Twist- User Timeline Tweet Classifier

Team Members

- Vaidy Venkat
- Sonali Sharma
- Priya Iyer

Mentor: Andy Schlaikjer

Project Goals

Currently twitter allows users to create custom Friend Lists: an example of individual categorization .e.g. @newyorktimes, @cnn are usually classified under "News" or "TellMeWhatsHappening" list. Generally, these lists are based on the user and not on the text of the tweet. The goal of our application is to:

a) Analyze the tweets on a given user's timeline, clean the tweets and then employ machine learning techniques to automatically classify the tweets under predefined categories based on one or more features of the tweets.
b) Create a user interface to display tweets on user's timeline, labeled with categories and create a pie to show percentage split of tweets amongst the categories.

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 is 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, in progress
- Text Mining: creation of custom algorithms to clean data specific to tweets
- Creation of Training Set and Test Set
- Creation of algorithms to train and test the SVM
- Creation of a basic user interface to display labeled tweets

Phase 2:

- Training and refinement of the machine with feedback: cross validation, include more features to train the SVM and train with more data/tweets
- Evaluation of effectiveness and efficiency
- Further improvements to the user interface

Workflow of the Project Activities

Input: User's twitter handle and the user's twitter stream

1. Read all the references and do initial research.

2. Create the architecture of the classifier.

3. Identify twitter handles for each of the categories, download their tweets and use the same as training set for that category. Also, stream tweets as a basis training data set.

4. Write an algorithm, which will use the training set, pre process it (text mining), create feature vectors and prepare an input for the Classifier. Classify the data into one of 5 to 6 categories such as: Sports, Finance, Entertainment, Technology, Personal, etc.

5. Create a web interface, which enables the basic functionalities such as:

    a.) Enter username

    b.) Classify tweets for that user

c.) Provide user with an ability to modify classifications if necessary (modify labels using a drop down)

6. Run the algorithm created as part of step 2 in order to initially classify the incoming twitter stream for a particular user on the web interface. Maximum number of tweets considered= 200 (limitation of the API).

7. User to verify the categories of classified tweets and reclassify if necessary.

8. Repeat steps 6 and 7 until at least 70% of the tweets are correctly classified. For example, if we have 200 tweets, 140 of them need to be correctly categorized. This step will continue until the end of the project.

Additional tasks :

1. Create documentation for the architecture of the classifier.

2. Create visualization, which shows the percentage of the categories for a given user using a pie/bar chart.

Accomplishments

- **Further refinement of project goals:**
  - Realized that classification of tweets is different from other kinds of text classification, so we decided to use tweets alone as our training and test data set and collected data from Twitter's recommended "Who to follow" list under various categories. (https://twitter.com/i/#!/who_to_follow/interests)
  - Based on the advice of our mentor, we decided to use categories, which were mutually exclusive and more granular. Our new categories are: Sports, Finance, Entertainment, Technology, and Personal.

- **Data Gathering:**

For initial training purposes, we were advised to use only two categories which are mutually exclusive and easy to identify. For this purpose we chose: sports and finance. Thus, the data collection was based on specific twitter handles in these categories.

For each of these categories we identified a set of top 10 handles and collected 200 tweets from each handle thereby making the total count of tweets for these two categories as 40000. The handles used for each of the categories are:

Sports: @SInow, @SportsCenter, @TwitterSports, @NBCSports, @SkySportsNews, @YahooSports, @SkySports, @FOXSports

Finance: @WSJ, @TheEconomist, @mutualofomaha, @YahooFinance, @Forbes, @daily_finance, @ftfinancenews, @GoogleFinSvcs, @CNNMoney

These tweets were collected using twitter4j API and stored in .txt files. We then analyzed to tweets to come up with a strategy to clean the data. For the fist phase of the project we did not want to use too much of data because our aim was to test the lib SVM classifier with just enough data so that any significant problems with the input file could be identified easily.

- **Algorithm Development:**

In the first phase of the project most of our algorithm building was around Text Mining and creation of feature vectors using tf-idf.

- Text Mining algorithm included Removal of special characters, tokenization, removal of redundant letters and words, spell check, stemming, removal of stop words, generation of bigrams and conversion to lower case.
- **Text Mining**
  - The pre-processing of the tweets is done in order to extract the features to be used for the classifier.
  - Initially, the tweets are tokenized and the special characters are removed. Only characters like #, ` and – are retained in the tweets.
  - Then the tweets are checked for spelling and then the resultant tokens are stemmed using Porters algorithm.
  - Stop words are also removed and then bigrams are generated from the tokens.

