Article

Construction and analysis of the word network based on the Random Reading Frame (RRF) method

WenJun Zhang
School of Life Sciences, Sun Yat-sen University, Guangzhou 510275, China
E-mail: zhwj@mail.sysu.edu.cn, wjzhang@iaees.org

Received 9 August 2018; *Accepted 15 January 2021*; *Published 1 September 2021*
(cc) BY

Abstract
In present study, a method was developed to construct and analyze the word network. The core of the method is Random Reading Frame (RRF) method. First, download or collect word files (in various formats, e.g., pdf, txt, doc, docx, rtf, html, etc.) from internet or local machine in terms of the concerned topics. All files were then combined in a final text file. Excepting for splitting words and stop words, all words were arranged in a word vector following their orders in the combined text file. In the RRF method, for a given pair of unique words (x, y), x, $y \in \{u_1,u_2,\ldots,u_m\}$, a reading frame with randomly changeable width is randomly placed on the vector to count the respective number of the two words in the frame. Randomly repeating the procedure p times, the paired numbers are thus achieved: (x_1, y_1), (x_2, y_2), …, (x_p, y_p). In such a way, the paired numbers for all pairs of unique words are achieved. Thereafter, for a given pair of unique words (x, y), Pearson correlation and Pearson partial correlation, Spearman rank correlation, or point correlation is used to calculate their correlation value according to their paired numbers (x_1, y_1), (x_2, y_2), …, (x_p, y_p), and the statistically significance can be determined by t-test (Pearson correlation, Pearson partial correlation, Spearman rank correlation) or χ^2-test (point correlation). In such a way, all statistically significant word pairs are achieved in terms of the correlation measure chosen by user. Finally, the word network, in terms of the correlation measure chosen, can be constructed based on these word pairs, and no links between statistically insignificant word pairs. Network analysis is conducted for the word network constructed from significant between-word positive correlations among all unique words. Word centrality measures, word tree, word chains, word modules, etc., can be calculated in the method. The Matlab software, wordNetwork for the method was given also.

Keywords word association; association rules; correlation measures; Random Reading Frame; network construction; network analysis; algorithm; text mining.

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

Word statistics in terms of internet resources and multiple files is of significant in analyzing word association, word clustering (Qi et al., 2018), etc. Furthermore, the construction of a word network based on internet

resources and multiple files helps to find new associations, new interactions, new paths among various things and events. In a sense, the word analysis based on internet resources and multiple files belongs to big data analytics. It stresses the significance and necessity of big data, that is, mining the huge value inside big data. 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 their earlier study, Qi et al. (2018) have developed a Java method to cluster words. So far, most of the methods on word analysis have focused on the word frequency, etc. In this study, I present a method for constructing and analyzing the word network. The core of the method is Random Reading Frame (RRF) method. The Matlab software for the method was given also.

2 Algorithm and Software

2.1 Algorithm

The following are the procedures for the algorithm, wordNetwork:

2.1.1 Collection of words

(1) Specifying the language (English, French, German, Spanish, Portuguese, Italian, Dutch, Russian, Latin, Indonesian, Chinese).

(2) Read files (*.txt, *.doc, *.docx, *.wps, *.pdf, *.csv, *.htm, *.html, *.shtm, *.shtml, *.xhtm, *.xhtml, *.asp, *.xml, etc.) from internet resources or local machine, or read a local file. If multiple files are used, they are combined in a text file.

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

Corresponding to the language used, default stop words are used. For instance, in English text, the words in the set {more, already, the, and, of, is, are, has, have, was, were, had, etc, to, too, be, for, with, or, a, let, an, all, over, across, must, as, any, each, can, could, by, become, what, there, when, which, where, that, if, then, thus, than, we, you, yours, us, it, its, they, will, would, shall, should, may, might, from, 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} are default words.

In addition, users can enter their own new stop and/or splitting words.

(4) Find all words in the text file, excepting for splitting and stop words. All words are naturally ordered according to their orders in the original text file, and they consititute a word vector, $w=(w_1,w_2,\ldots,w_n)$.

In addition, all unique words are arranged with their respective IDs (Zhang and Qi, 2020).

2.1.2 Word statistcs

In the word vector, all unique words are calculated for word occurrence frequency, and ranked from the greater to the smaller ones, $u=(u_1,u_2,\ldots,u_m)$, $u_i\in\{w_1,w_2,\ldots,w_n\}$, $i=1,2,\ldots,m$, $m\le n$. In detail, unique words, word occurrence frequencies and word proportions against total unique words, with occurrence frequency greater than 90%, are listed.

2.1.3 Random Reading Frame (RRF) method

In the word vector, closely associated words tend to occur simultaneously (Qi et al., 2018). Therefore, for a given pair of unique words (x, y), x, $y\in\{u_1,u_2,\ldots,u_m\}$, a reading frame with randomly changeable width (the maximum permissible width and minimum permissible width are the certain proportions of size n of the word vector w, which are given by users) is randomly placed on the vector to count the respective number of the two words in the frame. The random reading frame is a random fragment of the word vector $w=(w_1,w_2,\ldots,w_n)$: $(w_i,w_{i+1},\ldots,w_{i+k})$, where i denotes the location of the random reading frame and k represents the width of the

random reading frame, both i and k are random numbers, $1 \le i \le n-k$. Randomly repeating the procedure p times (i.e., p randomizations) (Zhang, 2011a, 2011b), the paired numbers are thus achieved: (x_1, y_1), (x_2, y_2), …, (x_p, y_p). In such a way, the paired numbers for all pairs of unique words are achieved.

2.1.4 Construction of word network: association rules

For a given pair of unique words (x, y), Pearson correlation and Pearson partial correlation (Zhang, 2012a, 2012b; Zhang et al., 2014; Zhang and Li, 2015a, 2015b), Spearman rank correlation (Zhang, 2014, 2015; Zhang and Li, 2015a), or point correlation (Zhang, 2016d, 2017b, 2018) is used to calculate their correlation value according to their paired numbers (x_1, y_1), (x_2, y_2), …, (x_p, y_p), and the statistically significance can be determined by t-test (Pearson correlation, Pearson partial correlation, Spearman rank correlation) or χ^2-test (point correlation). In such a way, all statistically significant word pairs are achieved in terms of the correlation measure chosen by user. Finally, the word network, in terms of the correlation measure chosen, can be constructed based on these word pairs, and no links between statistically insignificant word pairs.

2.1.5 Network analysis

Network analysis is conducted for the word network constructed from significant between-word positive correlations among all unique words.

Word importance: it represented by various centrality measures (Xing and Zhang, 2020; Zhang and Zhan, 2011; Zhang, 2012a, 2012c, 2016d, 2018; Zhang and Zhang, 2019), including degree centrality (the degree of a word, i.e., the number of links owned by a word, which represents the word importance), closeness centrality (a global centrality representing the independence of a word in the network. Closeness centrality is defined as reciprocal of the sum of the word's geodesic distances to all other words (the distances of the shortest paths) in the network), and betweenness centrality (it represents the word's ability to control the data flow in the network. This measure is the proportion of number of geodesic paths that pass through the given word to total number of geodesic paths between any pair of words in the network) (Li and Zhang, 2012; Shams and Khansari, 2014; Jiang and Zhang, 2015a, 2015b; Jiang et al., 2015; Nuwagaba and Hui, 2015).

Word tree: the tree with minimum total links. It represents the simplest topological structure that connects all words (Chan et al., 1982; Minty, 1965; Zhang, 2016c, 2017a).

Word modules: the words in the same module are more closely connected. Closerly connected link has greater weight (Zhang, 2016b, 2017c).

Word chains: a chain denotes a link series between two words (here it means the chain with minimum total links) (Floyd, 1962; Zhang, 2016a, 2018). Word chains between two words (a chain with minimum total links) for the first x% words in all unique words are listed.

