Sentiment Analysis: A Primary Implementation

Yichen PAN

2015 Big Data Lab Summer Internship

Abstract:

This report mainly talks about the ways to do sentiment analysis based on Twitter text. There are mainly three ways to conduct sentiment analysis. The paper is organized into four sections. The first section is about Data cleaning and data preprocessing. The second section talks about Lexicon-based approach. The third section is about supervised (Machine Learning) approach, where results of several machine learning algorithms are compared. The final section is about the combined approach.

1. Introduction:

Most current approaches for identifying the sentiment of text can be categorized into one of two main groups: supervised approaches, which uses a wide range of features and labeled data for training sentiment classifiers, and lexicon-based approaches, which make use of pre-built lexicons of words weighted with their sentiment orientations to determine the overall sentiment of a given text.

2. Data Cleaning and preprocessing

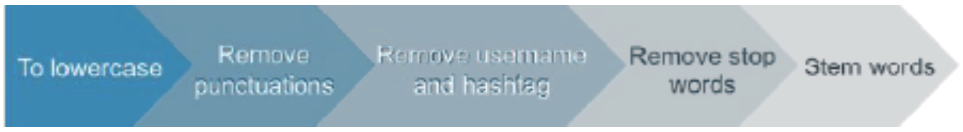
2.1 Raw data

Experiment data is tweet data concerning the topic “gain marriage” extracted by Twitter API and Node.JS.

The one sample raw data shows below:

2.2 Data preprocessing

In terms of the data processing, there are 5 steps:



1. To lowercase: Since text mining algorithms are case-sensitive, they may treat words like “little” and “lItTle” as different words. Therefore, characters need to be converted to lower case.

2. Remove punctuations: Punctuations are useless in text mining. All redundant punctuations in the raw tweets are removed while special symbols such as dash and hyphen are kept intact. URLs and links are also eliminated.

3. Remove usernames and hashtags: It is concerned with the privacy issues.

4. Eliminate the stop words: These stop words have relatively high frequency of occurrence and significantly increase the dimension of feature set, therefore they should be removed before feature extraction section.

5. Stem words: The process of stemming is to remove the prefix or postfix from a word, and extract it's stem. In the paper, SnowBall is used.

2.3 Experiment Data

We extracted 1000 tweets from the raw data pool and used them as the experiment dataset.

2.4 Manual codes

Related manual codes are named as *CleanData.java*、*CleanDataEXE.java*、*CSVUtil.java*

3. Lexicon-based approach:

Lexicon-based approach works on an assumption that the collective polarity of a document or sentence is the sum of polarities of the individual words or phrases.

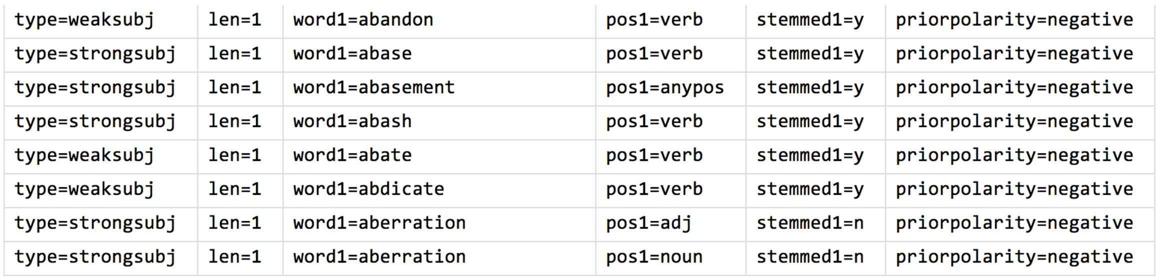
Sentimental Analysis can be phrase based in which the sentiment of a single phrase is taken in to consideration, sentence based in which the sentiment of the whole sentence is calculated or it can be document based in which an aggregated sentiment of the whole document is categorized as positive, negative or neutral.

Sentiment Analysis is generally carried out in three steps. First, the subject towards which the sentiment is directed is found then, the polarity of the sentiment is calculated and finally the degree of the polarity is assigned with the help of a sentiment score which denotes the intensity of the sentiment.

Popularity of lexicon-based approaches is rapidly increasing since they require no training data, and hence are more suited to a wider range of domains than supervised approaches.

3.1 Lexicon

The lexicon used here is lexicon of Theresa Wilson, Janyce Wiebe and Paul Hoffmann (2005).



type=weaksubj	len=1	word1=abandon	pos1=verb	stemmed1=y	priorpolarity=negative
type=strongsubj	len=1	word1=abase	pos1=verb	stemmed1=y	priorpolarity=negative
type=strongsubj	len=1	word1=abasement	pos1=anypos	stemmed1=y	priorpolarity=negative
type=strongsubj	len=1	word1=abash	pos1=verb	stemmed1=y	priorpolarity=negative
type=weaksubj	len=1	word1=abate	pos1=verb	stemmed1=y	priorpolarity=negative
type=weaksubj	len=1	word1=abdicate	pos1=verb	stemmed1=y	priorpolarity=negative
type=strongsubj	len=1	word1=aberration	pos1=adj	stemmed1=n	priorpolarity=negative
type=strongsubj	len=1	word1=aberration	pos1=noun	stemmed1=n	priorpolarity=negative

Table 1: Lexicon Details

Each line in the file contains one subjectivity clue.