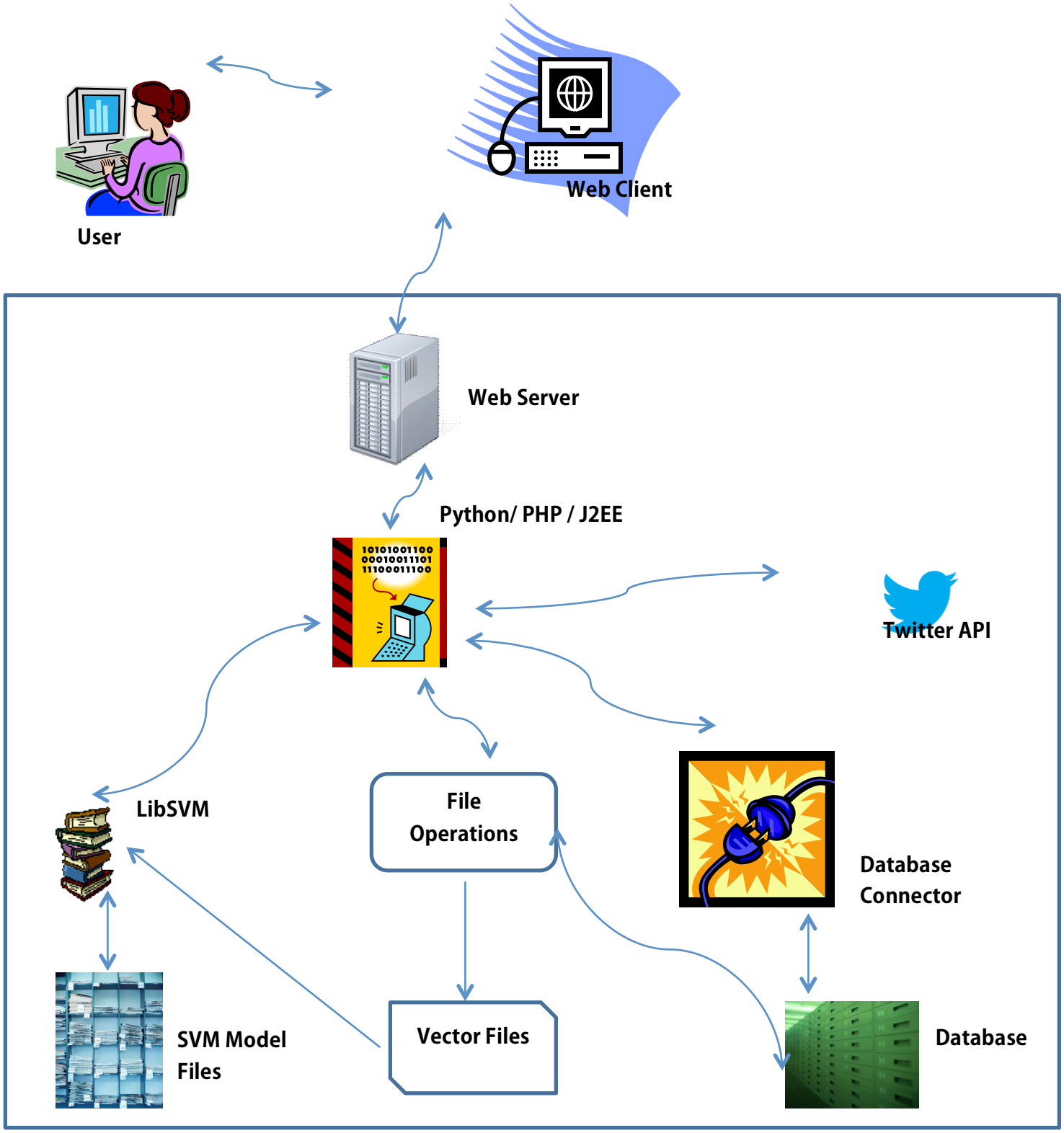
- **Insertion of processed data into the database**

- Since further processing and querying has to be done on the tokens, the processed tweets are stored in an Sqlite database rather than a text file.
- There are tables that store all unique words that occur in the tweets and also store the frequency of a word in a tweet.
- This DB is populated after the initial processing of the tweets.

- **Feature Extraction**
  - For now, the words in the tweets are used as the features.
  - We are using TFIDF to determine the weight of the words in the tweets and we use this value to classify the tweets.
  - For all the processed tokens in the DB, the TFIDF value is calculated and then stored into the database.
  - These values are later used as the input to the SVM Predict module.

- Creation of feature vectors for libSVM using TF-IDF (Term Frequency – Inverse Document Frequency).

- **Architecture:**



- **Coding**

  Used Python for developing algorithms specified above. Wrote separate scripts for each of the following tasks:

  Scripts are available at: https://github.com/rahmaniacc/twist.git

  - Data Collection
    - We require data for the purposes of training and testing. For this, we collected data from the public twitter stream which will be later used to test the classifier.
    - We also collected data from specific user handles to use as the training data.
    - We found a list of influential twitter users classified by topics and chose a set of users and collected tweets from those handles as train data for those categories.
  - Text Mining
  - Insertion of processed data into the database
  - Feature Extraction
  - Creation of input files for libSVM: Lib SVM accepts the input file in a particular format which is:

    <category> <index>:<feature> <index>:<feature> <index>:<feature>

    <category> <index>:<feature> <index>:<feature> <index>:<feature>

    <category> <index>:<feature> <index>:<feature> <index>:<feature>

    We wrote a python script that would read data from the database, fetch categories, words and their feature (tf idf) and generate the file in the format specified above.

  - Testing data for classification – The classifier was testing using the test data. This test data was separate from the training data. Testing was done by first using the training set to create a classifier model, using lib SVM train method. The output model was then used along with the test data set and svm predict method was called to classify the test data

- **Interface Design**

  The idea is to keep the user interface simple. Our final interface will provide an end user an ability to login via twitter and view a list of classified tweets on the page. There will be a small pie chart visualization showing the percentage split of user timeline tweets based on categories. For phase one we have created a webpage that takes user name and display home timeline tweets with an unlabeled category. This will be improved in the next phase where correct classification will be assigned to tweets.
  http://people.ischool.berkeley.edu/~sonali.sharma/ProjectWebsite/index.html

Next Steps:

1. Gathering more data sets adding more features
2. Using feedback to train the machine iteratively
3. Extending the algorithm to include other categories
4. Refining the user interface to enable user authentication and provide a mechanism for the users to re-classify incorrectly classified tweets

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,	First Deliverable

2012	
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 visualizations.
Dec 6, 2012	Final Deliverable

Percentage Contribution of Each Team Member

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

Literature Review

This has been attached as a separate document.