第 48 卷 第 11 期
2021 年 11 月

湖南大学学报（自然科学版）
Journal of Hunan University（Natural Sciences）

Vol. 48. No. 11.
November 2021

Text Mining Infrastructure in Erlang

Abbas Jkhayyir Kadhim

Southern Technical University, Missan, Iraq

Abstract: Nowadays, one of the most important aspects of our lives is intrinsic pillars, which are the fastest element to implement in order to obtain an efficient result and support decision making. This paper proves the fastest method of classification algorithm performance during the extraction of data knowledge. While analyzing the text file description, data mining extraction for the text file is always used in conjunction with various fields, such as natural language processing (NLP) and information data. The preparation of the text mining algorithms for the text file is based on two files that are derived from Erlang shell framework. The first file is the text file, which requires interpretation and results, while the second file contains the actual words from the English dictionary. Inter-process communication is required to obtain a result file on the classified data according to the correct or incorrect words in the English language. We also present analysis techniques using methods based on text count, text classification, and string kernels. The article provides two deductive tips, with the information retrieval data of the text file including the percentage of correct or incorrect words. Information retrieval must be performed as a Knowledge Discovery in Databases (KDD), using the frequent pattern analysis technique under the classification algorithm to provide us with a full useful analysis of the text file. Furthermore, the result test under the runtime demonstrates the code written by Erlang functional programming language, which provides high efficiency in the speed of its performance in implementation.

Keywords: machine learning, text mining, supervised learning, classification algorithm, Erlang functional programming language.

二郎中的文本挖掘基础设施

摘要:

如今，我们生活中最重要的方面之一是内在支柱，这是为了获得有效结果和支持决策制定的最快实施要素。本文证明了在数据知

识提取过程中分类算法性能最快的方法。在分析文本文件描述时，文本文件的数据挖掘提取总是与自然语言处理（NLP）和信息数据等各个领域结合使用。文本文件的文本挖掘算法的编写基于两个衍生自二郎壳框架的文件。第一个文件是文本文件，需要解释和结果，而第二个文件包含英语词典中的实际单词。需要进行进程间通信，根据英文单词的正确或错误来获取分类数据的结果文件。我们还介绍了使用基于文本计数、文本分类和字符串内核的方法的分析技术。文章提供了两个演绎技巧，文本文件的信息检索数据包括正确或错误单词的百分比。信息检索必须作为数据库中的知识发现 (凯德)

来执行，使用分类算法下的频繁模式分析技术为我们提供对文本文件的完整有用的分析。此外，运行下的结果测试展示了二郎函数式编程语言编写的代码，在实现的速度上提供了很高的效率。

关键词：机器学习、文本挖掘、监督学习、分类算法、二郎函数式编程语言。

Received: July 24, 2021 / Revised: September 15, 2021 / Accepted: October 13, 2021 / Published: November 30, 2021
About the author: Abbas Jkhayyir Kadhim, Ass. Lecturer, Southern Technical University, Missan, Iraq
Corresponding author Abbas Jkhayyir Kadhim, abbasjk@stu.edu.iq

1. Introduction

The requirements for the quality of the product being developed have rapidly increased in recent years [6]. Since the beginning of the software's product development, developers have been striving to monitor its quality. Software metrics and their visualization are two important features of measurement systems. In this paper, we introduce the framework for analyzing the quality of programs written in the Erlang programing language, which is built on the top of the refactor static analysis tool. Our goal is to analyze projects and then prepare the results of these measurements.

2. Definition

2.1. Erlang

Erlang is a functional language designed for highly parallel scalable applications requiring high uptime [1]. Several industrial and open-source products were implemented in Erlang. The tools to measure the complexity and quality of the source code are therefore highly desirable. The default compiler of Erlang is Erlang/OTP, based on the BEAM virtual machine. It takes an .erl file as the input and produces a .beam byte code as the output. To detect software defects, such as exception-raising code or hidden failures in Erlang code, a software tool called a dialyzer was developed, which detected a significant number of discrepancies that have gone undetected throughout years of extensive testing [3].