Below is an example:

type=strongsubj len=1 word1=abuse pos1=verb stemmed1=y priorpolarity=negative

a. type - either strongsubj or weaksubj
A clue that is subjective in most contexts is considered strongly subjective (strongsubj), and those that may only have certain subjective usages are considered weakly subjective (weaksubj).

b. len - length of the clue in words
All clues in this file are single words.

c. word1 - token or stem of the clue

d. pos1 - part of speech of the clue, may be anypos (any part of speech)

e. stemmed1 - y (yes) or n (no)
Is the clue word1 stemmed? If stemmed1=y, this means that the clue should match all unstemmed variants of the word with the corresponding part of speech. For example, "abuse", above, will match "abuses" (verb), "abused" (verb), "abusing" (verb), but not "abuse" (noun) or "abuses" (noun).

f. priorpolarity - positive, negative, both, neutral
The prior polarity of the clue. Out of context, does the clue seem to evoke something positive or something negative.

3.2 Sentiment Calculation

The sentiment calculation is based on a set of heuristics built on the sentiment orientation of the words. Blind negation words are extracted from the sentence. The presence of the blind negation words indicates negative sentiment. If the sentence contains a blind negation word then other steps are skipped and sentiment is blindly assigned as negative. Next, sentiment words are extracted. The sentiment polarity can be changed due to negation words.

```
Algorithm 1: ALGO_SENTICAL
Input: Tweets, SentiWord_Dictionary
Output: Sentiment (positive, negative or neutral)
1) BEGIN
2)For each tweet Ti
3){
4)   SentiScore=0;  NegationCount = 0;
5)   For each word Wj in Ti that exists in Sentiword_Dictionary
6)   {
7)        If polarity[Wj] = blindnegation
8)        Return negative; }
9)        Else {
10)            If (polarity[Wj] = positive && strength[Wj] = Weaksubj)
11)            {
12)                 SentiScore = SentiScore + 0.5;
13)            }
14)            Else If (polarity[Wj] = negative && strength[Wj] = Strongsubj)
15)            {
16)                 SentiScore = SentiScore – 1;
17)            }
18)            Else If (polarity[Wj] = negative && strength[Wj] = Weaksubj)
19)            {
20)                  SentiScore = SentiScore – 0.5;
21)            }
22)       }
23)       If polarity[Wj] = negation
24)       {
25)            Sentiscore = Sentiscore * -1
26)       }
27) }
28) If (Sentiscore of Ti >0)
29){
30)       Sentiment = positive
31) }
32) ElseIf (SentiscoreofTI<0)
33){
34)       Sentiment = negative
35) }
36) Else {
37)       Sentiment = neutral
38) }
39) Return Sentiment
40)}
```

41)END

3.3 Limitations

Lexicon-based approaches have two main limitations.
Firstly, the number of words in the lexicons is finite, which may constitute a problem when extracting sentiment from very dynamic environments such as Twitter, where new terms, abbreviations and malformed words constantly emerge.
Secondly and more importantly, sentiment lexicons tend to assign a fixed sentiment orientation and score.

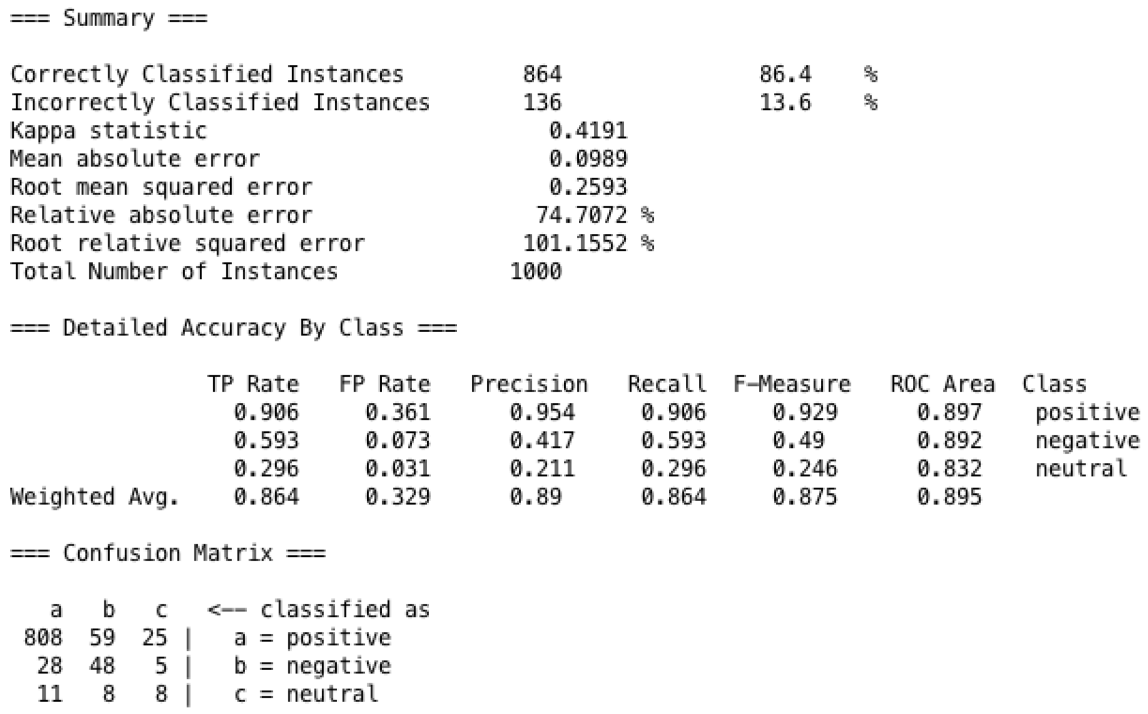
3.4 Manual codes

Related manual codes are named as *sentAnalysis_lexicon.R*

4. Supervised approach

We experimented with three standard algorithms: Naive Bayes classification, maximum entropy classification, and support vector machines. To implement these machine learning algorithms on our document data, we used the following standard bag-of-features framework. Let $\{f_1, . . . , f_m\}$ be a predefined set of m features that can appear in a document; examples include the word "still" or the bigram "really stinks". Let $n_i(d)$ be the number of times f_i occurs in document d. Then, each document d is represented by the document vector ~d := $(n_1(d), n_2(d), . . . , n_m(d))$.

