Secrets of the Secrets: Text Mining on Dao Canon

Yubing Yang, Jue Jin, Shui Wang and Le Wang*

School of Information Engineering, Ningbo Dahongying University, Ningbo, Zhejiang, China

Abstract: Dao Canon are ancient Chinese documents that recorded the never-ending efforts of Taoists to longevity and supernatural powers. Text mining analysis may help better understand these documents. This chapter performs text mining on Dao Canon synopsis text utilizing the *arulesSequences* and *tm* packages of R, and introduces some related techs & tricks in dealing with Chinese literals.

Keywords: Ancient chinese, dao canon, daoism, text mining.

1. INTRODUCTION

Daoism (or Taoism) is the ancient Chinese endeavor to defy the natural law of aging, to be immortal or to get supernatural powers. Unlike Buddhism or Christianity, although some of the Daoist literatures and methods have already spread to the rest of the world (like *Dao De Jing* and *Tai Ji Quan*), Daoism has not yet been a well-studied international academic topic. Compared with other similar subjects, there are several characteristics that make Daoism more intricate to laymen as well as to professionals, such as:

1. Dao literatures were written completely in ancient Chinese, with a complex jargon system, can hardly be scientifically studied by international scholars.
2. Most of the Dao literatures concerns practical exercising methods that can hardly be judged theoretically.
3. Some of the authors used riddle-like language to shield the "unqualified" learners from stealing their precious knowledge.

To perform systematic research on the literatures in Dao Canon, we introduce text mining methodology into this field; the reason of utilizing text mining is as follows:

1. The Canon includes enormous amount of documents, and text mining should help to reveal hidden knowledge.
2. Ancient authors tended to conceal their real names, and text mining helps to discover the relationship between documents.
3. The intension of jargons usually are ambiguous, sometimes even deliberately misrepresented; text mining helps to clarify this situation by utilizing pattern analysis or other mining processes such as clustering or association analysis, *etc.*

This chapter is a practical introduction on the use of R when data-mining the documents of Dao Canon, especially the techs & tricks in dealing with Chinese literals; it also includes some interoperation techniques of related software such as MySQL and PHP. The background setup of this chapter is that we can get digitized Canon data (Web pages) from several open resources, such as [1] and [2], and transform them to adequate format using other languages such as Java or PHP. We also have a dedicated Web site for depositing and sharing related data & result files [3].

2. PREPARING - PART 1: READING DATA FROM CSV FILES

Some existing Dao Canon related information can be found in CSV text format, such as the TOC (Table of Content) data in [4], formatted as the following:

```
序號，冊號—頁碼，經名，涵芬樓道藏冊頁，分類，卷數，作者
1，1—
1，靈寶無量度人上品妙經，0001天上004，洞真部本文類，六十一卷，
2，1—
417，元始無量度人上品妙經直音，0013洪上044，洞真部本文類，一卷，
3，1—425,，元始說先天道德經註解，0013洪上090,洞真部本文類，五卷,
    李嘉謀註
```

The above style is typical in current Dao canon literature. The first line is the title of the following data columns, and they are: serial no., scroll-page, canon title, scroll/page in Hanfenlou Colletion, category, number of volumes, and author. Note that the punctuation characters are in Chinese format, which also need to be transformed for data cleansing.

From the second line, are TOC texts following the style of the title line. Taken the second line as an example, it means that the Superior Lingbao Transcending Conon (靈寶無量度人上品妙經) is in scroll 1, page 1, classified as "0001天上004" in Hanfenlou Colletion, belongs to Dongzhen Category, and includes 61 volumes.

To remove the redundant information in the raw data above,we perform some formatting work to establish the header, and replace the values in fields "category" and "series" *etc* with unique numerical identifiers. The resulting data are like the following (full data file *dao.csv* can be downloaded at [5]):

```
serial no, category, series, scroll, page, title, volume, author, Hanfen-
lou_scroll, Hanfenlou_page
1,1,1,1,1,靈寶無量度人上品妙經,六十一卷,,1,天上004
2,1,1,1,417,元始無量度人上品妙經直音,一卷,,13,洪上044
3,1,1,1,425,元始說先天道德經註解,五卷,李嘉謀註,13,洪上090
```

The Reading from a CSV file can be done using R's "read" functions, such as *read.csv()* or *read.delim()*.Character encoding should be specified when reading form Chinese data files. Suppose that the above TOC data are stored in a CSV file located in "*D:/r/dao.csv*" with

"ANSI" encoding, we can use one of the following commands to read the TOC data from it:

> toc <- read.delim("D:/r/dao.csv", sep=',', header=TRUE, fileEncoding='GBK')

> toc <- read.csv("d:/r/dao.csv")

Note that if you saved the CSV file using encodings other than "ANSI", you must explicitly provide a value for *fileEncoding* parameter. In fact, the "Notepad" tool in Windows can save a file in 4 different encodings, and the corresponding parameter values for these encodings are:

- fileEncoding="UTF-16LE" for "Unicode" encoding;
- fileEncoding="UCS-2" for "Unicode big endian";
- fileEncoding="UTF-8" for "UTF-8";
- fileEncoding="GBK" for "ANSI".