Erlang contains the following data types [1], [2], [3]:

- Integers of unlimited size.
- Floats.
- Strings, placed within double quotes: "This is a string".
- Atoms. An atom is an element by itself. It starts with a lowercase letter and is built of letters, digits, and underscores, or it is any string placed within single quotes: atom1, 'Atom 2'.
- Lists are a comma-separated sequence of certain values placed within brackets: [abc, 123, "This is a string"].
- Tuples are a comma-separated sequence of certain values placed within braces: {abc, 123, "This is a string"}.
- Records are not a separate data type but are merely tuples with keys associated with each value. They are declared in a file and defined (given specific values) in the program.
- Binaries are placed within double angle brackets: «0, 128, 128, 255», «"It is some string"», «X:7, Y:5, Z:1». Binaries are series of bits—the number of bits in a binary has to be a multiple of 8.
- References are globally unique values.
- Pids stand for process identifiers.

2.2. Machine Learning

Over the past two decades, machine learning has become one of the major topics of artificial intelligence approaches [4], [5]. With ever-increasing amounts of data becoming available, there is good reason to believe that smart data analysis will become even more pervasive as a necessary ingredient for technological progress in relation to the machine learning approach. Machine learning can appear under many guises, and we will now discuss several applications and the types of data they deal with. Many forms of machine learning are reduced to a range of fairly different problems and a set of fairly narrow models [4], [5]. Much of the science of machine learning is centered around solving these problems and providing reliable guarantees for solutions. There are many research fields, such as statistics and machine learning, that are helpful for the development of various forms of data mining and knowledge discovery [5], [17]. Supervised learning means that the input data (X) and the output data (Y) are measured according to a possible function (f), which is expressed by applying the input as follows: X, Y = f (X).

2.3. Text Mining

The text mining discovery process involves using a native text file or web document to acquire helpful and comprehensible information [6], [7]. Today, the available information is increasing, with progress in data mining research that has led to the development of other forms of text mining methods for mining motivational patterns in big data. Text mining is one of the most vital and motivational research areas, and data analysis tools are particularly important in the text scope of natural language. The text can be any kind of content, such as social media posts, email messages, text messages, word documents, web content, blog posts, news articles, and other types of data. The performance of text mining algorithms, including classification, clustering, association, prediction, and other such algorithms, is related to the methods used to design the data warehouse and the methods through which the data is stored [6], [7], [8]. Detailed data is very important for obtaining proper reports. Therefore, the high rate of big data represents a great challenge in mining algorithms, as the goal is to obtain useful information from big data in order to give effective decision-making support. Text mining involves automated methods for analyzing the content of these documents and discovering the knowledge within them. Numerical data mining has long been used to examine patterns in numerical data and make predictions based on those patterns [6], [7], [8].

3. Literature Review

David Meyer [6] provides a package integrated with database backend support to minimize memory demands with preceding metadata management

performed on comprehensive text files, which relieves the usage on several traditional file formats: plain text, CSV text, or PDFs.

Ingo Feinerer [9] introduces text mining in R using framework text mining provided by the tm package, presents methods for data import and CRISP phases, pre-processing, creating term-document matrices, importing plain texts from files in a directory from a vector in R, pre-processing and transforming the texts for boosting performance and loading documents into the memory without their compression. It works for medium and large data.

Mohamed Yassine [10] provides a new perspective on studying the friendship emotions and feelings, dealing with their nature and the nature of the language used. This case study on Lebanese Facebook users used an unsupervised technique (k-means clustering algorithm). The result shows its high accuracy in determining the subjectivity of texts and predicting friendship.

Eduardo G. Altmann [11] worked on a statistical model and used the logicality for each word with less emphasis on pattern frequency or frequency of an occurrence. Furthermore, he developed a model of behavior that determines the dynamics of word usage.

4. Research Methodology

The major objectives of this paper are to study particular features and analyze the requirements, procedures through text mining algorithms based on two files called up by the Erlang shell framework. The first file is the text file needing interpretation and getting the result; the second one is the English dictionary words, the main file of code execution, and the result file.