4.1 Naïve Bayes



```
=== Summary ===

Correctly Classified Instances         864               86.4    %
Incorrectly Classified Instances       136               13.6    %
Kappa statistic                          0.4191
Mean absolute error                      0.0989
Root mean squared error                  0.2593
Relative absolute error                 74.7072 %
Root relative squared error            101.1552 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.906     0.361      0.954     0.906     0.929      0.897    positive
                 0.593     0.073      0.417     0.593     0.49       0.892    negative
                 0.296     0.031      0.211     0.296     0.246      0.832    neutral
Weighted Avg.    0.864     0.329      0.89      0.864     0.875      0.895

=== Confusion Matrix ===

   a   b   c   <-- classified as
 808  59  25 |   a = positive
  28  48   5 |   b = negative
  11   8   8 |   c = neutral
```

Table : results of Naïve Bayes classifier

4.2 Multinomial Naïve Bayes

With a multinomial event model, samples (feature vectors) represent the frequencies with which certain events have been generated by amultinomial $(p_1, \dots, p_n)$ where p_i is the probability that event i occurs (or K such multinomials in the multiclass case). A feature vector $\mathbf{x} = (x_1, \dots, x_n)$ is then a histogram, with x_i counting the number of times event i was observed in a particular instance. This is the event model typically used for document classification, with events representing the occurrence of a word in a single document (see bag of words assumption).
The likelihood of observing a histogram x is given by

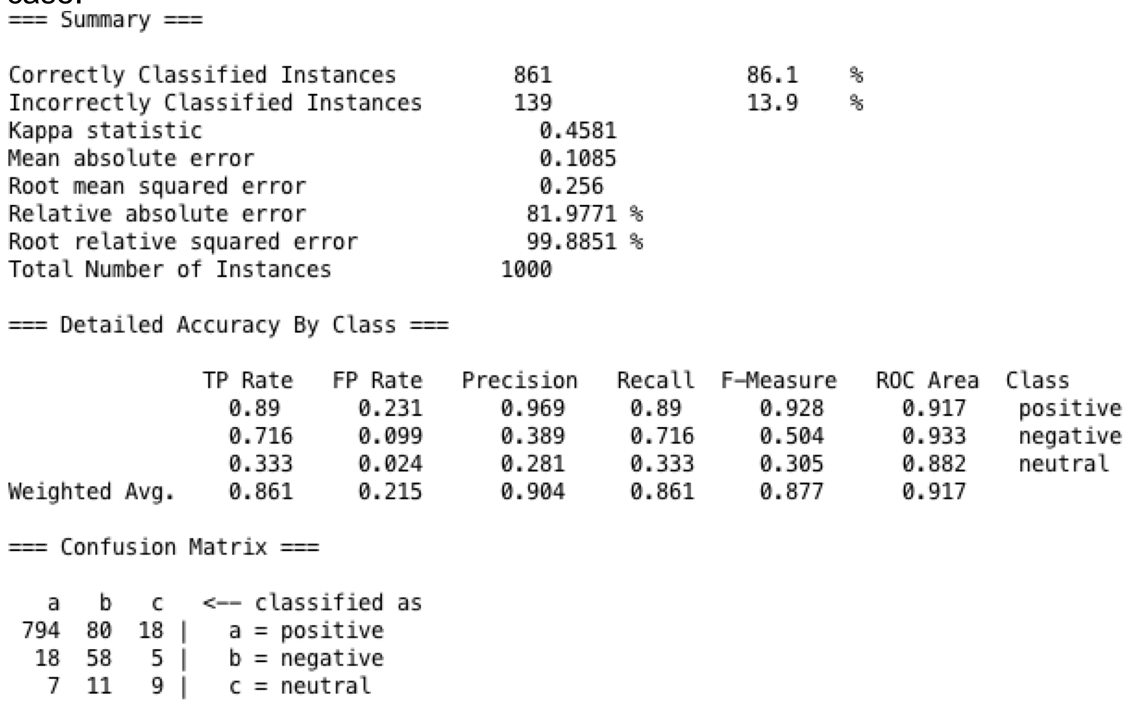
$$p(\mathbf{x}|C_k) = \frac{(\sum_i x_i)!}{\prod_i x_i!} \prod_i {p_{ki}}^{x_i}$$

The multinomial naive Bayes classifier becomes a linear classifier when expressed in log-space:[2]

$$\begin{aligned}\log p(C_k|\mathbf{x}) &\propto \log\left(p(C_k)\prod_{i=1}^{n} {p_{ki}}^{x_i}\right)\\ &= \log p(C_k) + \sum_{i=1}^{n} x_i \cdot \log p_{ki}\\ &= b + \mathbf{w}_k^\top \mathbf{x}\end{aligned}$$

where $b = \log p(C_k)$ and $w_{ki} = \log p_{ki}$.

