ISSN 1512-1232

NEW APPROACH OF TEXT MINING IN R

Mandal Shrabanti, Pal Anita
Departments of Mathematics,
National Institute of Technology Durgapur,
Durgapur-713209, West Bengal, India,
shrabmandal@gmail.com, anita.buie@gmail.com

Abstract

During the last decade text mining has become a widely used discipline utilizing statistical and machine learning methods. Text mining refers to the process of parsing a selection or corpus of text in order to identify certain aspects, such as the most frequently occurring word or phrase. The package named tm is available for text mining applications in R. tm package provides many important functions which are very efficiently handle the text mining. Here we try to append a method to this package for text correction & that helps us to create error free term-document matrix.

Keywords**:** *Text mining, tm package, term-document matrix.*

1. INTRODUCTION

Text mining encompasses a vast field of theoretical approaches and methods with one thing in common: text as input information. This allows various definitions, ranging from an extension of classical data mining to texts to more sophisticated formulations like the use of large online text collections to discover new facts and trends about the world itself" [1]. In general, text mining is an interdisciplinary field of activity amongst data mining, linguistics, computational statistics, and computer science. Standard techniques are text classification, text clustering, ontology and taxonomy creation, document summarization and latent corpus analysis. In addition a lot of techniques from related fields like information retrieval are commonly used.

Classical applications in text mining [2] come from the data mining community, like document clustering [3][4][5] and document classification [6]. For both the idea is to transform the text into a structured format based on term frequencies and subsequently apply standard data mining techniques. Typical applications in document clustering include grouping news articles or information service documents [7], whereas text categorization methods are used in, e.g., e-mail filters and automatic labeling of documents in business libraries [8]. Especially in the context of clustering, specific distance measures [9][10], like the Cosine, play an important role. With the advent of the World Wide Web, support for information retrieval tasks (carried out by, e.g., search engines and web robots) has quickly becomes an issue. Here, a possibly unstructured user query is first transformed into a structured format, which is then matched against texts coming from a data base. To build the latter, again, the challenge is to normalize unstructured input data to fulfill the repositories' requirements on information quality and structure, which often involves grammatical parsing.

During the last years, more innovative text mining methods have been used for analyses in various fields, e.g., in linguistic stylometry [11][12][13], where the probability that a specific author wrote a specific text is calculated by analyzing the author's writing style, or in search engines for learning rankings of documents from search engine logs of user behavior [14]. Latest developments in document exchange have brought up valuable concepts for automatic handling of texts.

The semantic web [15] propagates standardized formats for document exchange to enable agents to perform semantic operations on them. This is implemented by providing metadata and by annotating the text with tags. One key format is RDF [16] where efforts to handle this format have already been made in R [17] with the Bio-conductor project [18]. This development offers great edibility in document exchange. But with the growing popularity of XML based formats (e.g.,

ISSN 1512-1232
RDF/XML as a common representation for RDF) tools need to be able to handle XML documents and metadata. The benefit of text mining comes with the large amount of valuable information latent in texts which is not available in classical structured data formats for various reasons: text has always been the default way of storing information for hundreds of years, and mainly time, personal and cost contraints prohibit us from bringing texts into well structured formats (like data frames or tables).

Statistical contexts for text mining applications in research and business intelligence include latent semantic analysis techniques in bioinformatics [19], the usage of statistical methods for automatically investigating jurisdictions [20], plagiarism detection in universities and publishing houses, computer assisted cross-language information retrieval [21] or adaptive spam filters learning via statistical inference.

Further common scenarios are help desk inquiries [22], measuring customer preferences by analyzing qualitative interviews [23], automatic grading [24], fraud detection by investigating notification of claims, or parsing social network sites for specific patterns such as ideas for new products. Nowadays almost every major statistical computing product offers text mining capabilities, and many well-known data mining products provide solutions for text mining tasks.

2. THE tm PACKAGE

The **tm** package offers group of functionalities for managing text documents, abstracts the process of document manipulation and eases the usage of heterogeneous text formats in R. The package has integrated database backend support to minimize memory demands. An advanced meta data management is implemented for collections of text documents to alleviate the usage of large and with meta data enriched document sets. With the package ships native support for handling the Reuters 21578 data set, Gmane RSS feeds, e-mails, and several classic file formats (e.g. plain text, CSV text, or PDFs). The data structures and algorithms can be extended to fit custom demands, since the package is designed in a modular way to enable easy integration of new file formats, readers, transformations and filter operations. **tm** provides easy access to preprocessing and manipulation mechanisms such as whitespace removal, stemming, or conversion between file formats. Further generic filter architecture is available in order to filter documents for certain criteria, or perform full text search. The package supports the import of document collections to term-document matrices, and string kernels can be easily constructed from text documents. The following are the basic operations available in tm package [25].