2.2 Software

The following are the Matlab functions of the software, wordNetwork (Version 1.0; http://www.iaees.org/publications/journals/nb/articles/2021-11(3)/e-suppl/Zhang-Supplementary-Material.rar) (Fig. 1):

```
function wordNetwork
%Copyright Zhang WJ, 2018
clear
[lang,OK]=listdlg('liststring',{'English','French','German','Spanish','Portuguese','Italian','Dutch','Russian','Latin','Indonesian','Chin
ese'},'listsize',[230 200],'OkString','OK','CancelString','Cancel','promptstring','Language','selectionmode','single');
if (lang==1) delworda={'more' 'already' 'the' 'and' 'of' 'is' 'are' 'has' 'have' 'was' 'were' 'had' 'etc' 'to' 'too' 'be' 'for' 'with' 'or' 'a' 'let' 'an'
'all' 'over' 'across' 'must' 'as' 'any' 'each' 'can' 'could' 'by' 'become' 'what' 'there' 'when' 'which' 'where' 'that' 'if' 'then' 'thus' 'than' 'we'
'you' 'yours' 'us' 'it' 'its' 'they' 'will' 'would' 'shall' 'should' 'may' 'might' 'from' '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'}; printStr=['English',char(13)]; %English
elseif (lang==2) delworda={'more' 'already' 'le' 'et' 'de' 'are' 'have' 'a¨|t¨|' 'pour' 'avec' 'a' 'let' 'un' 'tout' 'devenir' 'quoi' 'l¨¤' 'quand' 'qui' 'o¨´' 'si' 'alors' 'donc' 'que' 'nous' 'vous' 'les' 'nous' 'they' 'will' 'would' 'should' 'devrait' 'may' 'pourrait' 'while' 'comment' 'une fois' 'encore' 'avant' 'apr¨¨s' 'entre' 'les deux' 'l"un' 'ou' 'l"autre' 'souvent' 'sans' 'seulement' 'la plupart' 'vont' 'vont' 'mais' 'certains' 'tels' 'ceux' 'ceux' 'this'}; printStr=['French',char(13)]; %French
elseif (lang==3) delworda={'mehr' 'schon' 'und' 'ist' 'ist' 'war' 'war' 'wurden' 'f¨¹r' 'mit' 'oder' 'ein' '¨¹ber' 'muss' 'als' 'jeder' 'kann' 'durch' 'werden' 'was' 'dort' 'wann' 'wo' 'dann' 'also' 'als' 'wir' 'du' 'dein' 'uns' 'es' 'Sie' 'k?nnen' 'von' 'w?hrend' 'nach' 'zwischen' 'beiden' 'entweder' 'nur' 'oft' 'gew?hnlich' 'ohne' 'verwenden' 'auch' 'hoch' 'niedrig' 'gro?' 'klein' 'gro?' 'nur' 'am meisten' 'ging' 'ging' 'aber' 'einige' 'wie' 'wer' 'wessen' 'unter' 'diese' 'das'}; printStr=['German',char(13)]; %German
elseif (lang==4) delworda={'m¨¢s' 'ya' 'el' 'y' 'de' 'es' 'son' 'tiene' 'tienen' 'se' 'fueron' 'se' 'tuvieron' 'etc' 'a' 'tambi¨¦n' 'se' 'para' 'con' 'o' 'a' 'let' 'an' 'all' 'over' 'across' 'must' 'como' 'any' 'cada' 'can' 'could' 'de' 'by' 'convertirse' 'qu¨¦' 'all¨¢' 'cuando' 'd¨®nde' 'que' 'si' 'luego' 'as¨¢' 'que' 'nosotros' 'tu' 'nosotros' 'es' 'su' 'they' 'will' 'would' 'shall' 'should' 'may' 'might' 'from' 'while' 'at' 'in' 'on' 'c¨®mo' 'once' 'again' 'ago' 'before' 'despu¨¦s' 'entre' 'ambos' 'o' 'simplemente' 'cada' 'a menudo' 'generalmente' 'sin' 'usar' 'tambi¨¦n' 'alto' 'bajo' 'grande' 'peque?o' 'grande' 'solo' 'm¨¢s' 'ir' 'yendo' 'se fue' 'pero' 'algo' 'tal' 'como' 'qui¨¦n' 'cuyo' 'debajo' 'estos' 'esos' 'esto'};
printStr=['Spanish',char(13)]; %Spanish
elseif (lang==5) delworda={'more' 'already' 'the' 'e' 'of' 'is' 'has' 'have' 'foram' 'foram' 'tinham' 'etc' 'to' 'too' 'be' 'para' 'com' 'ou' 'a' 'let' 'an' 'all' 'over' 'across' 'deve' 'como' 'any' 'each' 'can' 'could' 'by' ' tornar-se' 'o que' 'l¨¢' 'quando' 'que' 'onde' 'que' 'se' 'assim' 'do que' 'n¨®s' 'voc¨ª' 'seu' 'n¨®s' 'it' 'its' 'they' 'will' 'would' 'shall' 'should' 'pode' 'from' 'while' 'at' 'in' 'on' 'como' 'uma vez' 'novamente' 'ago' 'antes' 'after' 'entre' 'ambos' 'ou' 'apenas' 'cada' 'frequentemente' 'geralmente' 'sem' 'usar' 'tambem' 'alta' 'baixa' 'grande' 'pequena' 'grande' 'apenas' 'a maioria' 'vai' 'mas' 'alguns' 'tais' 'como' 'quem' 'de quem' 'sob' 'esses' 'aqueles' 'isso'};
printStr=['Portuguese',char(13)]; %Portuguese
elseif (lang==6) delworda={'pi¨´' 'gi¨¤' 'l"e' 'di' '¨¨' 'si' 'ha' 'avuto' 'si' 'erano' 'avuto' 'ecc' 'per' 'essere' 'con' 'o' 'lasciare' 'un' 'dover' 'dappertutto' 'come' 'ognuno' 'poteva' 'diventare' 'ci¨°' 'che' 'l¨¤' 'quando' 'quale' 'dove' 'se' 'poi' 'cos¨¬' 'come' 'noi' 'volont¨¤' 'dovrebbe' 'dovrebbe' 'potrebbe' 'da' 'mentre' '¨¨' 'dentro' 'su' 'come' 'ancora' 'una' 'volta' 'fa' 'prima' 'dopo' 'tra' 'entrambi' 'o' 'solo' 'spesso' 'spesso' 'senza' 'uso' 'anche' 'alto' 'basso' 'grande' 'grande' 'grande' 'solo' 'la' 'maggior' 'parte' 'andava,' 'ma' 'alcuni' 'come' 'chi' 'di' 'chi' 'sotto' 'questi'};
printStr=['Italian',char(13)]; %Italian
elseif (lang==7) delworda={'meer' 'al' 'de' 'en' 'van' 'is' 'zijn' 'heeft' 'was' 'etc' 'had' 'ook' 'om' 'mee' 'te' 'zijn' 'of' 'een' 'alles' 'over' 'te' 'laten' 'moet' 'zoals' 'elk' 'kan' 'dat' 'door' 'wat' 'te' 'worden' 'waar' 'wanneer' 'dat' 'dan' 'dan' 'hebben' 'wij' 'jou' 'de' 'jouwe' 'het' 'zijn' 'zij' 'zal' 'zou' 'moeten' 'zou' 'kunnen' 'moge' 'vanaf' 'hoe' 'lang' 'geleden' 'meestal' 'weer' 'zonder' 'gebruik' 'ook' 'hoog' 'laag' 'groot' 'klein' 'geweldig' 'alleen' 'meest' 'gaan' 'gaan' 'maar' 'enkelen' 'zoals' 'wie' 'van' 'wie' 'dit' 'onder' 'deze'};
printStr=['Dutch',char(13)]; %Dutch
elseif (lang==8) delworda={'уже' 'было' 'было' 'было' 'тоже' 'где' 'где' 'тогда' 'тогда' 'таким образом' 'чем' 'мы' 'вы' 'вы' 'мы' 'это' 'его' 'они' 'будут' 'должны' 'могли' 'от' 'до' 'когда' 'снова' 'назад' 'раньше' 'после' 'между' 'оба' 'либо' 'просто' 'каждый' 'часто' 'обычно' 'без' 'использования' 'также' 'высокий' 'низкий' 'большой' 'маленький' 'большой' 'только' 'большинство' 'чьи' 'под' 'эти' 'те' 'это'};
printStr=['Russian',char(13)]; %Russian
elseif (lang==9) delworda={'magis' 'Iam' 'ex' 'a' 'esse' 'non' 'habet' 'habetis' 'est' 'habuit' 'etc' 'enim' 'aut' 'fiat' 'omnis' 'in' 'per' 'debet' 'sicut' 'quis' 'quisque' 'potest' 'possem' 'a' 'in' 'quid' 'quod' 'ubi' 'quod si' 'nunc' 'haec' 'est' 'hoc' 'et' 'te' 'nostra' 'ut' 'suum' 'non' 'erit' 'si' 'eorum' 'debet' 'ut' 'potentia' 'dum' 'at' 'in' 'quam' 'quondam' 'adhuc' 'ante' 'prius' 'postquam' 'inter' 'neque' 'iustus' 'uterque' 'saepenumero' 'plerumque' 'absque' 'usus' 'etiam' 'altum' 'humilis' 'magnum' 'parva' 'magnus' 'solum' 'maximum' 'futurum' 'vade' 'autem' 'quidam' 'haec' 'est' 'qui' 'per' 'haec' 'et'}; printStr=['Latin',char(13)]; %Latin
elseif (lang==10) delworda={'adalah' 'punya' 'untuk' 'dengan' 'atau' 'a' 'biarkan' 'an' 'semua' 'menyeberang' 'harus' 'sebagai' 'setiap' 'setiap' 'dapat' 'bisa' 'oleh' 'menjadi' 'apa' 'di sana' 'kapan' 'yang' 'di mana' 'bahwa' 'jika' 'kemudian' 'demikian' 'daripada' 'kami' 'Anda' 'Anda' 'kami' 'itu' 'miliknya ' 'mereka' 'akan' 'akan' 'harus' 'seharusnya' 'mungkin' 'mungkin' 'dari' 'sementara' 'pada' 'dalam'

