akc: A Tidy Framework for Automatic Knowledge Classification in R

by Tian-Yuan Huang, Li Li, and Liying Yang

Abstract Knowledge classification is an extensive and effective approach for domain knowledge management. To automatically extract and organize knowledge from unstructured textualized data is desirable and appealing in various circumstances. In this paper, the tidy framework for automatic knowledge classification supported by the **akc** package is introduced. With the powerful support from the R ecosystem, the **akc** framework could handle multiple procedures in data science workflow, including text cleaning, keyword extraction, synonyms consolidation and data presentation. While focusing on bibliometric analysis, the **akc** package is extensible to be used in other contexts. This paper introduces the framework and its features in detail. Specific examples are given to guide the potential users and developers to participate in open science of text mining.

Introduction

Co-word analysis has long been used for knowledge discovery, especially in the field of library and information science (Callon et al., 1986). Based on co-occurrence relationships between words or phrases, this method could provide quantitative evidence of information linkages, mapping the association and evolution of knowledge over time. In conjunction with social network analysis (SNA), co-word analysis could be escalated and yield more informative results, such as topic popularity (Huang and Zhao, 2019) and knowledge grouping (Khasseh et al., 2017). Meanwhile, in the area of network science many community detection algorithms have been proposed to unveil the topological structure of the network (Fortunato, 2010; Javed et al., 2018). These methods have then been incorporated into the co-word analysis, assisting to group components in the co-word network. Currently, the co-word analysis based on community detection is flourishing across various fields, including information science, social science and medical science (Hu et al., 2013; Hu and Zhang, 2015; Leung et al., 2017; Baziyad et al., 2019).

For implementation, interactive software applications (e.g. CiteSpace and VOSviewer) have provided freely available toolkits for automatic co-word analysis, making this technique even more popular. Interactive software applications are generally friendlier to users, but they might not be flexible enough for the whole data science workflow. In addition, the manual adjustments could be variant, bringing extra risks to the research reproducibility. In this paper, we have designed a flexible framework for automatic knowledge classification, and presented an open software package **akc** supported by R ecosystem for implementation. Based on community detection in co-occurrence network, the package could conduct unsupervised classification on the knowledge represented by extracted keywords. Moreover, the framework would handle tasks such as data cleaning and keyword merging in the upstream of data science workflow, whereas in the downstream it provides both summarized table and visualized figure of knowledge grouping. While the package was first designed for academic knowledge classification in bibliometric analysis, the framework is rather general, so as to benefit a broader audience who are interested in text mining, network science and knowledge discovery.

Background

Classification could be identified as a meaningful clustering of experience, turning information into structured knowledge (Kwasnik, 1999). In bibliometric research, this method has been frequently used to group domain knowledge represented by author keywords, usually listed as a part of co-word analysis, keyword analysis or knowledge mapping (He, 1999; Hu et al., 2013; Leung et al., 2017; Li et al., 2017; Wang and Chai, 2018). While all named as (unsupervised) classification or clustering, the algorithm behind could vary widely. For instance, some researches have utilized hierarchical clustering to group keywords in into different themes (Hu and Zhang, 2015; Khasseh et al., 2017), whereas the studies applying VOSviewer have adopted a kind of weighted variant of modularity-based clustering with a resolution parameter to identify smaller clusters (Van Eck and Waltman, 2010). In the framework of **akc**, we have utilized the modularity-based clustering method known as community detection in network science (Newman, 2004; Murata, 2010). These functions are supported by the **igraph** package. Main detection algorithms implemented in **akc** include Edge betweenness (Girvan and Newman, 2002), Fastgreedy (Clauset et al., 2004), Infomap (Rosvall and Bergstrom, 2007; Rosvall et al., 2009), Label propagation (Raghavan et al., 2007), Leading eigenvector (Newman, 2006), Multilevel