2.1. DATA IMPORT

Corpus represents collection of text documents that is main structure for managing documents in tm package. A corpus offers an abstract concept, and there can exist several implementations in parallel. The default implementation is the so-called VCorpus (short for Volatile Corpus) which realizes semantics as known from most R objects: corpora are R objects held fully in memory. We denote this as volatile since once the R object is destroyed, the whole corpus is lost.

Such a volatile corpus can be created by using constructor VCorpus(x, readerControl). Another implementation is the PCorpus which stands for Permanent Corpus, i.e., the documents are physically stored outside of R (e.g., in a database), corresponding R objects are basically only pointers to external structures, and changes to the underlying corpus are reflected to all R objects associated with it. Compared to the volatile corpus the corpus encapsulated by a permanent corpus object is not destroyed if the corresponding R object is released.

Within the corpus constructor, x must be a Source object which abstracts the input location. tm provides a set of predefined sources, e.g., DirSource, VectorSource, or DataframeSource, which handle a directory, a vector interpreting each component as document, or data frame like structures (like CSV _les), respectively. Except DirSource, which is designed solely for directories on a file system, and VectorSource, which only accepts (character) vectors, most other implemented sources can take connections as input (a character string is interpreted as file path). getSources() lists available sources, and users can create their own sources.

ISSN 1512-1232

The second argument readerControl of the corpus constructor has to be a list with the named components reader and language. The first component reader constructs a text document from elements delivered by a source. The tm package ships with several readers (e.g., readPlain(), readPDF(), readDOC(), . . .). See getReaders() for an up-to-date list of available readers. Each source has a default reader which can be overridden. E.g., for DirSource the default just reads in the input files and interprets their content as text.Finally, the second component language sets the texts' language (preferably using ISO 639-2 codes). In case of a permanent corpus, a third argument dbControl has to be a list with the named components dbName giving the filename holding the sourced out objects (i.e., the database), and dbType holding a valid database type as supported by package filehash. Activated database support reduces the memory demand, however, access gets slower since each operation is limited by the hard disk's read and write capabilities

2.2. DATA EXPORT

Suppose we have created a corpus via manipulating other objects in R, thus do not have the texts already stored on a hard disk, and we want to save the text documents to disk, we can simply use function writeCorpus().

2.3. INSPECTING CORPORA

For inspecting the corpus tm package provides the print() which hide the raw amount of information. print() gives a concise overview whereas the full content of text documents is displayed with inspect().

2.4. TRANSFORMATIONS

Once we have a corpus we typically want to modify the documents in it, e.g., stemming, stop word removal, et cetera. In tm, all this functionality is subsumed into the concept of a transformation. Transformations are done via the tm_map() function which applies (maps) a function to all elements of the corpus. Basically, all transformations work on single text documents and tm_map() just applies them to all documents in a corpus.

2.5. FILTERS

Often it is of special interest to filter out documents satisfying given properties. For this purpose the function tm_filter is designed. It is possible to write custom filter functions which get applied to each document in the corpus. Alternatively, we can create indices based on selections and subset the corpus with them. E.g., the following statement filters out those documents having an ID equal to "237" and the string "INDONESIA SEEN AT CROSSROADS OVER ECONOMIC CHANGE" as their heading.

2.6. METADATA MANAGEMENT

Metadata is used to annotate text documents or whole corpora with additional information. The easiest way to accomplish this with tm is to use the meta() function. A text document has a few predefined attributes like author but can be extended with an arbitrary number of additional user-defined metadata tags. These additional metadata tags are individually attached to a single text document. From a corpus perspective these metadata attachments are locally stored together with each individual text document. Alternatively to meta() the function DublinCore() provides a full mapping between Simple Dublin Core metadata and tm metadata structures and can be similarly used to get and set metadata information for text documents

3. APPEND FUNCTION

From the above discussion about the tm package it is cleared to all the it has include different function for data import, data export, document transforming , corpus filtering and for creating term-document matrixes. After analyzing the functions of corpus transforming and filtering we have summarized these points.

ISSN 1512-1232

i. For searching the document we can easily remove the stop words, white space and concatenate token.
ii. By using the above character vector can be initialized by tokens.
iii. We can directly search the documents from the corpus by any token.

When user want to filter out the documents by using the entered token tm package offers the desired output. Suppose a user enter token with wrong spelling then the entire term-document matrix will be formed based on the given wrong token. Therefore required output may not be come.To overcome this scenario a function can be added for generating the list of possible tokens of character vector. Then depends on this character vector term- document matrix will be formed.

