# A Study on Text Classification: Term Weighting Algorithm Analysis

Kuan-Hua Tseng[1], Chun-Hung Richard Lin[1], Jain-Shing Liu[2], Chih-Ming Andrew Huang[1], Yue-Han Wang[1]
[1] Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan
[2] Department of Computer Science and Information Engineering, Providence University, Taiwan
ethan@mail.cse.nsysu.edu.tw, lin@cse.nsysu.edu.tw, chlliu@pu.edu.tw, andrewh232@gmail.com, wang15951@gmail.com

## Abstract

With the advancement of digital recording and storing technology, plus the huge growth of world wide web, people nowadays use digital texts instead of paper to write and record. In order to realize more text applications, the technology of text classification is gradually gaining attention recently. To achieve automatic text classification through machine learning, the related five technologies, including pre-processing, feature extraction, feature selection, term weighting and classification algorithm, are often discussed as well by many researches. In this paper, we are going to explore the impact of term weighting on text classification.

Term weighting is definitely a very important part of text classification. The calculated weight should directly reflect the importance of the term in entire text to allow machine learning to achieve the best classified result. We applied some common term weighting methods to several pre-defined datasets and conducted the experiments. Instead of intuitively considering that the value of weight represents how important it is, it turned out that the result shows the term actually may not as important as the high scored weight represents.

**Keywords:** Text classification, Term weighting, Supervised term weighting

## 1 Introduction

With the rapid development of internet, there are huge and still increasing amount of web content in texts which are exchanged between people. These unstructured web texts are everywhere: emails, instant messages, social media, web pages, and more. Web texts can be an extremely rich source of information, but due to its unstructured characteristic, it is difficult to extract useful key parts systematically from it. If we manage these textual data in a manual way, it will cost us too many materials and manpower to handle. Therefore, to manage these complicated web textual data in an effective way, is getting important. Here we utilize the machine learning text classification technology to help people automatically structure and analyze the text in a quickly and cost-effectively way [1-2].

To achieve automatic text classification, we introduced several widely used machine learning technology to predict the category of the target text. The machine learning algorithm, including Naïve-Bayes [3], Supporting Vector Machine [4], K-nearest Neighbors [5], Decision Tree [6], etc., is highly related to computational statistics. During the process of machine learning, every move we take such as feature selection, classification algorithm or weighting, can be a huge affection to the result. In order to get the best result, we must do some pre-research to optimize the learning process.

Most of the text classification cases are multi-label classification, which is an extension of multi-class classification [7]. That means one text can be categorized into one or several pre-defined classes.

In this paper, we focus on the categorization of text, not only automatically and efficiently but also higher precision. Term weighting plays the most important role of the classification process, we also want to learn about the effect of term weights to the classification process. Moreover, we want to figure out whether the high weight terms are significant or not for classification.

## 2 Related Work

### 2.1 Datasets and Framework

There are 7 datasets which are used in this paper for training and testing. They are Reuter 21578, Re0, Re1, Re52, k1a, k1b and RCV1. Table 1 shows the basic information about all datasets.

Reuter 21578 is a document collection appeared on Reuters news in 1987 [8], and is often used in text categorization. It contains multi-class and multi-label datasets with 90 categories and 10788 documents. We split them into two set for training and testing, which have 7769 documents and 3019 documents respectively.

*Corresponding Author: Kuan-Hua Tseng; E-mail: ethan@mail.cse.nsysu.edu.tw
DOI: 10.3966/160792642021032202007

**Table 1.** Base Information of Datasets

| Dataset | Documents | Effective Words | Classes |
|---|---|---|---|
| Reuters-21578 | 10788 | 9280 | 90 |
| Re0 | 1504 | 2886 | 13 |
| Re1 | 1657 | 3758 | 25 |
| Re52 | 9130 | 7977 | 52 |
| K1a | 2340 | 21839 | 20 |
| K1b | 2340 | 21839 | 6 |
| RCV1 | 804414 | 47236 | 103 |

Re0, Re1, K1a and K1b, are provided by Karypis Lab University of Minnesota.[1] The Re0 and Re1 dataset are subsets that derived from Reuter 21578. We also select some certain documents to create Re52 dataset. Re0 contains 13 categories with 1504 documents. Re1 has 25 categories with 1657 documents. Re52 is a single label subset of Reuter 21587. It contains 52 categories and 9130 documents. The k1a and k1b dataset are subsets of WebACE. They have up to a total 2340 documents and 21839 effective words.

Datasets mentioned above are small-scale datasets, that means the quantities of documents are in the range of thousands to ten thousand, and each dataset has around thousands of effective words. The number of documents and the number of effective words are not much difference hence the weight of word may be distorted due to the small term frequency. So, we introduced the Reuters Corpus Volume I (RCV1) [9], an archive of over 800000 manually categorized newswire stories made by Reuters, Ltd. RCV1 is a large-scale dataset which contains 103 categories and 804414 documents.

All these datasets are imbalanced. The skewed distribution makes many conventional machine learning algorithms less effective. Imbalanced data typically refers to the classification problem that the classes are not distributed equally. From the figures below, we can easily understand that there is no dataset with normal distribution.

Figure 1 shows that 76.2% documents of Re0 are in only three categories. Figure 2 shows that 42.3% documents of Re1 are in only two categories. Figure 3 shows that 43.6% documents of K1a are in only three categories and there are five categories which contain less than 1% documents of K1a. Figure 4 shows that 59.4% documents of K1b are in a single category. Figure 5 shows that 68.1% documents of Re52 are in only two categories. Figure 6 shows that 58.7% documents of Reuters-21578 are in only two categories. Figure 7 shows the category distribution of RCV, the CCAT category has over 380000 documents but the GMIL category only contains five documents.