If a given class and feature value never occur together in the training data, then the frequency-based probability estimate will be zero. This is problematic because it will wipe out all information in the other probabilities when they are multiplied. Therefore, it is often desirable to incorporate a small-sample correction, called pseudocount, in all probability estimates such that no probability is ever set to be exactly zero. This way of regularizing naive Bayes is called Laplace smoothing when the pseudocount is one, and Lidstone smoothing in the general case.



```
=== Summary ===

Correctly Classified Instances         861               86.1    %
Incorrectly Classified Instances       139               13.9    %
Kappa statistic                          0.4581
Mean absolute error                      0.1085
Root mean squared error                  0.256
Relative absolute error                 81.9771 %
Root relative squared error             99.8851 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.89      0.231      0.969     0.89      0.928      0.917    positive
                 0.716     0.099      0.389     0.716     0.504      0.933    negative
                 0.333     0.024      0.281     0.333     0.305      0.882    neutral
Weighted Avg.    0.861     0.215      0.904     0.861     0.877      0.917

=== Confusion Matrix ===

   a   b   c   <-- classified as
 794  80  18 |   a = positive
  18  58   5 |   b = negative
   7  11   9 |   c = neutral
```

Table : results of multinomial Naïve Bayes classifier

4.3 Decision tree J48

Decision tree J48 is the implementation of algorithm C4.5 developed by the WEKA project team.

C4.5 builds decision trees from a set of training data in the same way as ID3, using the concept of information entropy. The training data is a set $S = s_1, s_2, \cdots$ of already classified samples. Each sample s_i consists of a p-dimensional vector $(x_{1,i}, x_{2,i}, ..., x_{p,i})$, where the x_j represent attributes or features of the sample, as well as the class in which s_i falls.

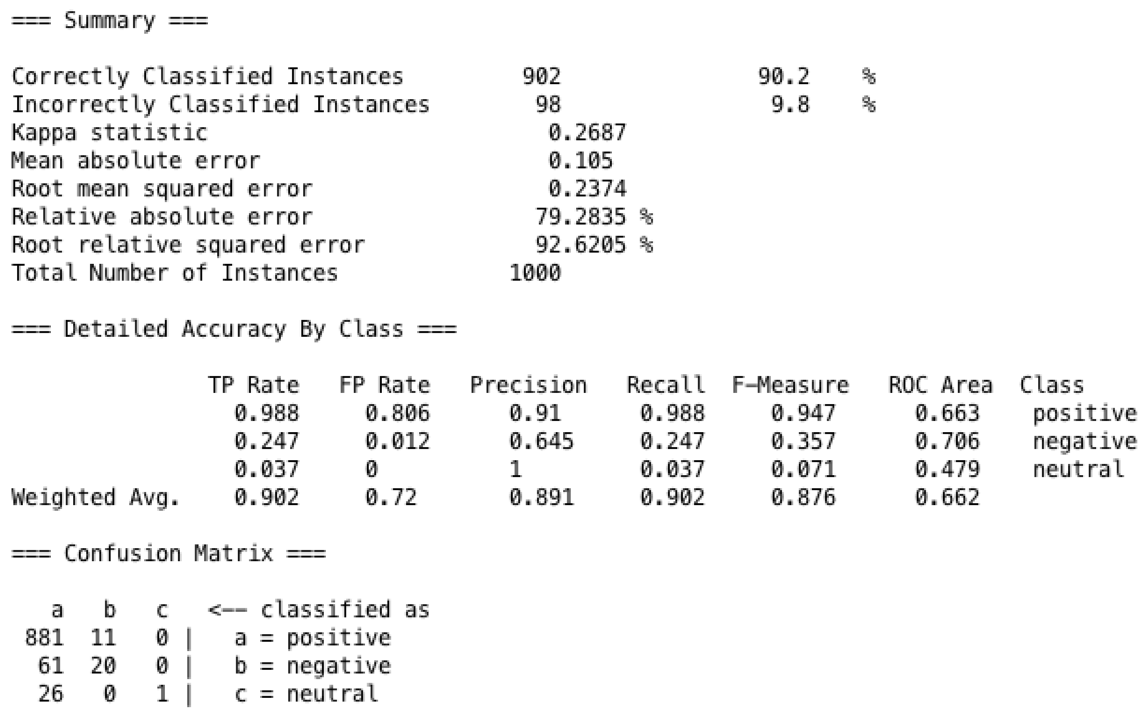
At each node of the tree, C4.5 chooses the attribute of the data that most effectively splits its set of samples into subsets enriched in one class or the other. The splitting criterion is the normalized information gain (difference in entropy). The attribute with the highest normalized information gain is chosen to make the decision. The C4.5 algorithm then recurs on the smaller sublists.

This algorithm has a few base cases:

- All the samples in the list belong to the same class. When this happens, it simply creates a leaf node for the decision tree saying to choose that class.
- None of the features provide any information gain. In this case, C4.5 creates a decision node higher up the tree using the expected value of the class.
- Instance of previously-unseen class encountered. Again, C4.5 creates a decision node higher up the tree using the expected value.

