



**FINAL REPORT  
MASTER IN INFORMATION AND  
COMMUNICATION TECHNOLOGY**

By  
**Bùi Đình Dương**  
Intake 2012-2014

Project:  
**Text Classification using Decision Tree and Maximum  
Entropy**

Supervisors: Dr. TRẦN THẾ TRUNG  
FPT Research Institute

**Hanoi, September 2013**

## **Acknowledgement**

I would like to thank Dr. Tran The Trung for his guidance and supervision throughout my research.

I would also like to thank members in FPT Research Institute for their support.

## Contents

<b>Acknowledgement.....</b>	<b>i</b>
<b>List of abbreviations .....</b>	<b>iv</b>
<b>List of Tables .....</b>	<b>v</b>
<b>List of Figures.....</b>	<b>vi</b>
<b>Chapter 1: Introduction.....</b>	<b>1</b>
<b>Chapter 2: Text classification .....</b>	<b>2</b>
<b>1. Formal Definition .....</b>	<b>2</b>
<b>2. General Model .....</b>	<b>3</b>
2.1. Training .....	4
2.2. Classifying.....	5
2.3. Text Preprocessing.....	5
2.4. Text Representation Model .....	6
2.4.1. Vector Space Model .....	6
2.4.2. Word Weighting.....	7
2.5. Classifier Evaluation .....	9
2.5.1. Macro-Averaging .....	10
2.5.2. Micro-Averaging.....	10
<b>3. Text Classification Methods .....</b>	<b>11</b>
3.1. Decision Tree.....	11
3.2. Maximum Entropy.....	20
<b>Chapter 3: Experiment .....</b>	<b>25</b>
<b>1. Introduction to Weka .....</b>	<b>25</b>
<b>2. Implementation text classification .....</b>	<b>25</b>
2.1. Decision Tree.....	25
<b>Chapter 4: Conclusion .....</b>	<b>32</b>

<b>References .....</b>	<b>33</b>
-------------------------	-----------

## List of abbreviations

CART	Classification And Regression Tree
CHAID	CHi-squared Automatic Interaction Detector
ID3:	Iterative Dichotomiser 3
NLP	Natural Language Processing
Q&A system	Question and Answer system
SVM	Support Vector Machine
TC	Text classification

## List of Tables

Table 01: Vector Space Model example .....	7
Table 02: Word Weighting example .....	8
Table 03: Table classifier evaluation .....	9

## List of Figures

Fig.1: Architecture of Q&A System.....	1
Fig.2: Example of text classification for web content categories.....	2
Fig.3: Training Phase Model .....	4
Fig.4: Training phase in detail .....	4
Fig.5: Classifying phase .....	5
Fig.6: Detail of classifying phase.....	5
Fig.7: Example of stop-word in English.....	6
Fig.8: Represent text vectors in 2D space .....	7
Fig.9: Example of Decision Tree .....	11
Fig.10: Pseudo code of ID3 by The MIT Press1999 .....	12
Fig.11: Example dataset in Decision Tree .....	13
Fig.12: Outlook is selected as root node .....	14
Fig.13: The final tree .....	15
Fig.14: Document to be classified .....	16
Fig.15: Text representation using feature vector for document in Fig.14. ....	17
Fig.16: Expanding root node of the tree .....	18
Fig.17: Final Tree for the example .....	19
Fig.18: Feature weights in MaxEnt modeling for the class “earnings”.....	24
Fig.19: Training Dataset .....	25
Fig.20: Sample movie review data .....	26
Fig.21: Loading data.....	26
Fig.22: Data preprocessing .....	27
Fig.23: Represent text to feature vector .....	27
Fig.24: Data is now represented by feature vector .....	28
Fig.25: Run 4 test models .....	28
Fig.26: Result after using training 66% and test 34% .....	29
Fig.27: Result after using training 80% and test 20% .....	29
Fig.28: Result 5-fold cross validation (test model 3) .....	30
Fig.29: Result 5-fold cross validation (test model 4) .....	30

## Chapter 1: Introduction

FPT Technology Research Institute is a member of FPT University, an environment for researchers and students of FPT University to do researches that apply to information technology industry and to support FPT Corporation technology.

FPT is now developing a Question – Answer system. The idea of the system is that, users can send their questions, and the system has to automatically response answers. The overview design architecture of Q&A System includes some modules: Question parsing, Query generation, Document retrieval, Answer extraction, Answer ranking and selection.

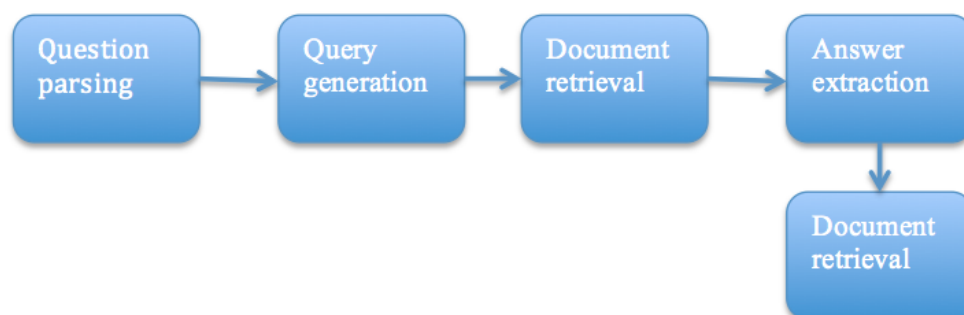


Fig.1: Architecture of Q&A System

The first module is very important in the system. All other modules depend on this module. The input of this module (and the whole system) is a question. A question is a text content. This module has to classify the type of question (text classification).

To classify question type, the system uses some popular algorithms such as Bernoulli, Naive Bayes, Rocchio use TF-IDF, TF-IDF Probability, Maximum Entropy, Decision Tree, Support Vector Machine (SVM). In fact, they have just implemented first three algorithms.

In the internship time, I have joined this Q&A system project. I researched text classification field with two algorithms: Decision Tree and Maximum Entropy.



## Chapter 2: Text classification

Text classification is also called document classification or document categorization. It is a sub-domain in Natural Language Processing (NLP). There are some problems in NLP such as: POS tagging, Word Sense Disambiguation, and Prepositional Phrase Attachment. In text classification, one document is an object need to interact.

The objective of text classification is to assign a document to one or more predefined classes (supervised). There are two general research approaches: supervised (classifying) and unsupervised learning (clustering).

The applications of text classification are:

- E-mail spam filtering
- Categorize newspaper articles into topics
- Organize Web pages into hierarchical categories
- Language identification: automatically determining the language of a text

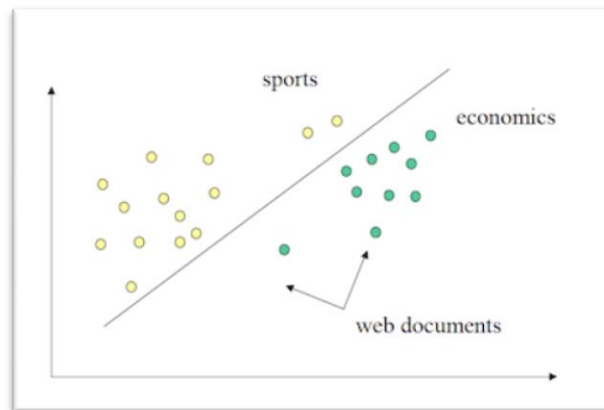


Fig.2: Example of text classification for web content categories

### 1. Formal Definition

In this document, we are talking about statistical classification

Given:

- A set of documents  $D = \{d_1, d_2, \dots, d_m\}$
- A predefined set of topics  $T = \{t_1, t_2, \dots, t_n\}$

Determine:

- The topic of  $d$ :  $t(d) \in T$ , where  $t(x)$  is a classification function ( $f$ ) whose domain is  $D$  and whose range is  $T$ .

There are two approaches of text classification: manual classification and automatic classification (machine learning-based approach).

- **Manual (rule-based approach):** we define a set of rules that classify documents.

For example: Classify documents into “sports” category

- “Ball” must be a word that is usually used in sports
  1. Rule 1: “ball”  $\in d \Rightarrow t(d) = \text{sports}$
- “Ball”, “Game”, “Play” must be three words that is usually used in sports:
  - Rule 2: “ball”  $\in d$  & “game”  $\in d$  & “play”  $\in d \Rightarrow t(d) = \text{sports}$ .

Advantage:

- Very accurate
- Need to write new rules for new data target.

Disadvantage:

- Use for small number of documents.

- **Automatic classification:** we use a set of sample documents (training data), which was classified into the predefined classes. A program automatically creates classifiers based on the training data with some algorithms. After that, we use the classifiers to test with new dataset (test data).

Advantage:

- With this approach, we can classify large number of documents.
- Domain independent (we can use for many domains of document)

Disadvantage:

- Need a training data (to train model)
- Difficult algorithms.