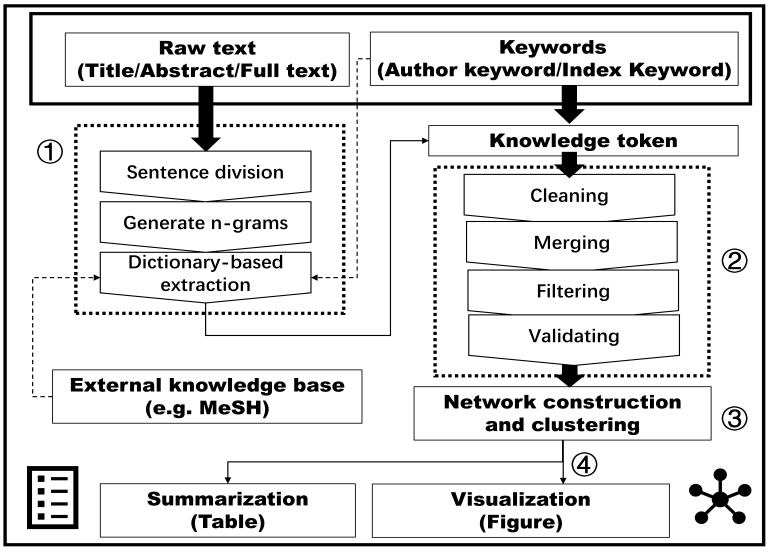

Figure 1: The design of akc framework.

(Blondel et al., 2008), Spinglass (Reichardt and Bornholdt, 2006) and Walktrap (Pons and Latapy, 2005). The details of these algorithms and their comparisons have been discussed in the previous studies (de Sousa and Zhao, 2014; Yang et al., 2016; Garg and Rani, 2017; Amrahov and Tugrul, 2018).

In practical application, the classification result is susceptible to data variation. The upstream procedures, such as information retrieval, data cleaning and word sense disambiguation, play vital roles in automatic knowledge classification. For bibliometric analysis, the author keyword field provides valuable source of scientific knowledge. These keywords are good representation of domain knowledge and could be used directly for analysis. In addition, such collection of keywords from papers published in specific fields could provide professional dictionary for information retrieval, such as keyword extraction from raw text in the title, abstract and full text of literature. While focuses on automatic knowledge classification based on community detection in keyword co-occurrence network, the **akc** framework also provides utilities for keyword-based knowledge retrieval, text cleaning, synonyms merging and data visualization in data science workflow. These tasks might have different requirements in specific backgrounds. Currently, **akc** concentrates on keyword-based bibliometric analysis of scientific literature. Nonetheless, the R ecosystem is versatile, and the popular tidy data framework is flexible to extend to various data science tasks from other different fields (Wickham et al., 2014; Wickham and Grolemund, 2016), which benefits both end-users and software developers. In addition, when users have more specific needs in their tasks, they could easily seek other powerful facilities from R community. For instance, **akc** provides functions to extract keywords using n-grams model (utilizing facilities provided by **tidytext**), but skip-gram modelling is not supported currently. This functionality, on the other hand, could be provided in **tokenizers** or **quanteda** package in R. A greater picture of natural language processing (NLP) in R could be found in the CRAN Task View: Natural Language Processing.

Framework

An overview of the framework is given in Figure 1. Note that the name **akc** refers to the overall framework for automatic keyword classification as well as the released R package in this paper. The whole workflow could be largely divided into four procedures: (1) Keyword extraction (optional); (2) Keyword preprocessing; (3) Network construction and clustering; (4) Results presentation.

(1) Keyword extraction (optional)

In bibliometric meta-data entries, the textual information of title, abstract and keyword are usually provided for each paper. If the keywords were used directly, there is no need to do information retrieval. Then we could directly skip this procedure and start from keyword preprocessing. However, sometimes the keyword field is missing, then we would need to extract the keywords from raw text in the title, abstract or full text with an external dictionary. At other times, one might want to get more keywords and their co-occurrence relationships from each entry. In such case, the keyword field could serve as an internal dictionary for information retrieval in the provided raw text.

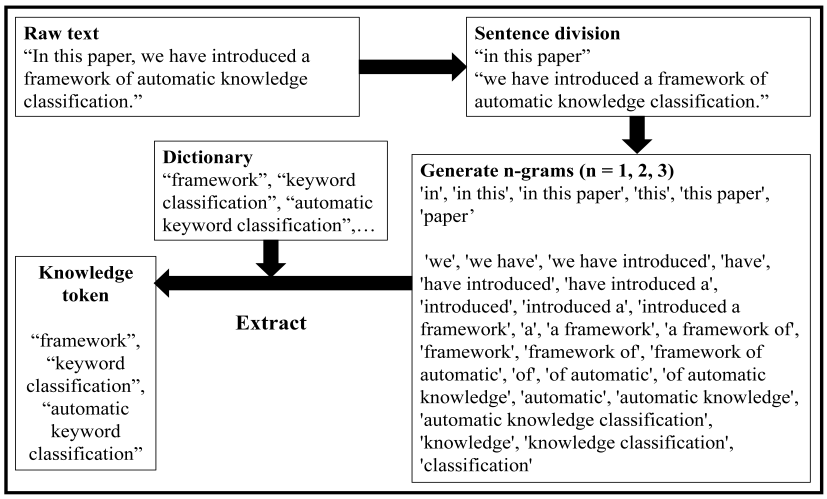

Figure 2: An example of keyword extraction procedure (the letters are automatically turned to lower case in sentence division).