[1] http://glaros.dtc.umn.edu/gkhome/cluto/cluto/download

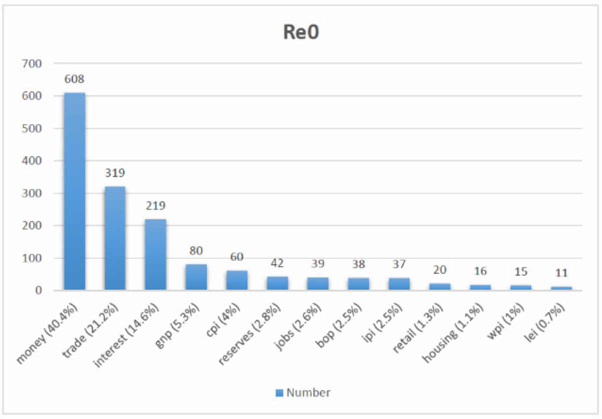


**Figure 1.** Category distribution of dataset Re0

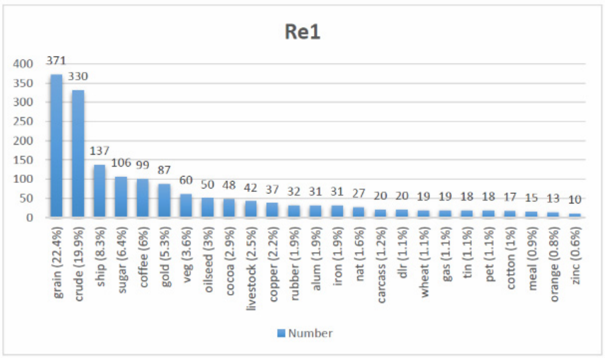


**Figure 2.** Category distribution of dataset Re1

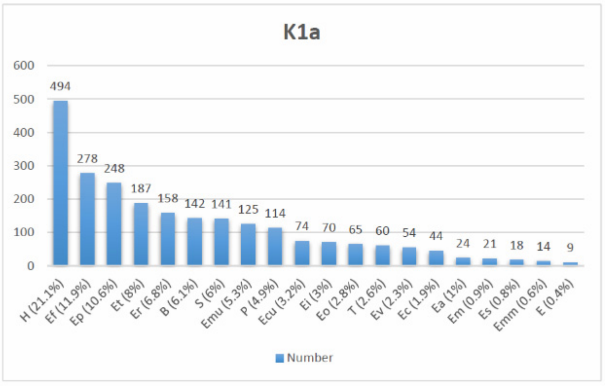


**Figure 3.** Category distribution of dataset K1a

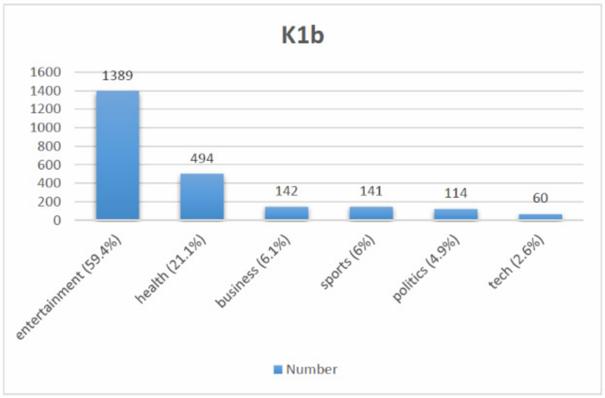


**Figure 4.** Category distribution of dataset K1b

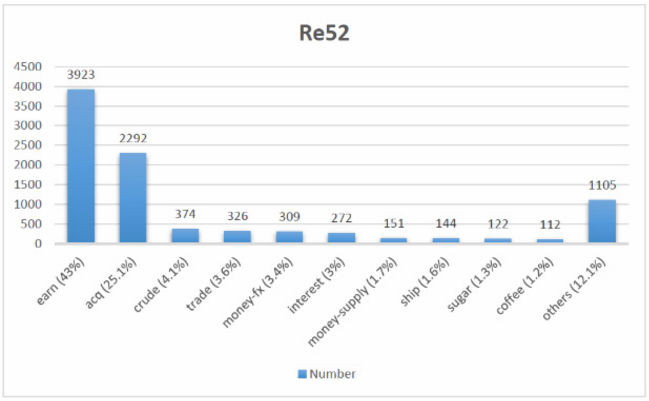


**Figure 5.** Category distribution of dataset Re52

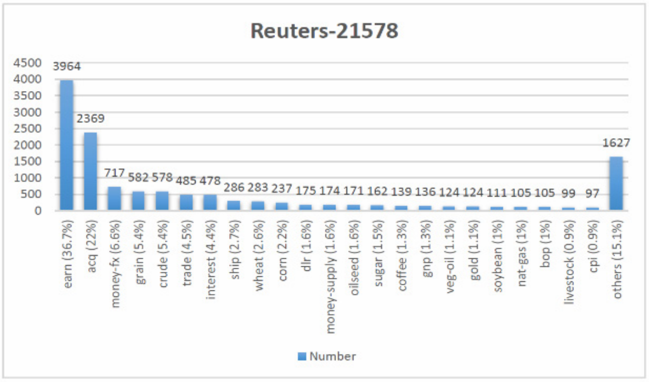


**Figure 6.** Category distribution of dataset Reuters-21578

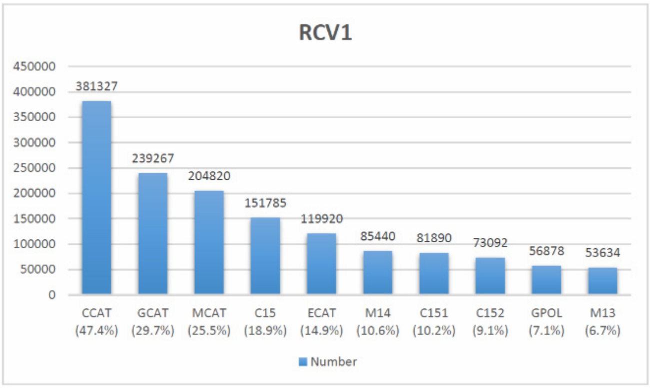


**Figure 7.** Category distribution of dataset RCV1

Figure 8 presents the framework of supervised text classification. At very first, we need to do pre-processing includes stop words removal, stemming [10] and lemmatization [11]. Then we use bag-of word model to transform words to vectors and give every term a weight. Feature reduction, including feature extraction and feature selection, is the next step. Feature extraction reduces the amount of resource which is required to describe a large set of data. Feature selection is the process of selecting a subset of relevant features that are used in model construction. After these procedures, we use training set to train the classifier models with machine learning algorithms.

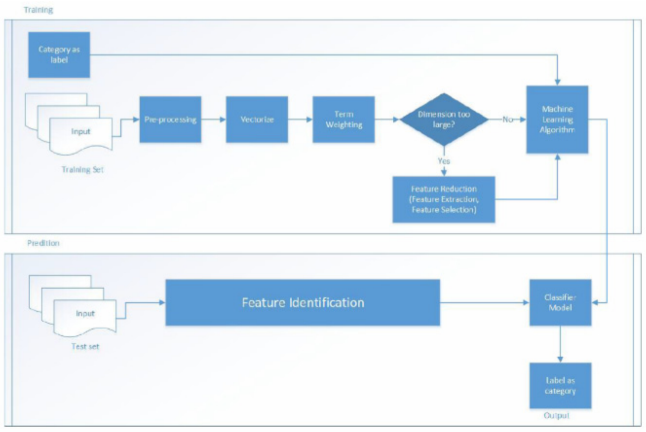


**Figure 8.** Framework for supervised text classification

For new test set, we have to do the feature identification, including picking up the features which are in the training set, and give these terms an adjusted weight. After that, we may predict the result of this new set through the classification model.

In this paper we will also introduce RFE method to figure out the relation between classification and term weights.

## 2.2 Pre-processing

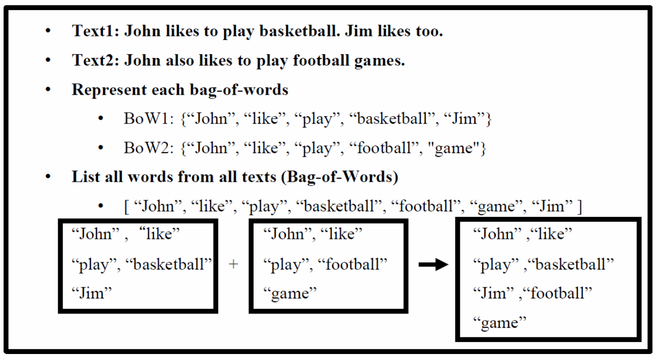
Here we use Reuters 21578 as an example to do pre-processing. We remove the stop words as well as stem words, then keep the meaningful words like nouns, verbs, adjectives, and adverbs. This can be easily done by NLTK package. There is a stop word table inside NLTK, it collects meaningless words including be (verb), etc. Not only the tense words, there are also many other words that carry the same meaning. To eliminate these words, we use stemming to shorten the lookup time and normalize the sentences. For example, ‘is’, ‘am’ and ‘are’ are three different be verbs. By stemming, they can be treated as one word ‘be’. Re0, Re1, k1a, k1b already been done this preprocessing, so we can skip this step on them.

## 2.3 Bag-of-words Model

The bag-of-words model is a way of representing words in a vector of occurrence counts of a vocabulary. It is widely used in natural language processing and information retrieval. In this model, a text is represented as a bag of its words only but disregards the grammar and the word order to keep the multiplicity. In our training set, we put all effective words into a bag which is used by the bag-of-model. Figure 9 shows the result after the bag-of-words model.

## 2.4 Term Weighting

After these previous steps, every word which is retrieved from the text will be given a weight through the weighting algorithm [12]. In this paper, we use eight different kinds of weighting algorithm to do the experiments and compare the results. Except TF,



- **Text1: John likes to play basketball. Jim likes too.**
- **Text2: John also likes to play football games.**
- **Represent each bag-of-words**
  - BoW1: {“John”, “like”, “play”, “basketball”, “Jim”}
  - BoW2: {“John”, “like”, “play”, “football”, “game”}
- **List all words from all texts (Bag-of-Words)**
  - [ “John”, “like”, “play”, “basketball”, “football”, “game”, “Jim” ]

“John” , “like” “play”, “basketball” “Jim”
+
“John”, “like” “play”, “football” “game”
“John” ,“like” “play” ,“basketball” “Jim” ,“football” “game”

**Figure 9.** An example of bag-of-words model

TFIDF and TFICF, we also introduce four additional supervised term weighting methods. Then we propose our new supervised term weighting method.

### 2.4.1 One-hot Encoding

A one-hot encoding is a representation of categorical variables as binary vectors. In natural language processing, a one-hot vector represents that a word is in the document or not. The vector consists of zero bits in all cells originally, then when the corresponding words are found in the text, these bits will mark 1 in the cells. Figure 10 shows the example of representation using one-hot encoding.

- **Text1: John likes to play basketball. Jim likes too.**
- **Text2: John also likes to play football games.**
- **Effective words from all texts**
  - [ "John", "like", "play", "basketball", "football", "game", "Jim"]
- **One-hot encoding vectors**
  - Text1: [1, 1, 1, 1, 0, 0, 1]
  - Text2: [1, 1, 1, 0, 1, 1, 0]

**Figure 10.** Example of representation using one-hot encoding

### 2.4.2 TFIDF (Term Frequency, Inverse Document Frequency)

TFIDF [13-14] is often used as a weighting algorithm in information retrieval, text mining and user modeling. It reflects how important a word to a document in a collection or corpus is. This algorithm is combined with TF part and IDF part.

TF (Term Frequency) means that the number of occurrences of a term in a document, and is simply proportional to the term frequency. In TF formula, ni,j is the number of the specify words that occurred in document dj, and the denominator denotes the number of all words that occurred in the document dj.

IDF (Inverse Document Frequency) represents a specified term factor that is quantified by the inverse function and the number of occurrences in documents. In the formula, |D| is the total number of documents in the corpus, the denominator is the number of documents that contains this specified term. To avoid a division-by-zero error, it’s common to adjust the denominator to 1+|{j:t$_i$ ∈ d$_j$}|. Figure 11 shows an example of representation using TFIDF.

- **Text1: John likes to play basketball. Jim likes too.**
- **Text2: John also likes to play football games.**
- **Effective words from all texts**
  - [ "John", "like", "play", "basketball", "football", "game", "Jim" ]
- **tf**
  - Text1: [1, 2, 1, 2, 0, 0, 1]
  - Text2: [1, 1, 1, 0, 1, 1, 0]
- **idf**
  - $[\log(\frac{2}{1+2}), \log(\frac{2}{1+2}), \log(\frac{2}{1+2}), \log(\frac{2}{1+1}), \log(\frac{2}{1+1}), \log(\frac{2}{1+1}), \log(\frac{2}{1+1})]$
- **Represent as vectors using TFIDF (tf * idf)**
  - Text1: [-0.176, -0.352, -0.176, -0.352, 0, 0, -0.176]
  - Text2: [-0.176, -0.176, -0.176, 0, -0.176, -0.176, 0]

**Figure 11.** Example of representation using TFIDF

$$tf_{i,j} = \frac{n_{i,j}}{\Sigma_k n_{k,j}} \tag{1}$$

$$idf_i = \log \frac{|D|}{|\{j : t_i \in d_j\}|} \tag{2}$$

$$tfidf_{i,j} = tf_{i,j} * idf_i \tag{3}$$

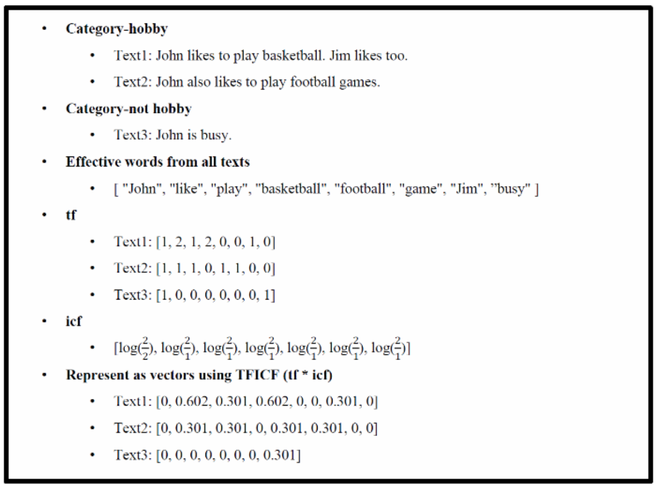
### 2.4.3 TFICF (Term Frequency, Inverse Category Frequency)

TFICF [15-17] is often used as a weighting algorithm factor too in information retrieval, text mining. In general speaking of text categorization, the fewer a term appears in one category, the more discriminative power this term has. This algorithm is combined by TF part and ICF part. The definition of TF here is the same as described previously.