```
'pada' 'bagaimana' 'sekali' 'lagi' 'yang lalu' 'sebelumnya' 'setelah' 'antara' 'keduanya' ' hanya' 'kebanyakan' 'pergi' 'pergi' 'pergi' 'tetapi' 'beberapa' 'seperti' 'sebagai' 'siapa' 'yang' 'di bawah' 'ini' 'mereka' 'ini'}; printStr=['Indonesian',char(13)]; %Indonesian
elseif (lang==11) delworda={'的' '得' '地' '已经' '在' '和' '而' '具' '必' '或' '者' '是' '有' '没' '等' '对' '应' '也' '希望' '尤其' '意义' '然' '可' '以' '仍' '从' '导致' '到' '同' '建立' '为' '与' '于' '人' '一' '二' '三' '四' '五' '六' '七' '八' '九' '十' '1' '2' '3' '4' '5' '6' '7' '8' '9' '0' '个' '全部' '过去' '未来' '结束' '开始' '东' '西' '南' '北' '上' '下' '左' '右' '年' '月' '日' '小时' '分钟' '秒' '我' '你' '他' '她' '它' '们' '不' '必须' '因为' '任何' '每' '若' '通过' '成' '什么' '还' '别' '变化' '时候' '这' '那' '其中' '即' '如果' '所以' '那么' '因' '此' '除' '了' '会' '能' '够' '将' '要' '应当' '应该' '可' '以' '为' '同时' '在' '怎' '什' '么' '再' '前' '后' '左' '右' '之' '间' '两' '只' '次' '经常' '使' '用' '另' '里' '外' '高' '低' '大' '小' '最' '来' '去' '但' '些' '例' '如' '谁' '男' '女' '别' '黑' '白' '越' '更' '太' '，' '。' '、' '“' '”' '？' '！'};
printStr=['Chinese',char(13)]; %Chinese
end
sdel='';
for i=1:length(delworda)
if (i~=1) sdel=strcat(sdel,',',delworda(i));
else sdel=strcat(sdel,delworda(i));
end
end
sdel=sdel{1};
h=helpdlg(sdel,'Existing Delimiter Words');
[newdel,OK]=listdlg('liststring',{'No','Yes'},'listsize',[300 100],'OkString','OK','CancelString','Cancel','promptstring','Are there any additional stop words?','selectionmode','single');
if (newdel==2)
del=inputdlg({'Additional Delimiter Words (Separated by comma)'},'',1,{''});
del=strrep(del,',',char(32));
del=del{1};
del=strtrim(del);
i=1;
while (i>0)
spac=findstr(del,' ');
if (length(spac)==0)
ws(i)=cellstr(del);
break; end
ws(i)=cellstr(del(1:(spac(1)-1)));
del=strtrim(del(spac(1):end));
i=i+1;
end
ws=lower(ws);
delworda=[delworda ws];
end
close(h);
[langs,OK]=listdlg('liststring',{'Pearson Correlation (Linear Correlation)','Spearman Correlation (Rank Correlation)','Point Correlation (0-1 Correlation)'},'listsize',[300 100],'OkString','OK','CancelString','Cancel','promptstring','Correlation Measure','selectionmode','single');
if (langs==1) cor=1; printStr=['Language: ',printStr,char(13),'Correlation Measure: Pearson Correlation (Linear Correlation)',char(13),char(13)];
elseif (langs==2) cor=2; printStr=['Language: ',printStr,char(13),'Correlation Measure: Spearman Rank Correlation (Rank
```

```
Correlation)',char(13),char(13)];
elseif (langs==3) cor=3; printStr=['Language: ',printStr,char(13),'Correlation Measure: Point Correlation (0-1 Correlation or
Association Coefficient)',char(13),char(13)];
end
answer=inputdlg({'Number of Random Samplings (e.g., 500)','Frequency Threshold ((0,1])','Significance Level
((0,0.1])','Proportion of Minimum Frame Size ((0,0.5])','Proportion of Maximum Frame Size ((0,0.5])','Proportion of Words for
Finding Word Chains ((0,1])'},'',1,{'500','0.9','0.001','0.001','0.1','0.1'});
ran=str2double(answer(1));
thr=str2double(answer(2));
sig=str2double(answer(3));
minf=str2double(answer(4));
maxf=str2double(answer(5));
thm=str2double(answer(6));
printStr=strcat(printStr,[char(13),char(13),'Number of Randamizations=',num2str(ran),char(13),'Frequency
Threshold=',num2str(thr),char(13),'Significance Level p=',num2str(sig),char(13),'Proportion of Minimum Frame
Size=',num2str(minf),char(13),'Proportion of Maximum Frame Size=',num2str(maxf)]);
[filloc,OK]=listdlg('liststring',{'Local File(s)','Internet URL(s)'},'listsize',[250
80],'OkString','OK','CancelString','Cancel','promptstring','File(s) Location','selectionmode','single');
%Read filenames
if (filloc==1)
[filename,pathname,filterindex]=uigetfile({'*.doc';'*.docx';'*.wps';'*.txt';'*.pdf';'*.csv';'*.html';'*.htm';'*.shtml';'*.shtm';'*.xhtml';'
*.xhtm';'*.asp';'*.xml';'*.*'},'Open one or more files','MultiSelect','on');
if isstr(filename) filsel=1;
else filsel=2;
end
elseif (filloc==2)
[filename,pathname]=uigetfile({'*.doc';'*.docx';'*.wps';'*.rtf';'*.txt';'*.pdf';'*.csv';'*.*'},'Open the URL(s) file');
ha=msgbox('In the URL(s) file, URLs are separated by ";", ",", or space.');
if isequal(filename,0) h=warndlg('The URL(s) file should be chosen','');
else file=fullfile(pathname,filename);
end
fid=fopen(file,'r');
[pathstr,name,ext]=fileparts(filename);
s='';
if (strcmpi(ext,'.txt') | strcmpi(ext,'.csv') )
sx=textread(file,'%s');
st='';
for i=1:size(sx,1);
st=strcat(st,[char(32) sx{i}]);
end
s=strcat(s,st);
elseif (strcmpi(ext,'.doc') | strcmpi(ext,'.docx') | strcmpi(ext,'.wps') | strcmpi(ext,'.rtf'))
try
Word=actxGetRunningServer('Word.Application');
catch
```