Text mining is the sub of data mining from text data collections [6], [7], [8], [12]. KDD is the process of determining useful information from a collection of data. KDD used data mining techniques, including data choice, data preparation, data cleaning, incorporating prior knowledge discovery on data sets, and interpreting precise solutions of the outcome by evaluation to obtain information knowledge [4]. The classification model is one of the data mining algorithms. The goal of the classification is a precise prediction of the target class for each case on data. Fig. 1 shows the KDD process steps indicated by arrows.

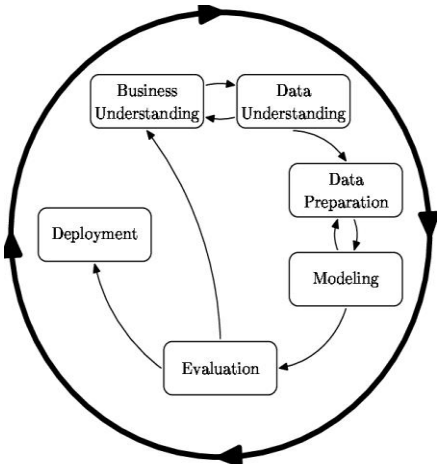

Fig. 1 CRISP-data mining process phases [7], [8], [12]

The aim of the discovery of data mining process can be divided further into two different goals/tasks performed by data mining system [12], [13]:

1. Descriptive modeling;
2. Predictive modeling.

In descriptive modeling, data are described completely in an understandable form [14], [15].

In predictive modeling, the value of one variable is predicted from unknown values, i.e., future data are predicted [13], [14].

Text mining methods' description and prediction are achieved by classification and frequent pattern data mining methods [13], [15].

4.1. Pre-Processing and Preparing the Data

i. Step-words

Delete all the punctuation marks ( ) [ ] { } ! ? , ; : .
for the words to be more understandable.

ii. End line by "Enter" to a new line, the text automatically adds \n; then, \n must be removed on whole lines in the text file.

iii. Choose the stop words (i, he, she, and, or, the, a, an, in, etc.) or delete them from the text. The author decided to keep them in this paper because the dictionary contains these stop words [8], [13], [15].

Following, is the code of imported data from a text file. This data will be sent to the process cleaning.

```
readTxt(FileName) ->
  {ok, File} = file:open(FileName,[read]),
  {ok, Txt}  = file:read(File,1024 * 1024),
   cleanLineText(Txt).

cleanLineText([ ])-> [ ];
cleanLineText([H|Tt])->
case (H==13) or (H==10) or (H==40) or
      (H==41) or (H==44) or (H==46) or
      (H==58) or (H==59) or (H==91) or
      (H==93) or (H==123) or (H==125) of
    true -> [ 32| cleanLineText(Tt)] ;
    false-> [ H| cleanLineText(Tt)]
              end.
```

4.2. Modeling and Evaluation

Design a model for the data classification and implementation of the algorithm. At first, the algorithm function calls the following files:

do_Code_Classification(MyFile, FileDictionaryEnglish, FileNameSaveResult)->

i. The text file of the input data: it is a text file consisting of words in the English language intended to be classified.

ii. English Dictionary: a file containing 28,000 words in the English language, with numbers from 0 to 9999.

iii. Results file: means the file in which the results will store the words as true or false, with the frequent pattern words, and the percentage words for true or false in the text input. Fig. 2 will present the model life cycle.

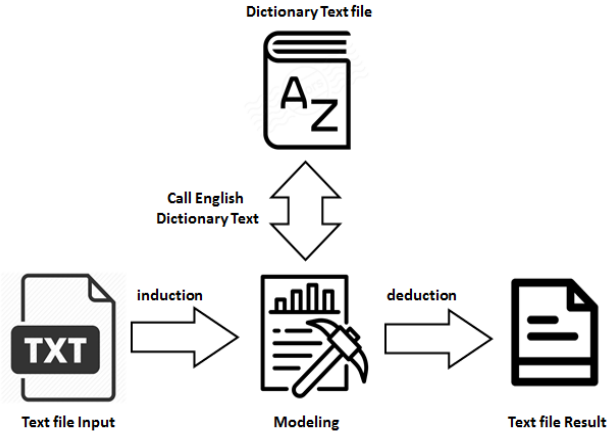

Fig. 2 The process phases [16]