ICF (Inverse Category Frequency) represents a specified term factor that can be quantified by the inverse function and the number of occurrences in categories. In the ICF formula, |C| is the total number of categories in the corpus, the denominator is the number of categories that contains this specified term. Figure 12 shows the example of representation using TFICF.

$$icf(ti) = \log(\frac{|C|}{cf(ti)}) \tag{4}$$

$$tficf_{i,j} = tf_{i,j} * icf_i \tag{5}$$



- **Category-hobby**
  - Text1: John likes to play basketball. Jim likes too.
  - Text2: John also likes to play football games.
- **Category-not hobby**
  - Text3: John is busy.
- **Effective words from all texts**
  - [ "John", "like", "play", "basketball", "football", "game", "Jim", "busy" ]
- **tf**
  - Text1: [1, 2, 1, 2, 0, 0, 1, 0]
  - Text2: [1, 1, 1, 0, 1, 1, 0, 0]
  - Text3: [1, 0, 0, 0, 0, 0, 0, 1]
- **icf**
  - $[\log(\frac{2}{2}), \log(\frac{2}{1}), \log(\frac{2}{1}), \log(\frac{2}{1}), \log(\frac{2}{1}), \log(\frac{2}{1}), \log(\frac{2}{1})]$
- **Represent as vectors using TFICF (tf * icf)**
  - Text1: [0, 0.602, 0.301, 0.602, 0, 0, 0.301, 0]
  - Text2: [0, 0.301, 0.301, 0, 0.301, 0.301, 0, 0]
  - Text3: [0, 0, 0, 0, 0, 0, 0, 0.301]

**Figure 12.** Example of representation using TFICF

### 2.4.4 Supervised Term Weighting

Supervised term weighting (STW) has been used in text classification for several years. Before the STW, we usually use binary classification for text classification. The STW becomes popular these years because it considers the characteristics of dataset and uses the prior information on training documents in predefined categories [18].

A traditional weight algorithm consists of a local weight and a global weight like previously mentioned TFIDF, which is the most commonly used one in text classification. STW uses the weight after feature-selected process to replace the global weight. Table 2 lists the fundamental information elements which are used for feature selection in text classification.

**Table 2.** Fundamental information elements

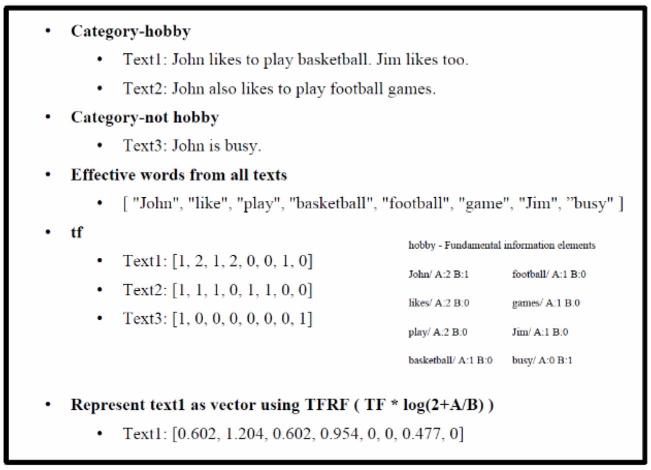
| | $C1$ | $\overline{C_1}$ |
|---|---|---|
| $t_k$ | $A$ | $B$ |
| $\overline{t_k}$ | $C$ | $D$ |

- A denotes the number of documents in the positive class that not contain term $t_k$ .
- B denotes the number of documents in the positive class that not contain term $t_k$ .
- C denotes the number of documents in the positive class that not contain term $t_k$ .
- D denotes the number of documents in the negative class that do not contain $t_k$ .
- The sum of A, B, C and D is the number of documents in the whole collection.

In text classification, there are several supervised term weighting algorithms except the traditional TF-like method. Table 3 lists four supervised term weighting methods we used in this paper including TFOdd [19], TFProb [20] and TFRF [21-22]. Figure 13 shows an example of representation using TFRF.

**Table 3.** STWs using fundamental infotmation element

| Methods | Mathematical form represented by information elements |
|---|---|
| ntf * Chi-square (ChiS) | $ntf * N(AD-BC)^2 /(A+C)(A+B)(B+D)(C+D)$ |
| ntf * Odd ratio (OddsR) [19] | $ntf * \log(AD/BC)$ |
| Probability based term weight (Prob.) [20] | $ntf * \log(1+\frac{A}{B} * \frac{A}{C})$ |
| Relevance frequency (rf) [21-22] | $ntf * \log(2+\frac{A}{B})$ |



- **Category-hobby**
  - Text1: John likes to play basketball. Jim likes too.
  - Text2: John also likes to play football games.
- **Category-not hobby**
  - Text3: John is busy.
- **Effective words from all texts**
  - [ "John", "like", "play", "basketball", "football", "game", "Jim", "busy" ]
- **tf**
  - Text1: [1, 2, 1, 2, 0, 0, 1, 0]
  - Text2: [1, 1, 1, 0, 1, 1, 0, 0]
  - Text3: [1, 0, 0, 0, 0, 0, 0, 1]

hobby - Fundamental information elements

| | |
|---|---|
| John/ A:2 B:1 | football/ A:1 B:0 |
| likes/ A:2 B:0 | games/ A:1 B:0 |
| play/ A:2 B:0 | Jim/ A:1 B:0 |
| basketball/ A:1 B:0 | busy/ A:0 B:1 |

- **Represent text1 as vector using TFRF ( TF * log(2+A/B) )**
  - Text1: [0.602, 1.204, 0.602, 0.954, 0, 0, 0.477, 0]

**Figure 13.** Example of representing in TFRF

## 2.5 Our Proposed Term Weighting Method

We newly propose a supervised term weighting scheme, Term Frequency-Category Relevance Frequency (TFCRF), which uses the odds of positive and negative class probabilities to improve results. Table 4 lists the Mathematical formula of TFCRF. This formula is used to improve probability-based term weight [20].

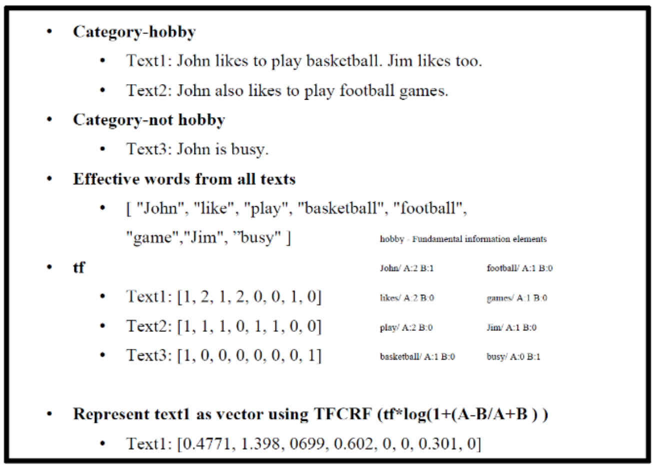
**Table 4.** The formula of proposed STW: TFCRF

| Methods | Mathematical form represented by information elements |
|---|---|
| TFCRF | $ntf * \log(1+\frac{A-B}{A+B})$ |

Probability based term weight considers fundamental information elements B and C. Element B is the number of documents which contain term $t_k$ but in the negative $C_i$ classes. Element C is the number of documents which don't have the term $t_k$ but in the positive $C_i$ class.

Considering that element C should not have a negative impact to the discriminatory power of term $t_k$ to class $C_i$, it can even be said to be irrelevant, we decide to treat element C as an insignificant factor, so we don't put C in our formula.

We take element B as the real matter to affect the term weighting result. Since A + B is the total appearance number of term $t_k$, we can say that A and B are mutually exclusive. That means when A gains more, the less B will have. In our proposed novel supervised term weighting method, we will treat B as a negative factor to the discriminating power of term $t_k$. Figure 14 shows the example of representation using TFCRF.