```
Word=actxserver('Word.Application');
end;
Document=Word.Documents.Open(file);
set(Word,'Visible',0);
s=strcat(s,get(Document.Content,'text'));
elseif strcmpi(ext,'.pdf')
try
Word=actxGetRunningServer('Word.Application');
catch
Word=actxserver('Word.Application');
end;
Document=invoke(Word.Documents,'Open',file);
%Document=Word.Documents.Open(file);
set(Word,'Visible',0);
s=strcat(s,get(Document.Content,'text'));
end
s=strrep(s,',',char(32));
s=strrep(s,';',char(32));
s=strrep(s,char(13),char(32));
s=strrep(s,'\n',char(32));
s=strrep(s,'\r\n',char(32));
s=strtrim(s);
i=1;
while (i>0)
space=findstr(s,' ');
if (length(space)==0)
urls(i)=cellstr(s);
break; end
urls(i)=cellstr(s(1:(space(1)-1)));
s=strtrim(s(space(1):end));
i=i+1;
end
filename=urls;
if (length(filename)==1)
filename=urls{1};
filsel=1;
else filsel=2;
end
close(ha);
end
%Number of files
if (filsel==2) filnum=size(filename,2);
else filnum=1;
end
htmlid=0;
```

```
s='';
for id=1:filnum
if (filloc==1)
if (filnum~=1) file=fullfile(pathname,filename{id});
fid=fopen(file,'r');
[pathstr,name,ext]=fileparts(filename{id});
else file=fullfile(pathname,filename);
fid=fopen(file,'r');
[pathstr,name,ext]=fileparts(filename);
end
elseif (filloc==2)
com.mathworks.mlwidgets.html.HTMLPrefs.setProxyHost;
if (filnum~=1) file=filename{id};
fid=fopen(file,'r');
[pathstr,name,ext]=fileparts(filename{id});
else file=filename;
fid=fopen(file,'r');
[pathstr,name,ext]=fileparts(filename);
end
end
try
if (strcmpi(ext,'.txt') | strcmpi(ext,'.csv'))
sx=textread(file,'%s');
st='';
for i=1:size(sx,1);
st=strcat(st,[char(32),sx{i}]);
end
s=strcat(s,st);
elseif (strcmpi(ext,'.doc') | strcmpi(ext,'.docx') | strcmpi(ext,'.wps') | strcmpi(ext,'.rtf'))
try
Word=actxGetRunningServer('Word.Application');
catch
Word=actxserver('Word.Application');
end;
Document=Word.Documents.Open(file);
set(Word,'Visible',0);
s=strcat(s,[char(32),get(Document.Content,'text')]);
elseif strcmpi(ext,'.pdf')
try
Word=actxGetRunningServer('Word.Application');
catch
Word=actxserver('Word.Application');
end;
Document=invoke(Word.Documents,'Open',file);
%Document=Word.Documents.Open(file);
```

```
set(Word,'Visible',0);
s=strcat(s,[char(32),get(Document.Content,'text')]);
elseif (strcmpi(ext,'.html') | strcmpi(ext,'.htm') | strcmpi(ext,'.shtml') | strcmpi(ext,'.shtm') | strcmpi(ext,'.xhtml') |
strcmpi(ext,'.xhtm') | strcmpi(ext,'.asp') | strcmpi(ext,'.xml'))
if (filloc==1) url=[strcat('file:///',file)];
elseif (filloc==2) url=urls{id};
end
s=strcat(s,[char(32),urlread(url,'get','','GBK')]);
if (htmlid==0)
ws={'xml' 'item' 'shtml' 'shtm' 'xhtml' 'xhtm' 'description' 'author' 'http' 'www' 'html' 'head' 'body' 'font' 'table' 'border' 'doctype' 'dtd'
'cellspacing' 'cellpadding' 'width' 'height' 'th' 'tr' 'td' 'div' 'align=' 'bgcolor' 'center' 'href=' '</' '/' '>' '<' 'h' 'br' 'hr' 'ul' 'ol' 'dl' 'dd' 'li' 'b'
'p' 'http://www.' 'script' 'size' 'i' 'u' 'em' 'sub' 'sup' 'src=' 'img' 'vlink' 'alink' 'background' 'frameset' 'rows' 'cols' 'form' 'input' 'type'
'value' 'name' 'option' 'textarea' 'xmlns' 'w3c' 'dtd' 'meta' 'cotent' 'asap' 'href' 'css' 'style' 'title' 'colspan' 'src' 'window' 'document' ""};
delworda=[delworda ws];
end
htmlid=1;
else
if (filloc==1) url=[strcat('file:///',file)];
elseif (filloc==2) url=urls{id};
end
s=strcat(s,[char(32),urlread(url,'get','','GBK')]);
end
catch
%Continue
end
end
fclose all;
hand=msgbox('Please be patience and wait for results...');
for i=1:length(delworda)
delworda(i)=strcat({' '},delworda(i),{' '});
del=char(delworda(i));
delwordA(i)=strcat({' '},upper(del(2)),del(3:end));
if (lang==11)                %For Chinese
delworda(i)=strtrim(delworda(i));
delwordA(i)=strtrim(delwordA(i));
end
end
%ASCII 32: space
%ASCII 48-57: 0-9
%ASCII 65-90: A-Z
%ASCII 97-122: a-z
%ASCII 0-31127: format characters
ch=zeros(1,128);
for i=1:32
ch(i)=i-1;
```

```
end;
ch(33)=127;
%ASCII 33, 34, 38, 39-41, 44, 46, 58, 59, 63: comma symbols
ch(34)=33; ch(35)=34; ch(36)=38; ch(37)=39; ch(38)=40; ch(39)=41;
ch(40)=44; ch(41)=46; ch(42)=58; ch(43)=59;ch(44)=63;
%ASCII 35, 36, 37: # $ %
ch(45)=35; ch(46)=36; ch(47)=37;
%ASCII 64, 96, 124, 126: @ ` | ~
ch(48)=64; ch(49)=96; ch(50)=124; ch(51)=126;
%ASCII 92, 94, 95: \ ^ _
ch(52)=92; ch(53)=94; ch(54)=95;
%ASCII 40, 41, 91, 93, 123, 125:    ( ) [ ] { }
ch(55)=40; ch(56)=41; ch(57)=91; ch(58)=93; ch(59)=123; ch(60)=125;
%ASCII 43, 45, 42, 47: + - * /
ch(61)=43; ch(62)=45; ch(63)=42; ch(64)=47;
%ASCII 60, 61, 62: < = >
ch(65)=60; ch(66)=61; ch(67)=62;
c=67;
%for i=1:10;ch(c+i)=47+i;end;c=77;
for i=1:c
s=strrep(s,char(ch(i)),char(32));
end
for i=1:length(delworda)
s=strrep(s,delworda(i),char(32));
end
s=s{1};
for i=1:length(delwordA)
s=strrep(s,delwordA(i),char(32));
end
s=s{1};
s=strtrim(s);
i=1;
while (i>0)
space=findstr(s,' ');
if (length(space)==0)
words(i)=cellstr(s);
break; end
words(i)=cellstr(s(1:(space(1)-1)));
s=strtrim(s(space(1):end));
i=i+1;
end
words=lower(words);
nn=length(words);
uniwords=unique(words);
d=length(uniwords);
```

```
for i=1:d
uni(i)=sum(ismember(words,uniwords(i)));
end
rank=sortrows([(1:d)' uni'],-2);
rt=uniwords(rank(:,1));
rk=rank(:,2)/nn;
m=0;
su=0;
for i=1:d
su=su+rk(i);
if (su>=thr) m=i; break; end
end
if (i==d) m=d; end
uniqwords=rt(1:m)';
fclose all;
printStr=strcat(printStr,[char(13),char(13),'There are totally ',num2str(nn),' unique words. ']);
printStr=strcat(printStr,[char(13),char(13),'Unique words, word occurrence frequencies and word proportions against total unique words, with occurrence frequency greater than ',num2str(thr*100),'%',char(13),char(13)]);
strr='';
for i=1:m
strr=strcat(strr,[char(13),num2str(i),'    ',cell2mat(uniqwords(i)),'    ',num2str(rank(i,2)),'    ',num2str(rk(i))]);
end
printStr=strcat(printStr,strr);
X=zeros(m,ran);
minframe=round(minf*nn+0.5);
if (minframe<2)
minframe=2;
end
maxframe=round(maxf*nn+0.5);
for k=1:ran
q=round(rand*(maxframe-minframe)+1)+minframe-1;
p=round(rand*(nn-q));
for i=1:m
for j=p+1:p+q
if isequal(uniqwords(i),words(j)) X(i,k)=X(i,k)+1; end;
end
end
end
%Pearson correlation
if (cor==1)
dim=size(X);
m=dim(1); n=dim(2);
%Pearson correlation matrix: r
r=corr(X');
tvalues=abs(r)./sqrt((1-r.^2)/(n-2));
```