From now, when talking about text classification, we understand it is automatic classification. We will research machine learning-based approach.

## 2. General Model

In general, there are two phases in statistical classification: **training phase** and **classifying phase**.

## 2.1. Training

We have a training set of objects (documents), each element are classified with one or more classes. They are encoded via a data representation model. Each object in the training set is represented in the form  $(\vec{X}, c)$ , where  $\vec{X}$  is a vector of measurements, and  $c$  is a class (label). After that, we define a model class and a training procedure. The class model is set parameters of classifiers and the training procedure is algorithm we use to select the best parameters from the set parameter of classifiers. After getting the classifier model, we need to evaluate that classifier model.

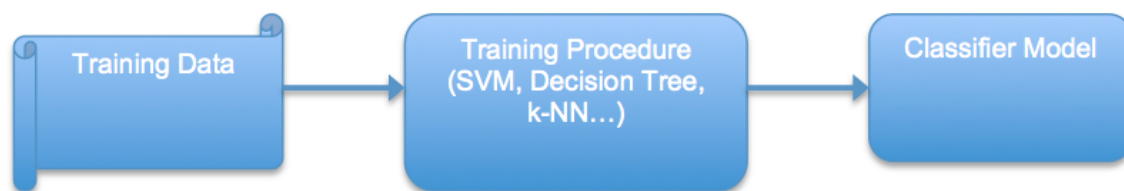


Fig.3: Training Phase Model

Examples of training procedure: Naive Bayes, K-Nearest-Neighbor (KNN), Decision Tree, Artificial Neural Network, and Support Vector Machine (SVM).

In detail, training phase needs some other process:

- Text preprocessing preprocesses the text document resources into a suitable format, structure to use.
- Text Representation Model encodes the document into a weighted model.
- Feature Extraction: removes stop words, and features that do not use to classify. This will improve the performance of training procedure.
- Classifier Evaluation: After implementing the classifier model, we need to evaluate its quality.

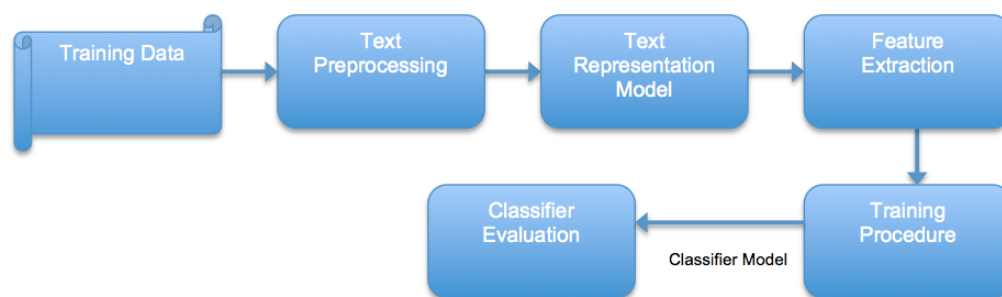


Fig.4: Training phase in detail

## 2.2. Classifying

After training phase, we have a classifier model. We use this classifier model to classify new text documents.

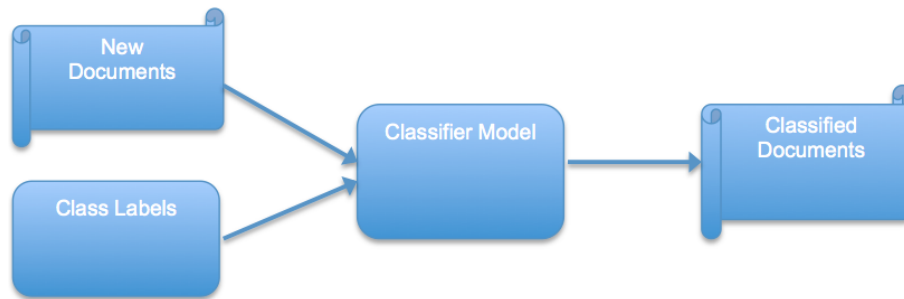


Fig.5: Classifying phase

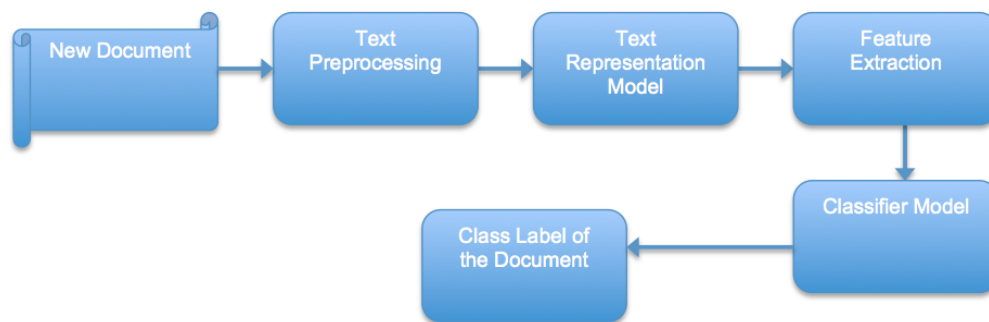


Fig.6: Detail of classifying phase

The classifier model depends on the algorithm. We will see the detail in part Decision Tree and Maximum Entropy part inside this document.

## 2.3. Text Preprocessing

Before presenting documents, it is need to do stemming (removing word but still keeping the document content) those documents.

This task is important because it reduces the quantity of words inside the documents. Therefore, it reduces the quantity of data to be represented.

Word analytic: the objective of this task is to determine the words occurred inside the document. The result of this process is a collection of each single word.

- Convert document to lower case.

- Remove words that rely occur in the document.
- Remove special character: [ ],[.], [,], [:], ["], ['], [,], [/], [[]], [~], [!], [!], [@], [#], [\$],[%],[^],[&],[\*],[()],[)]
- Removed suffix, prefix of word to get the root word. For example, “clusters”, “clustering”, “clustered” => cluster.
- Remove stop-words: Stop-word is the word occurred in the document but it is not used to classify document. Therefore, we need to remove stop-word to reduce the amount of data.

a	been	do
able	before	does
about	below	during
after	best	each
again	but	else
all	by	enough
almost	came	ever
also	can	except
am	cannot	few
and	clearly	for
are	come	former
as	consider	from
at	could	get
be	despite	goes
because	did	going

Fig.7: Example of stop-word in English

## 2.4. Text Representation Model

An original document is just a string. It needs to be represented in a suitable model to process. How it is represented depends on each algorithm. There are many approaches to represent a document such as Boolean Retrieval, Word Association, Vector Space Model, Probabilistic Retrieval, and Latent Semantic Indexing. One of the most popular methods is Vector Space Model because it is simple and efficient. In this report, we will focus on Vector Space Model only.

### 2.4.1. Vector Space Model

Each document is represented as a vector of word counts (possibly weighted). Each element in a vector is a word inside a document and the number of times it occurs in the document.

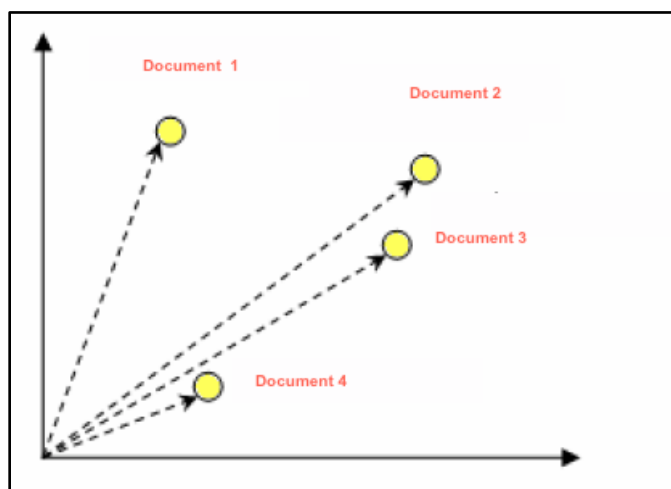


Fig.8: Represent text vectors in 2D space

In general, we have a document and it will be represented as vector  $\vec{X}(x_1, x_2, \dots, x_n)$ .

Where  $n$  is the number of time the word  $i$  ( $1 \leq i \leq n$ ) occurs in the document.

For example, we have 2 documents:

- DOC1: USTH students are very smart.
- DOC2: smart people agree that USTH students are smart.

After text preprocessing task, we represent them like this:

Word	DOC1	DOC2
Smart	1	2
Student	0	1
Agree	0	1
People	0	1
Very	1	0

Table 01: Vector Space Model example

To represent large number of documents in training data set,  $D_j$  is the document number  $j$  in the data set, and vector  $\vec{x}_j = (x_{1j}, x_{2j}, \dots, x_{nj})$  is the feature vector for document  $D_j$  and  $x_{ij}$  is the **Word Weighting** number  $i$  of vector  $\vec{x}_j$ .