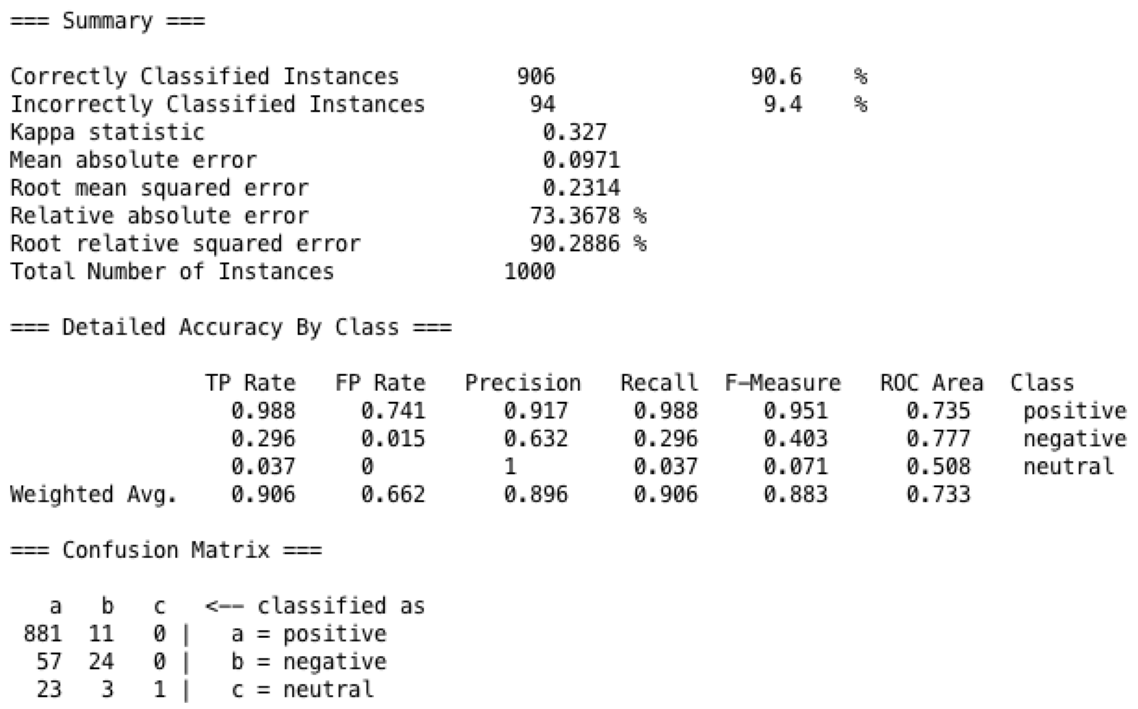
When executing WEKA for classification, one option of J48 classifier is confidence factor, which represents the confidence factor used for pruning (smaller values incur more pruning). It is the parameter that can affect the effectiveness of post-pruning. In the WEKA J48 classifier, lowering the confidence factor decreases the amount of post-pruning.

Following are the results of classification with confidence factor ranging from 0.1 to 1.0(results after confidence factor equals 0.6 are all the same as that with confidence factor equaling 0.6)



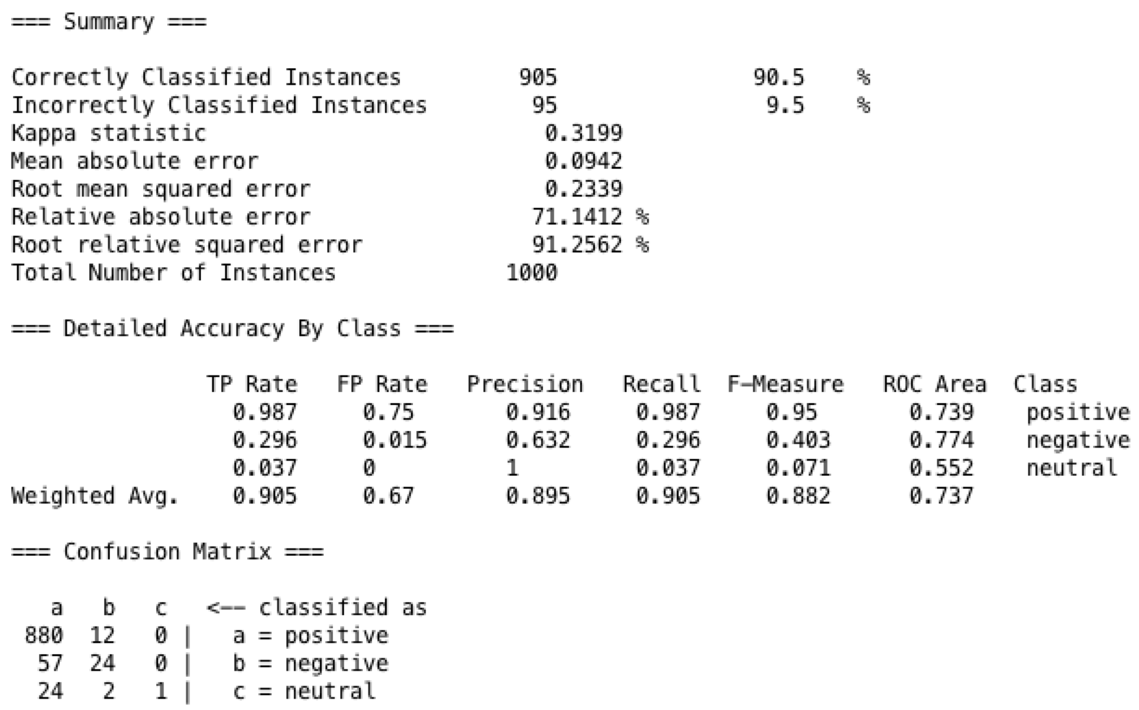
```
=== Summary ===

Correctly Classified Instances         902               90.2    %
Incorrectly Classified Instances        98                9.8    %
Kappa statistic                          0.2687
Mean absolute error                      0.105
Root mean squared error                  0.2374
Relative absolute error                 79.2835 %
Root relative squared error             92.6205 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.988     0.806      0.91      0.988     0.947      0.663    positive
                 0.247     0.012      0.645     0.247     0.357      0.706    negative
                 0.037     0          1         0.037     0.071      0.479    neutral
Weighted Avg.    0.902     0.72       0.891     0.902     0.876      0.662

=== Confusion Matrix ===

   a   b   c   <-- classified as
 881  11   0 |   a = positive
  61  20   0 |   b = negative
  26   0   1 |   c = neutral
```