```
alpha=(1-tcdf(tvalues,n-2))*2;
sigmat=alpha<sig;
sigmat=sigmat.*r-eye(m);
if (sigmat~=ones(m))
s='Word pairs with statistically significant Pearson correlation:';
printStr=sigPairs(s,printStr,sigmat,uniqwords);
else
printStr=strcat(printStr,['No significant word pairs',char(13),char(13)]);
end
printStr=netAnaly(printStr,sigmat,uniqwords,thm);
if (n>m)
%Partial correlation matrix: parr
inversr=inv(r);
for i=1:m-1; for j=i+1:m; parr(i,j)=-inversr(i,j)/sqrt(inversr(i,i)*inversr(j,j));end;end;
for i=1:m-1; for j=i+1:m; parr(j,i)=parr(i,j);end;end;
for i=1:m; parr(i,i)=1;end;
tvalues=abs(parr)./sqrt((1-parr.^2)/(n-m));
alpha=(1-tcdf(tvalues,n-m))*2;
sigmat=alpha<sig;
sigmat=sigmat.*parr-eye(m);
if (sigmat~=ones(m))
s='Word pairs with statistically significant Pearson partial correlation:';
printStr=sigPairs(s,printStr,sigmat,uniqwords);
else
printStr=strcat(printStr,[char(13),'No significant word pairs']);
end
printStr=netAnaly(printStr,sigmat,uniqwords,thm);
end
end
%Spearman rank correlation
if (cor==2)
dim=size(X);
m=dim(1); n=dim(2);
for u=1:m-1
for v=u+1:m
temx=X(u,1:n);
temy=X(v,1:n);
rx=zeros(1,n); ry=zeros(1,n); xx=zeros(1,n); yy=zeros(1,n);
for j=1:n
nx=1;ny=1;
for i=1:n
if (temx(i)<temx(j)) nx=nx+1; end
if (temy(i)<temy(j)) ny=ny+1; end
end
rx(j)=nx;
```

```
ry(j)=ny;
end
for j=1:n
if (rx(j)==(n+1)) continue; end
nx=rx(j);
ntie=-1;
for i=1:n
if (rx(i)~=nx) continue; end
ntie=ntie+1;
xx(i)=rx(i);
rx(i)=0;
end
for i=1:n
if (rx(i)~=0) continue; end
xx(i)=xx(i)+(ntie*0.5);
rx(i)=n+1;
end
end
for j=1:n
if (ry(j)==(n+1)) continue; end
ny=ry(j);
ntie=-1;
for i=1:n
if (ry(i)~=ny) continue; end
ntie=ntie+1;
yy(i)=ry(i);
ry(i)=0;
end
for i=1:n
if (ry(i)~=0) continue; end
yy(i)=yy(i)+ntie*0.5;
ry(i)=n+1;
end
end
rs=0;
rs=sum((xx-yy).^2);
r(u,v)=1-((6*rs)/(n*(n^2-1)));
r(v,u)=r(u,v);
end; end
for i=1:m
r(i,i)=1;
end
tvalues=abs(r)./sqrt((1-r.^2)/(n-2));
alpha=(1-tcdf(tvalues,n-2))*2;
sigmat=alpha<sig;
```

```
sigmat=sigmat.*r-eye(m);
if (sigmat~=ones(m))
s='Word pairs with statistically significant Spearman correlation:';
printStr=sigPairs(s,printStr,sigmat,uniqwords);
end
printStr=netAnaly(printStr,sigmat,uniqwords,thm);
end
%Point correlation
if (cor==3)
dim=size(X);
m=dim(1); n=dim(2);
for i=1:m-1
for j=i:m
x=X(i,:); y=X(j,:);
aa=sum((x==0) & (y==0));
bb=sum((x==0) & (y~=0));
cc=sum((x~=0) & (y==0));
dd=sum((x~=0) & (y~=0));
pointcorr(i,j)=(aa*dd-bb*cc)/sqrt((aa+bb)*(cc+dd)*(aa+cc)*(bb+dd));
chi2(i,j)=n*(aa*dd-bb*cc)^2/((aa+bb)*(cc+dd)*(aa+cc)*(bb+dd));
pointcorr(j,i)=pointcorr(i,j);
chi2(j,i)=chi2(i,j);
end; end
chi2test=chi2>chi2inv(1-sig,1);
%chi2=10.8 for alpha=0.001; chi2=6.635 for alpha=0.01; chi2=7.87 for alpha=0.005
sigmat=chi2test.*pointcorr-eye(m);
s='Word pairs with statistically significant Point correlation:';
printStr=sigPairs(s,printStr,sigmat,uniqwords);
printStr=netAnaly(printStr,sigmat,uniqwords,thm);
end
close(hand);
save=0;
[filename,pathname]=uiputfile({'*.doc';'*.docx';'*.txt';'*.*'},'Sava as');
file=fullfile(pathname,filename);
[pathstr,name,ext]=fileparts(filename);
if (strcmpi(ext,'.doc') | strcmpi(ext,'.docx'))
try
Word=actxGetRunningServer('Word.Application');
catch
Word=actxserver('Word.Application');
end;
set(Word,'Visible',0);
document=invoke(Word.Documents,'Add');
document.SaveAs(file);
content=document.Content;
```

```
selection=Word.Selection;
paragraphformat=selection.ParagraphFormat;
content.Start=0;
save=1;
content.Text=cell2mat(printStr);
document.Save;
elseif (strcmpi(ext,'.txt'))
fid=fopen(file,'wt');
save=2;
printStr=strrep(printStr,char(13),'\n');
fprintf(fid,cell2mat(printStr));
end
fclose all;

function printStr=wordModules(printStr,a,uniqwords)
dim=size(a); m=dim(1);
for i=1:m-1
for j=i+1:m
if (a(i,j)~=0)
r(i,j)=0;
for k=1:m
if ((a(i,k)==a(k,j)) & (a(i,k)~=0)) r(i,j)=r(i,j)+1; end
end
r(i,j)=r(i,j)/m; r(j,i)=r(i,j);
d(i,j)=1-r(i,j); d(j,i)=d(i,j);
else r(i,j)=0; r(j,i)=0; d(i,j)=1; d(j,i)=1;
end; end; end
d1=d;
bb1=1;
u(bb1)=0;
nu(bb1)=m;
for i=1:nu(bb1) x(bb1,i)=i; end
for i=1:nu(bb1) y(bb1,i)=1; end
while (nu(bb1)>1)
aa=1e+10;
for i=1:nu(bb1)-1
for j=i+1:nu(bb1)
if (d(i,j)<=aa) aa=d(i,j); end
end; end
aa1=0;
for i=1:nu(bb1)-1
for j=i+1:nu(bb1)
if (abs(d(i,j)-aa)<=1e-06)
aa1=aa1+1; v(aa1)=i; w(aa1)=j;
end; end; end
```

```
for i=1:nu(bb1) s(i)=0; end
nn1=0;
for i=1:aa1
if ((v(i)~=0) & (w(i)~=0))
nn1=nn1+1;
for j=1:aa1
if ((v(j)==v(i)) | (v(j)==w(i)) | (w(j)==w(i)) | (w(j)==v(i)))
s(v(j))=nn1; s(w(j))=nn1;
if (j~=i) v(j)=0; w(j)=0; end; end
end
v(i)=0; w(i)=0;
end; end
for i=1:nn1
y(bb1+1,i)=0;
for j=1:nu(bb1)
if (s(j)==i)
for k=1:m
if (x(bb1,k)==j) x(bb1+1,k)=i; end
end
y(bb1+1,i)=y(bb1+1,i)+y(bb1,j);
end; end; end
for i=1:nu(bb1)
if (s(i)==0)
nn1=nn1+1;
for k=1:m
if (x(bb1,k)==i) x(bb1+1,k)=nn1; end
end
y(bb1+1,nn1)=y(bb1,i); end
end;
bb1=bb1+1;
u(bb1)=aa;
nu(bb1)=nn1;
for i=1:nu(bb1)-1
for j=i+1:nu(bb1)
d(i,j)=-1e+10;
for k=1:m
if (x(bb1,k)==i)
for kk=1:m
if (x(bb1,kk)==j)
if (d1(k,kk)>d(i,j)) d(i,j)=d1(k,kk); end
end; end; end; end
d(j,i)=d(i,j);
end; end
end;
for k=1:m
```