At first, the file of input data (text input file) is imported by the read function (readTxt(MyFile), then a preprocessing is performed of the data text contained in it, also calling the dictionary file.

```
TxtDectionary1=readTxt(FileDectionaryEnglish),
TxtDectionary2=string:tokens(TxtDectionary1," "),
TextMy1= readTxt(MyFile),
TextMy2 =string:tokens(TextMy1," "),
```

After preprocessing the data, the algorithm will send each word in the text to the dictionary to check if this word is existing, true or false by the (classfy(Word,TxtDectionary2) function, then take each word as an item in text input to compare with all word items in dictionary text. This idea is similar to a one-to-many relationship in a database.

∃ Item (Text Input) == ∀ Item (Text Dictionary)

```
[classfy(Word,TxtDectionary2)||Word<- TextMy2 ]
    classfy(Word,[]) -> [Word,not_found];
    classfy(Word,[Word]) -> Word;
    classfy(Word,[HDec|TDec]) ->
          case Word==HDec of
              true -> [ Word | classfy(Word,TDec)];
      false->  classfy(Word,TDec)
          end.
```

The result in the variable (Result1) is stored, after doing preprocessing to get (Result2) as follows:

```
Result2  = ["this",true],
           ["zhu",false],
           ["models",true],
           ["used",true]
```

The algorithm performs a function on (Result2) to get frequent patterns of data mining for each correct word, after that, how many incorrect words there were appear in the text input.

```
frequent_patterns([])   -> [] ;
frequent_patterns(FullText) ->
Final= [  fpCount( X , FullText , [{0,X}] )    ||   X <- FullText    ],
lists:usort(Final).
fpCount( X , [ ] , [{Co,X}] ) -> [Co,X];
fpCount(   X   ,   [Htxt|Ttxt]   ,   [{Co,X}]   )   -> case X == Htxt of
        true  -> fpCount( X ,Ttxt , [{Co+1,X}] )   ;
        false -> fpCount( X ,Ttxt , [{Co,  X}] ) end.
```

The result of (frequent_patterns(Txt)) will be as :

```
'The pattern word True'
[182,"and"]
[162,"the"]
[131,"of"]
[85,"to"]
[60,"in"]
[49,"memory"]
[12,"have"]
[12,"also"]
[11,"this"]
[2,"locate"]
[1,"2003"]
```

The first item in the set refers to frequent patterns in each word in the text input, and the second item refers to the word.

At the end of the evaluation algorithm, to calculate the percentage for the whole text word Correct, Incorrect, run the (PercentTRUE()) function.

```
rateTRUE([], Rate)-> Rate;
rateTRUE([[First,Second]|Tail], Rate)->
case  Second == true of
    true -> rateTRUE(Tail, Rate+1) ;
    false-> rateTRUE(Tail, Rate)
   end.
     rateFALSE([], Rate)-> Rate;
```

4.3. Storing the Result

After processing, each input data produces output data. Some data serves the purpose of visualization and monitoring only. On the other hand, there is important data that should be saved in a file and for use future retrieval or analysis, which is dependent on the level of importance of that data.

In the algorithm below, it is very important to save the result in the text file output:

```
write_terms(Filename, Result) ->
  F = fun(Item) -> io_lib:format("~p ~n \r\n", [Item]) end,
    Text = lists:map(F, Result),
```

```
        file:write_file(Filename, Text).
    rateFALSE([[First,Second]|Tail], Rate)->
       case  Second == false of
        true -> rateFALSE(Tail, Rate+1) ;
    false-> rateFALSE(Tail, Rate)
    end.
```

As percentage, the result is printed as follows:

```
['The Result Percent OF True is %',75]
['The Result Percent OF False is %',25]
```

5. Result

This case is based on the importance of choosing the most proper method of implementation of algorithm code according to the fast runtime. Text mining on text file using Erlang functional program is one of the most intrinsic pillars depending on the speed runtime execution.