4. EXPERIMENT & RESULT

This module helps us to correct spelling of tokens. For correcting the spelling it generates the possible combination of input keyword .In the next module we will search all the possible tokens from the files of corpus. For doing so we use a matrix. This matrix has 26 rows and 3 columns .Every row contains that three alphabets which are similar sounding words and uses make frequently typing mistake because of similar looks and appearance in nearest distance in keyboard. Here we input the similar matrix which is given bellow:

$$\text{Simi1} = \begin{bmatrix} a & e & o \\ b & v & d \\ c & k & e \\ d & b & v \\ . & . & . \\ . & . & . \\ w & v & q \\ x & a & e \\ y & o & v \\ z & j & g \end{bmatrix}$$

Figure 1: simi1 matrix

The steps which are followed in text correction module are given below

Step 1: Accept the user keyword.
Step 2: Perform the dot (.) operation between each row of simil matrix and input keyword.
Step 3: The output of dot (.) operation is the coordinate of input keyword stored in input_vectector
Step 4: Perform the dot (.) operation between each category files and simil matrix.
Step 5: All the coordinates of the same files store in an individual_vector matrix.
Step 6: Calculate the distance between input_vector and each indivisual_vector.
Step 7: Identify the minimum distance and fetch keyword from the minimum distance location from each categorical_data_files.
Step 8: Store all words in text_list matrix.

```
text_list =

    'INDIA'
    'AID'
    'GRAIN'
    'MISE'
    'INDIA'
    'GANDHI'
```

Figure 2: text_list

ISSN 1512-1232

The objective is to provide the correct result for all times even if the input keyword is wrongly spelled. When a user enters the token then this function gives the list of tokens which are similar to the entered one.

Suppose when a user want to search a files which contains “India” then he /she give the “India” in search data field. And click on “search files” filed then the following output will appear

Figure 3: GUI for searching “India”

When a user gives the incorrect spelling of “India” suppose he gives “Indea”.

Then also same files will be opened. It is observed that always the same filed will be opened. This is shown in Figure 4.

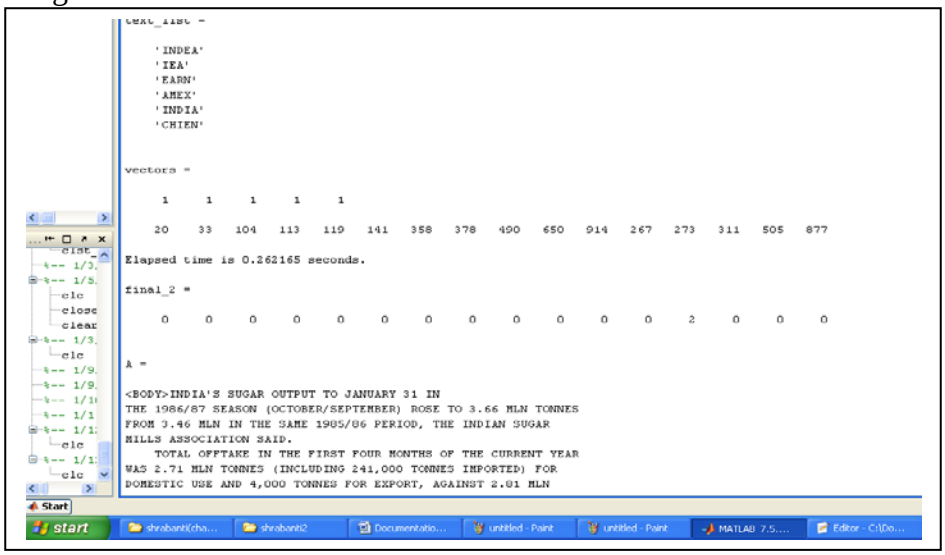

Figure 4: GUI for searching “Indea”

5. CONCLUSION & FUTURE WORK

The term text mining becomes more interesting when tm packaged has been lunched. Tm package provides all the possible functions for mining the text.Actuallly it has open a new door for text mining. It has given a facility to append the user function according to the requirement. So after pursue the tm package it is found that if a function like token correction will be appended to this package may provide the better output. There are many other techniques for correcting the text is

ISSN 1512-1232

available. We can append them. To analysis the term-document matrix is very challenging just because of huge data. In future we have to propose some easiest method to handle term-documents matrix easily.