Figure 2 has displayed an example of keyword extraction procedure. First, the raw text would be split into sub-sentences (clauses), this could suppress the generation of cross-clause n-grams. Then the sub-sentences would be tokenized into n-grams. The `n` could be specified by the users, inspecting the average number of words in keyword phrases might help decide the maximum number of `n`. Finally, a filter is made. Only tokens that have emerged in the user-defined dictionary are retained for further analysis. The whole keyword extraction procedure could be implemented automatically with `keyword_extract` function in **akc**.

(2) Keyword preprocessing

In practice, the textualized contents are seldom clean enough to implement analysis directly. Therefore, the upstream data cleaning process is inevitable. In keyword preprocessing procedure of **akc** framework, the cleaning part would take care of some details in the preprocess, such as convert the letters to lower case and remove parentheses and contents inside (optional). For merging part, **akc** help merge the synonymous phrases according to their lemmas or stems. While using lemmatization and stem extracting would get abnormal knowledge tokens, here in **akc** we have designed a conversion rule to tackle this problem. We first get the lemmatized or stemmed form of keywords, then group the keywords by their lemma or stem, and let the most frequent keyword in the group to represent the original keyword. This step could be realized by `keyword_merge` function in **akc** package. An example could be found in Figure 3. After keyword merging, there might still be too many keywords included in the analysis, which poses great burden for computation in the subsequent procedures. Therefore, a filter should be carried out here, it could exclude the infrequent terms, or extract top TF-IDF terms, or use any criteria that meets the need. Last, a manual validation should be carried out to ensure the final data quality.

(3) Network construction and clustering

Based on keyword co-occurrence relationship, the keyword pairs would form an edge list and be used to construct an undirected network. Then the facilities provided by **igraph** package would automatically group the nodes (representing the keywords). This procedure could be achieved by using `keyword_group` function in **akc**.

(4) Results presentation

Currently, there are two kinds of output presented by **akc**. One is a summarized result, namely a table with group number and keyword collections (attached with frequency). Another is network visualization, which has two modes. The local mode provides a keyword co-occurrence network by group (use facets in **ggplot2**), whereas the global mode displays the whole network structure. Note that one might include huge number of keywords and make a huge network, but in presentation the users could choose how many keywords from each group to be displayed. More details could be found in the following sections.

ID	Original form	Lemmatized form	Merged form
1	**higher education**	high education	**higher education**
2	**higher education**	high education	
3	high educations	high education	
4	higher educations	high education	
5	high education	high education	
6	**higher education**	high education	

Figure 3: An example of keyword merging conversion rule applied in akc. The keywords with the same lemma (or stem) would be merged to the highest frequency keyword in the original form.

The **akc** framework could never be built without the powerful support provided by R community. The **akc** package was developed under R environment, and main packages imported to **akc** framework include **data.table** (for high performance computing), **dplyr** (for tidy data manipulation), **ggplot2** (for data visualization), **ggraph** (for network visualization), **ggwordcloud** (for word cloud visualization), **igraph** (for network analysis), **stringr** (for string operations), **textstem** (for lemmatizing and stemming), **tidygraph** (for network data manipulation) and **tidytext** (for tidy tokenization). Getting more understandings on these R packages could help users utilize more alternative functions, so as to complete more specific and complex tasks. Hopefully, the users might also become potential developers of the **akc** framework in the future.