Table : results with confidence factor = 0.1



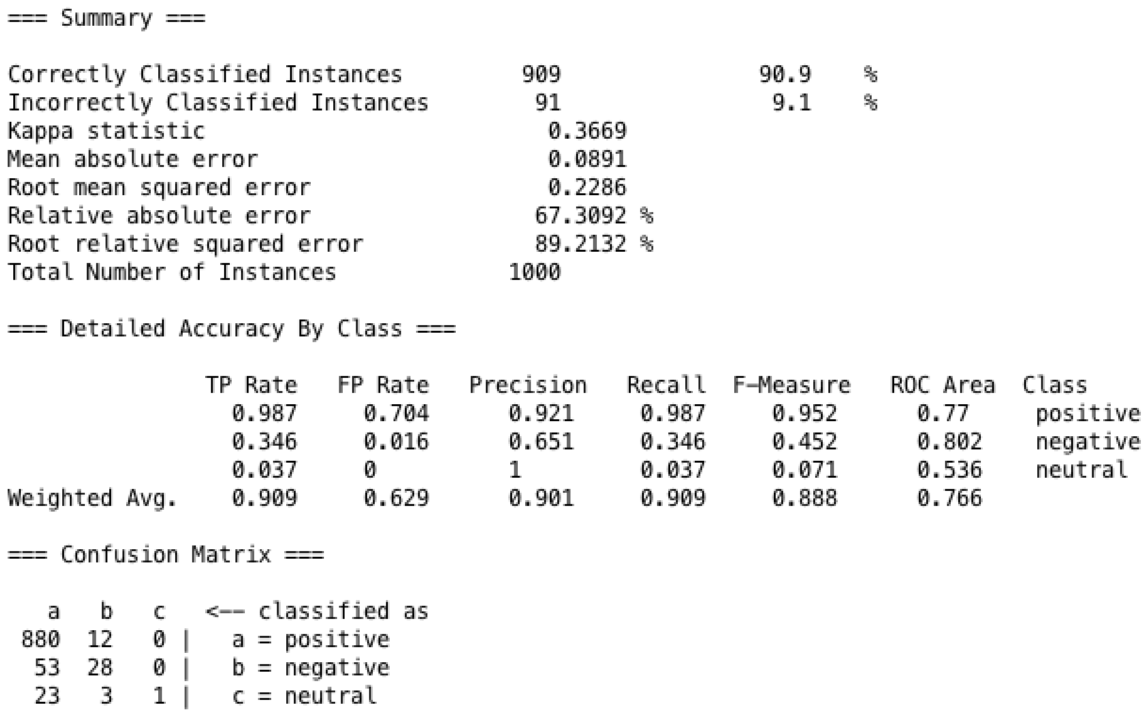
```
=== Summary ===

Correctly Classified Instances         906               90.6    %
Incorrectly Classified Instances        94                9.4    %
Kappa statistic                          0.327
Mean absolute error                      0.0971
Root mean squared error                  0.2314
Relative absolute error                 73.3678 %
Root relative squared error             90.2886 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.988     0.741      0.917     0.988     0.951      0.735    positive
                 0.296     0.015      0.632     0.296     0.403      0.777    negative
                 0.037     0          1         0.037     0.071      0.508    neutral
Weighted Avg.    0.906     0.662      0.896     0.906     0.883      0.733

=== Confusion Matrix ===

   a   b   c   <-- classified as
 881  11   0 |   a = positive
  57  24   0 |   b = negative
  23   3   1 |   c = neutral
```

