DOCUMENTATION

Summary of Method

1) Software Used: -

Weka API Toolkit
Eclipse Indigo

2) Features: -

Gini Index

3) Similarity/Distance Measures (if any) :--

N/A

3) Classifier (if you have used any standard classifier):-

Weka's Naive Bayes Classifier

4) References (if any) :-

http://stackoverflow.com/
http://en.wikipedia.org/wiki/Naive_Bayes_classifier
http://www.cs.waikato.ac.nz/ml/weka/
http://publib.boulder.ibm.com/infocenter/spssstat/v20r0m0/topic/com.ibm.spss.statistics.help/alg_naivebayes.htm
Machine Learning Tutorial, Technische Universität Darmstadt

Please describe the algorithms in details:

1) Pre-processing step:-

A case in the dataset is ignored if there is any missing value.

For example the tweet texts without any hashtags are not included in the hashtag classifier or considered for the prediction of the hashtag classifier.

2) Training Algorithm:

Input Format:-
Each training data is contained in the file Training.txt in the following format:
    <tweetid label tweettext>
where the 'label' is either 'Sports' or 'Politics' for the tweet identified by 'tweetid' and has text 'tweettext'

Output Format:-
Output.txt with the following format:
    <tweetid label>
where the 'label' is either 'Sports' or 'Politics' assigned to the tweet with identifier 'tweetid' by the algorithm.

D : Set of tuples

- Each tuple is an n-dimensional attribute vector
- X : $(x_1, x_2, x_3, ...x_n)$

Let there be m classes : $C_1, C_2 ... C_m$
X belongs to Class Ci iff:

$P(C_i/X) > P(C_j/X)$ for $1<=j<=m, j\neq i$

Probability distribution given as :

$$P\left(\frac{X}{Ci}\right) = \prod_{k=1}^{n} P\left(\frac{xk}{Ci}\right)$$

3) Validation and Parameter Tuning:

If P(Politics) > 0.8 the label = “Politics”
else label = “Sports”

The tweet texts satisfying the above condition were added to the training set to improve the prediction algorithm.

4) Testing Algorithm:

Steps :

1. Start
2. Obtain the tweet text and hashtag inside text, if any
3. Get the text classifier and hashtag classifier outputs and store them in result[] and hres[] respectively. result[0] gives probability of the text being in Politics category and result[1] the probability of the text being in Sports category. Similarly for hres[0] and hres[1].
4. If result[0] > 0.8 label = "Politics". Goto Step 12 else goto Step 5
5. If result[1] > 0.8 label = "Sports". Goto Step 12 else goto Step 6
6. If Length(hashtag) > 0 goto Step 7 else Step 11
7. If hres[0] > 0.6 label = "Politics". Goto Step 12 else goto Step 8
8. If hres[1] > 0.6 label = "Sports". Goto Step 12 else goto Step 9
9. If result[0] > 0.5 AND hres[0] > 0.5 label = "Politics". Goto Step 12 else goto Step 10
10. If result[1] > 0.5 AND hres[1] > 0.5 label = "Sports". Goto Step 12 else goto Step 11
11. Set label = "Sports"
12. Print out space separated tweet ID and label
13. Stop

Explanation of results on validation data:

Results of validation data are predicted on the basis of probabilities. If the probability P(Politics) > 0.7 it can generally be stated that the label for the tweet is Politics and similarly for Sports.

Why do you think your algorithm got the accuracy that it did on the validation data? Is scope for improvements?

The algorithm achieved an accuracy rating of 91.1587% for validation data. It achieved this by obtaining the text from training data set and using the probability of matching the validation text as a cutoff measure for the labelling of validation data set tweet text. Any hashtags inside the tweet text were also checked and classified, leading to better prediction of labels.

Ways in which algorithm could be improved:
1. Using external politics/sports related keywords to bias the classifier for the prediction
2. Using the prediction algorithm on particular phrases in the tweet text and checking it against the validation text.
3. Using better and more complex classifier if possible