Example

This section shows how **akc** can be used in a real case. A collection of bibliometric data of *R Journal* from 2009 to 2021 is used in this example. The data of this example can be accessed in the GitHub repository. Only **akc** package would be used in this workflow, first we would load the package and import the data in the R environment.

```
library(akc)
rj_bib = readRDS("./rj_bib.rds")
rj_bib
```

```
#> # A tibble: 568 x 4
#>       id title                                                abstract  year
#>    <int> <chr>                                                <chr>    <dbl>
#>  1     1 Aspects of the Social Organization and Trajectory of th~ Based p~  2009
#>  2     2 asympTest: A Simple R Package for Classical Parametric ~ asympTe~  2009
#>  3     3 ConvergenceConcepts: An R Package to Investigate Variou~ Converg~  2009
#>  4     4 copas: An R package for Fitting the Copas Selection Mod~ This ar~  2009
#>  5     5 Party on!                                              Random ~  2009
#>  6     6 Rattle: A Data Mining GUI for R                        Data mi~  2009
#>  7     7 sos: Searching Help Pages of R Packages                The sos~  2009
#>  8     8 The New R Help System                                  Version~  2009
#>  9     9 Transitioning to R: Replicating SAS, Stata, and SUDAAN ~ Statist~  2009
#> 10    10 Bayesian Estimation of the GARCH(1,1) Model with Studen~ This no~  2010
#> # ... with 558 more rows
```

`rj_bib` is a data frame with 4 columns, including *id* (Paper ID), *title* (Title of paper), *abstract* (Abstract of paper) and *year* (Publcation year of paper). Papers in *R Journal* do not contain keyword field, thus we would have to extract the keywords from the title or abstract field (first step in Figure 1). Here in our case, we would use abstract field as our data source. In addition, we would need a user-defined dictionary to extract the keywords, otherwise all the n-grams (meaningful or meaningless) would be extracted and the results would include redundant noise.

```
# import the user-defined dictionary
rj_user_dict = readRDS("./rj_user_dict.rds")
rj_user_dict
```

```
#> # A tibble: 627 x 1
#>    keyword
#>    <chr>
#>  1 seasonal-adjustment
#>  2 unit roots
#>  3 transformations
#>  4 decomposition
#>  5 combination
#>  6 integration
#>  7 competition
#>  8 regression
#>  9 accuracy
#> 10 symmetry
#> # ... with 617 more rows
```

Note that the dictionary should be a data.frame with only one column named "keyword". The user can also use `make_dict` function to build the dictionary data.frame with a string vector. This function removes duplicated phrases, turn them to lower case and sort them, which potentially improving the efficiency for the following processes.

```
rj_dict = make_dict(rj_user_dict$keyword)
```

With the bibliometric data (`rj_bib`) and dictionary data (`rj_dict`), we could start the workflow provided in Figure 1.

(1) Keyword extraction

In this step, we need a bibliometric data table with simply two informative columns, namely paper ID (*id*) and the raw text field (in our case *abstract*). The parameter *dict* is also specified to extract only keywords emerging in the user-defined dictionary. The implementation is very simple.

```
rj_extract_keywords = rj_bib %>%
  keyword_extract(id = "id",text = "abstract",dict = rj_dict)
```

By default, only phrases range 1 to 4 in length are included as extracted keywords. The user can update this range using parameters `n_min` and `n_max` in `keyword_extract` function. These is also a `stopword` parameter, allowing users to exclude specific keywords in the extracted phrases. The output of `keyword_extract` is a pretty displayed data.frame (tibble,`tbl_df` class provided by **tibble** package) with two columns, namely paper ID (*id*) and the extracted keyword (*keyword*).

(2) Keyword preprocessing

For the preprocessing part, `keyword_clean` and `keyword_merge` would be implemented. In the cleaning part, the `keyword_clean` function would: 1) Split the text with separators (If no separators exist, skip); 2) Remove the contents in the parentheses (including the parentheses, optional); 3) Remove white spaces from start and end of string and reduces repeated white spaces inside a string; 4) Remove all the null character string and pure number sequences (optional); 5) Convert all letters to lower case; 6) Lemmatization (optional). The merging part has been illustrated in the previous section, thus would not be explained again. In the tidy workflow, the preprocessing could be implemented via:

```
rj_cleaned_keywords = rj_extract_keywords %>%
  keyword_clean() %>%
  keyword_merge()
```

No parameters are used in these functions because **akc** has been designed to input and output tibbles with consistent column names. If the users have data tables with different column names, specify them in arguments (`id` and `keyword`) provided by the functions. More details could be found in the help document (use `?keyword_clean` and `?keyword_merge` in the console).

(3) Network construction and clustering

To construct a keyword co-occurrence network, only a data table with two columns (with paper ID and keyword) is needed. All the details have been taken care of in the `keyword_group` function.

