Article

Analysis of word occurrence frequency and word association in English text file: A big data analytics method

YanHong Qi[1], **GuangHua Liu**[2], **WenJun Zhang**[3]

[1]Sun Yat-sen University Libraries, Sun Yat-sen University, Guangzhou 510275, China

[2]Guangdong AIB Polytech College, Guangzhou 510507, China

[3]School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China

E-mail: qiyh@mail.sysu.edu.cn, zhwj@mail.sysu.edu.cn, ghliu@gdaib.edu.cn

Received 9 August 2017; *Accepted 15 October 2017*; *Published 1 September 2018*

Abstract

In present study, I presented an algorithm for analysis of word occurrence frequency and word association in English text file. Various delimiters were used for splitting words. In addition, common used grammatical words are ignored in word occurrence and association analysis. All different words were listed according to word occurrence frequency from the greater to the smaller. Word association was detected by using one-dimensional ordered cluster analysis. The words fallen in the same class may likely have strong association. Theoretically, various classes at distinct clustering hierarchical level may represent different hierarchical topics. Java software of the algorithm was provided.

Keywords big data analytics; word splitting; word occurrence frequency; word association; English text; algorithm; software.

Network Biology
ISSN 2220-8879
URL: http://www.iaees.org/publications/journals/nb/online-version.asp
RSS: http://www.iaees.org/publications/journals/nb/rss.xml
E-mail: networkbiology@iaees.org
Editor-in-Chief: WenJun Zhang
Publisher: International Academy of Ecology and Environmental Sciences

1 Introduction

Big data analytics is a quickly growing technology, which has been using in various areas, including network biology. About the definition and explanation of big data, there are many different opinions (Zhang et al., 2013; Zhang, 2017a-d, 2017f, 2018a-b). In the general sense, big data is a collection of data that cannot be perceived, acquired, managed, processed, and serviced with traditional IT technologies and hardware and software tools for a limited period of time. In 2010, the Apache Hadoop organization defined big data as large-scale data sets that ordinary computer software cannot capture, manage, and manage in an acceptable time frame. International Data Corporation (IDC) held that big data technology describes a new generation of technology and architecture that, through high-speed acquisition, discovery or analysis, we can extract the economic value of a large amount of data. From this point of view, the characteristics of big data can be summarized as 4Vs, namely volume (massive size), variety (numerous modals), velocity (generated quickly) and value (the value is very remarkable but the density is very low) (Zhang et al., 2013). The 4Vs definition

has been widely recognized. It stresses the significance and necessity of big data - that is, mining the huge value inside big data. Therefore, the most important topic of big data is how to extract valued information from large-scale, numerous, and rapidly growing data sets (Zhang, 2018).

In this study, I try to present an algorithm for analysis of word occurrence frequency and word association in English text file. Java is a powerful and widely used development tool (Zhang, 2011a-c; Zhang and Zhan, 2011; Zhang, 2012a; Zhang and Liu, 2012). Java software of the algorithm is provided.

2 Algorithm and Software

2.1 Algorithm

Read a text from a local English text file (*.txt), or from internet URL (http://www.../*.htm, or *.html). The following delimiters are used for splitting words:

Space, no-break space, soft hyphen, !, ", #, $, %, &, ', \r, \n, (,), * + , -, ., /, :, ;, <, =, >, ?, @, [,], ^, _, {, |, }, ~, ¯, °, ±, ·, ×, ÷ , <, >, ∧, ∨, ×, \

As the default setting, the following common used grammatical words are ignored in word occurrence analysis:

the,and,of,is,are,has,have,was,were,had,etc,to,be,for,with,or,a,let,an,all,over,must,as,any,each,can,by,become,what,there,when,which,where,that,if,then,thus,than,we,you,our,your,yours,us,it,its,they,will,would,shall,should,may,from,their,while,at,in,on,how,once,again,ago,before,after,between,both,either,just,each,often,usually,without,use,also,high,low,large,small,great,only,most,go,going,went,but,some,such,as,who,whose,under,these,those,this

Users can enter their own new words to be ignored in word occurrence analysis.

All different words are listed according to word occurrence frequency from the greater to the smaller.

Closely associated words tend to occur simultaneously and are expected to have the similar occurrence frequencies. Therefore, by using one-dimensional ordered cluster analysis (Zhang and Fang, 1982; Zhang, 2017; Qi, 2005), the words with similar occurrence frequencies may be clustered into the same class. By doing so, we can try to find all closely associated words in the text file.