```
y(bb1,k)=1; end
for i=bb1-1:-1:1
rr=0;
for j=1:nu(i+1)
ww=0;
for k=1:m
if (y(i+1,k)==j) ww=ww+1; v(ww)=k; end
end
vv=0;
for ii=1:ww
ee=0;
for jj=ii-1:-1:1
if (x(i,v(ii))==x(i,v(jj))) y(i,v(ii))=y(i,v(jj)); break; end
ee=ee+1;
end
if (ee==ii-1) vv=vv+1; y(i,v(ii))=rr+vv; end
end
rr=rr+vv;
end; end
for k=1:bb1
rs(k)=1-u(k);
end;
s=1; i=0;
while (m>0)
ss=1;
for j=s+1:bb1
if (rs(j)==rs(s)) ss=ss+1; end;
end
s=s+ss;
i=i+1;
la(i)=s-1;
if (s>=bb1) break; end
end
bb1=i;
yy=zeros(m);
for k=1:bb1
for i=1:nu(la(k))
for j=1:m
if (y(la(k),j)==i) yy(k,j)=i; end;
end; end; end
for k=1:bb1
rss(k)=rs(la(k)); uu(k)=u(la(k)); nuu(k)=nu(la(k));
end
linkWeightMat=zeros(m);
linkClusterIDs=zeros(m);
```

```
id=0;
for k=1:bb1
for i=1:nuu(k)
numm=0;
for j=1:m
if (yy(k,j)==i) numm=numm+1; temp(numm)=j; end
end
simi=max(rss(k),1e-10);
id=id+1;
for ii=1:numm-1
for jj=ii+1:numm
if ((a(temp(ii),temp(jj))~=0) & (linkWeightMat(temp(ii),temp(jj))==0))
linkWeightMat(temp(ii),temp(jj))=simi; linkWeightMat(temp(jj),temp(ii))=simi;
linkClusterIDs(temp(ii),temp(jj))=id; linkClusterIDs(temp(jj),temp(ii))=id;
end
end; end; end
end
dif=zeros(1,m);
su=0;
for i=1:m-1
for j=i+1:m
sm=0;
for k=1:su
if ((linkClusterIDs(i,j)~=0) & (linkClusterIDs(i,j)~=dif(k)))
sm=sm+1;
end; end
if (sm==su) su=su+1; dif(su)=linkClusterIDs(i,j); end
end; end
ma=max(max(linkClusterIDs));
for k=1:su
for i=1:m-1
for j=i+1:m
if ((linkClusterIDs(i,j)~=0) & (linkClusterIDs(i,j)==dif(k)) & (linkClusterIDs(i,j)~=ma))
linkClusterIDs(i,j)=k; linkClusterIDs(j,i)=k;
end; end; end; end
idnew=su;
for i=1:m-1
for j=i+1:m
if (linkClusterIDs(i,j)==ma)
idnew=idnew+1; linkClusterIDs(i,j)=idnew; linkClusterIDs(j,i)=idnew;
end
end; end
for k=1:idnew-1
s=0;
for i=1:m-1
```

```
for j=i+1:m
if (k~=linkClusterIDs(i,j)) s=s+1; end
end; end
if (s==(m*(m-1)/2))
for i=1:m-1
for j=i+1:m
if (linkClusterIDs(i,j)>k) linkClusterIDs(i,j)=linkClusterIDs(i,j)-1; linkClusterIDs(j,i)=linkClusterIDs(i,j); end;
end; end;
end; end
lab=zeros(1,idnew-1);
for k=1:idnew-1
s=0;
for i=1:m-1
for j=i+1:m
if (k==linkClusterIDs(i,j)) s=s+1; end
end; end;
lab(k)=s;
end
iss='';
for k=1:idnew-1
la=0;
for i=1:m-1
for j=i+1:m
if (linkClusterIDs(i,j)==k) weig=linkWeightMat(i,j); la=1; break; end;
end;
if (la==1) break; end
end;
if (lab(k)>1) iss=strcat(iss,[char(13),char(13),'Links in sub-network (cluster) ID ',num2str(k),' (Link weight=',num2str(weig),')']);
end;
if (lab(k)==1) iss=strcat(iss,[char(13),char(13),'Standalone link ID ',num2str(k),' (Link weight=',num2str(weig),')']); end;
for i=1:m-1
for j=i+1:m
if (k==linkClusterIDs(i,j)) iss=strcat(iss,['(',cell2mat(uniqwords(i)),',',cell2mat(uniqwords(j)),') ']); end
end; end;
end
printStr=strcat(printStr,iss);

function printStr=netAnaly(printStr,sigmat,uniqwords,thm)
%Word centrality, word tree, word modules, etc.
m=size(sigmat,1);
str='';
str=strcat(str,[char(13),'Network analysis for the correlational network constructed from significant between-word positive
correlations among above ',num2str(m),' words']);
str=strcat(str,[char(13),'Word importance (degree centrality: the degree of a word, i.e., the number of links owned by a word,
which represents the word importance.)']);
```

```
str=strcat(str,[char(13),'Word importance (Closeness centrality: a global centrality representing the independence of a word in the network. Closeness centrality is defined as reciprocal of the sum of the word¡¯s geodesic distances to all other words (the distances of the shortest paths) in the network.)']);
str=strcat(str,[char(13),'Word importance (Betweenness centrality: it represents the word¡¯s ability to control the data flow in the network. This measure is the proportion of number of geodesic paths that pass through the given word to total number of geodesic paths between any pair of words in the network.)']);
str=strcat(str,[char(13),'Word tree: the tree with minimum total links. It represents the simplest topological structure that connects all words.']);
str=strcat(str,[char(13),'Word modules: the words in the same module are more closely connected. Closerly connected link has greater weight.']);
str=strcat(str,[char(13),'Word chains: a chain denotes a link series between two words (here it means the chain with minimum total links)']);
str=strcat(str,[char(13),'']);
str=strcat(str,[char(13),'References:']);
str=strcat(str,[char(13),'Zhang WJ. Fundamentals of Network Biology. World Scientific Europe, London, 2018']);
str=strcat(str,[char(13),'Zhang WJ. Selforganizology: The Science of Self-Organization. World Scientific, Singapore, 2016']);
str=strcat(str,[char(13),'Zhang WJ. Computational Ecology: Graphs, Networks and Agent-based Modeling. World Scientific, Singapore, 2012']);
str=strcat(str,[char(13),'Zhang WJ. 2017. Finding the shortest tree in the network: A Matlab program and application in tumor pathway. Network Pharmacology, 2(1): 13-16']);
str=strcat(str,[char(13),'Zhang WJ. 2016. Screening node attributes that significantly influence node centrality in the network. Selforganizology, 3(3): 75-86']);
str=strcat(str,[char(13),'Zhang WJ. 2016. How to find cut nodes and bridges in the network? A Matlab program and application in tumor pathways. Network Pharmacology, 1(3): 82-85']);
str=strcat(str,[char(13),'Zhang WJ. 2016. A Matlab program for finding shortest paths in the network: Application in the tumor pathway. Network Pharmacology, 1(1): 42-53']);
str=strcat(str,[char(13),'Zhang WJ. 2016. Finding trees in the network: Some Matlab programs and application in tumor pathways. Network Pharmacology, 1(2): 66-73']);
str=strcat(str,[char(13),'Zhang WJ. 2016. A method for identifying hierarchical sub-networks / modules and weighting network links based on their similarity in sub-network / module affiliation. Network Pharmacology, 1(2): 54-65']);
str=strcat(str,[char(13),'Zhang WJ, Li X. 2016. A cluster method for finding node sets / sub-networks based on between- node similarity in sets of adjacency nodes: with application in finding sub-networks in tumor pathways. Proceedings of the International Academy of Ecology and Environmental Sciences, 6(1): 13-23']);
str=strcat(str,[char(13),'Zhang WJ. 2012. Several mathematical methods for identifying crucial nodes in networks. Network Biology, 2(4): 121-126']);
str=strcat(str,[char(13),'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']);
printStr=strcat(printStr,[char(13),str]);
sigmm=(((sigmat>0).*sigmat)~=0).*ones(m);
printStr=strcat(printStr,[char(13),char(13),'Word importance (Degree centrality: a local centrality; the number of a word¡¯s neighboring words in the network, reflecting the word influence on its neighbourhoods.):']);
deg=sum(sigmm);
rr=sortrows([(1:m)' deg'],-2);
strr='';
for i=1:m
```