Note that for some reason the "UTF-8" encoding of the Windows Notepad might not work smoothly with R, so you may want to use "ANSI" as your default file encoding.

Now we can take an overview of the result by

➢ *summary(toc)*

經名 Title	卷數 Scrolls	作者 Author
黃帝陰符經註 : 8 (Comments on Huangdi Yinfu Jing)	一卷 :1036 One scroll	:956
道德真經註 : 6 (Comments on Dao De Jing)	三卷 : 117 Three scrolls	杜光庭編 : 13 Eds. Du Guangting
(translations are add by the author to help non-Chinese readers and are omitted hereafter.)		
太上老君說常清靜經註: 5	二卷 : 88	杜光庭撰 : 8
周易參同契註 : 4	四卷 : 41	司馬承禎撰: 5
北斗七元金玄羽章 : 2	五卷 : 32	王 嚞撰 : 5
道德真經傳 : 2	十卷 : 29	俞琰註 : 5
(Other) : 1483	(Other): 167	(Other) :518

It is amazing to see that some of the statistical results come as a surprise: the most popular sutra may be *Yin Fu Jing* (陰符經) instead of *Dao De Jing* (as most of the people may tend to believe), because there are 8 documents with exactly the same name dedicating to commenting this sutra. Another 2 popular sutras are *Chang Qingjing Jing* (常清靜經) and *Zhouyi San Tong Qi* (周易參同契).

3. PREPARING - PART 2: ACCESSING DATA IN MYSQL

If we want to store & present data through a Web interface, the MySQL database system is an excellent choice for data storage. The following discusses procedures of using MySQL for Chinese information.

3.1. Create MySQL Tables for Chinese Information

To ensure compatibility with traditional Chinese literals, tables in MySQL must be created with UTF-8 encoding; an example is give as below:

```
CREATE TABLE `dao` (
`id` int(11) NOT NULL,
`category` varchar(45),
PRIMARY KEY (`id`)
) ENGINE=InnoDB DEFAULT CHARSET=utf8;
```

Please note that the parameter string "*DEFAULT CHARSET=utf8*" is mandatory.

Updating new data to the MySQL server from CSV files is straight forward and is not discussed in this chapter; it is safe to know that we already have the data prepared in [3] and the rest of this chapter will work on this base.

3.2. Reading Data from MySQL Table

Assume MySQL Data Source object name is *daotext-localhost*, and with the help of *RODBC* package [6], we can access MySQL data in an easy way:

> library(RODBC)

> ch <- odbcConnect(dsn="daotext-localhost", uid="daouser", pwd="XXXX", DBMSencoding="UTF-8");

*> daocategory <- sqlQuery(ch, "select * from daocanon-category limit 10")*

> daocategory

```
  id  category
1   1   洞真(Dongzhen Category)
2   2   洞玄(Dongxuan Category)
3   3   洞神(Dongshen Category)
4   4   太玄(Taixuan Category)
5   5   太平(Taiping Category)
6   6   太清(Taiqing Category)
7   7   正一(Zhengyi Category)
8   8   續道藏(Sequel Canon)
```

The above commands connects to the MySQL server represented by the Data Source Name "*daotext-localhost*" with user authentication specified by parameter "*uid*" and "*pwd*", and fetch at most 10 rows of data from table "daocanoncategory". Note that the value of the "*DBMSencoding*" parameter must coincide with that of the Data Source object, and with the actual encoding of the table.

4. FREQUENT SEQUENCES MINING ON TABLE OF CONTENT OF DAO CANON

Chinese sentences do not contain white space characters that can segment sentences into words in a "natural" way;

instead, context information as well as human experience play an important part in getting the real meaning from a sentence. Ancient Chinese documents do not have punctuations – and this makes the situation even worse. For example, for the famous "first sentence" of *Dao De Jing* writes "道可道非常道", there are at least two different ways of segmentation: "道可道, 非常道 (Truth that can be told is not the real truth)" and "道, 可道, 非常道 (Truth can be told only in an unusual way)",, and even more ways of interpreting them.

To speculate meaningful segmentation for ancient Chinese sentences, as a case study, we try to mine the frequent patterns of the title text of Dao Canon.

4.1. Package *arulesSequences*

Package *arulesSequences* [7] mines frequent sequences using *cSPADE* [8] algorithm.

The core method of arulesSequences is "cspade", with format like the following:

cspade(data, parameter = NULL, control = NULL, tmpdir = tempdir())

The meaning of its arguments are:

data: an object of class *transactions* with temporal information.

parameter: an object of class *SPparameter* or a named list with corresponding components.

control: an object of class *SPcontrol* or a named list with corresponding components.

tmpdir: a non-empty character vector giving the directory name where temporary files are written.