- **Category-hobby**
  - Text1: John likes to play basketball. Jim likes too.
  - Text2: John also likes to play football games.
- **Category-not hobby**
  - Text3: John is busy.
- **Effective words from all texts**
  - [ "John", "like", "play", "basketball", "football", "game","Jim", "busy" ]
- **tf**
  - Text1: [1, 2, 1, 2, 0, 0, 1, 0]
  - Text2: [1, 1, 1, 0, 1, 1, 0, 0]
  - Text3: [1, 0, 0, 0, 0, 0, 0, 1]

hobby - Fundamental information elements

| | |
|---|---|
| John/ A:2 B:1 | football/ A:1 B:0 |
| likes/ A:2 B:0 | games/ A:1 B:0 |
| play/ A:2 B:0 | Jim/ A:1 B:0 |
| basketball/ A:1 B:0 | busy/ A:0 B:1 |

- **Represent text1 as vector using TFCRF (tf*log(1+(A-B/A+B ) )**
  - Text1: [0.4771, 1.398, 0699, 0.602, 0, 0, 0.301, 0]

**Figure 14.** Example of representing in TFCRF

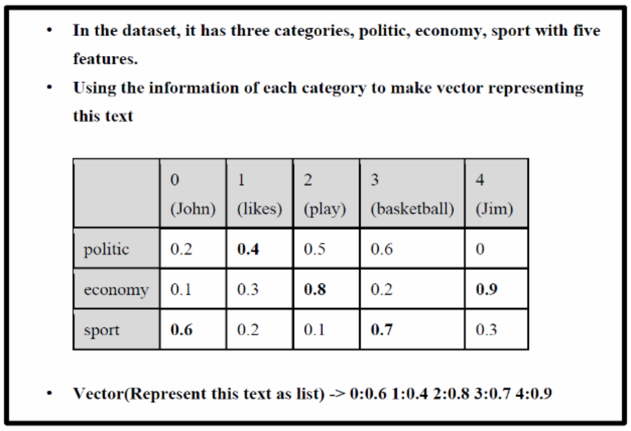
## 2.6 Test Documents Representation Method

How to represent our test documents with supervised term weighting is our next issue. Since there is no class information of test documents, we have to develop one to represent the test document in vector which is required by the team weighting scheme. We introduced four methods [23] here and described them in Table 5. Each method will be given an example. Figure 15 shows the example of W-Max method. Figure 16 shows the example of D-Max method. Figure 17 shows the example of D-TMax method. And the last Figure 18 shows the example of Hypo method.

**Table 5.** Four representation methods for test document

| | |
|---|---|
| W-Max (Word Max) | The term weight of each word is chosen based on the maximum value among \|C\| estimated term weights. |
| D-Max (Document Max) | The sum of all term weights in each vector is first calculated, and one vector with the maximum sum value is then selected as a representative vector. |
| D-TMax (Documents Two Max) | The sum of all term weights in each vector is calculated, and two vectors with the highest and second highest sum values are then selected. A vector is then created by choosing the term weight with the higher score between the two term weights of the selected vectors for each term. |
| Hypo (Hypothesis) | First we generated the \|C\| vectors according to the information of each class. Then we treat each category of the test document as a hypothesis label, that means all the test documents can be categorized to one or many of these labels. A text classifier is introduced here to calculate the prediction scores of all categories. Finally, we select the label with the highest prediction score as the predicted category label. |

**Table 5.** Four representation methods for test document (continue)

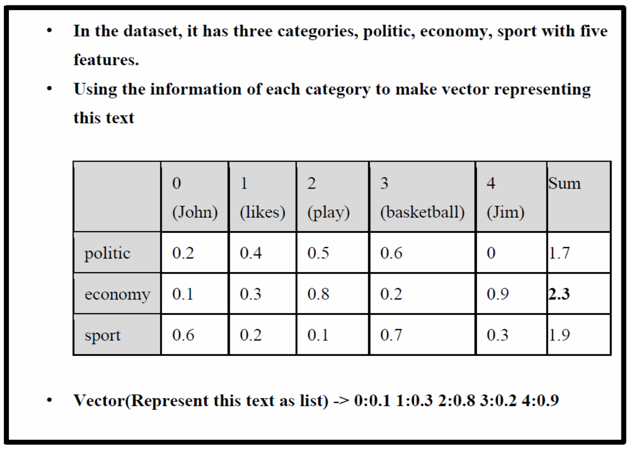


- **In the dataset, it has three categories, politic, economy, sport with five features.**
- **Using the information of each category to make vector representing this text**

| | 0 (John) | 1 (likes) | 2 (play) | 3 (basketball) | 4 (Jim) |
|---|---|---|---|---|---|
| politic | 0.2 | **0.4** | 0.5 | 0.6 | 0 |
| economy | 0.1 | 0.3 | **0.8** | 0.2 | **0.9** |
| sport | **0.6** | 0.2 | 0.1 | **0.7** | 0.3 |

- **Vector(Represent this text as list) -> 0:0.6 1:0.4 2:0.8 3:0.7 4:0.9**

**Figure 15.** Example of W-Max method

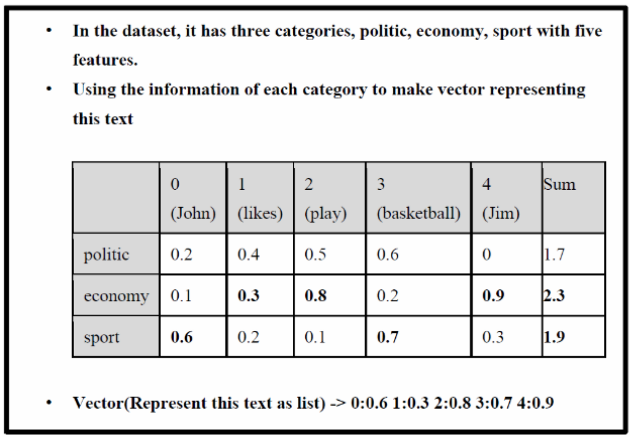


- **In the dataset, it has three categories, politic, economy, sport with five features.**
- **Using the information of each category to make vector representing this text**

| | 0 (John) | 1 (likes) | 2 (play) | 3 (basketball) | 4 (Jim) | Sum |
|---|---|---|---|---|---|---|
| politic | 0.2 | 0.4 | 0.5 | 0.6 | 0 | 1.7 |
| economy | 0.1 | 0.3 | 0.8 | 0.2 | 0.9 | **2.3** |
| sport | 0.6 | 0.2 | 0.1 | 0.7 | 0.3 | 1.9 |

- **Vector(Represent this text as list) -> 0:0.1 1:0.3 2:0.8 3:0.2 4:0.9**

**Figure 16.** Example of D-Max method

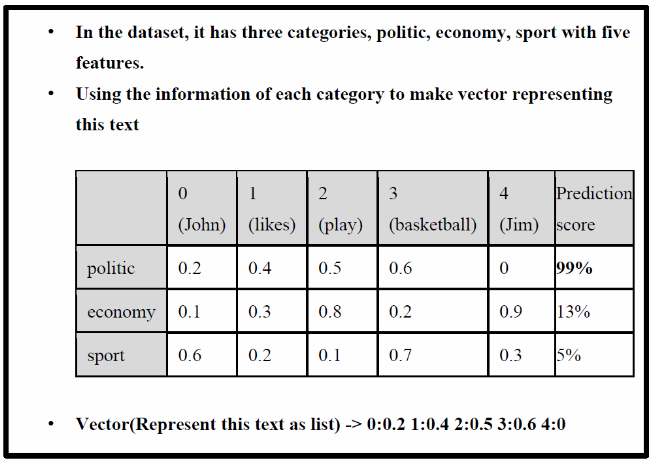


- **In the dataset, it has three categories, politic, economy, sport with five features.**
- **Using the information of each category to make vector representing this text**

| | 0 (John) | 1 (likes) | 2 (play) | 3 (basketball) | 4 (Jim) | Sum |
|---|---|---|---|---|---|---|
| politic | 0.2 | 0.4 | 0.5 | 0.6 | 0 | 1.7 |
| economy | 0.1 | **0.3** | **0.8** | 0.2 | **0.9** | **2.3** |
| sport | **0.6** | 0.2 | 0.1 | **0.7** | 0.3 | **1.9** |

- **Vector(Represent this text as list) -> 0:0.6 1:0.3 2:0.8 3:0.7 4:0.9**

**Figure 17.** Example of D-TMax method



- **In the dataset, it has three categories, politic, economy, sport with five features.**
- **Using the information of each category to make vector representing this text**

| | 0 (John) | 1 (likes) | 2 (play) | 3 (basketball) | 4 (Jim) | Prediction score |
|---|---|---|---|---|---|---|
| politic | 0.2 | 0.4 | 0.5 | 0.6 | 0 | **99%** |
| economy | 0.1 | 0.3 | 0.8 | 0.2 | 0.9 | 13% |
| sport | 0.6 | 0.2 | 0.1 | 0.7 | 0.3 | 5% |

- **Vector(Represent this text as list) -> 0:0.2 1:0.4 2:0.5 3:0.6 4:0**

**Figure 18.** Example of Hypo method

## 2.7 Feature Reduction

Feature reduction, also known as dimensionality reduction, is the process to reduce the feature size without losing important information. Feature reduction can be divided into two processes: feature selection and feature extraction. Feature selection returns a subset of relevant features from a large dataset, whereas feature extraction creates a new feature set which is reduced to a more manageable size for processing.

### 2.7.1 Feature Selection

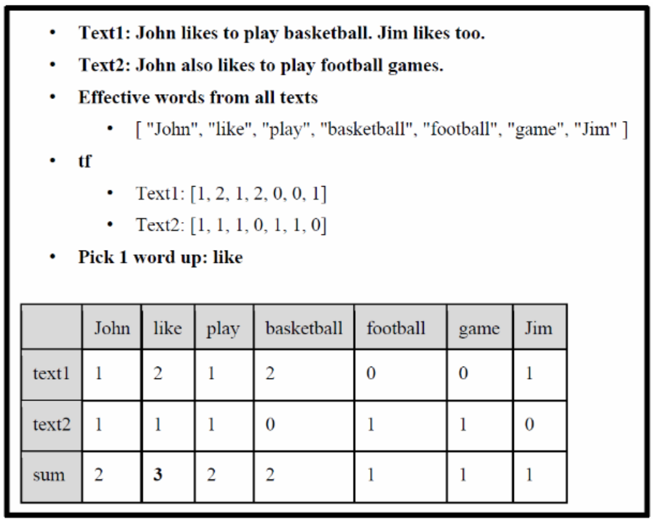
Feature selection is the process of selecting relevant features from raw dataset. This technique is useful because it simplifies the learning models and results, and make them easier to interpret by researchers. It also has shortened training time and enhanced generalization by reducing overfitting. There are three different kinds of method: wrapper method, filter method and embedded method, for the implementation of feature selection [24].