REFERENCES:

1. Hearst M , “Untangling Text Data Mining” , In \Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics, pp. 3-10. Association for Computational Linguistics, Morristown, NJ, USA. ISBN 1-55860-609-3,1999.
2. Weiss S, Indurkhya N, Zhang T, Damerau F , “Text Mining: Predictive Methods for Analyzing Unstructured Information” , Springer-Verlag. ISBN 0387954333, 2004
3. Zhao Y, Karypis G., “Topic-driven Clustering for Document Datasets." ,In \Proceedings of the 2005 SIAM International Conference on Data Mining (SDM05), pp. 358-369 , 2005.
4. Boley D.,” Hierarchical Taxonomies Using Divise Partitioning”, Technical Report 98-012, University of Minnesota , 1998.
5. Boley D, Gini M, Gross R, Han EH, Karypis G, Kumar V, Mobasher B, Moore J, Hastings K “Partitioning-based Clustering for Web Document Categorization”, Decision Support Systems, 27(3), 329-341. ISSN 0167-9236, 1999.
6. Sebastiani F., “Machine Learning in Automated Text Categorization”,"ACM Computing Surveys, 34(1), 1-47. ISSN 0360-0300, 2002.
7. Steinbach M, Karypis G, Kumar V,” A Comparison of Document Clustering Techniques." In KDD Workshop on Text Mining", 2000.
8. Miller TW,” Data and Text Mining”, Pearson Education International, 2005.
9. Zhao Y, Karypis G.,”Empirical and Theoretical Comparisons of Selected Criterion Functions for Document Clustering", Machine Learning, 55(3), 311-331. ISSN 0885-6125, 2004.
10. Strehl A, Ghosh J, Mooney RJ.,”Impact of Similarity Measures on Web-page Clustering”, In Proc. AAAI Workshop on AI for Web Search (AAAI 2000), Austin," pp.58-64. AAAI/MIT Press. ISBN 1-57735-116-9. , 2000.
11. Gir_on J, Ginebra J, Riba A .,” Bayesian Analysis of a Multinomial Sequence and Homogeneity of Literary Style", The American Statistician, 59(1), 19-30, 2005.
12. Nilo J, Binongo G .”Who Wrote the 15th Book of Oz? An Application of Multivariate Analysis to Authorship Attribution", Chance, 16(2), 9-17, 2003.
13. Holmes D, Kardos J.” Who was the Author? An Introduction to Stylometry", Chance, 16(2), 5-8, 2003.
14. Radlinski F, Joachims T ,,” Active Exploration for Learning Rankings from Clickthrough Data" , In Proceedings of the 13th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining," ACM, New York, NY, USA ., pp. 570-579 , 2007.
15. Berners-Lee T, Hendler J, Lassila O., “The Semantic Web”, Scientific American, pp.34-43, 2001.
16. Manola F, Miller E., “ RDF Primer”, World Wide Web Consortium., 2004.
17. R Development Core Team R.,” A Language and Environment for Statistical Computing”, R Foundation for Statistical Computing, Vienna, Austria. ISBN 3-900051-07-0, 2007.
18. Gentleman RC, Carey VJ, Bates DM, Bolstad B, Dettling M, Dudoit S, Ellis B, Gautier L, Ge Y, Gentry J, Hornik K, Hothorn T, Huber W, Iacus S, Irizarry R, Leisch F, Li C, Maechler M, Rossini AJ, Sawitzki G, Smith C, Smyth G, Tierney L, Yang JY, Zhang J,”Bioconductor: Open Software Development for Computational Biology and Bioinformatics", Genome Biology, 5(10), R80.1-16, 2004.

ISSN 1512-1232
19. Dong QW, Wang XL, Lin L. “Application of Latent Semantic Analysis to Protein Remote Homology Detection", Bioinformatics, 22(3), pp. 285-290., ISSN 1367-4803, 2006.
20. Feinerer I, Hornik K (2007). Text Mining of Supreme Administrative Court Jurisdictions." In C Preisach, H Burkhardt, L Schmidt-Thieme, R Decker (eds.), “Data Analysis, Machine Learning, and Applications “, Proceedings of the 31st Annual Conference of the Gesellschaft fur Klassi_kation e.V., Freiburg, Germany," Studies in Classification, Data Analysis, and Knowledge Organization. Springer-Verlag, March 7-9, 2007.
21. Li Y, Shawe-Taylor J. ,”Using KCCA for Japanese-English Cross-Language Information Retrieval and Classification" , Journal of Intelligent Information Systems, 2007.
22. Sakurai S, Suyama A ., “An E-mail Analysis Method Based on Text Mining Techniques" , Applied Soft Computing, 6(1),pp. 62-71, 2005.
23. Feinerer I, Wild F. , “Automated Coding of Qualitative Interviews with Latent Semantic Analysis”,In H Mayr, D Karagiannis (eds.), Proceedings of the 6th International Conference on Information Systems Technology and its Applications, Kharkiv, Ukraine," volume 107 of Lecture Notes in Informatics, pp. 66-77. Gesellschaft fur Informatik e.V., Bonn, Germany, May 23-25, 2007.
24. Wu YF, Chen X., “eLearning Assessment through Textual Analysis of Class Discussions", In Fifth IEEE International Conference on Advanced Learning Technologies," pp. 388-390. , 2005.
25. Feinerer Ingo, “Introduction to the tm Package”, Text Mining in R, June 10, 2014.

Article received: 2015-03-13