The most important argument is *data*, and it is an object of class transactions. A *transactions* class represents transaction data used for mining itemsets or rules. It is a direct extension of class *itemMatrix* (defined in package *arules*) to store a binary incidence matrix, item labels, and optionally transaction IDs and user IDs. Usually objects of this class are created by coercion from objects of other classes, as illustrated in the following example:

```
> a_list <- list(
+ c("a","b","c"),
+ c("a","b"),
+ c("a","b","d"),
+ c("c","e"),
+ c("a","b","d","e")
+ )
> ## set transaction names
> names(a_list) <- paste("Tr", c(1:5), sep = "")
> a_list
```

```
$Tr1
[1] "a" "b" "c"
$Tr2
[1] "a" "b"
$Tr3
[1] "a" "b" "d"
$Tr4
[1] "c" "e"
$Tr5
[1] "a" "b" "d" "e"
```

The above list can be coerced into transactions:

```
> trans <- as(a_list, "transactions")
> trans
```

```
transactions in sparse format with
5 transactions (rows) and
5 items (columns)
```

```
> inspect(trans)
```

```
Items transactionID
1 {a, b, c} Tr1
2 {a, b} Tr2
3 {a, b, d} Tr3
4 {c, e} Tr4
5 {a, b, d, e} Tr5
```

You can use the *summary* command for more detailed information of this *transactions* object:

```
> summary(trans)
```

```
transactions as itemMatrix in sparse format with
5 rows (elements/itemsets/transactions) and
5 columns (items) and a density of 0.56
most frequent items:
a b c d e (Other)
4 4 2 2 2 0
```

In *arulesSequences*, function *read_baskets* read transaction data in basket format (with additional temporal or other information) and create an object of class *transactions*.

4.2. Read in Transaction Data

Transaction data represent the kind of data people use in supermarket for shopping basket information. As an example, the "zaki" sample dataset in package *arulesSequences* has the following form:

```
1 10 2 C D
1 15 3 A B C
1 20 3 A D F
1 25 4 A C D F
2 15 3 A B F
2 20 1 E
3 10 3 A B F
4 10 3 D G H
4 20 2 B F
4 25 3 A G H
```

The first column is "*transaction id*", indicating different transactions; the second column is "*event id*", indicating the different "event" within a transaction; both the "transaction id" and the "event id" form the temporal information of transactions; the third column is the size of the transaction, meaning the number of "items" within this transaction. All other columns represent the shopping "items" of the transactions.

In package *arulesSequences* we use function *read_baskets* to read transaction data in basket format (with additional temporal or other information) and create an object of class transactions.

The format of *read_baskets* is:

```
read_baskets(con, sep = "[ \t]+", info = NULL, iteminfo = NULL)
```

where *con* is an object of class *connection* or file name; for example, we can construct a connection for the "zaki" sample dataset in package *arulesSequences*:

```
> conn <- system.file("misc", "zaki.txt", package="arulesSequences")
```

Argument *sep* of *read_baskets* is a regular expression specifying how fields are separated in the data file; the default separation characters include space and tab; argument *info* is a character vector specifying the header for columns with additional transaction information; argument *iteminfo* is a data frame specifying (additional) item information.

The following commands read in the *zaki* dataset and build a *transactions* object:

```
> x <- read_baskets(conn, info = c("sequenceID","eventID","SIZE") )
> as(x, "data.frame")
```

(The singular Chinese literals are taken from Canon titles.)

Transaction ID. Sequence ID	Transaction ID. Event ID	Transaction ID. SIZE	Items
1	1	10	2 {C,D}
2	1	15	3 {A,B,C}
3	1	20	3 {A,D,F}
4	1	25	4 {A,C,D,F}
5	2	15	3 {A,B,F}
6	2	20	1 {E}
7	3	10	3 {A,B,F}
8	4	10	3 {D,G,H}
9	4	20	2 {B,F}
10	4	25	3 {A,G,H}

4.3. Preprocess of the TOC Title Text

For our purpose – to find the frequent patterns of words across all title texts – we must first split the title text into single words and prepare them to create a *transactions* object. We leave this job to the PHP server because we have already put all the TOC information in the MySQL database as described in Section 2. We may suggest 3 additional tips if you want to do this yourself:

(a) Use UTF-8 encoding for your PHP web page (you can do this by adding the following HTML clause to the "<head>" section of you PHP file: <meta http-equiv="content-type" content="text/html;charset=utf-8">);

(b) Save you PHP file in UTF-8 format.