Wrapper methods use predictive models to score the features in subsets. These subsets will be used to train a model, and be tested on another hold-out set. By repeating this process, we can find best score one among the subsets. The wrapper methods are computationally intensive and may cost a lot of computational resource, but also give us the best performance dataset.

Filter methods use proxy measures instead of the error rate to score the features in subsets. These measures are chosen according to their simplicity of computing, and the usefulness to the feature sets. There are several common measures such as the chi-squared stats and the mutual information. The filter method is often less computationally intensive than the wrapper one, but provides the feature set which may not tuned to a specific type compare to predictive model. It often gives a lower prediction performance than wrapper.

Embedded methods perform feature selection as a part of the machine learning model construction process. Therefore, we call them embedded methods. Common embedded methods include LASSO method and RFE method. In terms of computational complexity, the embedded methods are between wrappers and filters.

Reuters-21578 has 9280 features and 10788 documents. We can easily aware that there are too many features and documents which is not efficient to classify. In order to reduce the computational power, we introduced the simple feature selection to reduce the number of features. By that we choose 4000 features which have higher term frequency value for experiment. Figure 19 shows the example of feature selection using TF.



- **Text1: John likes to play basketball. Jim likes too.**
- **Text2: John also likes to play football games.**
- **Effective words from all texts**
  - [ "John", "like", "play", "basketball", "football", "game", "Jim" ]
- **tf**
  - Text1: [1, 2, 1, 2, 0, 0, 1]
  - Text2: [1, 1, 1, 0, 1, 1, 0]
- **Pick 1 word up: like**

| | John | like | play | basketball | football | game | Jim |
|---|---|---|---|---|---|---|---|
| text1 | 1 | 2 | 1 | 2 | 0 | 0 | 1 |
| text2 | 1 | 1 | 1 | 0 | 1 | 1 | 0 |
| sum | 2 | **3** | 2 | 2 | 1 | 1 | 1 |

**Figure 19.** Example of feature selection with TF

### 2.7.2 Feature Extraction

If the input data is too large to be used in machine learning, then we transform it into a reduced set of features. Here we introduced the feature extraction to reduce the amount of required resources for describing a large dataset.

A large size of variables may cause classification algorithm overfits on training samples and is generalize poorly to new samples. Feature extraction can be implemented by many methods such as LSA (Latent Semantic Analysis) and PCA (Principal Content Analysis). We use PCA plus logistic regression to do the experiment in this paper. There are two advantages using PCA as our major feature extraction method. First, it reduces the processing dimension and because of this, it saves the model training time. The second is solving the collinearity problem and pick up the features which are independent. The collinearity problem happens when independent variables in a regression model are too much correlated. Independent variables should be just as the name itself says, if the level of correlation between variables is too high, it will be a problem when user is trying to fit the model or interpreting the result.

Principal components analysis (PCA) [25] is a statistical process. It uses the orthogonal transformation to convert a set of observation value from possibly correlated variables into a set of linearly uncorrelated variables which are called principal components. The transformation is defined in such way that the first principal component has the largest possible variance, and then following by each succeeding component in turn. These succeeding components have the highest variance possible under the constraint that they are orthogonal to their preceding components. The resulting vectors are an uncorrelated orthogonal basis set.

In this paper, by the tremendous help of PCA, we reduced the dimensions to the level that the percentage of variance explained is over 90%. Table 6 shows the number of reduced dimensions with PCA. After extracting features, we use logistic regression to classify.

**Table 6.** Reduced dimension by PCA

| Dataset | | Re0 | Re1 | K1a | K1b | Re52 | Reuters-21578 |
|---|---|---|---|---|---|---|---|
| Dimension | | 2886 | 3758 | 21839 | 21839 | 7977 | 4000 |
| Reduced Dimension | TF | 233 | 400 | 864 | 864 | 750 | 711 |
| | TFIDG | 493 | 723 | 1284 | 1284 | 1765 | 1420 |
| | TFICF | 342 | 461 | 955 | 955 | 1229 | 1038 |
| | TFChi | 4 | 6 | 1 | 1 | 10 | 11 |
| | TFOdd | 177 | 256 | 786 | 786 | 498 | 421 |
| | TFRF | 294 | 424 | 883 | 883 | 368 | 339 |
| | TFProb | 13 | 14 | 20 | 20 | 17 | 20 |

## 2.8 Recursive Feature Elimination (RFE)

In this paper, we analysis the relation between term weighting and classifying. Recursive feature elimination (RFE) [26] is involved here. RFE picks feature up in classification step which is different from other feature selection methods.

Recursive feature elimination (RFE) is a feature selection algorithm. It takes the advantage of reducing the redundant and recursive features and can be used in many different machine learning classification algorithms. The Major concept of RFE is sorting out the influence of features by excluding the features that have the least influence on the target. The estimator is first trained by the initial dataset, then the feature who has the smallest weight are removed from the set. By repeating this procedure, the desired size of feature will be reached. Figure 20 depicts the framework of RFE.

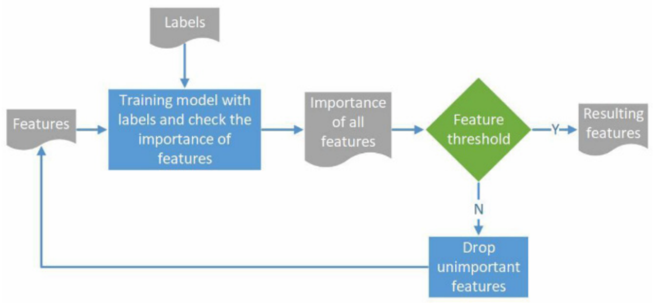

Labels
Features
Training model with labels and check the importance of features
Importance of all features
Feature threshold
Y
Resulting features
N
Drop unimportant features

**Figure 20.** RFE Framework

## 2.9 F1-measure

There are many ways to evaluate a model such as accuracy, error rate, precision, recall and F1-measure. They are widely used in the machine learning evaluation but sometimes we may not easy to distinguish good model from other models by them. Here is an example. If we have a model to predict earthquake, and it has 99% accuracy (TP/TP+TN) score but due to the near zero frequency of earthquake, it is very easy to design the model which predicts no earthquake at all. In this case, although the accuracy is very high, we can't say that it is a good model.

Then the precision and recall are developed for another aspect of evaluation. Precision focuses on true positive of predicted positive. Recall focuses on true positive of actual positive. But still, in extreme situations, they are both insufficient to determine whether the model is good or not. How do we explain the significance of a model with high precision and low recall and a model with high recall and low precision, respectively? The former one can be regarded as a more cautious model. Although it is not often to predict positive entities, but as long as there is a predicted positive, it is almost correct (Precision high), while the latter one is a loose model, although sometimes it predicts the wrong result, but almost everything that should be predicted positive are actually predicted positive (Recall high).

Now, the F1-measure are designed to consider both. F1-measure [27] is a measure of statistical analysis of binary classification. It considers both the precision and the recall. Table 7 lists four results of classification in Confusion Matrix. Precision is the rate that the number of true positive divides by the total number of predicted positive. Recall is the rate that the number of true positive divides by the total number of actual positive. We can use precision and recall to calculate F1-measure. The macro-average gives weight equally to all the classes, that means it is an arithmetic mean of the F1-scores of all classes. The micro-average gives weight equally to all the texts, it simply looks at all the classes together. In this paper, we use micro F1 and macro F1 to evaluate the results.

**Table 7.** Confusion Matrix

| | Condition Positive | Condition Negative |
|---|---|---|
| Predicted Condition Positive | TP (True Positives) | FP (False Positives) |
| Predicted Condition Negative | FN (False Negatives) | TN (True Negatives) |

$$\text{Precision: } \frac{TP}{TP+FP} \quad \textbf{(6)}$$

$$\text{Recall: } \frac{TP}{TP+FN} \quad \textbf{(7)}$$

$$\text{F1- measure: } \frac{2PR}{P+R} \quad \textbf{(8)}$$

$$\text{Macro Precision: } \frac{1}{n}\sum_{i=1}^{n} R_i \quad \textbf{(9)}$$

$$\text{Macro Recall: } \frac{1}{n}\sum_{i=1}^{n} R_i \quad \textbf{(10)}$$

$$\text{Macro F1- measure: } \frac{1}{n}\sum_{i=1}^{n} F_i \quad \textbf{(11)}$$

$$\text{Micro Precision: } \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FP_i} \quad \textbf{(12)}$$

$$\text{Micro Recall: } \frac{\sum_{i=1}^{n} TP_i}{\sum_{i=1}^{n} TP_i + \sum_{i=1}^{n} FN_i} \quad \textbf{(13)}$$

$$\text{Micro F1- measure: } \frac{2 * \mathit{Micro\ Precision} * \mathit{Micro\ Recall}}{\mathit{Micro\ Precision} + \mathit{Micro\ Recall}} \quad \textbf{(14)}$$

## 3 Results and Discussion

In this section, we implemented eight term weighting methods to the predefined six datasets as previously mentioned. The results are shown in section 3.1 and followed by the analysis and discussion part in section 3.2.