#### 2.4.2. Word Weighting

There are some ways to calculate the weighting for a feature vector of a document.

- Word frequency weighting
- Boolean weighting
- TF\*IDF weighting
- Entropy weighting

In this report, we focus into two simple approaches: Word frequency weighting and TF\*IDF. In other word, it is the number of time that a word appears in a document.

Three values that are used to calculate the weighting are Term frequency, Document frequency, and Collection frequency.

- Term frequency

$tf_{ij}$  is the number of time word  $W_i$  appears in document  $D_j$ . The higher  $tf_{ij}$ , the better word to describe its document. Example: if the word “ball” appears many times in “Sport” category => the word “ball” is good to describe “sport” category.

- Collection frequency

$cf_{ij}$  is the number of time word  $W_i$  appears in the whole document collection.

- Document frequency

$df_i$  is the number of document containing word  $D_i$ . This value is an indicator of information. A semantically focused word will often occur several times in a document if it occurs at all other documents. Semantically unfocussed words are spread out homogeneously over all documents.

For example, we consider a collection of documents which belong to three categories: Economic, Finance, Stock.

Word	Collection frequency (cf)	Document frequency (df)
Stock	10400	3978
Money	10432	8770

Table 02: Word Weighting example

We can see that, the word Stock and Money have similar cf value, but the df value of stock is a half of Money. The reason is that, the word Stock is used in small boundary while the word Money can be use in almost categories.

TF\*IDF is the combination of tf and df into one. The formula is:

$$weight(i, j) = \begin{cases} (1 + \log(tf_{ij})) \log\left(\frac{N}{df_i}\right) & \text{if } tf_{ij} \geq 1 \\ 0 & \text{if } tf_{ij} = 0 \end{cases}$$

Where: N is the total number of documents.  $W_{ij} = 0$  if word does not appear in document.

## 2.5. Classifier Evaluation

After having classifier model, it is needed to evaluate quality of the model. We use new test dataset. Now, we simple consider a binary classifier model (2 class label). We usually make a table to evaluate classifier model:

	YES is correct	NO is correct
YES was assigned	a	b
NO was assigned	c	d

Table 03: Table classifier evaluation

Where:

- a: the number of document objects belong to the classifying class and they are assigned to the class by classifier model (correct assignment)
- b: the number of object does not belong to the classifying class but they are assigned to the class by classifier model (incorrect assignment).
- c: the number of document objects belong to the classifying class but they are removed from the class (incorrect remove).
- d: the number of object does not belong to the classifying class and they are removed from the class (correct remove).

To evaluate the quality of classifier model, we often use two values:

- **Accuracy:** the proportion of correctly classified objects.

$$\circ \frac{a+d}{a+b+c+d}$$

- **Error**

$$\circ \frac{c+b}{a+b+c+d}$$

In fact, we often consider only the correct assignment of classifier model. Therefore, there are some other values to evaluate classifier model:

- **Precision** is defined as a measure of the proportion of selected items that the system got right:

$$\circ \frac{a}{a+b}$$

- **Recall** is defined as the proportion of the target items that the system selected:

$$\circ \frac{a}{a+c}$$



- **Fallout:** the proportion of no targeted items that were mistakenly selected:

- $\frac{b}{b+a}$

In some situations, Precision and Recall are combined (F-measure):

- $F1 = 2 * \text{Precision} * \text{Recall} / (\text{Precision} + \text{Recall})$

Those measures above are usually used in binary classifier model. However, in fact, classifier model is used for many class labels. Therefore, in order to evaluate classifier model, we need two other methods: Macro-Averaging and Micro-Averaging.

### 2.5.1. Macro-Averaging

This is the method to calculate the average of precision and recall of each class. After making the table to evaluate classifier model, every class need to recalculate precision and recall.

$$P_{macro} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{a_i}{a_i + b_i}$$

$$R_{macro} = \frac{1}{|C|} \sum_{i=1}^{|C|} \frac{a_i}{a_i + c_i}$$

Where:  $|C|$  is to total number of class labels need to be classified.

### 2.5.2. Micro-Averaging

After making the table to evaluate classifier model, we calculate the sum of all classes for each value. Finally, we calculate precision and recall.

$$P_{micro} = \frac{\sum_{i=1}^{|C|} a_i}{\sum_{i=1}^{|C|} (a_i + b_i)}$$

$$R_{micro} = \frac{\sum_{i=1}^{|C|} a_i}{\sum_{i=1}^{|C|} (a_i + c_i)}$$

Where:  $|C|$  is to total number of class labels need to be classified.

### 3. Text Classification Methods

#### 3.1. Decision Tree

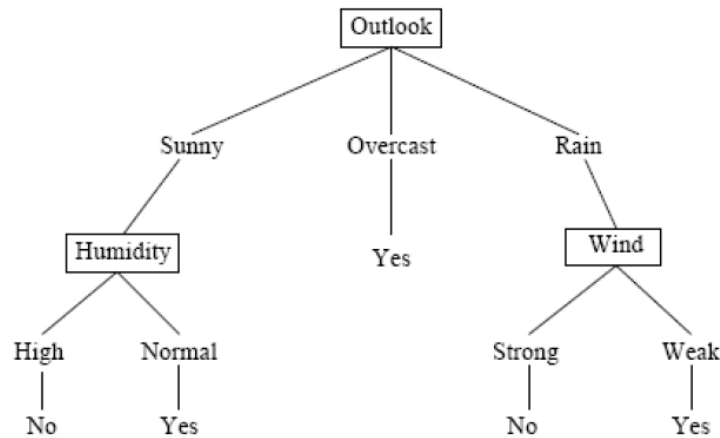


Fig.9: Example of Decision Tree

A decision tree is a tree whose internal nodes are tests and whose leaf nodes are class label (category). In decision tree, each internal node test one attribute and each branch from a node selects one value for the attribute. The attribute used to make the decision is not predefined. Therefore, we need to use an attribute, which gives maximum information. The leaf node is a class label. In Fig.9, it is an example of decision tree whose root node is “Outlook”. The leaf node is “Yes” and “No” which are class labels. It is famous example about someone will go tennis or not.

There are many decision-tree algorithms:

- ID3 (Iterative Dichotomiser 3)
- C4.5 (successor of ID3)
- CART (Classification And Regression Tree)
- CHAID (CHi-squared Automatic Interaction Detector). Performs multi-level splits when computing classification trees.
- MARS: Extend decision trees to better handle numerical data.

In this report, we will focus only on ID3 algorithm.

#### ❖ ID3 Algorithm

ID3 invented by Ross Quinlan, and it is used to generate a decision tree from a dataset. ID3 is the precursor to the C4.5 algorithm.

General steps:

1. From the dataset S, calculate the entropy of every attribute
2. Split the set S into subsets using the attribute for which entropy is minimum (or information gain is maximum).
3. Expanding decision tree by adding a node containing that attribute
4. Recursive on subsets using remaining attributes

The algorithm stops when there is no remaining attribute or the whole data set has been classified.

Pseudo code:

```

GenerateTree(X)
  If NodeEntropy(X) <  $\theta_I$  //  $I_m = -\sum_{i=1}^K p_m^i \log_2 p_m^i$  Stop criterion
    Create leaf labelled by majority class in X
    Return
   $i \leftarrow \text{SplitAttribute}(X)$ 
  For each branch of  $x_i$ 
    Find  $X_i$  falling in branch
    GenerateTree( $X_i$ ) } Depth first search

SplitAttribute(X)
  MinEnt  $\leftarrow$  MAX
  For all attributes  $i = 1, \dots, d$ 
    If  $x_i$  is discrete with  $n$  values
      Split X into  $X_1, \dots, X_n$  by  $x_i$ 
       $e \leftarrow \text{SplitEntropy}(X_1, \dots, X_n)$  //  $I_m = -\sum_{j=1}^n \frac{N_{mj}}{N_m} \sum_{i=1}^K p_{mj}^i \log p_{mj}^i$ 
      If  $e < \text{MinEnt}$  MinEnt  $\leftarrow e$ ; bestf  $\leftarrow i$ 
    Else //  $x_i$  is numeric
      For all possible splits
        Split X into  $X_1, X_2$  on  $x_i$ 
         $e \leftarrow \text{SplitEntropy}(X_1, X_2)$  //
        If  $e < \text{MinEnt}$  MinEnt  $\leftarrow e$ ; bestf  $\leftarrow i$ 
  Return bestf
  
```

Fig.10: Pseudo code of ID3 by The MIT Press1999

## ❖ Entropy

Entropy is a measure the impurity/inhomogeneity of a data set. In other word, entropy is the indicator of how much information inside a data set.