(c) Make sure to have the following statements at the beginning of your each PHP file that tries to communicate with MySQL:

```
// Read UTF-8 encoded text correctly from MySQL:
mysql_query("SET character_set_server=utf8");
mysql_query("SET character_set_results=utf8");
```

The result is provided in [9] (the *canon_title_csv.php* page), and it has the form like this:

```
0,1,靈,寶,無,量,度,人,上,品,妙,經
1,1,元,始,無,量,度,人,上,品,妙,經,直,音
2,1,元,始,說,先,天,道,德,經,註,解
3,1,無,上,內,秘,真,藏,經
4,1,太,上,無,極,總,真,文,昌,大,洞,仙,經
5,1,上,清,大,洞,真,經
6,1,大,洞,玉,經
7,1,太,上,三,十,六,部,真,經
8,1,太,上,一,乘,海,空,智,藏,經
9,1,高,上,玉,皇,本,行,集,經
10,1,高,上,玉,皇,本,行,集,經
```

The first column is "sequence ID", indicating different transactions; the second column is "event ID", indicating the temporal order within a transaction. Save the above data to a local file (such as "*D:/r/title.csv*") and execute the following commands:

```
> library(arulesSequences)
> x <- read_baskets("D:/r/title.csv",sep=",",info=list('sequenceID', 'eventID'))
> as(x, "data.frame")
```

Transaction ID.s Equence ID	Transaction ID. Event ID	Items
1	0	1{寶,度,經,量,靈,妙,品,人,上,無}
2	1	1{度,經,量,妙,品,人,上,始,無,音,元,直}
3	2	1 {道,德,解,經,始,說,天,先,元,註}
4	3	1 {藏,經,秘,內,上,無,真}
5	4	1 {昌,大,洞,極,經,上,太,文,無,仙,真,總}
6	5	1 {大,洞,經,清,上,真}
7	6	1 {大,洞,經,玉}
8	7	1 {部,經,六,三,上,十,太,真}
9	8	1 {藏,乘,海,經,空,上,太,一,智}
10	9	1 {本,高,皇,集,經,上,行,玉}

As we can see, the single characters in one "item" are rearranged automatically by the function to a different order other than the original order as the title text.

4.4. Frequent Pattern Mining of the TOC Title Text

Now we can perform the mining (the "*parameter*" argument in *cspade* function specifies that we want the minimum support to be 10%):

> y <- cspade(x, parameter=list(support=0.1))

> as(y, "data.frame")

```
sequence support
1 <{寶}> 0.1438038
2 <{道}> 0.1086812
3 <{洞}> 0.1669980
4 <{經}> 0.4254473
5 <{靈}> 0.1709742
6 <{清}> 0.1285620
7 <{上}> 0.3638171
8 <{太}> 0.2982107
9 <{天}> 0.1053678
10 <{玄}> 0.1590457
11 <{真}> 0.2670643
12 <{經,真}> 0.1391650
13 <{上,真}> 0.1013917
14 <{經,太}> 0.1981445
15 <{上,太}> 0.2511597
16 <{經,上,太}> 0.1769384
17 <{洞,上}> 0.1086812
18 <{經,上}> 0.2345924
19 <{清,上}> 0.1033797
20 <{寶,靈}> 0.1166335
21 <{洞,經}> 0.1047051
```

The result is self-explaining, but some of the frequent sequences, like {上(holy),太(superior)}, is confusing. Obviously, the order of appearance of "太(superior)" and "上(holy)" is incorrect; this is the side effect of the *read_baskets* function when creating the *transactions* object. To solve this problem, we have to split one title text into multiple "event" to ensure the order. The resulting dataset is provided in [10] (the *canon_title_multievent_csv.php* page) and has the following form:

```
0, 1, 靈
0, 2, 寶
0, 3, 無
0, 4, 量
0, 5, 度
0, 6, 人
0, 7, 上
0, 8, 品
0, 9, 妙
0, 10, 經
1, 1, 元
1, 2, 始
1, 3, 無
1, 4, 量
1, 5, 度
1, 6, 人
1, 7, 上
1, 8, 品
1, 9, 妙
1, 10, 經
1, 11, 直
1, 12, 音
```

Save the above data to a file (such as "D:/r/title1.csv") and perform the mining process:

> x <- read_baskets("D:/r/title1.csv",sep=",",info=list('sequenceID', 'eventID'))

> x

```
transactions in sparse format with
12388 transactions (rows) and
987 items (columns)
```

> s2 <- cspade(x, parameter=list(support=0.1))

> s2

```
set of 19 sequences
```

> as(s2, "data.frame");

```
sequence support
1 <{寶}> 0.1438038
2 <{道}> 0.1086812
3 <{洞}> 0.1669980
4 <{經}> 0.4254473
5 <{靈}> 0.1709742
6 <{清}> 0.1285620
7 <{上}> 0.3638171
8 <{太}> 0.2982107
9 <{天}> 0.1053678
10 <{玄}> 0.1590457
11 <{真}> 0.2670643
12 <{太},{上}> 0.2405567
13 <{上},{清}> 0.1007290
14 <{洞},{經}> 0.1047051
15 <{上},{經}> 0.2345924
16 <{太},{經}> 0.1974818
17 <{真},{經}> 0.1371769
18 <{太},{上},{經}> 0.1729622
19 <{靈},{寶}> 0.1153082
```

Note: if you get a "system invocation failed" when calling *cspade*, try reduce the support value and re-execute the mining command.