Table : results with confidence factor = 0.2



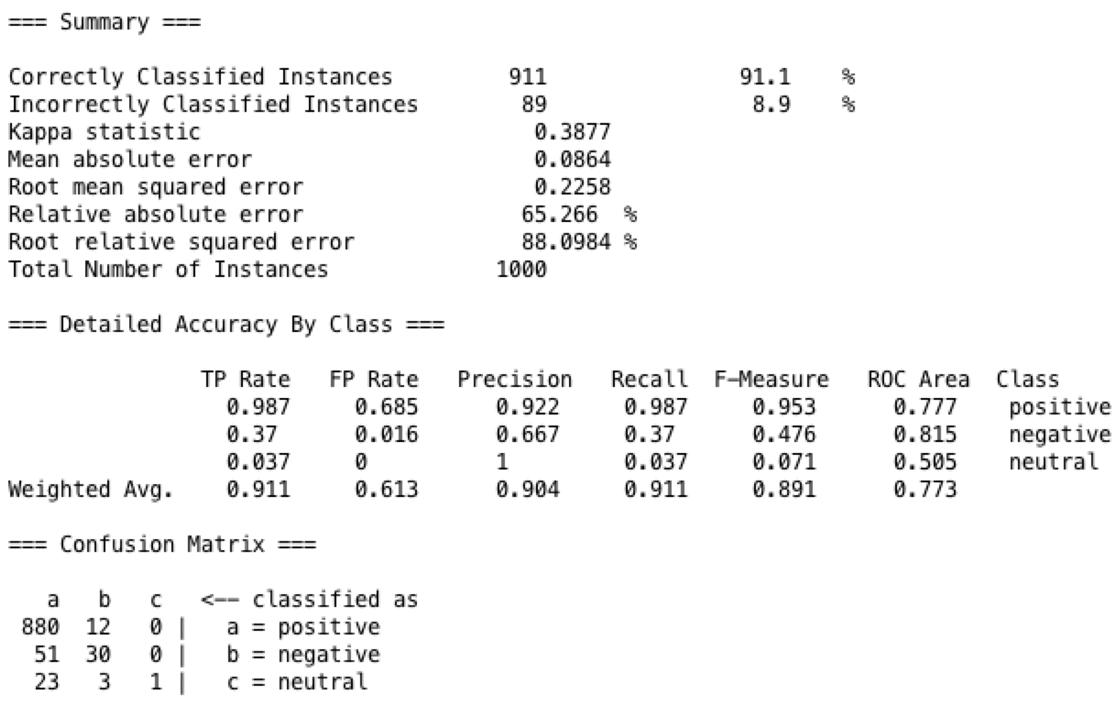
```
=== Summary ===

Correctly Classified Instances         905               90.5    %
Incorrectly Classified Instances        95                9.5    %
Kappa statistic                          0.3199
Mean absolute error                      0.0942
Root mean squared error                  0.2339
Relative absolute error                 71.1412 %
Root relative squared error             91.2562 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.987     0.75       0.916     0.987     0.95       0.739    positive
                 0.296     0.015      0.632     0.296     0.403      0.774    negative
                 0.037     0          1         0.037     0.071      0.552    neutral
Weighted Avg.    0.905     0.67       0.895     0.905     0.882      0.737

=== Confusion Matrix ===

   a   b   c   <-- classified as
 880  12   0 |   a = positive
  57  24   0 |   b = negative
  24   2   1 |   c = neutral
```

Table : results with confidence factor = 0.3



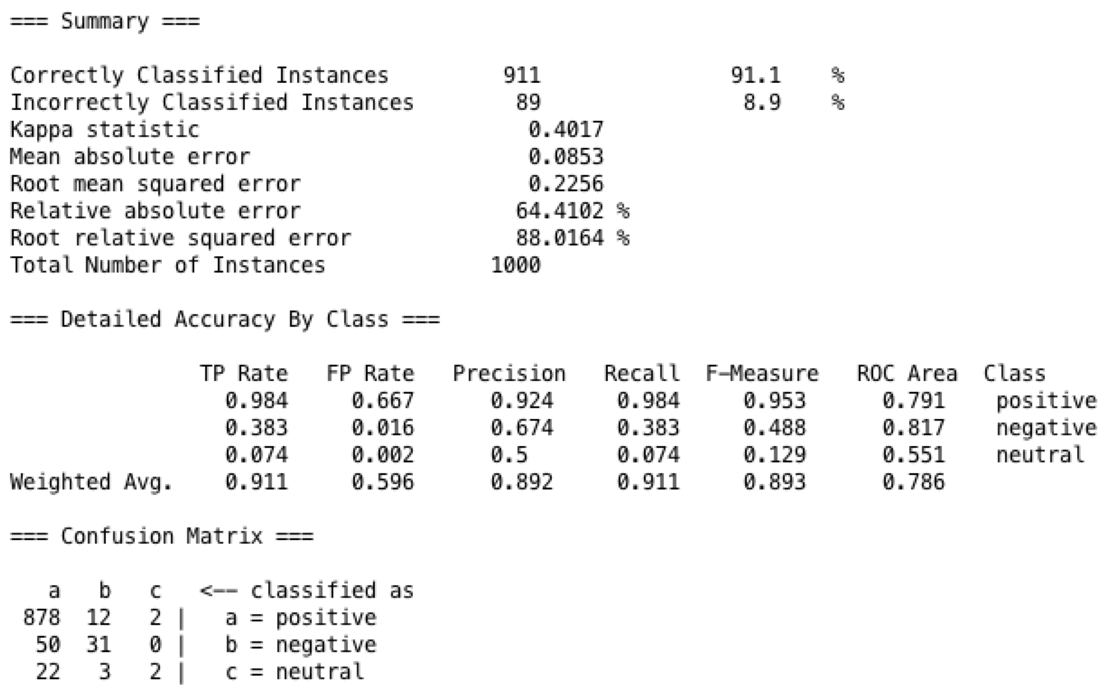
```
=== Summary ===

Correctly Classified Instances         909               90.9    %
Incorrectly Classified Instances        91                9.1    %
Kappa statistic                          0.3669
Mean absolute error                      0.0891
Root mean squared error                  0.2286
Relative absolute error                 67.3092 %
Root relative squared error             89.2132 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.987     0.704      0.921     0.987     0.952      0.77     positive
                 0.346     0.016      0.651     0.346     0.452      0.802    negative
                 0.037     0          1         0.037     0.071      0.536    neutral
Weighted Avg.    0.909     0.629      0.901     0.909     0.888      0.766

=== Confusion Matrix ===

   a   b   c   <-- classified as
 880  12   0 |   a = positive
  53  28   0 |   b = negative
  23   3   1 |   c = neutral
```