For example, considering two strings: “a dsa w dass dd dldadkl kok ds” and “a dog is bigger than a cat”. The first string is generated randomly letter. It has no information with us. The second string is not randomly, it contains information; every letter does not appearing completely randomly. Somehow, we can guess when a letter appears.

$$Entropy(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

Where:

- S is the set of training data
- C number of class labels
- Pi: the rate of elements belong class ci

❖ **Information Gain:**

Information gain is the measures of reducing entropy in S by an attribute in S.

$$Gain(S, A) = Entropy(S) - \sum_{v \in Value(A)} \frac{|S_v|}{S} Entropy(S_v)$$

Where:

- Value (A): set of A values.
- Sv: subset of S.

Now, we will go in detail how entropy and gain work. Let look at example below:

Day	Outlook	Temperature	Humidity	Wind	Play Tennis
D1	Sunny	Hot	High	Weak	No
D2	Sunny	Hot	High	Strong	No
D3	Overcast	Hot	High	Weak	Yes
D4	Rain	Mild	High	Weak	Yes
D5	Rain	Cool	Normal	Weak	Yes
D6	Rain	Cool	Normal	Strong	No
D7	Overcast	Cool	Normal	Strong	Yes
D8	Sunny	Mild	High	Weak	No
D9	Sunny	Cool	Normal	Weak	Yes
D10	Rain	Mild	Normal	Weak	Yes
D11	Sunny	Mild	Normal	Strong	Yes
D12	Overcast	Mild	High	Strong	Yes
D13	Overcast	Hot	Normal	Weak	Yes
D14	Rain	Mild	High	Strong	No

Fig.11: Example dataset in Decision Tree

Follow the entropy formula, we have:

In S, we have number of No=5, number of Yes=9.

$$Entropy(S) = -\frac{9}{14} \log_2 \frac{9}{14} - \frac{5}{14} \log_2 \frac{5}{14} = 0.940$$

Now, we calculate Information gain for Wind attribute:

Wind has two values Weak and Strong. Weak=8, Strong =6.

$$\begin{aligned} Gain(S, Wind) &= Entropy(S) - \sum_{v \in \{Weak, Strong\}} \frac{|S_v|}{S} Entropy(S_v) \\ &= Entropy(S) - \frac{8}{14} Entropy(S_{weak}) - \frac{6}{14} Entropy(S_{strong}) \\ &= 0.940 - \frac{8}{14} * 0.811 - \frac{6}{14} * 1.00 = 0.048 \end{aligned}$$

In root node, what attribute will be selected {Outlook, Temperature, Humidity, Wind}?

We calculate information gain for all attributes:

- Gain(S, Outlook) = ... = 0.246
- Gain(S, Temperature) = ... = 0.029
- Gain(S, Humidity) = ... = 0.151
- Gain(S, Wind) = ... = 0.048

The attribute has max gain will be selected. In this case, Outlook will be selected to test in root node.

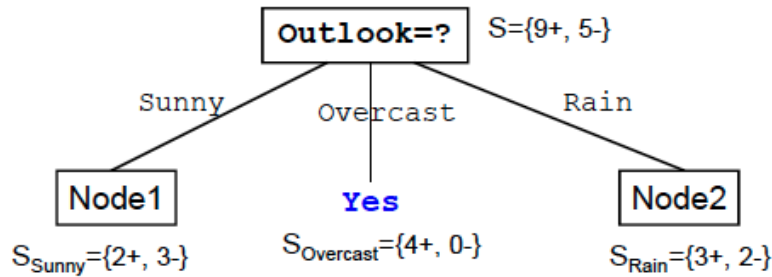


Fig.12: Outlook is selected as root node

In Node 1, what attribute will be selected?

Note: Outlook has been selected at root node; so, it will not be selected anymore.

- Gain(sunny, Temperature)=...= 0.57

- $\text{Gain}(\text{sunny}, \text{Humidity}) = \dots = 0.97$
- $\text{Gain}(\text{sunny}, \text{Wind}) = \dots = 0.019$

Humidity has max gain; therefore it will be selected at Node 1.

Calculating  $\text{Gain}(\text{rain}, \text{Temperature})$ ,  $\text{Gain}(\text{rain}, \text{Wind})$  we will have final tree:

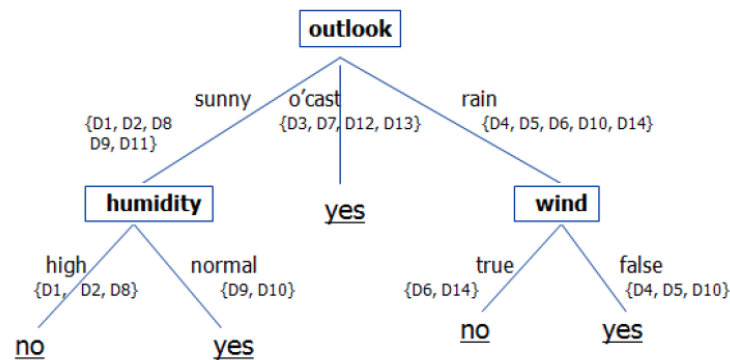


Fig.13: The final tree

Final Tree is the classifier model. We can use it to classify new data.

For example, we have new data as follow:

$D = \{\text{sunny}, \text{cool}, \text{high}, \text{strong}\}$

Apply classifier model:

- Root Node: D is SUNNY => Turn Left
    - Humidity Node: D is HIGH => Turn Left
- ⇒ D belong "NO" class (No one play tennis in this weather condition).

#### ❖ Apply Decision Tree in Text Classification

This section, we will see how to apply decision tree algorithm to classify documents. We will take an example from the book "Foundations of Statistical Natural Language Processing".

Here is the sample of documents about REUTER article:

```

<REUTERS NEWID="11">
<DATE>26-FEB-1987 15:18:59.34</DATE>
<TOPICS><D>earn</D></TOPICS>
<TEXT>
<TITLE>COBANCO INC &lt;CBCO> YEAR NET</TITLE>
<DATELINE> SANTA CRUZ, Calif., Feb 26 - </DATELINE>
<BODY>Shr 34 cts vs 1.19 dlrs
      Net 807,000 vs 2,858,000
      Assets 510.2 mln vs 479.7 mln
      Deposits 472.3 mln vs 440.3 mln
      Loans 299.2 mln vs 327.2 mln
      Note: 4th qtr not available. Year includes 1985
      extraordinary gain from tax carry forward of 132,000 dlrs,
      or five cts per shr.
      Reuter
</BODY></TEXT>
</REUTERS>

```

Fig.14: Document to be classified

### ➤ Text representation

First step is to represent the document. In this example, the document is represented in a vector of features  $\vec{x}_j = (w_{1j}, w_{2j}, \dots, w_{kj})$ .  $w_{ij}$  is the weight (term weighting) of features in the document.

$$w_{ij} = \text{round} \left( 10 \times \frac{1 + \log(tf_{ij})}{1 + \log(l_j)} \right)$$

Where:

$tf_{ij}$  is the number of times a word  $i$  appears in document  $j$ .

$l_j$  is the length of document  $j$

$w_{ij} = 0$  if the word  $i$  does not appear in document  $j$ .

For instance, if a word "student" appears in the document 6 times and the length of the document is 89 words. The weight of word "student" is  $w_{ij} = \text{round} \left( 10 \times \frac{1 + \log 6}{1 + \log 89} \right) = 5$ .

Documents are represented by vector of features, but there are many features in the vector. How to select the best features is also a problem. We usually select the top features, which have biggest weights (the words that appear a lot in documents). However, in some situations, that way is not the best. Therefore, it is one more way to select the best feature in document. It is **chi-square test**.

$$\chi^2 = \sum \frac{(\text{observation} - \text{expected})^2}{\text{expected}}$$

Where:

Observation is the measure value.

Expected is expected value.

vs	5	$\vec{x} =$	$c = 1$
mln	5		
cts	3		
	3		
6	3		
000	4		
loss	0		
	0		
"	0		
3	4		
profit	0		
dlrs	3		
1	2		
pct	0		
is	0		
s	0		
that	0		
net	3		
lt	2		
at	0		

Fig.15: Text representation using feature vector for document in Fig.14.

### ❖ Training Phase

After represent document, we use data to train model. In this example, the author uses 7681 documents. There are 2304 documents belonging to “earnings” and 5377 is others.

The root node will have 7681 documents.

<b>node 1</b>
7681 articles
$P(c n_1) = 0.300$
split: cts
value: 2

$P(c|n)$ : probability of documents in node 1, which belong to class C.

$$P(c|n) = \frac{n_c + 1}{n + |C|}$$

Where  $|C|$  is the total number of class.



To expand from root node, with ID3 algorithm, we need to calculate the information gain of each feature of the document. In here, feature vector is the continuous values. Therefore, to split the vector, we need to determine where to split. In fact, we use a Heuristic algorithm to determine the best value to split for each feature vector of a document.

