University of Engineering & Technology, Lahore

Department of Electrical Engineering

Artificial Intelligence Lab 1

1: Text Classification

Submitted by:

Hadia Hameed
2013-EE-2

Maham Nasir Khan
2013-EE-6

Submitted to:
Sir Muhammad Wasim

February 28, 2017

1 Sentiment Analysis:

1.1 Specifications:

The zip file of that data set contains pos and neg folders for positive and negative sentiments each containing 1000 documents. Your task is to split the dataset into 980 and 20 documents for both negative and positive class. 980 documents will be used for training and 20 documents for testing. The accuracy of 20+20=40 (positive/negative) documents showed be shown to the user. You have to show how many percent accuracy is being achieved.

1.2 Pseudo-Code:

Algorithm for Training:

```
→ for i=1:980
    → file = read training file "cv_train(i)".
    → unique_words = scan the text to determine unique vocabulary.
    → freq = determine the frequency of each unique word.
    → selected_words = remove monosyllables and select long significant
    words.
    → I = determine the indices of the most frequent words.
    → bag_of_words(i,1) = store frequent words for text file"i".
→ end
→ Repeat this for both positive and negative training examples.
```

Algorithm for Testing:

```
→ for i=1:20
    → Repeat above steps [2-6] for the test files.
    → Compare the test words with the training bag of words.
    → If intersection>0 give it class pos/neg and increment true_positive or
    true_negative(depending on the bag of words being used for comparison)
→ end
→ Repeat this for both positive and negative test instances.
→ Calculate accuracy = $\frac{100 * (true\_positive + true\_negative)}{40}$.
```

1.3 MATLAB Code: Sentiment Analysis:

```
clc;
close all;
clear all;

train_files = 1:980;
test_files = 1:20;
threshold = 4;

%TRAINING FOR POSITIVE CLASS INSTANCES
pos_bag_of_words = cell(length(train_files),1);
pos_bag_of_words(:) = {''};
for i=train_files

    %OPENING AND SCANNING TRAINING TEXT FILE WITH 'POS' CLASS
    file = sprintf('cv_train_pos (%d).txt',i);
    selected_words = sentiment_analysis(file,threshold);

    %LONG WORDS CHOSEN
    [~,I] = max(cell2mat(selected_words(:,2)));

    %BAG OF WORDS FOR POSITIVE CLASS
    pos_bag_of_words(i,1) = selected_words(I);

end

%TRAINING FOR NEGATIVE CLASS INSTANCES
neg_bag_of_words = cell(length(train_files),1);
neg_bag_of_words(:) = {''};
for j=train_files
    %OPENING AND SCANNING TRAINING TEXT FILES WITH 'POS' CLASS
    file = sprintf('cv_train_neg (%d).txt',j);
    selected_words = sentiment_analysis(file,threshold);

    %LONG WORDS CHOSEN
    [~,I] = max(cell2mat(selected_words(:,2)));

    %BAG OF WORDS FOR POSITIVE CLASS
    neg_bag_of_words(j,1) = selected_words(I);

end

%REMOVE DUPLICATE WORDS FROM THE BAG OF WORDS
pos_bag_of_words = unique(pos_bag_of_words);
neg_bag_of_words = unique(neg_bag_of_words);
```

```
true_positive=0;true_negative=0;

%TESTING POSITIVE TEST FILES AND INCREMENTING TRUE_POSITIVE
for k=test_files
    %OPENING AND SCANNING TEST FILE WITH 'POS' CLASS
    file = sprintf('cv_test_pos (%d).txt',k);

    selected_words = sentiment_analysis(file,threshold);

    %LONG WORDS CHOSEN
    [~,I] = max(cell2mat(selected_words(:,2)));

    %COMPARING COMMON WORDS IN TRAIN AND TEST BAGS
    common_words = intersect(pos_bag_of_words,selected_words(I));
    common_words = length(common_words);
    if(common_words>0)
        true_positive = true_positive+1;
    end
end

%TESTING NEGATIVE TEST FILES AND INCREMENTING TRUE_NEGATIVE
for k=test_files
    %OPENING AND SCANNING TEST FILE WITH 'POS' CLASS
    file = sprintf('cv_test_neg (%d).txt',k);

    selected_words = sentiment_analysis(file,threshold);

    %LONG WORDS CHOSEN
    [~,I] = max(cell2mat(selected_words(:,2)));

    %COMPARING COMMON WORDS IN TRAIN AND TEST BAGS
    common_words = intersect(neg_bag_of_words,selected_words(I));
    common_words = length(common_words);
    if(common_words>0)
        true_negative = true_negative+1;
    end
end
accuracy = 100*(true_positive+true_negative)/40
```

```
%****************SENTIMENT ANALYSIS FUNCTION****************

function [selected_words] = sentiment_analysis(file,threshold)
    %OPENING AND SCANNING TEXT FILES
    fid = fopen(file);
    s=textscan(fid,'%s');
    fclose(fid);

    %DETERMINING UNIQUE WORDS AND THEIR FREQUENCIES
    stringified=s{:};
    [unique_words,unique_words_i,unique_words_j]=unique(stringified);
    freq=hist(unique_words_j,(1:numel(unique_words_i))')';
    unique_vocab = [unique_words num2cell(freq)];

    %REMOVE MONOSYLLABLES AND SHORT WORDS e.g 'a','the','is'
    word_length = cellfun('length',unique_vocab);
    long_words = word_length(:,1) >= threshold;

    %WORDS LONGER THAN 'threshold' CHOSEN
    selected_words = unique_vocab(long_words,:);

end
```