Table 1 below shows different methods of execution of the code of algorithm implementing the framework given by Erlang shell according to the time of runtime execution:

the code for time:timer:tc(text,do_Code_Classification,["file.txt","dectionary.txt","fileResult.txt"]):

Table 1 Methods of execution of the code of algorithm implementing the framework given by Erlang shell according to the time of runtime execution

No.	Input file words	Time of runtime execution
1	14	1.7 seconds
2	224	5 seconds
3	1013	18 seconds
4	9183	243 seconds
5	18098	731 seconds

Fig. 3, 4, and 5 show us the runtime execution of processing the algorithm from the start to the completion of the process. Fig. 6 shows the text file result.

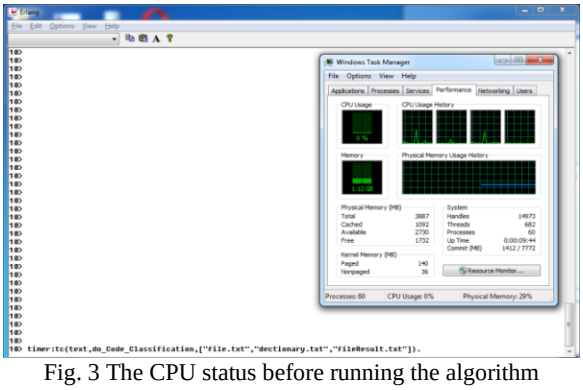

Fig. 3 The CPU status before running the algorithm

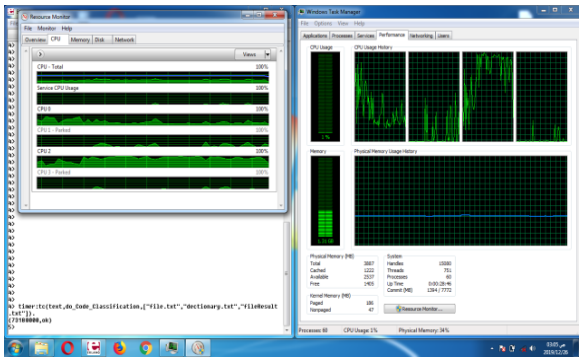

Fig. 4 The CPU status while running the algorithm

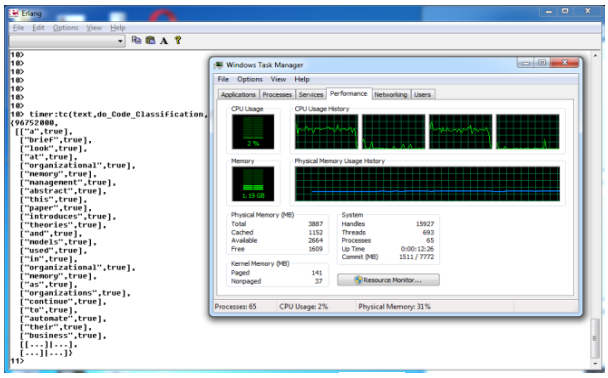

Fig. 5 The CPU status after ending the algorithm

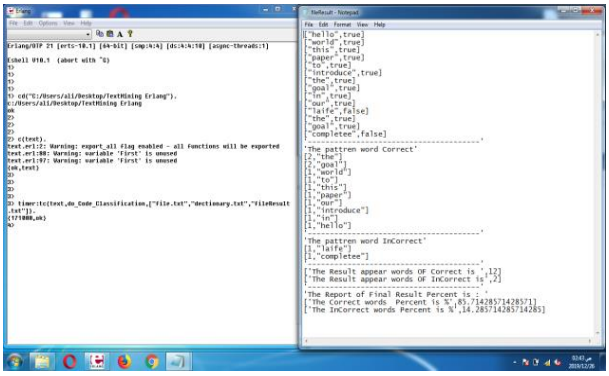

Fig. 6 The text file result

6. Conclusion

- This paper provides an applied study in the field of knowledge discovery using text mining. The optimum text document is a precise all words inclusive, with the classification of the text mining technique will achieve a proper word text. In addition, it describes how text data mining is always used in various fields like natural language processing (NLP), information data extraction, and information retrieval as the knowledge.
- The techniques used for text mining include information of classification of the data, with frequent pattern analysis.
- The Erlang language used in the algorithm is a functional language. It is different from the other program languages used in text mining and the data mining area. Most of the frameworks have a code for most algorithms, such as the Weka framework, which is often inflexible to edit. Writing the code in the Erlang language gives us more flexible algorithm methods.