The information gain of each feature is calculated as follow:

$$G(a,y) = H(t) - H(t|a) = H(t) - (p_L H(t_L) + p_R H(t_R))$$

Where :

- a is the attribute we split on
- y is the value of a we split on
- t is the distribution of the node that we split
- $p_L$  and  $p_R$  are the proportion of elements that are passed on to the left and right nodes,
- $t_L$  and  $t_R$  are the distributions of the left and right nodes
- $H(t)$  is the entropy in current node.
- $H(t_R)$  is the entropy right node
- $H(t_L)$  is the entropy left node

Look at Fig.15, we select the “cts” feature and the best split value is 2. The information gain of “cts”:

The Entropy Node 1:  $H(t) = -0.3 \cdot \log(0.3) - 0.7 \cdot \log(0.7) = 0.611$

The entropy left node  $H(t_L) = -0.116 \cdot \log(0.116) - 0.884 \cdot \log(0.884) = 0.359$

The entropy right node  $H(t_R) = -0.943 \cdot \log(0.943) - 0.057 \cdot \log(0.057) = 0.219$

$$\Rightarrow \text{Gain (cts,2)} = H(t) - H(t|\text{cts}) = 0.611 - \left( \frac{5977}{7681} \cdot 0.359 + \frac{1704}{7681} \cdot 0.219 \right) = 0.283$$

We calculate for other features of the document. The information gain of “cts” is the biggest.

Therefore, we will select “cts” to split the document collection with the split value is 2.

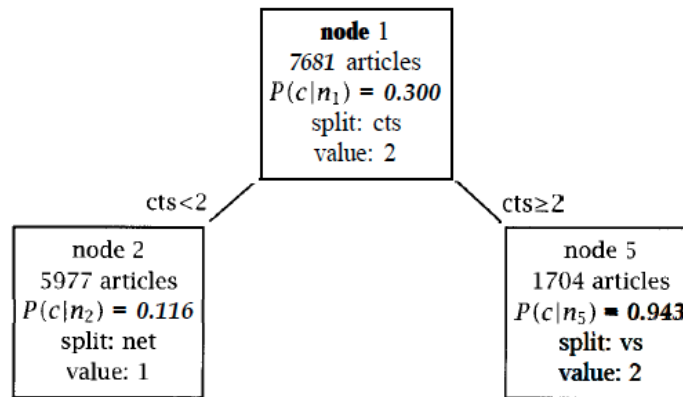


Fig.16: Expanding root node of the tree

We calculate those steps above for left and right node of the tree until all nodes are classified. Nodes contains documents of one class only is also the stop condition of ID3 algorithm.

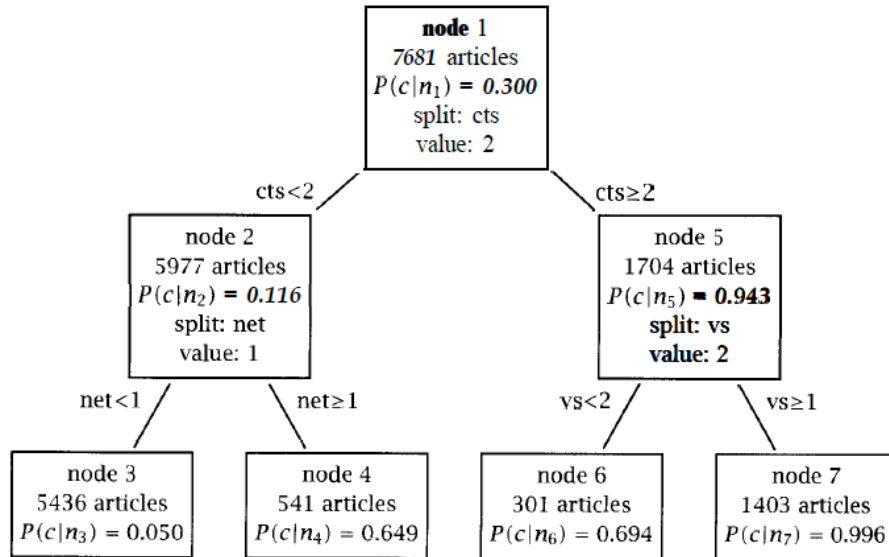


Fig.17: Final Tree for the example

### ❖ Pruning

The pruning step is to avoid *overfitting* of the tree and to optimize performance on new data. *Overfitting* is the problem that the tree can classify correctly for the training data but difficult to generalize to other data. At every step, we select the remaining leaf node we expect by one or some criterions to be least helpful for accurate classification. One example of pruning criterion is to compute a measure of confidence that indicates how much evidence the node is helpful. In each step in the process we save a classifier, which corresponds to the decision tree with the nodes remaining at this time.

### ❖ Cross-validation

To maximize the accurate classification, we use Cross-validation method to train the tree. From the training dataset, instead of creating one tree, we create some trees. For example, we create 5 trees (5-fold cross-validation). Each tree uses 80% data to train and 20% to test but not the same train data in training dataset. After pruning 5 trees, we calculate the average size of 5 trees (average tree). Now, we create a new tree and use 100% training dataset, pruning the new tree to fit with average tree. Finally, we can use new tree to test with new data set.

### ❖ Classifying Phase

From the tree we can use to test new dataset. Base on the classifier model, from the root node to leaf node check the path suitable for the new test data.

### 3.2. Maximum Entropy

Maximum entropy is used for processing many natural languages. It becomes popular and can be used to replace many other methods because its efficiency.

The main idea of maximum entropy is that one should prefer the most uniform models that also satisfy any given constraints. For example, we consider a text classification with 4 class labels. We have one constraint that averages 40% percent of documents in category “faculty” contains word “professor”. Therefore, if we know any document contains word “professor”, we have 40% that document belong to “faculty” class and 20% the ability that document belongs to 3 remaining classes.

What happen if the number of constraints is large? At that time, there are many models satisfied with all constraints and we need a technique to find optimal solution (best model). That technique is called Maximum Entropy.

#### Condition random variable

We consider a random variable  $X$  with the distribution:

$X$	$x_1$	$x_2$	$x_3$	$\dots$	$x_M$
$P$	$p_1$	$p_2$	$p_3$	$\dots$	$p_M$

Entropy of  $X$  is to measure amount of uncertain information of  $X$ .

$$H(X) = H(p_1, p_2, \dots, p_M) = -\sum_{i=1}^M p_i \log_2(p_i)$$

This entropy is in range of 0 and  $\log(M)$ .

The entropy is maximum when  $P_1 = P_2 = \dots = P_M$ .

Example:

- A string has only character “a” has entropy 0. We know exactly where and when the character “a” will appears in the string.
- A string has “0” and “1” character which appears randomly has entropy 1. We don’t know where and when a character appears in the string.

### Constraints and Features

In MaxEnt, we use the training dataset to build constraints on the conditional distribution. Every constraint expresses a characteristic of the training data that should also be present in the learned distribution. Each constrain is presented by a feature function:

$$f_i(\vec{x}_j, c) = \begin{cases} 1, & \text{if } w_{ij} > 0 \text{ and } c = 1 \\ 0, & \text{otherwise} \end{cases}$$

Where:

- $\vec{x}_j$  Is the feature vector to represent document j
- C is a class label.
- $w_{ij}$  is the weight of word i in document j

In here, we use a binary feature function to present if a word appears in a document or not. However, in fact we can use a function with real value to describe weight of features.

### Maximum Entropy Model

There are many models used in MaxEnt method. In this report, we will focus on the log-linear model.

$$p(\vec{x}, c) = \frac{1}{Z} \prod_{i=1}^K \alpha_i^{f_i(\vec{x}, c)}$$

Equation: P\*

Where:

- K is the number of features.
- $\alpha_i$  is the weight of feature
- $f_i$  and Z are normal constant used to ensure that a probability distribution results.

The objective of this model is to determine the best  $\alpha_i$  to classify data.

For example, we calculate  $P(\vec{x}, 0)$  and  $P(\vec{x}, 1)$ , choose the class label with the greater probability.

### Generalized iterative scaling

In this section, we will use some signs:

- $S$ : training dataset
- $\tilde{p}(x)$ : empirical probability of  $x$  in  $S$
- $p(x)$ : probability of  $x$
- $E_{\tilde{p}}f_i$ : The empirical expectation of features  $f_i$
- $E_p f_i$ : The expected value of features  $f_i$

Generalized iterative scaling is a method to find the maximum entropy distribution of form  $p^*$  (in beginning of section Maximum Entropy Model) of this log-linear model obeys the following set of constraints:

$$E_p f_i = E_{\tilde{p}} f_i$$

The expected value will be computed:

$$E_p f_i = \sum_{\vec{x}, c} p(\vec{x}, c) f_i(\vec{x}, c)$$

In general, the maximum entropy distribution  $E_p f_i$  cannot be calculated exactly since it would involve summing over all possible combinations of  $\vec{x}$  and  $c$ , a set can be very large. Instead, we use the following approximation:

$$E_p f_i \approx \sum_{\vec{x}, c} \tilde{p}(\vec{x}, c) f_i(\vec{x}, c)$$

Where:

$$\tilde{p}(c|\vec{x}) = \frac{\exp(\sum_i \alpha_i f_i(\vec{x}, c))}{\sum_c \exp(\sum_i \alpha_i f_i(\vec{x}, c))}$$

The empirical expectation of feature will be computed as:

$$E_{\tilde{p}} f_i = \sum_{\vec{x}, c} \tilde{p}(\vec{x}, c) f_i(\vec{x}, c) = \frac{1}{N} \sum_{j=1}^N f_i(\vec{x}_j, c)$$

Where:  $N$  is the total number of documents in training dataset.