Suppose there are totally n different words, arranged with occurrence frequency rank from the greater to the smaller, x_1, x_2, …, x_n. The distance between adjacent two words is the absolute of frequency difference between them

$$r_{i\,i+1} = |x_i - x_{i+1}| \quad i=1,2,\ldots,n-1$$

Find the shortest distance $r_{i\,i+1}$, i=1,2,…,n-1, and combine the two words i and i+1 into the same cluster. Similarly, take the minimum between-word distance as the between-cluster distance. Find the two clusters with minimum between-cluster distance and combine into a single cluster. Finally all words are combined into a single cluster. The minimum between-cluster distance at a clustering hierarchical level is defined as the between-cluster distance of this hierarchical level. Theoretically, various classes at distinct clustering hierarchical level may represent different hierarchical topics.

Users can enter the required minimum difference of word occurrence frequency for cluster analysis.

2.2 Software

The software of the algorithm above, parseEnglish, is a Java applet (Fig. 1), developed using JDK (Java Development Kit). It includes the following major methods:

urlConnection(): Connect specified URL and obtain online HTML text.
parseArray(): Split words and store them in a vector. Delimiters above are used to separate words. Grammatical words above are ignored. The new words entered by the user are ignored also.
ranking(): Ranking words according to their occurrence frequency, from the greater to the smaller.
stat(): Calculate occurrence frequency of distinct words.
oneDimCluster(): Make one-dimensional ordered clustering for word series. Only the words with the occurrence frequency not less than specified minimum threshold frequency are clustered.

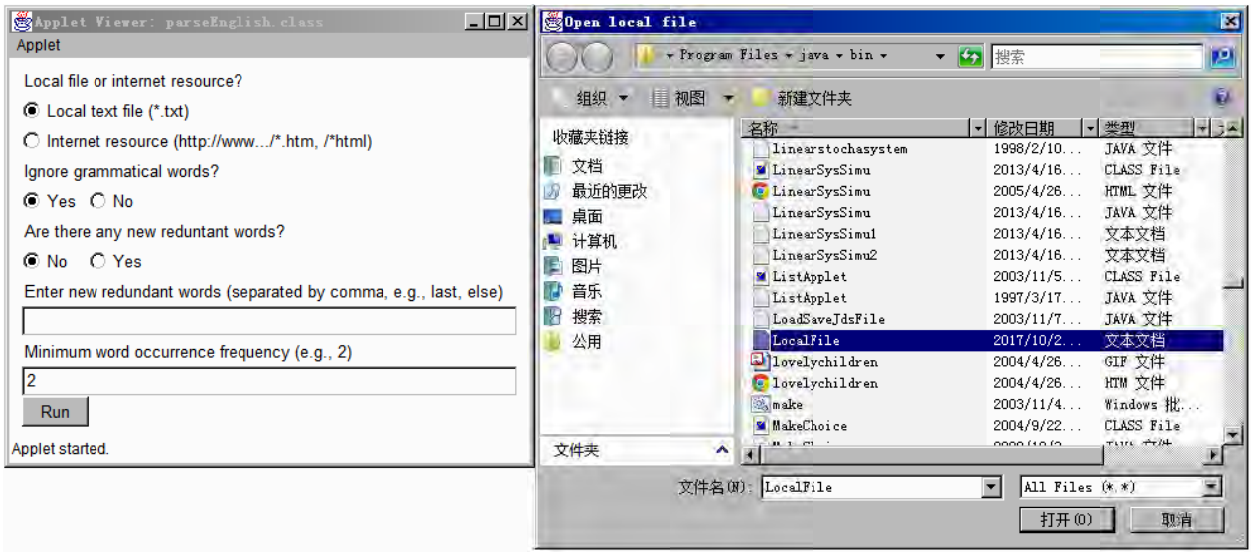

Fig. 1 Java applet of the algorithm.