### 3.1 Results

All the datasets have already been dealt with pre-processing and feature extraction so we can just apply the term weighting methods and give each feature a weight.

Following tables are the results of all datasets under SVM and logistic regression with one-hot encoding, the traditional and the supervised term weighting methods.

#### 3.1.1 Re0

We found that in Re0, TFRF gets a better macro-F1 score and TFIDF gets a better micro-F1 score under SVM classification as Table 8 Shows. When using PCA and logistic regression, TFRF gets better macro-F1 score and TFProb gets a better micro-F1 score. The result tells TFRF and TFProb get better performance. Its fundamental information elements, A, B, C, provide much more useful information for classifying. TFIDF gets a not bad result, which shows factor term frequency and document frequency are helpful in Re0. Although there are also many other algorithms that may provide us more information, but through the experiment we find that these formulas don't bring us better results in this dataset.

**Table 8.** Result of Re0

| Method | PCA+logistic regression | | SVM | |
|---|---|---|---|---|
| | macro-F1 | micro-F1 | macro-F1 | micro-F1 |
| One-hot encoding (Base) | 76.93% | 85.24% | 76.26% | 83.31% |
| TF | 73.88% (-3.05%) | 84.11% (-1.13%) | 75.06% (+1.2%) | 83.25% (-0.06%) |
| TFIDF | 61.64% (-15.29%) | 81.85% (-3.39%) | 83.00% (+6.74%) | 87.76% (+4.45%) |
| TFICF | 66.60% (-10.33%) | 78.66% (-6.58%) | 73.83% (-2.43%) | 78.92% (-4.39%) |
| TFChi | 75.51% (-1.42%) | 82.51% (-2.73%) | 69.82% (-6.44%) | 74.20% (-9.11%) |
| TFOdd | 77.80% (+0.87%) | 81.58% (-3.66%) | 73.30% (-2.96%) | 72.54% (-10.77%) |
| TFProb | 39.95% (-40.98%) | 92.80% (+7.56%) | 49.51% (-26.75%) | 74.93% (-8.38%) |
| TFRF | 75.92% (-1.01%) | 86.30% (+1.06%) | 83.15% (+6.89%) | 86.17% (+2.86%) |
| TFCRF | 61.99% (-14.94%) | 82.58% (-2.66%) | 71.92% (-4.34%) | 81.98% (-1.33%) |

#### 3.1.2 Re1

Table 9 shows the Re1 results of using TFRF under SVM gets the best macro-F1 score, and micro-F1 score as well. After applying PCA and logistic regression, we found that TFRF gets a better macro-F1 score and

TFRF again gets a better micro-F1 score. According to the foregoing results, TFRF gets best performance. It tells fundamental information elements A and B are much more useful for classifying in k1a. Although they are helpful by TFRF here, their importance is reduced in other methods.

**Table 9.** Result of Re1

| Method | PCA+logistic regression | | SVM | |
|---|---|---|---|---|
| | macro-F1 | micro-F1 | macro-F1 | micro-F1 |
| One-hot encoding (Base) | 69.72% | 84.67% | 70.43% | 83.77% |
| TF | 75.04% (+5.32%) | 84.54% (+1.87%) | 71.61% (+1.18%) | 83.04% (-0.73%) |
| TFIDF | 50.98% (-18.74%) | 77.43% (-2.76%) | 72.23% (+1.80%) | 86.06% (+2.29%) |
| TFICF | 78.01% (+8.29%) | 86.96% (+2.29%) | 75.18% (+4.75%) | 83.16% (-0.61%) |
| TFChi | 73.01% (+3.29%) | 83.83% (-0.84%) | 74.38% (+3.95%) | 81.11% (-2.66%) |
| TFOdd | 79.33% (+9.61%) | 87.45% (+2.78%) | 75.10% (+4.67%) | 86.42% (+2.65%) |
| TFProb | 36.61% (-33.11%) | 68.13% (-16.54%) | 70.72% (+0.29%) | 83.34% (-0.43%) |
| TFRF | 77.37% (+7.65%) | 88.89% (+4.22%) | 75.81% (+5.38%) | 87.81% (+4.04%) |
| TFCRF | 72.84% (+3.12%) | 87.14% (+2.47%) | 76.71% (+6.28%) | 87.87% (+4.10%) |

### 3.1.3 K1a

Table 10 shows the K1a results of using TFRF under SVM gets the best macro-F1 score and micro-F1 score as well. After applying PCA and logistic regression, we found that TFRF gets a better macro-F1 score and TFRF again gets a better micro-F1 score. According to the foregoing results, TFRF gets best performance. It tells fundamental information elements A and B are much more useful for classifying in k1a. Although they are helpful by TFRF here, their importance is reduced in other methods.

**Table 10.** Result of K1a

| Method | PCA+logistic regression | | SVM | |
|---|---|---|---|---|
| | macro-F1 | micro-F1 | macro-F1 | micro-F1 |
| One-hot encoding (Base) | 76.93% | 85.24% | 76.2% | 83.31% |
| TF | 71.92% (-5.01%) | 86.45% (+1.21%) | 72.78% (-3.48%) | 87.31% (+4.0%) |
| TFIDF | 52.29% (-24.64%) | 79.36% (-5.55%) | 68.41% (-7.85%) | 85.81% (+2.50%) |
| TFICF | 63.43% (-13.50%) | 82.35% (-2.89%) | 60.55% (-15.71%) | 78.16% (-5.15%) |
| TFChi | 64.42% (-12.51%) | 79.91% (-5.33%) | 64.13% (-12.13%) | 74.86% (-8.48%) |
| TFOdd | 67.33% (-9.60%) | 82.65% (-2.59%) | 45.13% (-30.98%) | 67.05% (-16.26%) |
| TFProb | 35.76% (-41.17%) | 56.62% (-28.62%) | 45.28% (-36.90%) | 59.66% (-23.65%) |
| TFRF | 71.95% (-4.98%) | 87.86% (+2.62%) | 78.08% (+1.82%) | 88.72% (+5.41%) |
| TFCRF | 65.66% (-11.27%) | 85.73% (+0.49%) | 72.08% (-4.18%) | 86.03% (+2.72%) |

### 3.1.4 K1b

Table 11 shows the K1b result of using TFICF under SVM gets the best macro-F1 score and TFRF gets the best micro-F1 score. After applying PCA and logistic regression, we found that TFOdd gets the best macro-F1 score and TFICF again gets the best micro-F1 score. According to the foregoing results, TFRF, TFICF and TFOdd have better performance to this dataset. We can say that fundamental information elements and factor category frequency are very useful for classifying in K1b.

**Table 11.** Result of K1b

| Method | PCA+logistic regression | | SVM | |
|---|---|---|---|---|
| | macro-F1 | micro-F1 | macro-F1 | micro-F1 |
| One-hot encoding (Base) | 69.72% | 84.67% | 70.43% | 83.77 |
| TF | 9359% (+23.87%) | 97.52% (+12.85%) | 95.08% (+24.65%) | 97.52% (+13.75%) |
| TFIDF | 68.39% (-1.33%) | 89.96% (+5.29%) | 87.27% (+16.84%) | 95.94% (+12.170%) |
| TFICF | 95.75% (+26.03%) | 98.25% (+13.58%) | 96.98% (+26.55%) | 98.08% (+14.31%) |
| TFChi | 90.69% (+20.97%) | 95.04% (+10.37%) | 87.55% (+17.12%) | 89.23% (+5.46%) |
| TFOdd | 97.56% (+27.84%) | 95.38% (+10.71%) | 90.02% (+19.59%) | 95.60% (+11.83%) |
| TFProb | 48.57% (-21.15%) | 84.06% (-0.61%) | 51.92% (-18.51%) | 82.95% (-0.82%) |
| TFRF | 96.07% (+26.35%) | 98.16% (+13.49%) | 96.90% (+26.47%) | 98.63% (+14.86%) |
| TFCRF | 87.78% (+18.06%) | 94.91% (+10.24%) | 67.61% (-2.82%) | 88.25% (+4.48%) |

### 3.1.5 Re52

Table 12 shows the Re52 result of using TFRF under SVM gets both the best macro-F1 score and micro-F1 score. As the result of PCA and logistic regression, TFICF gets both the best macro-F1 score and micro-F1 score. According to the foregoing results, TFRF and TFICF performed well in this dataset. We can say that fundamental information elements and factor category frequency are very helpful for classifying in Re52.

**Table 12.** Result of Re52

| Method | PCA+logistic regression | | SVM | |
|---|---|---|---|---|
| | macro-F1 | micro-F1 | macro-F1 | micro-F1 |
| One-hot encoding (Base) | 11.07% | 18.15% | 11.39% | 17.83% |
| TF | 64.83% (+56.76%) | 93.15% (+75.0%) | 66.59% (+55.20%) | 91.25% (+73.42%) |
| TFIDF | 39.73% (+28.66%) | 87.67% (+69.52%) | 66.85% (+54.46%) | 93.15% (+75.32%) |
| TFICF | 74.16% (+63.09%) | 94.20% (+76.05%) | 67.40% (+56.01%) | 89.81% (+71.98%) |
| TFChi | 17.25% (+6.18%) | 46.58% (+28.43%) | 46.26% (+34.87%) | 82.94% (+65.11%) |
| TFOdd | 68.84% (+57.77%) | 89.72% (+71.57%) | 66.14% (+54.75%) | 85.21% (+67.38%) |
| TFProb | 14.80% (+3.73%) | 76.46% (+58.31%) | 49.10% (+37.62%) | 83.89% (+66.06%) |
| TFRF | 54.66% (+43.59%) | 92.57% (+74.42%) | 68.99% (+57.60%) | 93.27% (+75.44%) |
| TFCRF | 23.12% (+12.05%) | 84.16% (+66.01%) | 68.82% (+57.43%) | 91.87% (+74.04%) |

### 3.1.6 Reuters-21578

Table 13 shows the Reuters-21578 result of using TFRF under SVM gets the best macro-F1 score and TFICF gets the best micro-F1 score. After applying PCA and logistic regression, TFOdd now gets the best macro-F1 score and TFRF gets the best micro-F1 score. According to the foregoing results, TFRF, TFOdd and TFICF work well in this dataset. We can say that fundamental information elements and factor category frequency are very helpful for classifying in Reuters-21758.