The algorithm needs that the sum of the all features for each possible  $p(\vec{x}, c)$  be equal to a constant  $C$ , with  $C$  is declared as:

$$\forall \vec{x}, c \quad \sum_i f_i(\vec{x}, c) = C$$

$C$  needs to be the biggest value in the set of all features to find optimal solution:

$$C \stackrel{\text{def}}{=} \max_{\vec{x}, c} \sum_{i=1}^K f_i(\vec{x}, c)$$

However, it is not all of the feature functions will have the value equal to  $C$  ( $\leq$ ). Therefore, we will add one feature function  $f_{K+1}$  into each feature function:

$$f_{K+1}(\vec{x}, c) = C - \sum_{i=1}^K f_i(\vec{x}, c)$$

#### ❖ Generalized Iterative Algorithm

1. Initialize the set of weight  $(\alpha_i^{(1)})$  ( $1 \leq i \leq K+1$ ).  $(1)$  is the set of weight for the first initial time. It will increase in each initial time. We usually initialize with the value 1.  
Compute the  $E_{\vec{p}} f_i$ .
2. Calculate  $\mathbf{p}^{(n)}(\vec{x}, c)$  for the distribution  $\mathbf{p}^{(n)}$  given by  $(\alpha_i^{(n)})$  for each element  $(\vec{x}, c)$  in training data. It will be computed as:

$$\mathbf{p}^{(n)}(\vec{x}, c) = \frac{1}{Z} \prod_{i=1}^{K+1} (\alpha_i^{(n)})^{f_i(\vec{x}, c)}$$

3. Compute the expected value  $E_p f_i$  (with  $1 \leq i \leq K+1$ ) for all feature functions with the equation of  $E_p f_i$ .

$$E_p f_i = \frac{1}{N} \sum_{j=1}^N \sum_c p(c | \vec{x}) f_i(\vec{x}_j, c)$$

4. Update the parameters  $(\alpha_i)$ :

$$\alpha_i^{(n+1)} = \alpha_i^n \left( \frac{E_{\vec{p}} f_i}{E_{\mathbf{p}^{(n)}} f_i} \right)^{\frac{1}{C}}$$

5. If the parameters of the procedure have converged, stop; otherwise, incrementing  $n$  and go to step 2.

The output after running the algorithm is the set of  $(a_i)$ .

### Apply in text classification

#### ❖ Text representation

We use the same way to represent text document that is feature vector, but the weight is the log  $(a_i)$ , which is generated by maximum entropy modeling.

Word $w^i$	Feature weight $\log \alpha_i$
vs	0.613
mln	-0.110
cts	1.298
	-0.432
&	-0.429
000	-0.413
loss	-0.332
	-0.085
"	0.202
3	-0.463
profit	0.360
dlrs	-0.202
l	-0.211
pct	-0.260
is	-0.546
s	-0.490
that	-0.285
net	-0.300
It	1.016
at	-0.465
$f_{K+1}$	0.009

Fig.18: Feature weights in MaxEnt modeling for the class “earnings”

We see  $f_{k+1}$  is in the end of feature vector.

#### ❖ Classifying phase

The input is a feature vector of a document:  $\vec{x}=(w_1, w_2, \dots, w_j)$ . We use trained feature model  $(a_i)$  above, compute two probability  $P(c, \vec{x})$  and  $P(\neg c, \vec{x})$ .

$$p(c|\vec{x}) = \frac{\exp(\sum_i \alpha_i f_i(\vec{x}, c))}{\sum_c \exp(\sum_i \alpha_i f_i(\vec{x}, c))}$$

The document belongs to class, which has higher probability.

## Chapter 3: Experiment

### 1. Introduction to Weka

“Weka is a collection of machine learning algorithms for data mining tasks. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. It is also well-suited for developing new machine learning schemes” (Weka Homepage).

Weka is developed by University of Waikato. It is famous software for machine learning.

### 2. Implementation text classification

We will make example to understand steps in text classification. In here, we only try with decision tree because it is supported by Weka.

#### 2.1. Decision Tree

Training Dataset: Movie review data. The data includes two classes: negative (bad review) and positive review (good review). There are 1000 negative reviews and 1000 positive reviews.

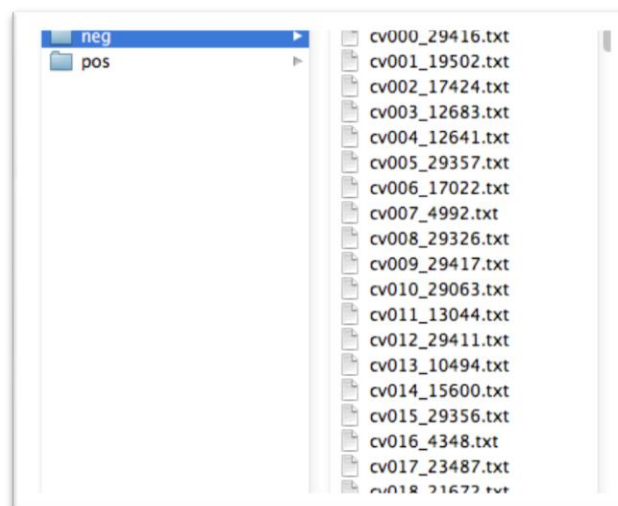
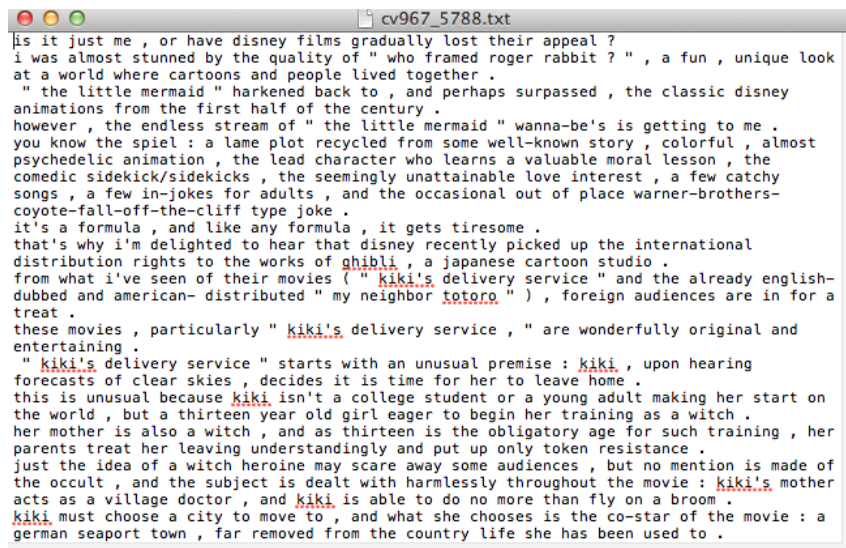


Fig.19: Training Dataset





is it just me , or have disney films gradually lost their appeal ?  
i was almost stunned by the quality of " who framed roger rabbit ? " , a fun , unique look  
at a world where cartoons and people lived together .  
" the little mermaid " harkened back to , and perhaps surpassed , the classic disney  
animations from the first half of the century .  
however , the endless stream of " the little mermaid " wanna-be's is getting to me .  
you know the spiel : a lame plot recycled from some well-known story , colorful , almost  
psychedelic animation , the lead character who learns a valuable moral lesson , the  
comedic sidekick/sidekicks , the seemingly unattainable love interest , a few catchy  
songs , a few in-jokes for adults , and the occasional out of place warner-brothers-  
coyote-fall-off-the-cliff type joke .  
it's a formula , and like any formula , it gets tiresome .  
that's why i'm delighted to hear that disney recently picked up the international  
distribution rights to the works of ghibli , a japanese cartoon studio .  
from what i've seen of their movies ( " kiki's delivery service " and the already english-  
dubbed and american- distributed " my neighbor totoro " ) , foreign audiences are in for a  
treat .  
these movies , particularly " kiki's delivery service , " are wonderfully original and  
entertaining .  
" kiki's delivery service " starts with an unusual premise : kiki , upon hearing  
forecasts of clear skies , decides it is time for her to leave home .  
this is unusual because kiki isn't a college student or a young adult making her start on  
the world , but a thirteen year old girl eager to begin her training as a witch .  
her mother is also a witch , and as thirteen is the obligatory age for such training , her  
parents treat her leaving understandingly and put up only token resistance .  
just the idea of a witch heroine may scare away some audiences , but no mention is made of  
the occult , and the subject is dealt with harmlessly throughout the movie : kiki's mother  
acts as a village doctor , and kiki is able to do no more than fly on a broom .  
kiki must choose a city to move to , and what she chooses is the co-star of the movie : a  
german seaport town , far removed from the country life she has been used to .