The following are Java codes of three methods, parseArray(), ranking(), stat(), and oneDimCluster():

```
public String[] parseArray(String s)
{
    Vector vector = new Vector();
    String s1 = s;
    int i = 0x186a0;
    String as1[] = {
        "the", "and", "of", "is", "are", "has", "have", "was", "were", "had",
        "etc", "to", "be", "for", "with", "or", "a", "let", "an", "all",
        "over", "must", "as", "any", "each", "can", "by", "become", "what", "there",
        "when", "which", "where", "that", "if", "then", "thus", "than", "we", "you",
        "our", "your", "yours", "us", "it", "its", "they", "will", "would", "shall",
        "should", "may", "from", "their", "while", "at", "in", "on", "how", "once",
        "again", "ago", "before", "after", "between", "both", "either", "just", "each", "often",
        "usually", "without", "use", "also", "high", "low", "large", "small", "great", "only",
        "most", "go", "going", "went", "but", "some", "such", "as", "who", "whose",
        "under", "these", "those", "this"
    };
    char ac[] = {
        '\u0020','\u00A0','\u00AD','\u0021','\u0022','\u0023','\u0024','\u0025','\u0026','\u0027','\r','\n','\t','\u0028','\u0029'
```

```
        ,'\u002A','\u002B','\u002C','\u002D','\u002E','\u002F','\u003A','\u003B','\u003C','\u003D','\u003E','\u003F','\u004
        0','\u005B','\u005D','\u005E','\u005F','\u007B','\u007C','\u007D','\u007E','\u00AF','\u00B0','\u00B1','\u00B7','\u0
        0D7','\u00F7','\u02C2','\u02C3','\u02C4','\u02C5','\u02DF','\\'
    };
    String as[];
    if(choword == 1)
    {
        int j2 = stword.length();
        int k2 = as1.length;
        String as2[] = new String[j2];
        String s3 = stword;
        int j = 0;
        int k1;
        do
        {
            k1 = s3.indexOf(',');
            if(k1 >= 0)
            {
                as2[j] = s3.substring(0, k1).trim();
                s3 = s3.substring(k1 + 1).trim();
                j++;
            } else
            {
                as2[j] = s3;
            }
        } while(k1 >= 0);
        as = new String[k2 + j + 1];
        for(int l = 0; l < j + k2 + 1; l++)
            if(l < k2)
                as[l] = as1[l];
            else
                as[l] = as2[l - k2];
    } else
    {
        as = new String[as1.length];
        as = as1;
    }
    int i2;
label0:
    do
    {
        i2 = i;
        for(int i1 = 0; i1 < ac.length; i1++)
            if((s.indexOf(ac[i1]) < i2) & (s.indexOf(ac[i1]) >= 0))
                i2 = s.indexOf(ac[i1]);
        if(i2 == i)
            break;
        if(i2 >= 0)
        {
            String s2 = s.substring(0, i2).trim();
            s = s.substring(i2 + 1).trim();
            if(cho == 1)
            {
                for(int j1 = 0; j1 < as.length; j1++)
                    if(s2.equalsIgnoreCase(as[j1]))
                        continue label0;
            }
            vector.addElement(s2);
```

```
            } else
            {
                vector.addElement(s);
            }
        } while(i2 >= 0);
        int l2 = vector.size();
        args1 = new String[l2];
        String as3[] = new String[l2];
        for(int k = 0; k < l2; k++)
            as3[k] = vector.elementAt(k).toString();
        return as3;
    }

    public String[][] ranking(String as[][])
    {
        int i = as.length;
        int ai[] = new int[i];
        int ai1[] = new int[i];
        String as1[] = new String[i];
        String as2[][] = new String[i][2];
        for(int j = 0; j <= i - 1; j++)
        {
            as1[j] = "";
            for(int k = 0; k <= 1; k++)
                as2[j][k] = "";
        }
        for(int l = 0; l <= i - 1; l++)
            ai1[l] = Integer.parseInt(as[l][1]);
        for(int i1 = 0; i1 <= i - 1; i1++)
        {
            ai[i1] = ai1[i1];
            as1[i1] = as[i1][0];
        }
        for(int j1 = 0; j1 <= i - 2; j1++)
        {
            int k1 = j1;
            for(int j2 = j1; j2 <= i - 2; j2++)
                if(ai[j2 + 1] >= ai[k1])
                    k1 = j2 + 1;
            int k2 = ai[j1];
            String s = as1[j1];
            ai[j1] = ai[k1];
            as1[j1] = as1[k1];
            ai[k1] = k2;
            as1[k1] = s;
        }
        for(int i2 = 0; i2 <= i - 1; i2++)
        {
            as2[i2][0] = as1[i2];
            as2[i2][1] = Integer.toString(ai[i2]);
        }
        return as2;
    }

    public String[][] stat(String as[])
    {
        int i = as.length;
        String as1[][] = new String[i][2];
```

```
            for(int j = 0; j <= i - 1; j++)
            {
                for(int k = 0; k <= 1; k++)
                    as1[j][k] = "";
            }
            l1 = 0;
label0:
            for(int l = 0; l <= i - 1; l++)
            {
                for(int i1 = 0; i1 <= l1; i1++)
                    if(as1[i1][0].equalsIgnoreCase(as[l]))
                        continue label0;
                as1[l1][0] = as[l];
                int j1 = 0;
                for(int i2 = l; i2 <= i - 1; i2++)
                    if(as[l].equalsIgnoreCase(as[i2]))
                        j1++;
                as1[l1][1] = Integer.toString(j1);
                l1++;
            }
            String as2[][] = new String[l1][2];
            for(int k1 = 0; k1 <= l1 - 1; k1++)
            {
                for(int j2 = 0; j2 <= 1; j2++)
                    as2[k1][j2] = as1[k1][j2];
            }
            args2 = new String[l1][2];
            return as2;
        }

        public void oneDimCluster(int ai[], String as[])
        {
            int i = ai.length;
            String as1[] = new String[i + 2];
            int ai1[][] = new int[i + 2][i + 2];
            int ai4[] = new int[i + 2];
            int ai2[] = new int[i + 2];
            double ad[] = new double[i + 2];
            int ai3[] = new int[i + 2];
            double ad1[] = new double[i + 2];
            double ad2[] = new double[i + 2];
            for(int k = 1; k <= i; k++)
                ai4[k] = ai[k - 1];
            String s = "";
            for(int l = 1; l <= i - 1; l++)
                ad[l] = Math.abs(ai4[l] - ai4[l + 1]);
            for(int i1 = 1; i1 <= i - 1; i1++)
                ad1[i1] = ad[i1];
            for(int j1 = 1; j1 <= i - 2; j1++)
            {
                int j4 = j1;
                for(int j3 = j1; j3 <= i - 2; j3++)
                    if(ad1[j3 + 1] <= ad1[j4])
                        j4 = j3 + 1;
                double d = ad1[j1];
                ad1[j1] = ad1[j4];
                ad1[j4] = d;
            }
```

```
                ad1[0] = 0.0D;
                int j = 1;
        label0:
                for(int k4 = 0; k4 <= i - 1; k4++)
                {
                    for(int k1 = 0; k1 <= k4 - 1; k1++)
                        if(Math.abs(ad1[k1] - ad1[k4]) <= 9.9999999999999995E-007D)
                            continue label0;
                    for(int i2 = 1; i2 <= i - 1; i2++)
                        if(ad[i2] - ad1[k4] <= 9.9999999999999995E-007D)
                            ai3[i2] = 1;
                        else
                            ai3[i2] = 0;
                    for(int j2 = 1; j2 <= i; j2++)
                        ai1[j][j2] = 0;
                    int k2 = 1;
                    ai2[j] = 1;
        label1:
                    while(k2 <= i - 1)
                    {
                        if(ai3[k2] == 0)
                        {
                            ai1[j][k2] = ai2[j];
                            ai1[j][k2 + 1] = ai2[j] + 1;
                        } else
                        if(ai3[k2] == 1)
                        {
                            for(int k3 = k2; k3 <= i - 1; k3++)
                            {
                                if(ai3[k3] == 0)
                                {
                                    k2 = k3;
                                    continue label1;
                                }
                                ai1[j][k3] = ai2[j];
                                ai1[j][k3 + 1] = ai2[j];
                                if(k3 == i - 1)
                                    break label1;
                            }
                        }
                        k2++;
                        ai2[j]++;
                    }
                    s = s + "\n";
                    s = s + "Cluster fineness (between-word frequency difference)=" + String.valueOf((double)(int)(ad1[k4] *
10000D) / 10000D) + "\n";
                    ad2[j] = ad1[k4];
                    String s1 = "";
                    for(int l2 = 1; l2 <= ai2[j]; l2++)
                    {
                        s1 = s1 + "(";
                        for(int l3 = 1; l3 <= i; l3++)
                            if(ai1[j][l3] == l2)
                                s1 = s1 + as[l3 - 1] + " ";
                        s1 = s1 + ")    ";
                    }
                    s = s + s1;
                    j++;
```

```
        }
        j--;
        output.editt1.appendText(s + "\n");
        String s2 = "One-dimensional Ordered Cluster";
        int l4 = 1;
        for(int i3 = 1; i3 <= ai2[1]; i3++)
        {
            for(int i4 = 1; i4 <= i; i4++)
                if(ai1[1][i4] == i3)
                {
                    as1[l4] = as[i4 - 1];
                    l4++;
                }
        }
        (new GraphicsFrame(new ClusterGraphics(as1, j, i, i, ad2[j], ad2, ai2, ai1), s2)).resize(710, 560);
    }
```