**Table 13.** Result of Reuters-21578

| Method | PCA+logistic regression | | SVM | |
|---|---|---|---|---|
| | macro-F1 | micro-F1 | macro-F1 | micro-F1 |
| One-hot encoding (Base) | 6.39% | 16.60% | 8.86% | 16.79% |
| TF | 42.74% (+36.35%) | 85.41% (+68.81%) | 45.83% (+36.97%) | 86.16% (+69.37%) |
| TFIDF | 16.86% (+10.47%) | 77.67% (+61.07%) | 49.89% (+41.03%) | 86.86% (+70.07%) |
| TFICF | 46.84% (+40.45%) | 85.46% (+68.86%) | 42.10% (+33.24%) | 87.00% (+70.21%) |
| TFChi | 11.25% (+4.86%) | 36.68% (+17.08%) | 17.04% (+8.18%) | 31.52% (+14.73%) |
| TFOdd | 56.52% (+50.13%) | 80.91% (+64.31%) | 48.31% (+39.45%) | 81.91% (+65.12%) |
| TFProb | 10.11% (+3.72%) | 62.21% (+45.61%) | 50.71% (+41.85%) | 74.86% (+58.07%) |
| TFRF | 37.75% (+29.36%) | 85.59% (+67.99%) | 54.52% (+45.66%) | 86.63% (+69.84%) |
| TFCRF | 17.82% (+11.43%) | 75.84% (+59.24%) | 52.98% (+44.12%) | 82.29% (+65.50%) |

## 3.2 Discussion

As the predicted results of all data sets under different term weighting methods shown in Section 3.1, we found that there are at least two suitable term weighting methods can be adopted in each case, depending on macro F1 or micro F1 respectively. In Reuters 21578, we think that the traditional term weighting method still works fine with the multi-label text classification. In the case of Re1, our newly proposed term weighting method TFCRF gave us a better result.

Although introducing term weighting method may have some improvements to certain dataset, but still there are some datasets which haven't shown significant effect. Term weighting methods here are not so obviously effective.

We use REF to find important terms for classification with SVM and logistic regression in each dataset. From Figure 21 to Figure 26, we can find the overlap rate in all datasets. The information of the table contains 1. the overlap rate between important terms for logistic regression and important terms for SVM, 2. the overlap rate between high weight terms and important terms for logistic regression, and 3. the overlap rate between high weight terms and important terms for SVM. We also try to use the less importance terms to do the same experiment, the range of term quantity is from 25% to 5% of total terms.

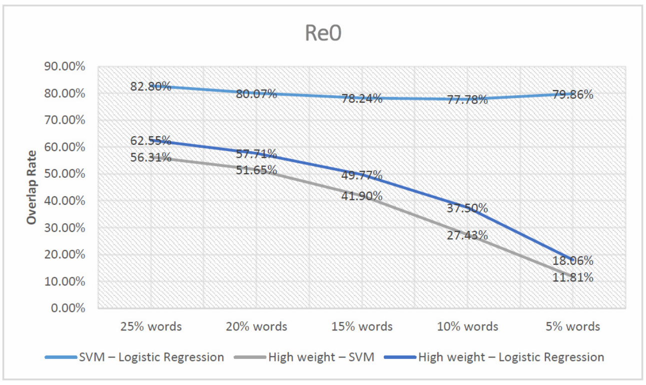


**Figure 21.** Overlap Rate with TFRF in Re0

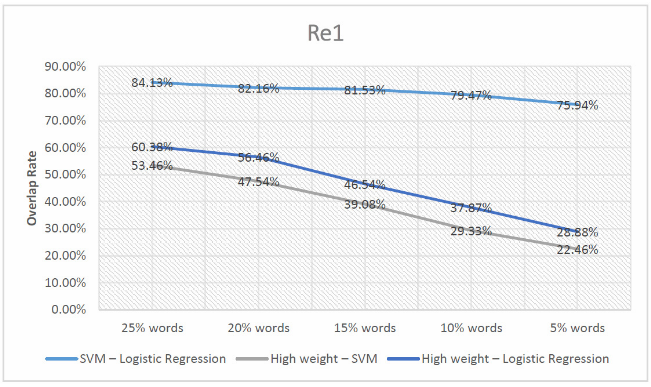


**Figure 22.** Overlap Rate with TFRF in Re1

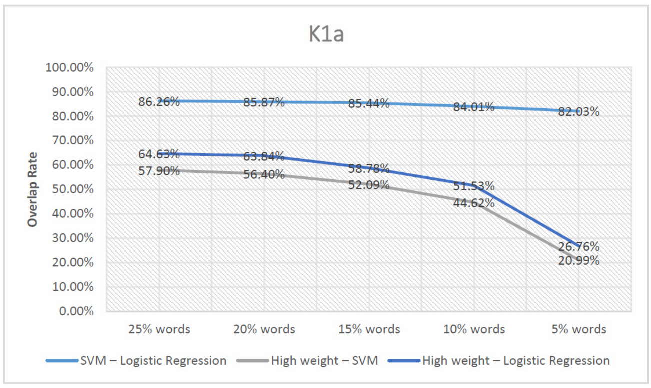


**Figure 23.** Overlap Rate with TFRF in K1a

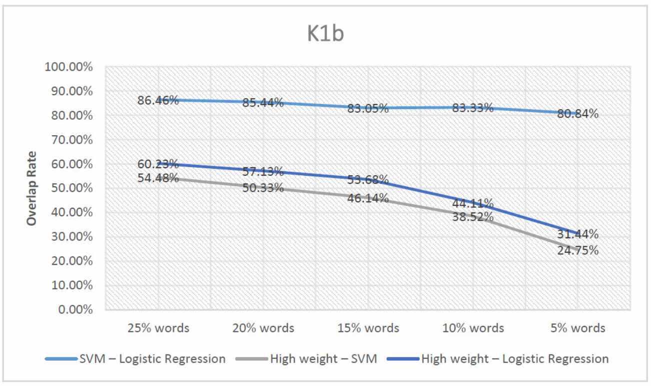


**Figure 24.** Overlap Rate with TFRF in K1b

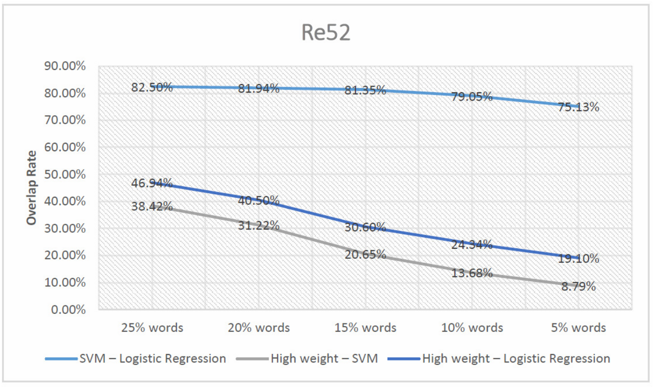


**Figure 25.** Overlap Rate with TFRF in Re52

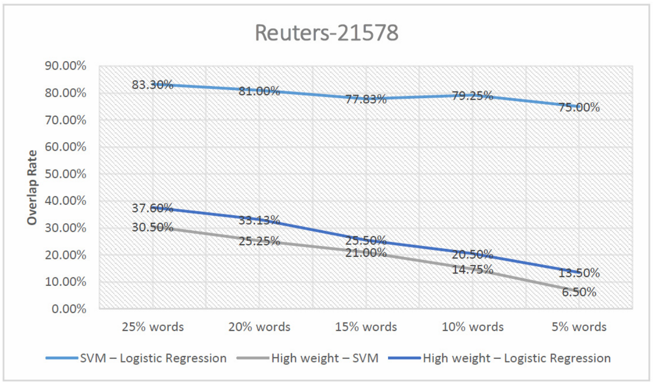


**Figure 26.** Overlap Rate with TFRF in Reuters-21678

From these figures, we find that the overlap rate can reach up to 70% between different term weighting methods. But the overlap rate between high weight terms and important terms for classification are significant lower. The lowest overlap rate can down to lower than 10%. The result also shows under different term quantity, the overlap rate between different term weighting methods is slightly decreased, but the overlap rate between high weight terms and important terms has notable decreasing up to over 30%.

This tells us that the actual importance of a term in document sometimes may not as much as the term`s calculated weight shows.

We use a simple feature selection to reduce the number of features for better performance. It selects features base on the value of term weight, the higher the value is, the easier it will pick. In previous paragraph, we concluded that high weight terms may not that important to the document. This may cause by the features we used for term weighting are inappropriate, the features we choose cannot actually represent the importance of the term in document.

The other potential issue exists in the term weighting methods is that maybe a term weighting method is suitable for a dataset, but doesn't mean that it is suitable for another dataset. And a much more complex method may not perform well than a simple one. For example, one-hot encoding even works better than almost all the other methods in dataset Re0. We also find that the importance of probability-based element

A and B are reduced in K1a.

These term weighting methods utilize many features but output the result in just a single value. The information must have some distortions or losses during the processing. So we think term weighting method should play an assist role instead of representing the importance of a term.

## 4 Conclusion

### 4.1 Information Missing

Bag-of-words model is a simplifying representation which is widely used in NLP. But it disregards grammar and word order in order to keep the multicity. It is the first reason that causes information missing.