```
strr=strcat(strr,[char(13),cell2mat(uniqwords(rr(i,1))),'    ',num2str(rr(i,2))]);
end
printStr=strcat(printStr,strr);
try
printStr=strcat(printStr,[char(13),char(13),'Word importance (Closeness centrality: a global centrality representing the independence of a word in the network. Closeness centrality is defined as reciprocal of the sum of the word¡¯s geodesic distances to all other words (the distances of the shortest paths) in the network.):']);
%Closeness centrality
[ss,pat,distances,paths]=Dijkstra(sigmm);
for i=1:m
deg(i)=1/sum(distances(i,:));
end
rr=sortrows([(1:m)' deg'],-2);
strr='';
for i=1:m
strr=strcat(strr,[char(13),cell2mat(uniqwords(rr(i,1))),'    ',num2str(rr(i,2))]);
end
printStr=strcat(printStr,strr);
catch
printStr=strcat(printStr,[char(13),'Word importance (Closeness centrality) is ignored due to data problem, e.g., too stricter significance level']);
end
try
printStr=strcat(printStr,[char(13),char(13),'Word importance (Betweenness centrality): it represents the word¡¯s ability to control the data flow in the network. This measure is the proportion of number of geodesic paths that pass through the given word to total number of geodesic paths between any pair of words in the network.):']);
%Betweenness centrality
[ss,pat,distances,paths]=Dijkstra(sigmm);
for i=1:m
deg(i)=pat(i)/ss;
end
rr=sortrows([(1:m)' deg'],-2);
strr='';
for i=1:m
strr=strcat(strr,[char(13),cell2mat(uniqwords(rr(i,1))),'    ',num2str(rr(i,2))]);
end
printStr=strcat(printStr,strr);
catch
printStr=strcat(printStr,[char(13),'Word importance (Betweenness centrality) is ignored due to data problem, e.g., too stricter significance level']);
end
if (rr(m,2)==0)
printStr=strcat(printStr,[char(13),char(13),'Isolated words: ',char(13),cell2mat(uniqwords(rr(i,1)))]);
end
strr='';
```

```
for i=m-1:-1:1
if (rr(i,2)~=0) break; end
strr=strcat(char(13),strr,',',cell2mat(uniqwords(rr(i,1))));
end
printStr=strcat(printStr,strr);
printStr=strcat(printStr,[char(13),char(13),'Word tree (the tree with minimum total links): ']);
try
tree=minTree(sigmm);
su=0;
strr='';
for i=1:m-1
for j=i+1:m
if (tree(i,j)==0) continue; end
su=su+1;
if (su~=1)
strr=strcat(strr,', (',uniqwords(i),',',uniqwords(j),')');
else
strr=strcat(char(13),strr,'(',uniqwords(i),',',uniqwords(j),')');
end
end; end
printStr=strcat(printStr,strr);
if isempty(strr) printStr=strcat(printStr,[char(13),'No minimum tree']); end
catch
printStr=strcat(printStr,[char(13),'Tree ignored due to data problem, e.g., insufficient random samplings']);
end
printStr=strcat(printStr,[char(13),char(13),'Word modules:']);
printStr=wordModules(printStr,sigmm,uniqwords);
printStr=strcat(printStr,[char(13),char(13),'Word chain between two words (a chain with minimum total links) for the first
',num2str(thm*100),'% words in ',num2str(m),' words:']);
try
thm=round(thm*m);
printStr=shortPath(sigmm,thm,uniqwords,printStr);
catch
printStr=strcat(printStr,[char(13),'Chains ignored due to data problem, e.g., insufficient random samplings']);
end
printStr=strcat(printStr,[char(13),char(13),'Copyright: Zhang WJ, 2018']);

function printStr=shortPath(d,thm,uniqwords,printStr)
%Floyd algorithm
%d=(dij)v*v, adjacency matrix of the weighted network, where v is the number of nodes in the network. dij=1 for unweighted
network and dij=wij for weighted network (wij is the weight for the link vi to vj), if vi and vj are adjacent, and dij=0, if vi and vj
are not adjacent; i, j=1,2,¡-, m
v=size(d,1);
a=zeros(v);
b=zeros(1,v*(v-1)/2);
```

```
e=zeros(1,v*(v-1)/2);
h=zeros(v*(v-1)/2);
inf=Inf;
strr='';
for i=1:v
for j=1:v
if ((d(i,j)==0) & (i~=j)) d(i,j)=inf; end
end; end
for i=1:v
for j=1:v
if (d(i,j)~=inf) a(i,j)=j; end
end; end
for i=1:v
for j=1:v
for k=1:v
c=d(j,i)+d(i,k);
if (c<d(j,k)) d(j,k)=c; a(j,k)=i; end
end; end; end
for p=1:v
for q=1:v
if (p==q) continue; end
u=a(p,q);
m=1;
b(1)=u;
while (v>0)
m=m+1;
b(m)=a(b(m-1),q);
if (q==b(m)) break; end
if (b(m)==b(m-1)) break; end
if (m>v) break; end
end
n=1;
e(1)=u;
while (v>0)
n=n+1;
e(n)=a(p,e(n-1));
if (p==e(n)) break; end
if (e(n)==e(n-1)) break; end
if (n>v) break; end
end
for i=1:m+n-1
if (i==1) h(i)=p; end
if ((i<=n) & (i>1)) h(i)=e(n-i+1); end
if ((i>n) & (i<(m+n-1))) h(i)=b(i-n+1); end
if (i==(m+n-1)) h(i)=q; end
```

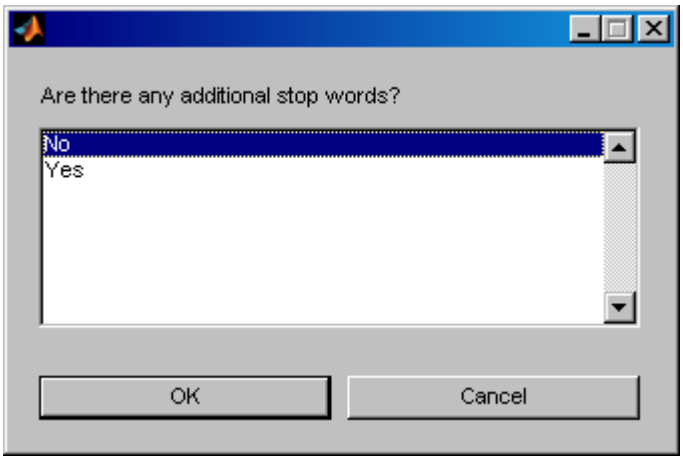
```
end
if ((q>p) & (p<=thm) & (q<=thm))
strr=strcat(strr,[char(13),'Chain between ',cell2mat(uniqwords(p)),' and ',cell2mat(uniqwords(q)),': ']);
for i=1:m+n-1
if ((h(i)~=0) & (d(p,q)~=inf))
if ((h(i)==h(i+1)) & (i<m+n-1)) continue; end
if (i<m+n-1)
strr=strcat(strr,uniqwords(h(i)),'->');
end
if (i>=m+n-1)
strr=strcat(strr,cell2mat(uniqwords(h(i))));
end
end; end
if (d(p,q)==inf)
strr=strcat(strr,[char(13),'No chain']);
end
end
end; end
printStr=strcat(printStr,strr)