These frequent patterns provide valuable information for understanding and segmentation (tokenization) of ancient Chinese text.

5. SEGMENTATION OF CHINESE TEXT

As we have mentioned before, segmentation of Chinese text is more complicated than that of English. Fortunately there are several segmentation algorithms that have already be well implemented and ready for use for our application. One of them is "mmseg4j" [11], an MMSEG algorithm [12] implementation in Java language.

The reason that we can use a Java program for our R project is that we can call Java functions within the R script. Package "rJava" [13] provides us with this probability.

You must utilize at least two functions of rJava to create a Java function call:

(1) *.jnew*: this function creates a new Java object. The calling format of *.jnew* is:

.jnew(class, ..., check=TRUE, silent=check)

Where class is a fully qualified class name in JNI notation (*e.g.* "java/lang/String"); "…" may be any parameters that will be passed to the corresponding constructor; the parameter types are determined automatically and/or taken from the *jobjRef* object. This function returns the reference (*jobjRef*) to the newly created object or null-reference if something went wrong.

Here is a straight forward example to show a minimized java AWT window:

f <- .jnew("java/awt/Frame","Hello")

.jcall(f,,"setVisible",TRUE)

(2) .jcall: calls a Java method with the supplied arguments. The calling format is:

.jcall(obj, returnSig = "V", method, ..., evalArray = TRUE,

evalString = TRUE, check = TRUE, interface = "Rcall-Method",

simplify = FALSE, use.true.class = FALSE)

Arguments:

obj: Java object (jobjRef as returned by .jcall or .jnew) or fully qualified class name in JNI notation (*e.g.* "java/lang/String").

returnSig: Return signature in JNI notation (*e.g.* "V" for void, "[I" for int[] *etc.*). For convenience additional type "S" is supported and expanded to "Ljava/lang/String;", re-mapping "T" to represent the type short.

method: The name of the method to be called.

Any parameters that will be passed to the Java method. The parameter types are determined automatically and/or taken from the jobjRef object. All named parameters are discarded.

evalString: This flag determines whether string result is returned as characters or as Java object reference. It should be set to FALSE to because the Chinese string can not be evaluated correctly and must be returned as a Java object, and it can later be fetched using *.jstrVal.*

An Example:

> *.jcall("java/lang/System","S","getProperty","os.name")*

```
[1] "Windows XP"
```

By studying the source code of mmseg4j, we can find that the core function for segmentation is the "*segWords*" method in Complex.java in package *com.chenlb.mmseg4j. example*, and it has the form:

```
public String segWords(String txt, String wordSpilt) throws IOException {
  return segWords(new StringReader(txt), wordSpilt);
}
```

It is obvious that we can get the segmentation result by calling *segWords* method of a *Complex* object. So we write a R script function to do this chore:

```
mmseg_java <- function(text="请提供要分词的中文文本参数(Input Chinese text for segmentation)", delimiter=" ")
{
library(rJava)
.jpackage(name="rJava",jars="mmseg4j-all-1.8.5-with-dic.jar")
c <- .jnew("com/chenlb/mmseg4j/example/Complex");
outRef <- .jcall(c, "S", "segWords", text, delimiter, evalString = FALSE)
#.jstrVal returns the content of a string reference
.jstrVal(outRef)
}
```

Make sure you already have the "mmseg4j" library file (a jar file) copied to the "java" subdirectory of the *rJava* package before you call this R function.

The following are some straight forward examples of this function call:

> *mmseg_java()*

```
[1] "请 提供 要 分词 的 中文 文本 参数"
The input Chinese sentence are correctly segmented to words; more examples are given in the following:
```

> *mmseg_java("*道藏文献文本分析*")*

```
[1] "道藏 文献 文本 分析"
```

> *mmseg_java("*太上老君說常清靜經註*")*

```
[1] "太上 老君 說 常 清 靜 經 註"
```

> *mmseg_java("*太上老君说常清静经注*")*

```
[1] "太上 老君 说 常 清静 经 注"
```

While the first two examples give perfect results, if you carefully compare the last two examples, you can find that there are some slight differences between them, where the characters "清静 (serenity)" can be identified as a word, but the traditional form "清靜 (another writing style of 'serenity')" can not. This is a serious problem when perform text mining tasks on ancient Chinese. One probable way of solving this problem might be first sequential pattern mining to determine the proper word segmentation approach and build a customized dictionary system. The customized dictionary file can then be put in a directory specified by the "*mmseg.dic.path*" directive so it will be included for word segmentation.