3 Application Example

Set the required minimum difference of word occurrence frequency for cluster analysis as 2. Table 1 and Fig. 2 show partial results of word occurrence frequencies of the mixed abstract text of Zhang (2012, 2015, 2016).

Table 1 Partial results of word occurrence frequencies of the mixed abstract text of Zhang (2012b, 2015, 2016)

Word	Frequency	%	Word	Frequency	%
network	16	4.507	degree	3	0.845
node	13	3.661	Various	3	0.845
Community	11	3.098	initial	3	0.845
species	9	2.535	Lamda	3	0.845
I	8	2.253	factor	3	0.845
evolution	8	2.253	dynamics	3	0.845
assembly	7	1.971	y	2	0.563
rule	7	1.971	x	2	0.563
present	6	1.69	emergency	2	0.563
connection	6	1.69	nodes	2	0.563
proposed	5	1.408	phenomena	2	0.563
model	5	1.408	disconnection	2	0.563
probability	5	1.408	composition	2	0.563
connections	5	1.408	think	2	0.563
organization	4	1.126	theory	2	0.563
self	4	1.126	rules	2	0.563
attraction	4	1.126	e	2	0.563
method	4	1.126	extinction	2	0.563
study	4	1.126	general	2	0.563
taxon	3	0.845	parameters	2	0.563
property	3	0.845	Effects	2	0.563
natural	3	0.845	simplified	2	0.563
growth	3	0.845	invasion	2	0.563
process	3	0.845	changes	2	0.563
random	3	0.845	number	2	0.563
Barabasi	3	0.845	different	2	0.563
Albert	3	0.845	1999	2	0.563
based	3	0.845	dependent	2	0.563
power	3	0.845	generate	2	0.563
addition	3	0.845	type	2	0.563
mechanism	3	0.845	Modeling	2	0.563

All words with frequency of 1 are omitted.

