models is that sometimes it may not be appropriate to calculate the probability of a single word, as words often appear in groups and in different forms, and so this causes two problems. First is that since words often appear together, we cannot apply the Naive Bayes' assumption of independence. Second, we don't count synonyms as the same word, although we sometimes should. For example, the tweet:

> "Freelancejobs $3 per 300 words!! by writeworks: Only Italian Writer!! I need you to write 50 articles of 300 wor... http://bit.ly/5HCszt"

is a publicly aimed tweet but was miscategorized until we transformed it into the following annoted tweet

> "_JOB _PRICE per 300 words by writeworks: only italian writer _I need _YOU write 50 articles of 300 wor _LINK ly/5hcszt"

6.2 Feature Detection Error

One of the things we noticed is that increasing the number of negative samples in the training set (to about three time the number of positive samples), led to an increase of about 10% in negative accuracy but a decrease of about 15-20% in positive accuracy depending on the type of classifier. The case being SVM's which have negative accuracies of over 90%, yet positive accuracy drops to about 50%. This shows that the system merits a lot more positive data else the classifier is heavily biased towards negative features and even certain tweets like "i'm going to the Bon jovi concert 2night, anyone wanna come?", due to the presence of the phrase "i'm", make the system reject it due to a greater negative weight attached to the phrase even though the tweet also has positive connotations in the form of an event type i.e. concert and a broadcast to the public through the use of the word "anyone".

Using the idea of WordNet to translate from words to a simpler common type in terms of place, time and event type leads to about a 10% increase in positive accuracy and 5% increase in negative accuracy for logisitic regression.

Similarly features like detection of some sort of sale/offer or a price/discount match leads to a 10% increase in positive accuracy and detection of phrases like "i", "i'm", "ur" which in a lot of cases point to personal tweets leads to a 5% increase in negative accuracy, both for SVM's. Hence a lot of our features end up helping us strongly distinguish between positive and negative tweets.

6.3 Data Set Error

Since we classified tweets by hand, there were a lot of misclassifications, most of which were removed by making a second pass through the classified data. For example, on the

first pass, we classified the following tweet as positive:

"Hello everyone!! Eating pizza!!! Love it"

It seemed like a message targeted at the general public due to the presence of "everyone" but on second glance, it is simply a personal message to friends about the user's status. This and many other tweets are examples of how the problem we are trying to solve can be tricky, even for humans.

The size of the data sets are also another area of concern. It's unreasonable to expect a training set of just over 1000 tweets to contain all the words, bigrams, and features that will be seen in any possible test set. The data that we present in this paper comes from using our largest training set, but it was not that large to begin with. We began with only about 500 tweets in the data set, but quickly realized the importance of a large data set, so we added more.

7. Event Information Extraction

Once tweets were classified as positive, we then devised a mechanism to extract the critical information in order to index it and then offer it to the user in a structured format. The characteristics displayed back to the user are date/time, place, price, event_type and general description. In order to extract them we basically used a combination of matching certain parts of the tweet via regular expressions (as in the case of date and time), use of wordnet to detect places/event types. In order to summarize the tweet we also Part of Speech tagged it and extracted the parts that represented any form of a noun or verb to be included in the general description. For e.g. the following tweet "Live in Birmingham, this Friday 4th June with The Destroyers - tickets here:http://lnk.ms/9NWV9 http://lnk.ms/9NWV9", is displayed as:

DATE_TIME: 4th Friday June
PLACE: Birmingham
DESCRIPTION: link-link-http://lnk.ms/9NWV9 Live Destroyers tickets

8. Conclusion

In order to make the system more robust there is definitely a need to gather more positive training data. However we feel that even with our current results from the four different types of classifiers we have a good enough system at hand which can be further improved by simply determining the final class of the tweet as a voted decision by all the classifiers. Once a tweet has been classified as containing relevant information it becomes fairly simple to summarize the event information from the tweet based on the features that were already extracted by us from it. Also if a user were to provided a search term such as "movies", it would be even simpler to match it against a repository of positive tweets, either

by term matching or using WordNet to match synonyms. Hence this could be a be a very useful application given the increasing importance of real-time search for consumers in our everyday lives.

9. Bibliography

a) Ravi Parikh and Matin Movassate. Sentiment Analysis of User-Generated Twitter Updates using Various Classification Techniques. cs224n Project 2009

b) Christpher Manning. Lexical Semantics. cs224n Lecture Slides, 2010

c) Ralph Grishman. Event Extraction: Learning From Corpora. NYU, July 2003

Collaboration: Divij worked on the svm code, feature extraction (with use of wordnet) , logistic regression, naive bayes (feature-based approach), and part of speech tagging for extracting event information from positive tweets. Chanh worked on feature extraction, naive bayes (language-based approach), boosted decision trees and the dataset construction.