The dictionary file for *mmseg4j* is simply a text file (UTF-8 format) with one word a line. Suppose we have such a file named *words-my.dic* (the file name must start with "words" and end with ".dic") and its content is:

```
太上老君(Taishang Laojun, an important God in Daoism)
註解(commentary)
```

(The first line is left blank intentionally to avoid the UTF-8 "BOM" issue in Windows system). The above dictionary file add 2 customized words for segmentation. Suppose we would like to put this file in *C:/dic*, the R code to activate this customized dictionary is:

> library(rJava)

> .jinit()

> p <- .jcall("java/lang/System", "S", "setProperty", "mmseg.dic.path", "C:/dic");

This script invokes Java System.setProperty method to set the "*mmseg.dic.path*" environment variable. Then we can use mmset_java for segmentation again:

> mmseg_java("太上老君说常清静经註解")

```
[1] "太上老君 说 常 清静 经 註解"
```

Although the customized dictionary can produce better performance for the segmentation process, the dictionary itself may take much more effort to built, especially when the ancient Chinese characters may have multiply allograph (variants, such as "注(commentary)" and "註(another writing of 'commentary')", "靜(quietness)" and "静(another writing of 'quietness')"). Systematic research on stop words, allograph and general lexicon are needed for the study of ancient Dao documents.

6. TEXT MINING ON CHINESE CORPUS

For a simple tryout, we now perform some text mining tasks upon Dao Canon Synopsis text. These text can be obtained from [14], but we have imported them into our MySQL database and you can easily download them as plain text data [15] (the *canon_synopsis.php* page).

The synopsis of one single Dao document has the form as the following (the text is taken from the synopsis of the 3rd document of Dao Canon):

```
元始說先天道德經註解五卷共27頁
南宋李嘉謀注解，元人張善淵刊行。五卷，收入《道藏》
洞真部本文類。本書經文《先天道德經》係模仿老子《道德經》
而作。分為妙、元、神、真、道五篇。每篇九章，一千字；共四
十五章、五千字。其內容晦澀，大致為解釋妙、元、神、真、
道五詞含義及其相生關係，並據此闡述妙徼、有無、動靜、陰陽、
五行、八卦、九宮、道德、道法自然、清靜無為之道理，
以及修真養生原理。李氏注解引述儒釋道三
教經書，逐章逐段解釋原文，較為明白曉暢。認為天地萬物生成
順序為由"妙"(虛無)而至"徼"(實有)，修道者應從徼而返歸妙本。
此為全書宗旨。
```

We use tm package [16] for text mining tasks. *tm* requires that a text corpus should be built before performing the mining tasks. A corpus can be built from multiple type of sources, such as a *data.frame* object (called *'DataframeSource'*), or a directory of document files (called *'DirSource'*), or other kind of sources (*URISource*, *XMLSource*, *etc*). The most commonly used Source type is the *DirSource*, where all document files are stored in a single directory. Because *tm* can not handle Chinese document directly, we have to transform the above text into a collection of "segmented" words, like this:

```
元. 始. 說. 先天. 道德. 經. 註. 解. 五卷. 共. 27. 頁. 南宋. 李. 嘉. 謀.
注解. 元. 人. 張. 善. 淵. 刊行. 五卷. 收入. 道藏. 洞. 真. 部. 本文. 類.
本. 書. 經. 文. 先天. 道德. 經. 係. 模仿. 老子. 道德. 經. 而作. 分. 為.
妙. 元. 神. 真. 道. 五. 篇. 每. 篇. 九章. 一. 千字. 共. 四十五. 章. 五千.
字. 其. 內. 容. 晦. 澀. 大致. 為. 解. 釋. 妙. 元. 神. 真. 道. 五. 詞. 含. 義.
及其. 相生. 關. 係. 並. 據. 此. 闡. 述. 妙. 徼. 有. 無. 動. 靜. 陰. 陽.
五行. 八卦. 九. 宮. 道德. 道. 法. 自然. 清. 靜. 無. 為. 之. 道理. 以及.
修. 真. 養. 生. 原理. 李. 氏. 注解. 引述. 儒. 釋. 道. 三. 教. 經. 書. 逐.
章. 逐. 段. 解. 釋. 原文. 較. 為. 明白. 曉. 暢. 認. 為. 天地. 萬. 物. 生成.
順. 序. 為. 由. 妙. 虛. 無. 而至. 徼. 實. 有. 修道. 者. 應. 從. 徼. 而. 返.
歸. 妙. 本. 此. 為. 全. 書. 宗旨
```