Table : results with confidence factor = 0.4



```
=== Summary ===

Correctly Classified Instances         911               91.1    %
Incorrectly Classified Instances        89                8.9    %
Kappa statistic                          0.3877
Mean absolute error                      0.0864
Root mean squared error                  0.2258
Relative absolute error                 65.266  %
Root relative squared error             88.0984 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.987     0.685      0.922     0.987     0.953      0.777    positive
                 0.37      0.016      0.667     0.37      0.476      0.815    negative
                 0.037     0          1         0.037     0.071      0.505    neutral
Weighted Avg.    0.911     0.613      0.904     0.911     0.891      0.773

=== Confusion Matrix ===

   a   b   c   <-- classified as
 880  12   0 |   a = positive
  51  30   0 |   b = negative
  23   3   1 |   c = neutral
```

Table : results with confidence factor = 0.5



```
=== Summary ===

Correctly Classified Instances         911               91.1    %
Incorrectly Classified Instances        89                8.9    %
Kappa statistic                          0.4017
Mean absolute error                      0.0853
Root mean squared error                  0.2256
Relative absolute error                 64.4102 %
Root relative squared error             88.0164 %
Total Number of Instances             1000

=== Detailed Accuracy By Class ===

               TP Rate   FP Rate   Precision   Recall  F-Measure   ROC Area  Class
                 0.984     0.667      0.924     0.984     0.953      0.791    positive
                 0.383     0.016      0.674     0.383     0.488      0.817    negative
                 0.074     0.002      0.5       0.074     0.129      0.551    neutral
Weighted Avg.    0.911     0.596      0.892     0.911     0.893      0.786

=== Confusion Matrix ===

   a   b   c   <-- classified as
 878  12   2 |   a = positive
  50  31   0 |   b = negative
  22   3   2 |   c = neutral
```

Table : results with confidence factor = 0.6

4.4 Manual codes

Related manual codes are named as *sentAnalysis.R*

5. Lexicon-based Ensemble Sentiment Classification

Lexicon building is thus a hard task in terms of avoiding numerical and statistical artifacts occurring due to certain characteristics of data sampling methods. A way of overcoming this problem was to use supervised learning methods, which would learn of provided data sets to differentiate the input space well enough to infer a text's sentiment. Such methods often secure results without the need to understand the nature of input data, yet remain more intractable for ever growing data sets. Supervised methods advantage lies in the fact that they are provided as ready-to-use functions in many machine learning libraries, while constructing lexicon-based methods requires more work to setup.

We have combined the best of both worlds - a mixture of a frequency-related measure used to build lexicons and an ensemble method to infer from lexicon based results. Such a combination provided efficiency on par with supervised learning while providing a major speed improvement.

5.1 Ensemble Learning

Ensemble learning technique combines multiple weak learners in an attempt to produce one strong learner.

5.1.1 Stacking

Stacking (sometimes called stacked generalization) involves training a learning algorithm to combine the predictions of several other learning algorithms. First, all of the other algorithms are trained using the available data, then a combiner algorithm is trained to make a final prediction using all the predictions of the other algorithms as additional inputs. If an arbitrary combiner algorithm is used, then stacking can theoretically represent any of the ensemble techniques described in this article, although in practice, a single-layer logistic regression model is often used as the combiner.

5.2 Proposed Method

5.2.1 Step 1: Prepare Lexicon

- SM(simplest method):
  - 2-word list consisting of strong sentiment words - good/bad, and we call the method based on it
- SL(simple lists):
  - based on a recently presented lexicon [13] based on IMDB review data
- PF
  - positive opinions are more frequently expressed with present and future tenses, whereas negative with past tenses.
- PF+:
  - sum of PF and SL
- BL(Bing Liu Opinion Lexicon)
- HLTEMNLP05_lexicon

5.2.2 Step 2: Lexicon Based Ensemble Classification

All of the described lexicons were used for assigning sentiment polarity based on Bag-of-Words (BoW) model. This model takes individual words w in a document as features, assuming their conditional independence. For each word w in review text t = {w1 , w2 , ..., wk } this unigram model predicts document sentiment based on occurrence of words from lexicons. Each word w that occurs in a lexicon l has a numeric value, i.e., ”1” for positive word and ”-1” for negative. For words outside lexicon value ”0” is assigned.

Let:

pos(l, t) = # of positive words from l that occur in t

neg(l, t) = # of negative words from l that occur in t

sum(l, t) = pos(l, t) - neg(l, t)

Then the sentiment sl(t) of a review with text t under a lexicon l is assigned using the following formula:

$$s_l(t) = \begin{cases} 1 & if\ sum(l,t) > 0 \\ -1 & if\ sum(l,t) < 0 \\ 0 & otherwise \end{cases}$$

Thus we obtain a sentiment polarity matrix for an input set of documents D = {d1,...,dn} and the defined set of lexicons L = {l1,...,lm}:

$$\mathcal{S}(\mathcal{L},\mathcal{D}) = \begin{pmatrix} s_{l_1}(d_1) & s_{l_1}(d_2) & \cdots & s_{l_1}(d_n) \\ s_{l_2}(d_1) & s_{l_2}(d_2) & \cdots & s_{l_2}(d_n) \\ \vdots & \vdots & \ddots & \vdots \\ s_{l_m}(d_1) & s_{l_m}(d_2) & \cdots & s_{l_m}(d_n) \end{pmatrix}$$

5.2.3 Step 3: Train Classifier

In the final step of our method we train a strong classifier, such as the C4.5 decision tree method, on the sentiment polarity matrix S with respective document texts and use such trained classifier to predict the sentiment of new documents.

5.3 Manual codes

Related manual codes are named as *sentAnalysis_lexicon+super.R*