- As with the most intrinsic pillars in general life, it is the fastest implementation and provides efficient results to support making decisions, whereas functional programming languages are quick to perform in implementation.
- Work has not been touched in a parallel way. Therefore, this work can be done in the future due to the importance and the high efficiency in the execution time of the runtime.

References

[1] ARMSTRONG J., VIRDING R., WIKSTRÖM C., and WILLIAMS M. *Concurrent Programming in Erlang.* 2nd ed. Prentice Hall, Herfordshire, 1996.

[2] ARMSTRONG J. A history of Erlang. Proceedings of the 3rd ACM SIGPLAN Conference on History of Programming Languages, 2007.

[3] LINDAHL T., & SAGONAS K. TYPER: A Type Annotator of Erlang Code. Proceedings of the ACM SIGPLAN Workshop on Erlang, Tallinn, 2005, pp. 17–25. https://doi.org/10.1145/1088361.1088366

[4] LISON, P. *An introduction to machine learning.* Language Technology Group, Edinburgh, 2015.

[5] WITTEN I. H., & FRANK E. *Data Mining: Practical Machine Learning Tools and Techniques.* 2nd ed. Morgan Kaufmann Publishers, San Francisco, California, 2005. https://academia.dk/BiologiskAntropologi/Epidemiologi/DataMining/Witten_and_Frank_DataMining_Weka_2nd_Ed_2005.pdf

[6] MEYER D., HORNIK K., and FEINERER I. Text mining infrastructure in R. *Journal of Statistical Software*, 2008, 25(5): 1-54. https://doi.org/10.18637/jss.v025.i05

[7] LIU L., & ÖZSU M. T. *Encyclopedia of database systems*, Vol. 6. Springer, New York, 2009. https://doi.org/10.1007/978-1-4614-8265-9

[8] HIPP J., & GÜNTZER U. Is pushing constraints deeply into the mining algorithms really what we want? - An alternative approach for association rule mining. *ACM SIGKDD Explorations Newsletter*, 2002, 4(1): 50-55. https://doi.org/10.1145/568574.568582

[9] FEINERER I. *Introduction to the tm Package Text Mining in R.* 2013. http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf

[10] YASSINE M., & HAJJ H. A framework for emotion mining from text in online social networks. Proceedings of the IEEE International Conference on Data Mining Workshops, Sydney, 2010, pp. 1136-1142. https://doi.org/10.1109/ICDMW.2010.75

[11] ALTMANN E. G., PIERREHUMBERT J. B., and MOTTER A. E. Beyond word frequency: Bursts, lulls, and scaling in the temporal distributions of words. *PLoS ONE*, 2009, 4(11): e7678. https://doi.org/10.1371/journal.pone.0007678

[12] FAYYAD U., PIATETSKY-SHAPIRO G., and SMITH P. Knowledge Discovery and Data Mining: Towards a Unifying Framework. KDD-96 Proceedings, 1996. https://aaai.org/Papers/KDD/1996/KDD96-014.pdf

[13] PADHY N., MISHRA P., and PANIGRAHI R. The Survey of Data Mining Applications and Feature Scope. *International Journal of Computer Science, Engineering and Information Technology*, 2012, 2(3): 43-58. https://doi.org/10.5121/ijcseit.2012.2303

[14] BOLASCO S., CANZONETTI A., CAPO F. M., RATTA-RINALDI F. D., and SINGH B. K. *Understanding Text Mining: A Pragmatic Approach.* Roam, 2002.

[15] ANANIADOU S., & MCNAUGHT J. (eds.) *Text Mining.* 2006.