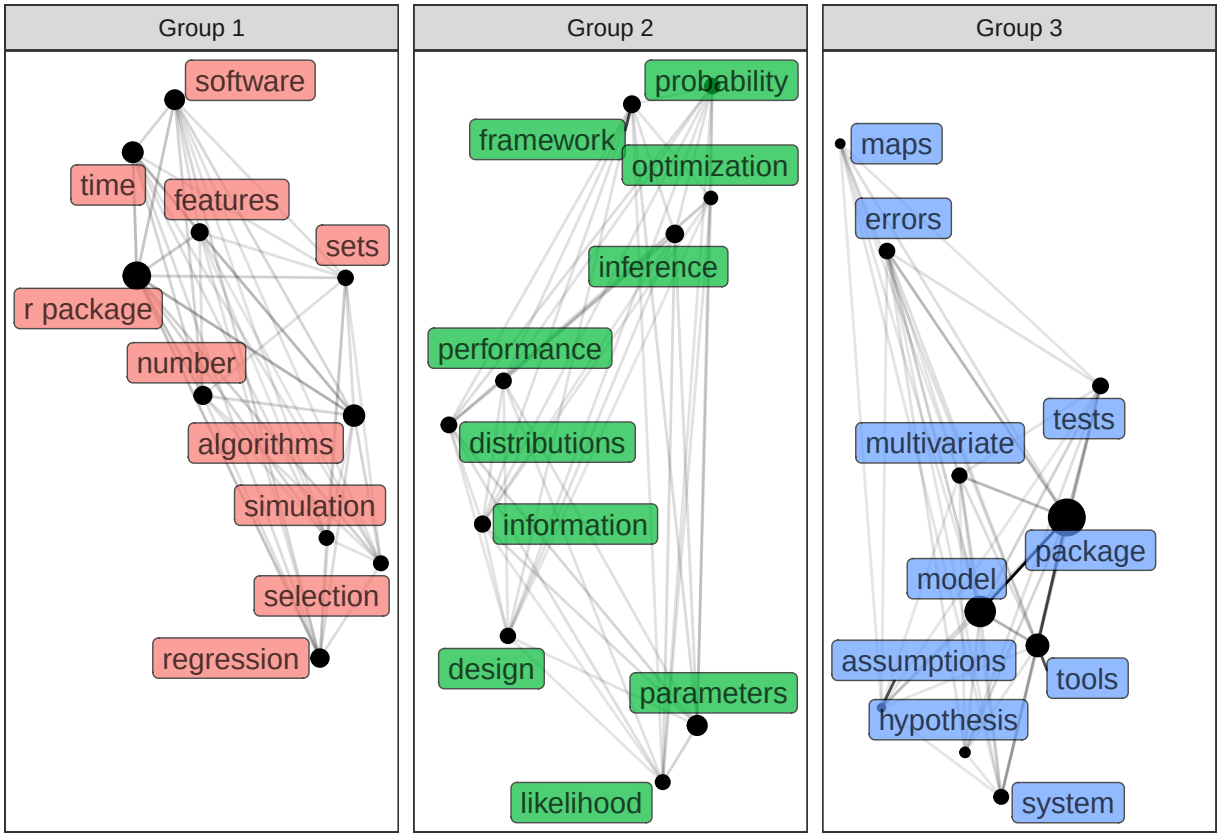

Figure 4: Network visualization provided by akc.

However, the user could specify: 1) the community detection function (use `com_detect_fun` argument); 2) the filter rule of keywords according to frequency (use `top` or `min_freq` argument, or both). In our example, we would use the default settings (utilizing Fastgreedy algorithm, only top 200 keywords by frequency would be included).

```
rj_network = rj_cleaned_keywords %>%
  keyword_group()
```

The output object `rj_network` is a `tbl_graph` class supported by **tidygraph**, this is a tidy data format containing the network data. Based on this data, we can present the results in various forms in the next section.

(4) Results presentation

Currently, there are two major ways to display the classified results in **akc**, namely network and table. A fast way to gain the network visualization is using `keyword_vis` function:

```
rj_network %>%
  keyword_vis()
```

In Figure 4, the keyword co-occurrence network is clustered into three groups. The size of nodes is proportional to the keyword frequency, while the transparency degree of edges is proportional to the co-occurrence relationship between keywords. For each group, only top 10 keywords by frequency are showed in each facet. If the user wants to dig into Group 1, `keyword_network` could be applied. Also, `max_nodes` parameter could be used to control how many nodes to be showed (in our case, we show 20 nodes in the visualization displayed in Figure 5).

```
rj_network %>%
  keyword_network(group_no = 1,max_nodes = 20)
```

Another displayed form is using table. This could be implemented by `keyword_table` via:

```
rj_table = rj_network %>%
  keyword_table()
```