Fig.20: Sample movie review data

Now, we will use this dataset in Weka, with ID3 text classification.

Steps:

1. Load training dataset into Weka

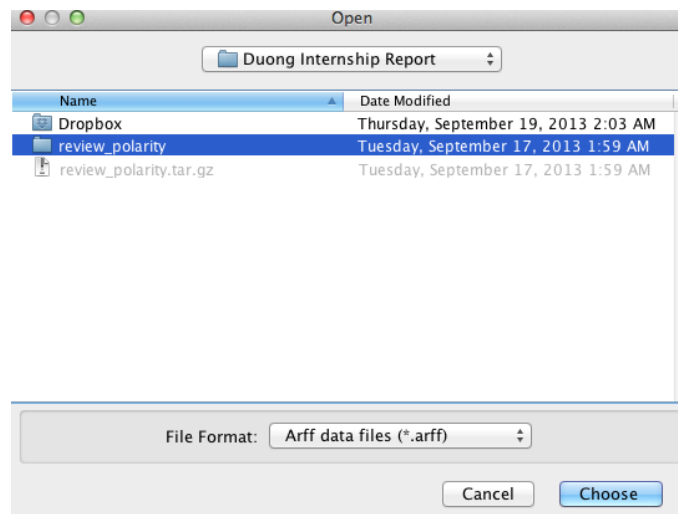


Fig.21: Loading data

## 2. Data Preprocessing

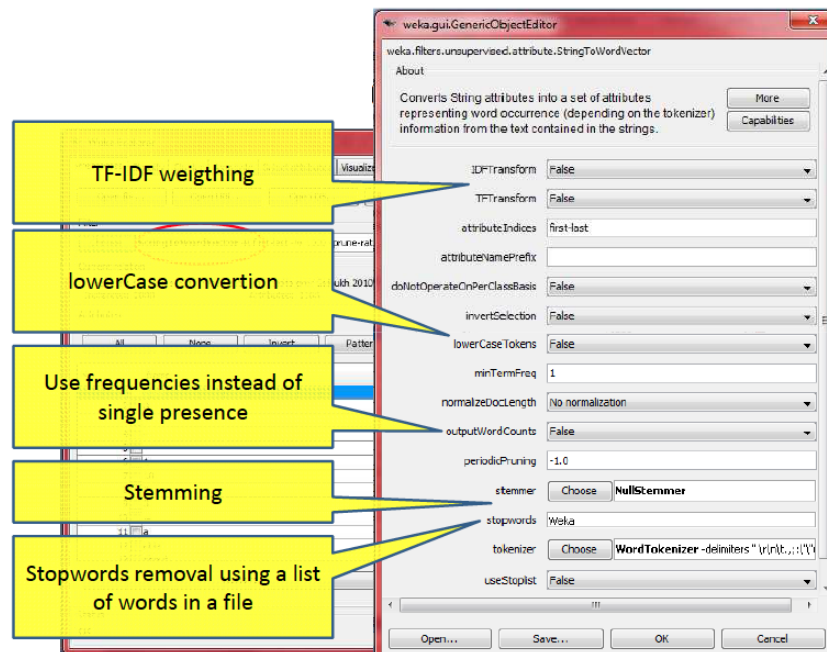


Fig.22: Data preprocessing

## 3. Represent data from text to feature vector

Using **StringToWord** in Weka to represent a text to feature vector  $\vec{V}=[v_1, v_2, v_3, \dots, v_n, class]$

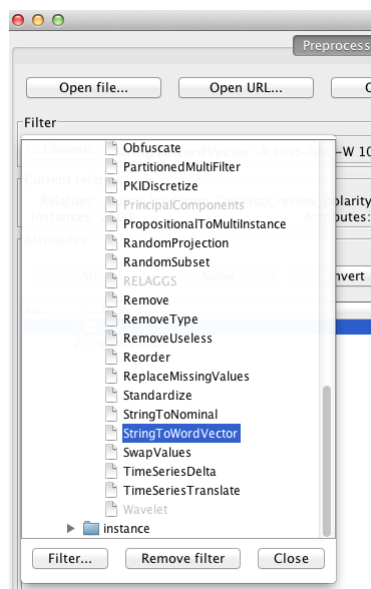
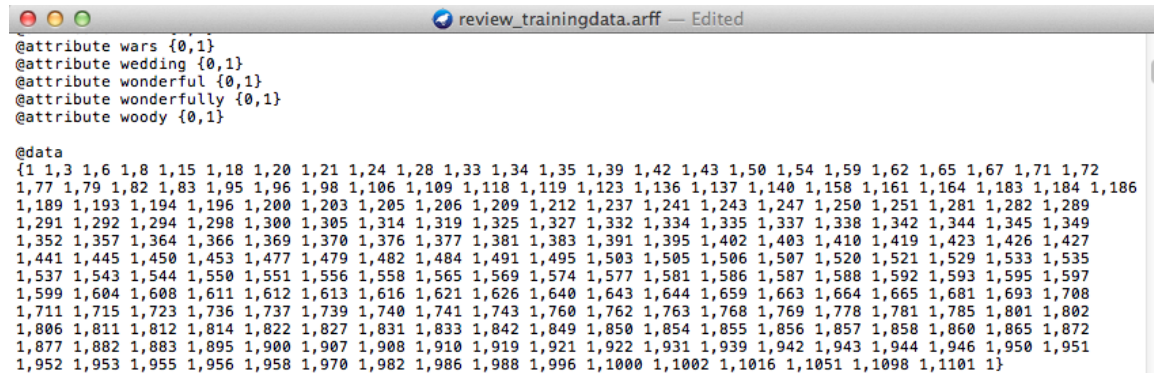


Fig.23: Represent text to feature vector

We can see word and term weighting now represent the data in weka.



```
@attribute wars {0,1}
@attribute wedding {0,1}
@attribute wonderful {0,1}
@attribute wonderfully {0,1}
@attribute woody {0,1}

@data
{1 1,3 1,6 1,8 1,15 1,18 1,20 1,21 1,24 1,28 1,33 1,34 1,35 1,39 1,42 1,43 1,50 1,54 1,59 1,62 1,65 1,67 1,71 1,72
1,77 1,79 1,82 1,83 1,95 1,96 1,98 1,106 1,109 1,118 1,119 1,123 1,136 1,137 1,140 1,158 1,161 1,164 1,183 1,184 1,186
1,189 1,193 1,194 1,196 1,200 1,203 1,205 1,206 1,209 1,212 1,237 1,241 1,243 1,247 1,250 1,251 1,281 1,282 1,289
1,291 1,292 1,294 1,298 1,300 1,305 1,314 1,319 1,325 1,327 1,332 1,334 1,335 1,337 1,338 1,342 1,344 1,345 1,349
1,352 1,357 1,364 1,366 1,369 1,370 1,376 1,377 1,381 1,383 1,391 1,395 1,402 1,403 1,410 1,419 1,423 1,426 1,427
1,441 1,445 1,450 1,453 1,477 1,479 1,482 1,484 1,491 1,495 1,503 1,505 1,506 1,507 1,520 1,521 1,529 1,533 1,535
1,537 1,543 1,544 1,550 1,551 1,556 1,558 1,565 1,569 1,574 1,577 1,581 1,586 1,587 1,588 1,592 1,593 1,595 1,597
1,599 1,604 1,608 1,611 1,612 1,613 1,616 1,621 1,626 1,640 1,643 1,644 1,659 1,663 1,664 1,665 1,681 1,693 1,708
1,711 1,715 1,723 1,736 1,737 1,739 1,740 1,741 1,743 1,760 1,762 1,763 1,768 1,769 1,778 1,781 1,785 1,801 1,802
1,806 1,811 1,812 1,814 1,822 1,827 1,831 1,833 1,842 1,849 1,850 1,854 1,855 1,856 1,857 1,858 1,860 1,865 1,872
1,877 1,882 1,883 1,895 1,900 1,907 1,908 1,910 1,919 1,921 1,922 1,931 1,939 1,942 1,943 1,944 1,946 1,950 1,951
1,952 1,953 1,955 1,956 1,958 1,970 1,982 1,986 1,988 1,996 1,1000 1,1002 1,1016 1,1051 1,1098 1,1101 }
```

Fig.24: Data is now represented by feature vector

#### 4. Training and Classifying

Now, we will use 4 test models and compare the result of each test.