Obviously this is not the ideal segmentation result (*e.g.*, the word "注解(commentary)" is separated into two singular literals), perhaps you can deal with this problem later, but right now we have to build our text mining experiments on this base. The following R script retrieves synopsis data from MySQL, segments each text, and saves them as separate text files (as *1.txt, 2.txt, 3.txt*, *etc*.) to *D:/corpus*:

```
# first load Java support for R
library(rJava);
.jpackage(name="rJava",jars="mmseg4j-all-1.8.5-with-dic.jar")
# create the object for Chinese segmentation
segObj <- .jnew("com/chenlb/mmseg4j/example/Complex");
# load ODBC support for accessing MySQL
library(RODBC)
# connect to database, using UTF-8 encoding
ch <- odbcConnect("daotext-localhost", DBMSencoding="UTF-8");
# fetch all synopsis text
x <- sqlQuery(ch,"select * from daocanonsynopsis ")
# get the total number of documents
n <- dim(x)[1]
# set corpus directory (this directory must already exist)
cdir <- "D:/corpus/"
# using a loop to save segmented synopsis to files
i <- 1
while(i<=n){
s <- paste(as.character(x[i,2]), ". ", as.character(x[i,3]));
outRef<-.jcall(segObj, "S", "segWords", s, ". ",evalString = FALSE)
# output the segmentation result to a file
cat(.jstrVal(outRef), file=paste(cdir, x[i,1], ".txt", sep=""), sep="" )
i<-i+1;
}
```

After the corpus has been set up, we can build a *Source* object linked to this corpus:

> library(tm)

> txt <- DirSource("D:/corpus ", encoding = "GBK");

> intros <- Corpus(txt);

> inspect(intros);

```
A corpus with 1485 text documents
The metadata consists of 2 tag-value pairs and a data frame
Available tags are:
create_date creator
Available variables in the data frame are:
MetaID
$`10.txt`
高. 上. 玉皇. 本行. 集. 經. 三. 卷. 共. 14. 頁. 簡. 稱. 玉皇. 經. 撰. 人.
不. 詳. …
$`100.txt`
太上. 昇. 玄. 說. 消. 災. 護. 命. 妙. 經. 註. 一卷. 共. 4. 頁. 宋. 末. 元.
初. …
……
$`999.txt`
太上. 老君. 說. 常. 清. 靜. 經. 頌. 注. 一卷. 共. 3. 頁. 金朝. 道士. 默然.
子.
```

The next step is to build the *TermDocumentMatrix*; the syntax is:

TermDocumentMatrix(x, control = list())

Where *x* is a corpus, and *control* is a named list of control options. Note that because we inserted a dot (period) after every word when segmentation to let *tm* correctly differentiate the separated words, we have to carefully set the *removePunctuation* option to TRUE for control to get a correct result. Other options that could affect the result include *stopwords* and *wordLengths* (the latter is an integer vector of length 2. Words shorter than the minimum word length *wordLengths[1]* or longer than the maximum word length *wordLengths[2]* are discarded. Defaults to c(3, Inf), *i.e.*, a minimum word length of 3 characters). See the following example:

```
> dtm <- TermDocumentMatrix(intros,
+ control = list(wordLengths = c(2, Inf),
+    removePunctuation = TRUE,
+    removeNumbers = TRUE,
+    stopwords=TRUE) )
```

The above command constructs the term-document matrix for all the words with length longer that 2 after removes all punctuations and numbers and stop words.

Once the term-document matrix has been built, we can use it for further analysis; the simplest one is to find out the frequent words of a certain range:

```
> findFreqTerms(dtm, lowfreq=200, highfreq=Inf)
```

```
[1] "北宋" "本文" "道藏" "道教" "道士" "方法" "金丹" "南北朝"
[9] "收入" "太上" "文字" "修道" "一部" "一卷" "以及" "真人"
[17] "之道" "之一" "作者"
```

Or you can find associations in a term-document matrix for those terms that correlate with a term (*e.g.*, "丹田(center of super power)") more than a certain limit (*e.g.*, 20%):

```
> findAssocs(dtm, "丹田", 0.2)
```

```
丹田 自下 小成 假性 小童 住所 天庭 於下
1.00 0.47 0.39 0.38 0.38 0.38 0.33 0.28
之神 自上 赤子 春夏秋冬 三寸 有假 居住 安心
0.28 0.28 0.27 0.27 0.27 0.27 0.25 0.23
面部 有大 只在 老人
0.22 0.21 0.21 0.20
```

```
> dtm <- removeSparseTerms(dtm,0.99)
> findAssocs(dtm, "丹田", 0.05)
```

```
丹田 之神 皆有 中下 人身 三元 神化 性命 魂魄 天真 修道
1.00 0.28 0.16 0.16 0.10 0.10 0.09 0.09 0.08 0.08 0.07
亦有 之士 大有 二十四 九年 日月 升天 十六 四十 晚唐 一卷
0.07 0.07 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06 0.06
引述 之中 六腑 文言 先天 延年
0.06 0.06 0.05 0.05 0.05 0.05
```

Combined with other packages, such cluster, *kernlab*, class, *etc*., you can do cluster analysis, kernel principal components analysis, k-nearest neighbour classification, *etc*., with just one or two lines of code, and even use visualized graphics to present the result. Considering the confusion and ambiguity in understanding the ancient documents by human intuition, the application potential of text mining on Dao Canon is promising.