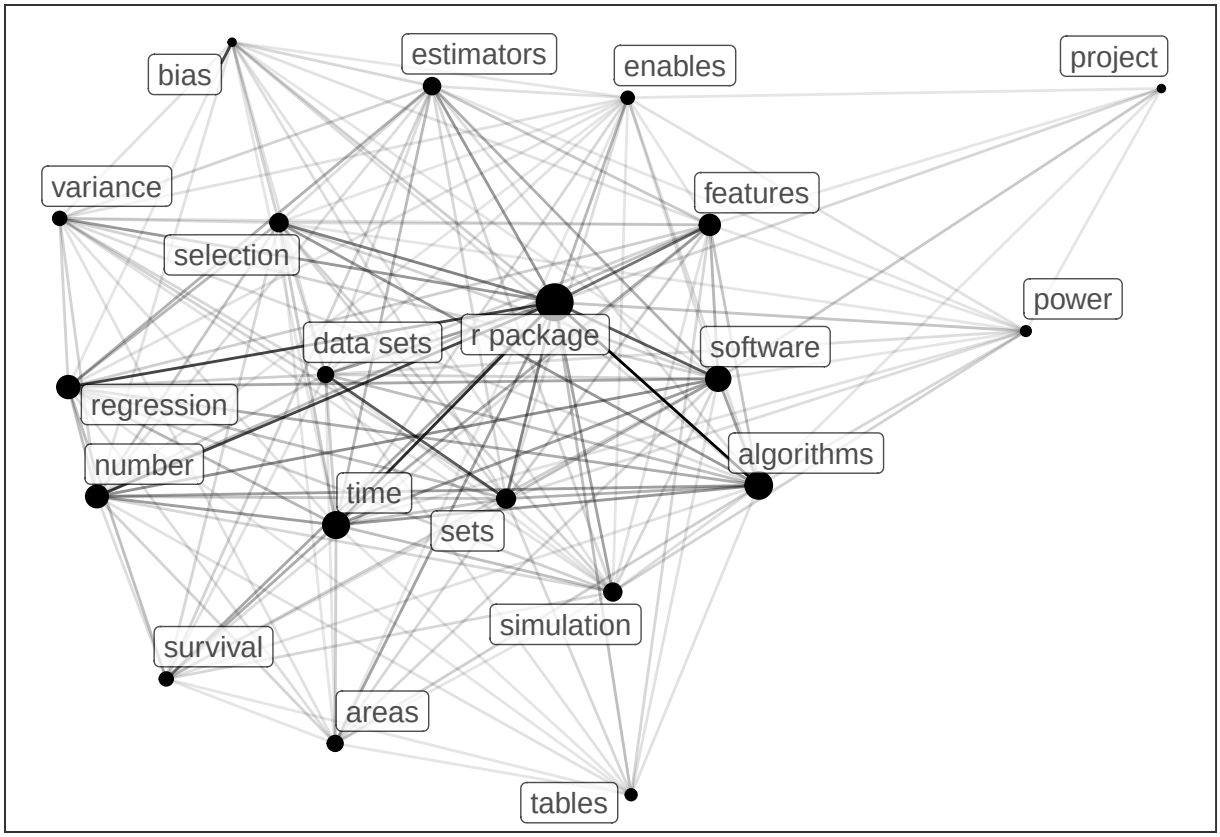

Figure 5: Focus on one cluster (Group 1) of the network. Top 20 keywords by frequency are showed.

Table 1: Informative table exported by akc.

Group	Keywords(TOP 10)
1	r package (238); algorithms (117); time (109); software (93); regression (75); number (72); features (60); sets (45); selection (41); simulation (40)
2	parameters (98); inference (65); framework (58); information (51); distributions (48); performance (47); probability (45); design (44); likelihood (41); optimization (31)
3	package (505); model (310); tools (140); tests (48); errors (46); multivariate (42); system (41); hypothesis (18); maps (16); assumptions (15)

This would return a data.frame with two columns (see Table 1), one is the group number, the other is the keywords (by default, only top 10 keywords by frequency would be displayed, and the frequency information is attached).

Word cloud visualization is also supported by **akc** via **ggwordcloud** package, this could be implemented by using `keyword_cloud` function.