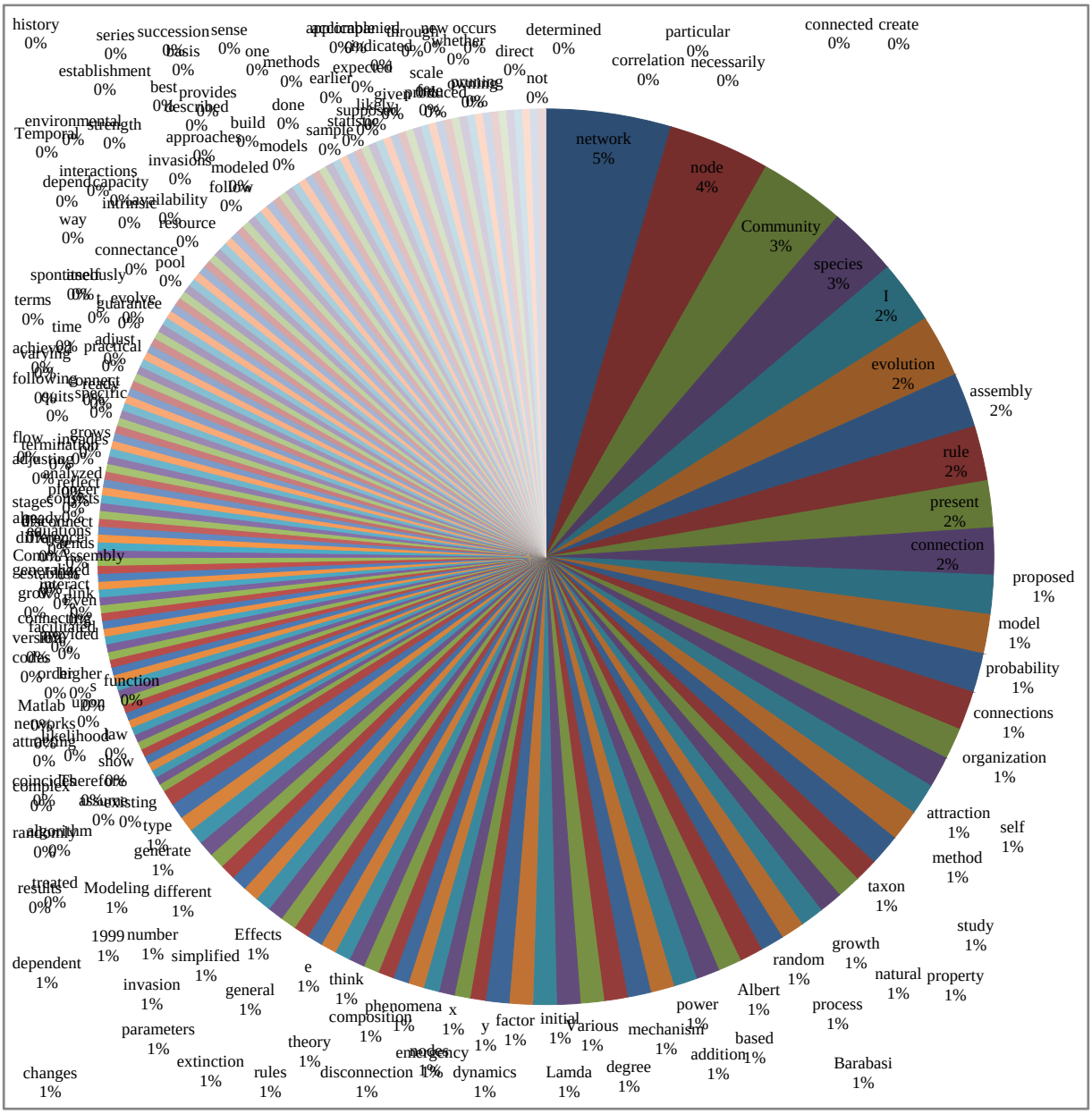

Fig. 2 Pie illustration of percentage of word occurrence.

The following and Fig. 3 show the results of one-dimensional ordered cluster analysis:

Cluster fineness (between-word frequency difference)=0.0

(network) (node) (Community) (species) (I evolution) (assembly rule) (present connection) (proposed model probability connections) (organization self attraction method study) (taxon property natural growth process random Barabasi Albert based power addition mechanism degree Various initial Lamda factor dynamics) (y x emergency nodes phenomena disconnection composition think theory rules e extinction general parameters Effects simplified invasion changes number different 1999 dependent generate type Modeling)

Cluster fineness (between-word frequency difference)=1.0

(network) (node) (Community) (species I evolution assembly rule present connection proposed model probability connections organization self attraction method study taxon property natural growth process random Barabasi Albert based power addition mechanism degree Various initial Lamda factor dynamics y x emergency nodes phenomena disconnection composition think theory rules e extinction general parameters Effects simplified invasion changes number different 1999 dependent generate type Modeling)

Cluster fineness (between-word frequency difference)=2.0

(network) (node Community species I evolution assembly rule present connection proposed model probability connections organization self attraction method study taxon property natural growth process random Barabasi Albert based power addition mechanism degree Various initial Lamda factor dynamics y x emergency nodes phenomena disconnection composition think theory rules e extinction general parameters Effects simplified invasion changes number different 1999 dependent generate type Modeling)

Cluster fineness (between-word frequency difference)=3.0

(network node Community species I evolution assembly rule present connection proposed model probability connections organization self attraction method study taxon property natural growth process random Barabasi Albert based power addition mechanism degree Various initial Lamda factor dynamics y x emergency nodes phenomena disconnection composition think theory rules e extinction general parameters Effects simplified invasion changes number different 1999 dependent generate type Modeling)