[16] SALMAN A. D., AL-FARTTOOSI H. A. D., and KADHIM A. J. Study impact the latitude on Covid-19 spread virus by data mining algorithm. *Journal of Physics: Conference Series*, 2020, 1664(1): 012109. https://doi.org/10.1088/1742-6596/1664/1/012109

[17] MOSTAFA A. M., IDREES A. M., KHEDR A. E., and HELMY Y. M. A Proposed Architectural Framework for Generating Personalized Users' Query Response. *Journal of Southwest Jiaotong University*, 2020, 55(5). https://doi.org/10.35741/issn.0258-2724.55.5.3

参考文:

[1] ARMSTRONG J.、VIRDING R.、WIKSTRÖM C. 和 WILLIAMS M. 二郎中的并发编程。第二版。普伦蒂斯霍尔，赫福德郡，1996。

[2] ARMSTRONG J. 二郎的历史。2007 年第三届 ACM信号计划编程语言历史会议论文集。

[3] LINDAHL T., & SAGONAS K. 打字机：二郎代码的类型注释器。关于二郎的 ACM信号计划研讨会论文集，塔林，2005，第 17-25 页。 https://doi.org/10.1145/1088361.1088366

[4] LISON, P. 机器学习简介。语言技术集团，爱丁堡，2015。

[5] WITTEN I. H., & FRANK E. 数据挖掘：实用机器学习工具和技术。第二版。摩根考夫曼出版社，加利福尼亚州旧金山，2005。https://academia.dk/BiologiskAntropologi/Epidemiologi/DataMining/Witten_and_Frank_DataMining_Weka_2nd_Ed_2005.pdf

[6] MEYER D.、HORNIK K. 和 FEINERER I. R. 统计软件杂志中的文本挖掘基础设施，2008，25(5)：1-54. https://doi.org/10.18637/jss。v025.i05

[7] LIU L., & ÖZSU M. T. 数据库系统百科全书，卷。 6. 斯普林格，纽约，2009。https://doi.org/10.1007/978-1-4614-8265-9

[8] HIPP J., & GÜNTZER U. 将约束深入推入挖掘算法真的是我们想要的吗？ - 关联规则挖掘的替代方法。ACM SIGKDD 探索通讯，2002，4(1)：50-55。 https://doi.org/10.1145/568574.568582

[9] FEINERER I. R. 2013 中的 tm 包文本挖掘介绍。http://cran.r-project.org/web/packages/tm/vignettes/tm.pdf

[10] YASSINE M., & HAJJ H. 在线社交网络文本情感挖掘框架。IEEE 国际数据挖掘研讨会论文集，悉尼，2010，第 1136-1142 页。 https://doi.org/10.1109/ICDMW.2010.75

[11] ALTMANN E. G.、PIERREHUMBERT J. B. 和 MOTTER A. E. 超越词频：词的时间分布中的突发、间歇和缩放。公共科学图书馆一，2009，4(11)：e7678. https://doi.org/10.1371/journal.pone.0007678

[12] FAYYAD U.、PIATETSKY-SHAPIRO G. 和 SMITH P. 知识发现和数据挖掘：迈向统一框架。凯德-96会议录，1996。https://aaai.org/Papers/KDD/1996/KDD96-014.pdf
[13] PADHY N.、MISHRA P. 和 PANIGRAHI R. 数据挖掘应用和特征范围调查。国际计算机科学、工程与信息技术学报, 2012, 2(3): 43-58. https://doi.org/10.5121/ijcseit.2012.2303
[14] BOLASCO S.、CANZONETTI A.、CAPO F. M.、RATTA-RINALDI F. D. 和 SINGH B. K. 理解文本挖掘：一种实用的方法。漫游，2002。
[15] ANANIADOU S., & MCNAUGHT J. (编辑。) 文本挖掘。2006。
[16] SALMAN A. D.、AL-FARTTOOSI H. A. D. 和 KADHIM A. J. 研究通过数据挖掘算法影响新冠肺炎传播病毒的纬度。物理学杂志：会议系列, 2020, 1664(1): 012109. https://doi.org/10.1088/1742-6596/1664/1/012109
[17] MOSTAFA A. M.、IDREES A. M.、KHEDR A. E. 和 HELMY Y. M. 生成个性化用户查询响应的拟议架构框架。西南交通大学学报, 2020, 55(5). https://doi.org/10.35741/issn.0258-2724.55.5.3