Discussion

The core functionality of the akc framework is to automatically group the knowledge pieces (keywords) using modularity-based clustering. Because this process is unsupervised, it could be difficult to evaluate the outcome of classification. Nevertheless, the default setting of community detection algorithm was selected after empirical tests via benchmarking. It was found that: 1) Edge betweenness and Spinglass algorithm are most time-consuming; 2) Edge betweenness and Walktrap algorithm could potentially find more local clusters in the network; 3) Label propagation could hardly divide the keywords into groups; 4) Infomap has high standard deviation of node number across groups. In the end, Fastgreedy was chosen as the default community detection algorithm in **akc**, because its performance is relatively stable, and the number of groups increases proportionally with the network size.

Though **akc** currently focuses on automatic knowledge classification based on community detection in keyword co-occurrence network, this framework is rather general in many natural language

processing problems. One could utilize part of the framework to complete some specific tasks, such as word consolidating (using keyword merging) and n-gram tokenizing (using keyword extraction with a null dictionary), then export the tidy table and work in another environment. As long as the data follows the rule of tidy data format (Wickham et al., 2014), the **akc** framework could be easily decomposed and applied in various circumstances. For instance, by considering the nationalities of authors as keywords, **akc** framework could also investigate the international collaboration behavior in the specific domain.

In the meantime, the **akc** framework is still in active development, trying new algorithms to carry out better unsupervised knowledge classification under the R environment. The expected new directions include more community detection functions, new clustering methods, better visualization settings, etc. Note that except for the topology-based community detection approach considering graph structure of the network, there is still another topic-based approach considering the textual information of the network nodes (Ding, 2011), such as hierarchical clustering (Newman, 2003), latent semantic analysis (Landauer et al., 1998) and Latent Dirichlet Allocation (Blei et al., 2003). These methods are also accessible in R, the relevant packages could be found in the CRAN Task View: Natural Language Processing. With the tidy framework, **akc** could assimilate more nutrition from the modern R ecosystem, and move forward to create better reproducible open science schemes in the future.

Conclusion

In this paper, we have proposed a tidy framework of automatic knowledge classification supported by a collection of R packages integrated by **akc**. While focusing on data mining based on keyword co-occurrence network, the framework also supports other procedures in data science workflow, such as text cleaning, keyword extraction and consolidating synonyms. Though in the current stage it aims to support analysis in bibliometric research, the framework is quite flexible to extend to various tasks in other fields. Hopefully, this work could attract more participants from both R community and academia to get involved, so as to contribute to the flourishing open science in text mining.

Acknowledgement

This study is funded by The National Social Science Fund of China "Research on Semantic Evaluation System of Scientific Literature Driven by Big Data" (21&ZD329). This article is created using **rjtools** (O'Hara-Wild et al., 2022) in R. The source code and data for reproducing this paper can be found at: https://github.com/hope-data-science/RJ_akc.

Bibliography

S. E. Amrahov and B. Tugrul. A community detection algorithm on graph data. In *2018 International Conference on Artificial Intelligence and Data Processing (IDAP)*, pages 1–4, 2018. [p2]

H. Baziyad, S. Shirazi, S. Hosseini, and R. Norouzi. Mapping the intellectual structure of epidemiology with use of co-word analysis. *Journal of Biostatistics and Epidemiology*, 2019. [p1]

D. M. Blei, A. Y. Ng, and M. I. Jordan. Latent dirichlet allocation. *Journal of Machine Learning Research*, 3:993–1022, 2003. [p8]

V. D. Blondel, J.-L. Guillaume, R. Lambiotte, and E. Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008. [p2]

M. Callon, A. Rip, and J. Law. *Mapping the dynamics of science and technology: Sociology of science in the real world*. Springer, 1986. [p1]

A. Clauset, M. E. J. Newman, and C. Moore. Finding community structure in very large networks. *Physical review E*, 70:066111, Dec 2004. [p1]

F. B. de Sousa and L. Zhao. Evaluating and comparing the igraph community detection algorithms. In *2014 Brazilian Conference on Intelligent Systems*, pages 408–413. IEEE, 2014. [p2]

Y. Ding. Community detection: Topological vs. topical. *Journal of Informetrics*, 5(4):498–514, 2011. [p8]

S. Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010. [p1]

N. Garg and R. Rani. A comparative study of community detection algorithms using graphs and r. In *2017 International Conference on Computing, Communication and Automation (ICCCA)*, pages 273–278. IEEE, 2017. [p2]

M. Girvan and M. E. Newman. Community structure in social and biological networks. *Proceedings of the National Academy of Sciences*, 99(12):7821–7826, 2002. [p1]