1. Split data training 66% and test 34% (test model 1)
2. Split data training 80% and test 20% (test model 2)
3. Using 5-fold cross validation (test model 3)
4. Using 10-fold cross validation (test model 4)

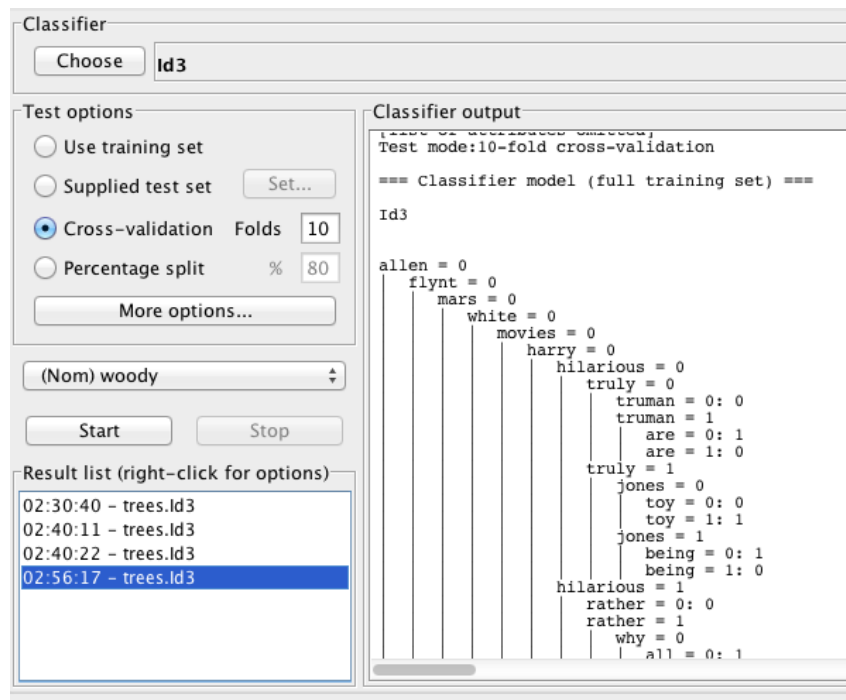


Fig.25: Run 4 test models

**Result:**

- ❖ Training 66% and test 34%

```

=== Evaluation on test split ===
=== Summary ===

Correctly Classified Instances      651          95.7353 %
Incorrectly Classified Instances    29           4.2647 %
Kappa statistic                    0.3865
Mean absolute error                0.0426
Root mean squared error            0.2065
Relative absolute error            61.8594 %
Root relative squared error        103.7757 %
Total Number of Instances         680

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.983    0.643    0.973      0.983    0.978      0.67      0
                0.357    0.017    0.476      0.357    0.408      0.67      1
Weighted Avg.    0.957    0.617    0.952      0.957    0.954      0.67

=== Confusion Matrix ===
  a  b  <-- classified as
641 11 |   a = 0
 18 10 |   b = 1

```

Fig.26: Result after using training 66% and test 34%

- ❖ Training 80% and test 20%

```

Correctly Classified Instances      383          95.75 %
Incorrectly Classified Instances    17           4.25 %
Kappa statistic                    0.3521
Mean absolute error                0.0425
Root mean squared error            0.2062
Relative absolute error            56.5302 %
Root relative squared error        96.6154 %
Total Number of Instances         400

=== Detailed Accuracy By Class ===

                TP Rate  FP Rate  Precision  Recall  F-Measure  ROC Area  Class
                0.992    0.737    0.964      0.992    0.978      0.628      0
                0.263    0.008    0.625      0.263    0.37       0.628      1
Weighted Avg.    0.958    0.702    0.948      0.958    0.949      0.628

=== Confusion Matrix ===
  a  b  <-- classified as
378  3 |   a = 0
 14  5 |   b = 1

```

Fig.27: Result after using training 80% and test 20%

## ❖ Using 5-fold cross validation (test model 3)

Correctly Classified Instances	1912	95.6	%				
Incorrectly Classified Instances	88	4.4	%				
Kappa statistic	0.3004						
Mean absolute error	0.044						
Root mean squared error	0.2098						
Relative absolute error	67.3817	%					
Root relative squared error	116.5722	%					
Total Number of Instances	2000						
=== Detailed Accuracy By Class ===							
	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.978	0.687	0.976	0.978	0.977	0.646	0
	0.313	0.022	0.333	0.313	0.323	0.646	1
Weighted Avg.	0.956	0.664	0.955	0.956	0.955	0.646	
=== Confusion Matrix ===							
a	b	<-- classified as					
1891	42	a = 0					
46	21	b = 1					

Fig.28: Result 5-fold cross validation (test model 3)

## ❖ Using 10-fold cross validation (test model 4)

Correctly Classified Instances	1921	96.05	%				
Incorrectly Classified Instances	79	3.95	%				
Kappa statistic	0.4272						
Mean absolute error	0.0395						
Root mean squared error	0.1987						
Relative absolute error	60.5459	%					
Root relative squared error	110.4499	%					
Total Number of Instances	2000						
=== Detailed Accuracy By Class ===							
	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0.977	0.522	0.982	0.977	0.98	0.727	0
	0.478	0.023	0.421	0.478	0.448	0.727	1
Weighted Avg.	0.961	0.506	0.963	0.961	0.962	0.727	
=== Confusion Matrix ===							
a	b	<-- classified as					
1889	44	a = 0					
35	32	b = 1					

Fig.29: Result 5-fold cross validation (test model 4)

Comments:

After running 4 test models, we see that:

- The result of test model 2 is better than model 1. In other word, the bigger training data the higher accuracy classification (Precision, recall, F1).
- The result of test model 4 is better than model 3. We see that, if we create bigger number of cross-validation to train model, the better standard model we will have.

➤ **Advantage of decision tree**

- The decision tree is easy to understand, interpret and explain
- Easily to trace the path from root to leaf node.

➤ **Disadvantage**

- The main problem is *overfitting*. Therefore decision tree needs pruning.

## Chapter 4: Conclusion

There are billions websites in the Internet, that amount of data is very big. There data increase day by day. Therefore, text classification is very important especially to manage, search the data.

In general, TC includes two phases: training phase and classifying phase. Training phase is the process to get a model from training data. That model will be use to apply in the test data. Training phase includes steps: text preprocessing, data representation, training model and finally evaluate the classifier model before classify phase. This is the first time I work with text classification field, and I saw its importance. However, the time is not enough for me to research the detail of TC problems.

The project showed the overview of TC also the two algorithms applying in training phase.

After that, the project showed examples of TC in Weka to understand the processes of TC and its terms.

After the project, I understand what the TC is, what are its processes, what is the objective of each process. I also understand how the decision tree and maximum entropy work in TC.

In the Q&A system, I have contributed the data which are questions about “Sport” category for training model.

### Future Works

I will continue to research text classification problems. There are many other problems, which are not discussed in this report.

First, I need to understand clearly both decision tree and maximum entropy with its problems.

Second, I will research other popular methods such as SVM, Naive Bayes, Rocchio use TF-IDF, TF-IDF Probability.

Finally, I will implement my own code in programming language. I will make a tool to test the decision tree algorithm. It is the best way to understand completely the problem.

## References

### Web:

- [1]. Document classification: [http://en.wikipedia.org/wiki/Document\\_classification](http://en.wikipedia.org/wiki/Document_classification)
- [2]. Document Representation: [http://en.wikipedia.org/wiki/Vector\\_space\\_model](http://en.wikipedia.org/wiki/Vector_space_model)
- [3]. Entropy: [http://en.wikipedia.org/wiki/Entropy\\_\(information\\_theory\)](http://en.wikipedia.org/wiki/Entropy_(information_theory))
- [4]. Information Gain: [http://en.wikipedia.org/wiki/Information\\_gain\\_in\\_decision\\_trees](http://en.wikipedia.org/wiki/Information_gain_in_decision_trees)
- [5]. Chi-squared: [http://en.wikipedia.org/wiki/Chi-squared\\_test](http://en.wikipedia.org/wiki/Chi-squared_test)
- [6]. Weka Homepage: <http://www.cs.waikato.ac.nz/ml/weka/>
- [7]. Weka Data <http://www.cs.cornell.edu/people/pabo/movie-review-data/>

### Book:

- [8]. Christopher D.Manning, Hinrich Schutze, “**Foundations of Statistical Natural Language Processing**”
- [9]. Kamal Nigam, John Lafferty, Andrew McCallum, “**Using Maximum Entropy for Text Classification**”, In IJCAI-99 Workshop on Machine Learning for Information Filtering.
- [10]. Kostas Fragos, Yannis Maistros, Christos Skourlas, “**A Weighted Maximum Entropy Language Model for Text Classification**”, NLUCS 2005: p.55-67.
- [11]. Tom M. Mitchell, “**Machine learning**”, Published by McGraw-Hill, Maidenhead, U.K., International Student Edition, 1997. ISBN: 0-07-115467-1.