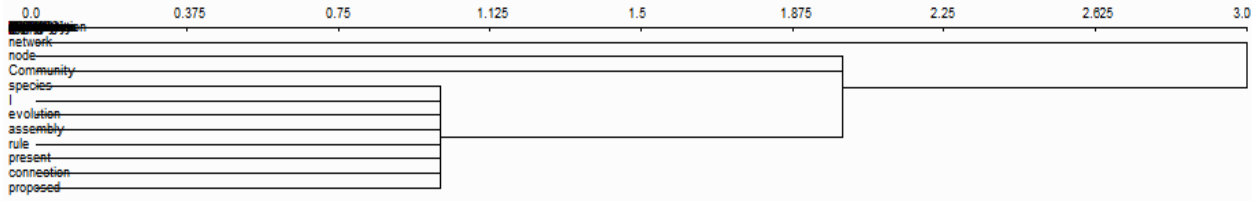

Fig. 3 One-dimensional ordered cluster.

It can be found that the words, network, node, community and species are the most frequent occurred words in the three abstracts.

Acknowledgment

We are thankful to the support of Discovery and Crucial Node Analysis of Important Biological and Social Networks (2015.6-2020.6), from Yangling Institute of Modern Agricultural Standardization, and High-Quality Textbook *Network Biology* Project for Engineering of Teaching Quality and Teaching Reform of Undergraduate Universities of Guangdong Province (2015.6-2018.6), from Department of Education of Guangdong Province, China.

References

Qi YH. 2005. The web software for ordered cluster analysis of sequential information. Information Science, 23(Suppl.): 99-101

Zhang WJ. 2011a. A Java algorithm for non-parametric statistic comparison of network structure. Network Biology, 1(2): 130-133

Zhang WJ. 2011b. A Java program for non-parametric statistic comparison of community structure. Computational Ecology and Software, 1(3): 183-185

Zhang WJ. 2011c. A Java program to test homogeneity of samples and examine sampling completeness. Network Biology, 1(2): 127-129

Zhang WJ. 2012a. A Java software for drawing graphs. Network Biology, 2(1): 38-44

Zhang WJ. 2012b. Modeling community succession and assembly: A novel method for network evolution. Network Biology, 2(2): 69-78

Zhang WJ. 2015. A generalized network evolution model and self-organization theory on community assembly. Selforganizology, 2(3): 55-64

Zhang WJ. 2016. A random network based, node attraction facilitated network evolution method. Selforganizology, 3(1): 1-9

Zhang WJ. 2017a. Network pharmacology of medicinal attributes and functions of Chinese herbal medicines: (I) Basic statistics of medicinal attributes and functions for more than 1100 Chinese herbal medicines. Network Pharmacology, 2(2): 17-37

Zhang WJ. 2017b. Network pharmacology of medicinal attributes and functions of Chinese herbal medicines: (II) Relational networks and pharmacological mechanisms of medicinal attributes and functions of Chinese herbal medicines. Network Pharmacology, 2(2): 38-66

Zhang WJ. 2017c. Network pharmacology of medicinal attributes and functions of Chinese herbal medicines: (III) Canonical correlation functions between attribute classes and linear eignmodels of Chinese herbal medicines. Network Pharmacology, 2(3): 67-81

Zhang WJ. 2017d. Network pharmacology of medicinal attributes and functions of Chinese herbal medicines: (IV) Classification and network analysis of medicinal functions of Chinese herbal medicines. Network Pharmacology, 2(3): 82-104

Zhang WJ. 2017e. Phase recognition in network evolution. Selforganizology, 4(3): 35-40

Zhang WJ. 2017f. Some correlations between eight types of malignant neoplasms: A hint from cancer dynamics of 31 European countries in 20 years. Network Biology, 7(3): 76-79

Zhang WJ. 2018a. Fundamentals of Network Biology. World Scientific, London, Singapore

Zhang WJ. 2018b. Global pesticide use: Profile, trend and cost / benefit analysis. Proceedings of the International Academy of Ecology and Environmental Sciences, 8(1): 1-26

Zhang WJ, Liu GH. 2012. Creating real network with expected degree distribution: A statistical simulation. Network Biology, 2(3): 110-117

Zhang WJ, Zhan CY. 2011. An algorithm for calculation of degree distribution and detection of network type: with application in food webs. Network Biology, 1(3-4): 159-170

Zhang Y, Chen M, Liao XF. 2013. Big Data Applications: A Survey. Journal of Computer Research and Development, 50(Suppl.): 216-233

Zhang YT, Fang KT. 1982. Introduction to Multivariate Statistics. Science Press, Beijing, China