Q. He. Knowledge discovery through co-word analysis. *Library Trends*, 48(1):133–159, 1999. [p1]

C.-P. Hu, J.-M. Hu, S.-L. Deng, and Y. Liu. A co-word analysis of library and information science in china. *Scientometrics*, 97(2):369–382, 2013. [p1]

J. Hu and Y. Zhang. Research patterns and trends of recommendation system in china using co-word analysis. *Information Processing & Management*, 51(4):329–339, 2015. [p1]

T.-Y. Huang and B. Zhao. Measuring popularity of ecological topics in a temporal dynamical knowledge network. *PloS One*, 14(1):e0208370, 2019. [p1]

M. A. Javed, M. S. Younis, S. Latif, J. Qadir, and A. Baig. Community detection in networks: A multidisciplinary review. *Journal of Network and Computer Applications*, 108:87–111, 2018. [p1]

A. A. Khasseh, F. Soheili, H. S. Moghaddam, and A. M. Chelak. Intellectual structure of knowledge in imetrics: A co-word analysis. *Information Processing & Management*, 53(3):705–720, 2017. [p1]

B. H. Kwasnik. The role of classification in knowledge representation and discovery. *Library Trends*, 48 (1):22–47, 1999. [p1]

T. K. Landauer, P. W. Foltz, and D. Laham. An introduction to latent semantic analysis. *Discourse Processes*, 25(2-3):259–284, 1998. [p8]

X. Y. Leung, J. Sun, and B. Bai. Bibliometrics of social media research: A co-citation and co-word analysis. *International Journal of Hospitality Management*, 66:35–45, 2017. [p1]

X. Li, E. Ma, and H. Qu. Knowledge mapping of hospitality research- a visual analysis using citespace. *International Journal of Hospitality Management*, 60:77–93, 2017. [p1]

T. Murata. *Detecting Communities in Social Networks*, pages 269–280. Springer US, Boston, MA, 2010. ISBN 978-1-4419-7142-5. [p1]

M. E. Newman. The structure and function of complex networks. *SIAM Review*, 45(2):167–256, 2003. [p8]

M. E. Newman. Fast algorithm for detecting community structure in networks. *Physical Review E*, 69 (6):066133, 2004. [p1]

M. E. Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical Review E*, 74(3):036104, 2006. [p1]

M. O'Hara-Wild, S. Kobakian, H. S. Zhang, D. Cook, and C. Dervieux. *rjtools: Tools for Preparing, Checking, and Submitting Articles to the R Journal*, 2022. URL https://github.com/rjournal/rjtools. R package version 1.0.0. [p8]

P. Pons and M. Latapy. Computing communities in large networks using random walks. In *International symposium on computer and information sciences*, pages 284–293. Springer, 2005. [p2]

U. N. Raghavan, R. Albert, and S. Kumara. Near linear time algorithm to detect community structures in large-scale networks. *Physical Review E*, 76(3):036106, 2007. [p1]

J. Reichardt and S. Bornholdt. Statistical mechanics of community detection. *Physical Review E*, 74(1): 016110, 2006. [p2]

M. Rosvall and C. T. Bergstrom. An information-theoretic framework for resolving community structure in complex networks. *Proceedings of the National Academy of Sciences*, 104(18):7327–7331, 2007. [p1]

M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, 2009. [p1]

N. J. Van Eck and L. Waltman. Software survey: Vosviewer, a computer program for bibliometric mapping. *Scientometrics*, 84(2):523–538, 2010. [p1]

M. Wang and L. Chai. Three new bibliometric indicators/approaches derived from keyword analysis. *Scientometrics*, 116(2):721–750, 2018. [p1]

H. Wickham and G. Grolemund. *R for data science: import, tidy, transform, visualize, and model data*. " O'Reilly Media, Inc.", 2016. [p2]

H. Wickham et al. Tidy data. *Journal of statistical software*, 59(10):1–23, 2014. [p2, 8]

Z. Yang, R. Algesheimer, and C. J. Tessone. A comparative analysis of community detection algorithms on artificial networks. *Scientific Reports*, 6(1):1–18, 2016. [p2]

Tian-Yuan Huang
National Science Library, Chinese Academy of Sciences
Beijing, China
ORCiD: 0000-0002-4151-3764
huangtianyuan@mail.las.ac.cn

Li Li
National Science Library, Chinese Academy of Sciences
Beijing, China
lili2020@mail.las.ac.cn

Liying Yang
National Science Library, Chinese Academy of Sciences
Beijing, China
yangly@mail.las.ac.cn