CONCLUSION

This Chapter performs basic text mining tasks on synopsis text of ancient Chinese Dao Canon documents, explains some techs & tricks in dealing with Chinese literals. While these experiments do provide us with some interesting perspectives on the ancient Daoist wisdom, it is still far from practical insights or innovative findings – and we don't even touch the scripture text themselves, whose formal digitization is still not complete. To advance these studies, it seems that the most urgent task is to build a reliable customized dictionary specially for the ancient Dao documents; and while this task can hopefully be achieved with the help of frequent pattern mining, a lot of hard work must be done before we can get an applicable result. The content of this Chapter is obviously only a beginning on utilizing text mining techniques in Dao Canon studies, but we hope it will lead a way to further research works to reveal more secrets of this ancient Chinese mysterious philosophy – the way to defy aging, and the way to better life.

CONFLICT OF INTEREST

The authors confirm that this article content has no conflict of interest.

ACKNOWLEDGEMENTS

This work is supported by Ningbo Natural Science Foundation (2013A610115) and Zhejiang Social Science Research Project (14NDJC056YB).

REFERENCES

[1] M. Jiang, "*Digital Resources of Traditional Chinese Daoism Culture for Free Download*," 2011. [Online] Available From: http://www.byscrj.cn/. [Accessed: July 1, 2012].

[2] Daoism Academic Information Website, "*Digital Dao Canon Database*," 2012. [Online] Available From: http://www.ctcwri.idv.tw/CMT000.htm. [Accessed: May 5, 2012].

[3] S. Wang, "Dao Canon Academic Research," 2012. [Online] Available From: http://www.daotext.org/index.html. [Accessed: July 1, 2012].

[4] M. Jiang, "*Tao Sutra Contents*," 2009. [Online] Available From: http://www.byscrj.cn/jmm/Tao_Sutra_Contents.htm. [Accessed: July 20, 2012, 2012].

[5] S. Wang, "*Dao Canon TOC CSV*," 2012. [Online] Available From: http://www.daotext.org/data/dao.csv. [Accessed: July 20, 2012].

[6] B. Ripley and M. Lapsley, "*RODBC: ODBC Database Access*," 2012. [Online] Available From: http://CRAN.R-project.org/package=RODBC. [Accessed: July 20, 2012].

[7] C. Buchta, M. Hahsler and D. Diaz, "*arulesSequences: Mining frequent sequences*," 2012. [Online] Available From: http://CRAN.R-project.org/package=arulesSequences. [Accessed: July 20, 2012].

[8] M. J. Zaki, "SPADE: an efficient algorithm for mining frequent sequences," *Machine Learning Journal*, vol. 42, pp. 31-60, 2001.

[9] S. Wang, "*Transaction data for Dao Canon TOC, single event*," 2012. [Online] Available From: http://www.daotext.org/canon_title_csv.php. [Accessed: July 1, 2012].

[10] S. Wang, "*Transaction data for Dao Canon TOC, multiple event*," 2012. [Online] Available From: http://localhost/canon_title_multievent_csv.php. [Accessed: July 1, 2012].

[11] B. Chan, "*mmseg4j: MMSEG for java lucene chinese analyzer, or for solr,*" 2012. [Online] Available From: http://code.google.com/p/mmseg4j/. [Accessed: July 23, 2012].

[12] C. Tsai, "*MMSEG: A Word Identification System for Mandarin Chinese Text Based on Two Variants of the Maximum Matching Algorithm*," 1996. [Online] Available From: http://technology.chtsai.org/mmseg/. [Accessed: July 23, 2012].

[13] S. Urbanek, "*rJava: Low-level R to Java interface*," 2011. [Online] Available From: http://CRAN.R-project.org/package=rJava. [Accessed: June 4, 2012].

[14] Daoism Academic Information Website, "*Synopsis of Zhengtong Daozang Contents - Dongzhen Volume*," 2012. [Online] Available From: http://www.ctcwri.idv.tw/CMT09道藏經目簡介/CMTS00-1.htm. [Accessed: May 5, 2012].

[15] S. Wang, "*Synopsis of Dao Canon*," 2012. [Online] Available From: http://www.daotext.org/canon_synopsis.php. [Accessed: July 20, 2012].

[16] I. Feinerer, K. Hornik and D. Meyer, "Text Mining Infrastructure in R," *Journal of Statistical Software*, vol. 25, pp. 30-36, 2008.

Received: June 10, 2015 Revised: July 29, 2015 Accepted: August 15, 2015