Term weighting method combines several useful information elements to a single value. This process may eliminate lots of the individual information from these elements. It is the second reason that causes information missing. On the other hands, the word2vec model uses a vector instead of a single value to represent a word. The formula of term weighting method which used in word2vec is similar to a cost function. The larger value the cost function has, the more important the corresponding word is. However, we observed that the importance of a word in a document is equal to the importance of the same word in classification. Maybe it is a big mistake using Term Frequency and disregarding the grammar and order or something else in previous research. From our experiments, the results do strongly support our viewpoint. In our future work, we will drop out the formula and try to keep all information elements within a vector to represent a word.

### 4.2 Conclusion

There are two main contributions in this research. First, we propose a new weighting algorithm, TFCRF, which considers the term that equally separated in all classes as a negative effect to the discriminating power. The TFCRF successfully brings us a better result in Re1. Second, we have shown that the discriminating power of a term may not be as strong as we thought, especially for those who have a higher weight in classification.

The result tells that term weighting methods can be adopted to improve the results of text classification in some case. Different datasets may suit different term weighting methods. It regards how the model should be evaluated and what the type of the text and category should be considered.

Although we get a better result from TFCRF, we still think that the term weighting methods are not as useful as we thought. In some cases, most of term weighting methods cannot get a satisfied or even worse result. Sometimes it costs a lot of resources to find a suitable weight, but the result may not show us an obvious improvement.

Term weighting is designed to express the importance of a word in a text and it is often used in text classification. We can easily say that high weight words may not reflect on the importance in classification. A high weight word may play a significant role to a text, but it may not be same important in classification.

The research of term weighting method recently may have encountered a bottleneck due to the information missing problem. Our future work will focus on how to design representation vectors of words to keep more information. Word2vec uses vectors to represent a word and it becomes a popular method in NLP. Referring to word2vec, designing a new method to keep the factors' information and make good use could be a topic. Also, how to deal with an imbalanced dataset is another issue. Imbalanced dataset problem may cause classification having bad results in certain condition. Finally, the multi-label texts nowadays appear more and more frequently in our life. It is necessary to improve the accuracy of this kind of classification for future application.

## References

[1] C. Xing, D. Wang, X. Zhang, C. Liu, Document Classification with Distributions of Word Vectors, *Signal and Information Processing Association Annual Summit and Conference (APSIPA)*, Siem Reap, Cambodia, 2014, pp. 1-5.

[2] A. D. Patel, V. N. Pandya, Web page classification based on context to the content extraction of articles, *2nd International Conference for Convergence in Technology (I2CT)*, Mumbai, India, 2017, pp. 539-541.

[3] I. Rish, An empirical study of the naive Bayes classifier, *International Joint Conferences on Artificial Intelligence (IJCAI) 2001 workshop on empirical methods in artificial intelligence*, Seattle, Washington, USA, 2001, Vol. 3, No. 22, pp. 41-46.

[4] C.-W. Hsu, C.-J. Lin, A Comparison of Methods for Multiclass Support Vector Machines, *IEEE Transactions on Neural Networks*, Vol. 13, No. 2, pp. 415-425, March, 2002.

[5] G. Guo, H. Wang, D. Bell, Y. Bi, K. Greer, KNN Model-Based Approach in Classification, in: R. Meersman, Z. Tari, D. C. Schmidt (Eds.), *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE*, Springer, Berlin, Heidelberg, 2003, pp. 986-996.

[6] J. R. Quinlan, Simplifying decision trees, *International Journal of Man-Machine Studies*, Vol. 27, No. 3, pp. 221-234, September, 1987.

[7] E. Gibaja, S. Ventura, A Tutorial on Multilabel Learning, *ACM Computing Surveys*, Vol. 47, No. 3, pp. 1-38, April, 2015.

[8] D. Lewis, Reuters-21578 text categorization test collection, *AT&T Labs-Res.*, Florham Park, NJ, USA, distribution 1.0,

1997.

[9] D. D. Lewis, Y. Yang, T. G. Rose, F. Li, RCV1: A new benchmark collection for text categorization research, *Journal of Machine Learning Research*, Vol. 5, pp. 361-397, December, 2004.

[10] V. Korde, Text Classification and Classifiers: A Survey, *International Journal of Artificial Intelligence & Applications*, Vol. 3, No. 2, pp. 85-99, March, 2012.

[11] E. Leopold, J. Kindermann, Text Categorization with Support Vector Machines-How to Represent Texts in Input Space?, *Machine Learning*, Vol. 46, No. 1-3, pp. 423-444, January, 2002.

[12] S. S. Samant, N. L. B. Murthy, A. Malapati, Improving Term Weighting Schemes for Short Text Classification in Vector Space Model, *IEEE Access*, Vol. 7, pp. 166578-166592, November, 2019.

[13] K. Chen, Z. Zhang, J. Long, H. Zhang, Turning from TF-IDF to TF-IGM for term weighting in text classification, *Expert Systems with Applications*, Vol. 66, pp. 245-260, December, 2016.

[14] G. Salton, M. J. McGill, *Introduction to Modern Information Retrieval*, McGraw-Hill, Inc., 1986.

[15] V. Lertnattee, T. Theeramunkong, Analysis of inverse class frequency in centroid-based text classification, *IEEE International Symposium on Communications and Information Technology (ISCIT) 2004*, Sapporo, Japan, 2004, pp. 1171-1176.

[16] J. W. Reed, Y. Jiao, T. E. Potok, B. A. Klump, M. T. Elmore, A. R. Hurson, TF-ICF: A New Term Weighting Scheme for Clustering Dynamic Data Streams, *2006 5th International Conference on Machine Learning and Applications (ICMLA'06)*, Orlando, FL, USA, 2006, pp. 258-263.

[17] D. Wang, H. Zhang, Inverse-Category-Frequency Based Supervised Term Weighting Schemes for Text Categorization, *Journal of Information Science and Engineering*, Vol. 29, No. 2, pp. 209-225, March, 2013.

[18] F. Carvalho, G. P. Guedes, TF-IDFC-RF: A Novel Supervised Term Weighting Scheme, https://arxiv.org/abs/ 2003.07193, 2020.

[19] D. Mladenic, M. Grobelnik, Feature Selection for Unbalanced Class Distribution and Naive Bayes, *Proceedings of the Sixteenth International Conference on Machine Learning*, Bled, Slovenia, 1999, pp. 258-267.

[20] Y. Liu, H. Loh, A. Sun, Imbalanced text classification: A term weighting approach, *Expert Systems with Applications*, Vol. 36, No. 1, pp. 690-701, January, 2009.

[21] M. Lan, C.-L. Tan, H.-B. Low, Proposing a new term weighting scheme for text categorization, *Proceedings of the National Conference on Artificial Intelligence*, Boston, Massachusetts, USA, 2006, pp. 763-768.

[22] M. Lan, C. L. Tan, J. Su, Y. Lu, Supervised and traditional term weighting methods for automatic text categorization, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 31, No. 4, pp. 721-735, April, 2009.

[23] H. Zhou, Y. Zhang, H. Liu, Y. Zhang, Feature Selection Based on Term Frequency Reordering of Document Level, *IEEE Access*, Vol. 6, pp. 51655-51668, September, 2018.

[24] Y. Ko, A New Term Weighting Scheme for Text Classification using the Odds of Positive and Negative Class Probabilities, *Journal of the Association for Information Science and Technology (ASIS&T)*, Vol. 66, No. 12, pp. 2553-2565, December, 2015.

[25] K. Pearson, On lines and planes of closest fit to systems of points in space, *Philosophical Magazine*, Vol. 2, No. 11, pp. 559-572, 1901.

[26] X. Lin, F. Yang, L. Zhou, P. Yin, H. Kong, W. Xing, X. Lu, L. Jia, Q. Wang, G. Xu, A support vector machine-recursive feature elimination feature selection method based on artificial contrast variables and mutual information, *Journal of chromatography B, Analytical technologies in the biomedical and life sciences*, Vol. 910, pp. 149-155, December, 2012.

[27] D. M. Powers, Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation, *Journal of Machine Learning Technologies*, Vol. 2, No. 1, pp. 37-63, 2011.

## Biographies

**Kuan-Hua Tseng** received the M.S. degree from the Department of Computer Science and Information Engineering, Da-Yeh University, Taiwan. He is now pursuing his Ph.D. in National Sun Yat-sen University, Taiwan. His research interests include AI, Machine Learning, mobile communication networks, Internet of things and embedded system.

**Chun-Hung Richard Lin** received Ph.D. degree from Computer Science Department, University of California, Los Angeles (UCLA). He is currently a Professor of the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan. His research interests include the design and control of mobile communication networks, Internet of things, edge computing and device AI, and embedded operating system design and implementation.

**Jain-Shing Liu** received the Ph.D. degree from the Department of Computer and Information Science, National Chiao Tung University, Taiwan. He currently is a professor with the Department of Computer Science and Information Engineering, Providence University, Taiwan. His research interests include design and performance analysis of wireless communication protocols, wireless local area networks, wireless sensor networks, and wireless rechargeable networks.

**Chih-Ming Andrew Huang** received the M.S. degree from the Department of Science of Statistics, National Chiao Tung University, Hsinchu, Taiwan. Currently, he is a Ph.D. student of Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan. His research interests include computer communication technologies, Internet of things, biometrics technologies and embedded system.

**Yue-Han Wang** received the B.S. degree from the Department of Computer Science and Information Engineering, Tunghai University, Taiwan in 2015 and the M.S. degree from the Department of Computer Science and Engineering, National Sun Yat-sen University, Taiwan in 2019. His research interests include Machine Learning, NLP, data engineering.