Accuracy= 77.5 %
Elapsed Time= 10 sec

2 Question Classification:

2.1 Specifications:

Each question in the dataset has its own class associated with it at the start of the question. Each individual question shall be considered a separate document. You have to show the accuracy of the algorithm based on the number of instances correctly classified. You will be using Naive Bayes for classification. No library should be used.

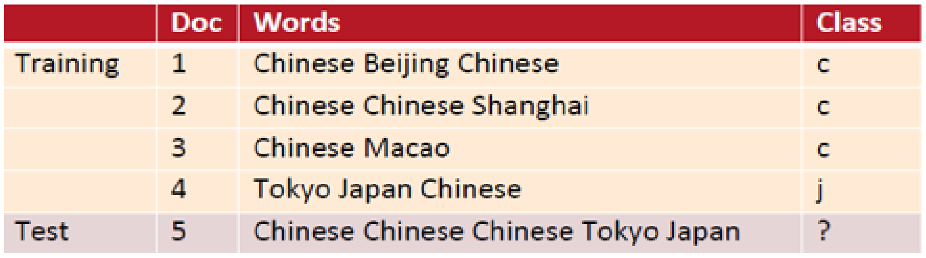
2.2 Pseudo-Code:

Algorithm for Naive-Bayes Classification:

```
→ train_data = load training data with train_instances number of rows; test_data = load test data with test_instances number of rows.
→ V = count unique vocabulary in the train_data.
→ classes = determine unique classes in the data; class_number = determine number of unique classes in the data.
→ for i=1:test_instances
    →words_in_a_row = count words in the test instance(i).
        → for j=1:class_number
            →b = number of times class(j) occurs in the train_data.
            →rows = train_data rows in which class(j) occurs.
            →class_word_count = total words in all the rows in which class(i) occurs.
                →for k=1:words_in_a_row.
                    →count = number of times word(k) occurs in the rows in which class(j) exists.
                    →Calculate $conditional\_probability = \frac{(count+1)}{(class\_word\_count+V)}$
                →end
            →posterior_prb = $\frac{b}{train\_instances}$prod(conditional_prbs(j,:)
        →end
    →I = maximum of posterior_probabilities
    →predicted_class = classes(I)
→end
```

2.3 Code Verification:

The code was first verified using the following example:

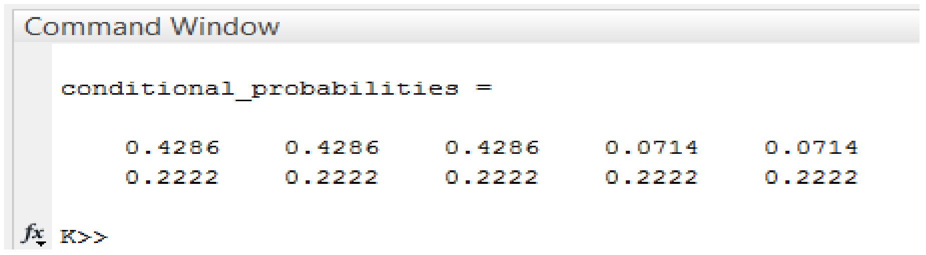


	Doc	Words	Class
Training	1	Chinese Beijing Chinese	c
	2	Chinese Chinese Shanghai	c
	3	Chinese Macao	c
	4	Tokyo Japan Chinese	j
Test	5	Chinese Chinese Chinese Tokyo Japan	?

Figure 1: Example for verification

Results:

Columns represent words in the test instance(i.e. Chinese, Chinese, Chinese, Tokyo, Japan and the rows represent classes i.e. c, j.



```
conditional_probabilities =

    0.4286    0.4286    0.4286    0.0714    0.0714
    0.2222    0.2222    0.2222    0.2222    0.2222

K>>
```

Figure 2: Results for the highlighted test instance

2.4 MATLAB Code: Text Classification using Naive-Bayes:

```
clc;
clear all;
close all;

[~,train_data,~] = xlsread('xl_data.xlsx',1);
[~,test_data,~] = xlsread('Testing_data.xlsx',1);

[train_instances,n] = size(train_data);
[test_instances,o] = size(test_data);

%COUNT OF UNIQUE VOCABULARY IN THE ENTIRE TRAINING DATA. REMOVE ANY EMPTY WHITE
%SPACE PRESENT IN THE VOCABULARY
V = unique(train_data(:,2:end));
[blanks] = find(strcmp('',V));V(blanks,:) = [];
[V,~] = size(V); %NUMBER OF UNIQUE WORDS PRESENT

%DETERMINING UNIQUE CLASSES
x = train_data(:,1);
[x1,x2,x2]=unique(x);
x2=accumarray(x2,1);
classes = [x1,num2cell(x2)];%UNIQUE CLASSES
[classes_number,~] = size(classes);%NUMBER OF UNIQUE CLASSES

%FOR STORING THE PREDICTED CLASSES
final_table = cell(test_instances,1);
final_table(:) = {''};

%FOR STORING POSTERIOR PROBABILITIES
post_prob = zeros(1,classes_number);

for p=1:test_instances
    %WORDS IN AN INSTANCE E.G {'How','are','you'}
    words_in_a_row = sum(~strcmp(test_data(p,2:end),''));
    conditional_probabilities = zeros(classes_number,words_in_a_row);
    for q=1:classes_number %CONDITIONAL PROBABILITIES FOR EACH WORD:CLASS

        %RUNNING FOR e.g. CLASS 'C'
        class_word_count = 0;

        %TOTAL NUMBER OF TIMES CLASS 'C' OCCURS
        b = cell2mat(classes(q,2));

        %ROWS IN WHICH CLASS 'C' OCCURS
        [rows] = find(strcmp(classes(q),train_data));
        [g,~] = size(rows);

        %TOTAL WORDS IN ALL THE ROWS IN WHICH THE CLASS 'C' OCCURS
        for i=1:g
          class_word_count = sum(~strcmp(train_data(rows(i),2:end),''))+class_word_count;
        end

        for r=1:words_in_a_row
            %RUNNING FOR e.g 'How'
            count = 0;

            test_word = test_data(p,(r+1));
            if (~isempty(rows))
                for i=1:g
                    for j=1:n
                        if(strcmp(test_word,train_data(rows(i),j)))
                            count = count + 1;
                        end
                    end
                end
            else
                count=0;
            end
            conditional_probabilities(q,r) = (count+1)/(class_word_count+V);
        end
        post_prob(1,q) = (b/train_instances)*prod(conditional_probabilities(q,:));
    end
 [~,I] = max(post_prob);
 final_table(p,1) =  classes(I,1);
end
[rows] = find(strcmp(final_table(:,1),test_data(:,1)));
[tp,~] = size(rows);
accuracy = 100*tp/test_instances
```

Accuracy= 57.2 %
Elapsed Time= 56 min