function tree=minTree(a)
%Kruskal algorithm to calculate the shortest tree in a network.
%a: between-node weights matrix of the weighted network (e.g., adj.xls, etc. The matrix is a=(aij)v*v, where v is the number of nodes in the network. aij is the weight between the nodes i and j, i, j=1,2,¡-, v
v=size(a,1);
cc=0;
t=zeros(v); t1=zeros(v); b=zeros(1,10000000);
k=1;
for i=1:v-1
for j=i+1:v
if (a(i,j)>0)
b(k)=a(i,j);
kk=1;
for l=1:k-1
if (b(k)==b(l))
kk=0;
break;
end; end
k=k+kk;
end; end; end
k=k-1;
for i=1:k-1
for j=i+1:k
if (b(j)<b(i))
cc=b(j);
```

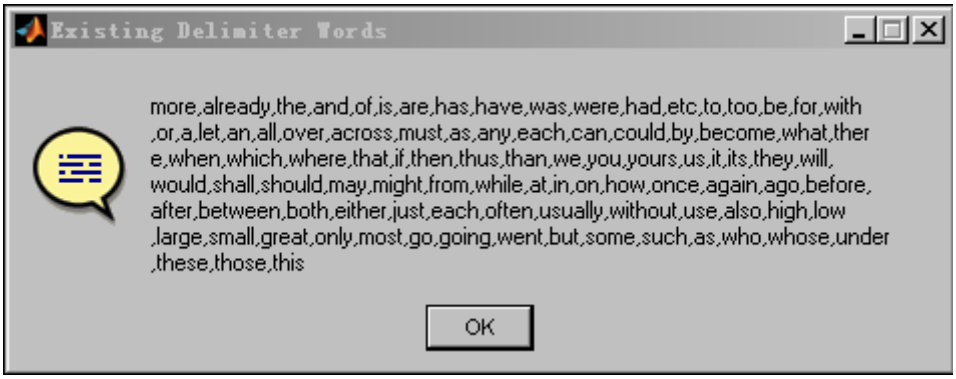
```
b(j)=b(i);
b(i)=cc;
end; end; end
m=0;
for l=1:k
if (m==v) break; end
for i=1:v-1
for j=i+1:v
if (a(i,j)==b(l))
t(i,j)=b(l);
t(j,i)=b(l);
for i1=1:v;
for j1=1:v;
t1(i1,j1)=t(i1,j1);
end; end
while(v>0)
in=1;
c=0;
for i1=1:v
kk=0;
for j1=1:v
if (t1(i1,j1)>0)
kk=kk+1;
c=j1;
end; end
if (kk==1)
t1(i1,c)=0;
t1(c,i1)=0;
in=0;
end
end
if (in~=0) break; end
end
in=0;
for i1=1:v-1
for j1=i1+1:v
if (t1(i1,j1)>0)
in=1;
break;
end; end; end
if (in~=0)
t(i,j)=0;
t(j,i)=0;
else m=m+1; end
end; end; end; end
```

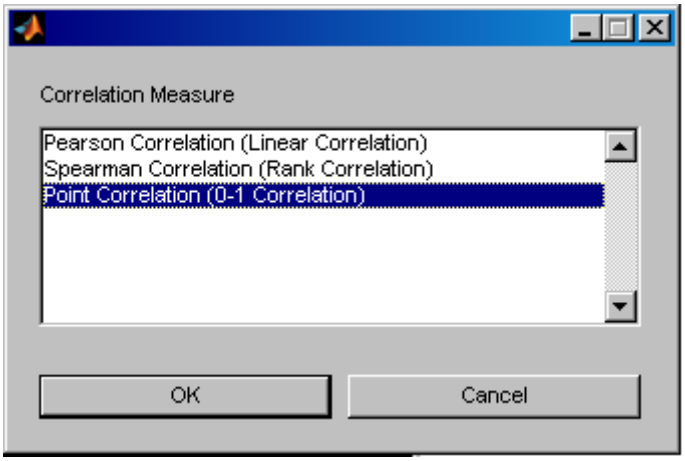
```
tree=t;
%tree: shortest tree

function [ss,pat,distances,paths]=Dijkstra(d)
% d: weighted adjacency matrix; ss: total number of paths; pat: number of paths passing through each node; distances: matrix of distances between different nodes; paths: string of paths and distances between any of two nodes
v=size(d,1);
p=zeros(1,v); w=zeros(1,v); a=zeros(1,v); b=zeros(1,v);
pat=zeros(1,v);
distances=zeros(v);
for i=1:v
for j=1:v
if ((d(i,j)==0) & (i~=j)) d(i,j)=inf; end
end; end
paths='';
su=0;
for j=1:v-1
for k=j+1:v
for i=1:v
p(i)=0; w(i)=0;
a(i)=inf;
end
a(j)=0; w(j)=1; n=j; h=0;
while (v>0)
ma=inf;
for i=1:v
if (w(i)==1) continue; end
iv=d(n,i)+a(n);
if (iv<a(i)) a(i)=iv; b(i)=n; end
if (a(i)>ma) continue; end
ma=a(i); h=i;
end
w(h)=1;
if (h==k) break; end
n=h;
end
sd=a(k); p(1)=k; c=k;
for i=2:v
if (c==j) break; end
p(i)=b(c); c=b(c);
end
paths=strcat(paths,'Shortest path from ',num2str(j),' to ',num2str(k),':\n');
for i=v:-1:1
if ((p(i)~=0) & (sd~=inf))
if (i>1) paths=strcat(paths,num2str(p(i)),'->'); end
```

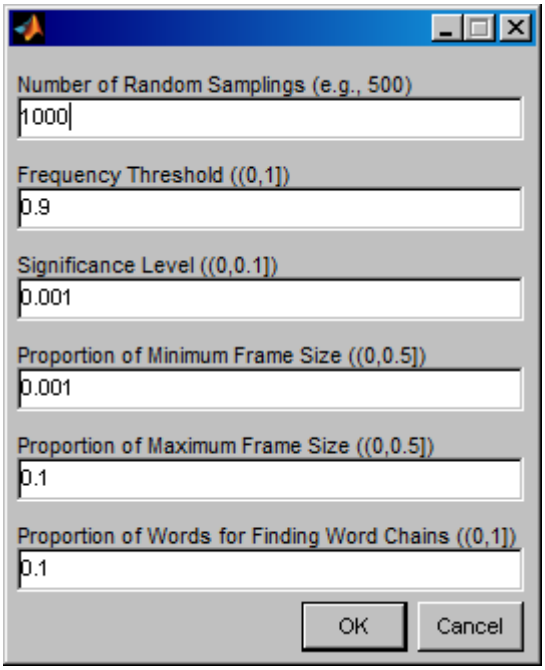
```
if (i<=1) paths=strcat(paths,num2str(p(i)),'\n'); end
end; end
for i=v:-1:1
for h=1:v
if (p(i)==h) pat(h)=pat(h)+1; break; end
end; end
if (sd~=inf) paths=strcat(paths,'Distance=',num2str(sd),'\n'); distances(j,k)=sd; distances(k,j)=sd; end
if (sd==inf) paths=strcat(paths,'No path','\n'); su=su+1; end
end; end
ss=v*(v-1)/2-su;

function printStr=sigPairs(s,printStr,sigmat,uniqwords)
m=size(sigmat,1);
str='';
str=strcat(str,[char(13),'The relational network constructed from significant between-word positive and negative correlations among above ',num2str(m),' words']);
str=strcat(str,[char(13),'References']);
str=strcat(str,[char(13),'Zhang WJ. 2011. Constructing ecological interaction networks by correlation analysis: hints from community sampling. Network Biology, 1(2): 81-98']);
str=strcat(str,[char(13),'Zhang WJ. 2012. How to construct the statistic network? An association network of herbaceous plants constructed from field sampling. Network Biology, 2(2): 57-68']);
str=strcat(str,[char(13),'Zhang WJ. Fundamentals of Network Biology. World Scientific Europe, London, 2018']);
str=strcat(str,[char(13),'Zhang WJ. Computational Ecology: Graphs, Networks and Agent-based Modeling. World Scientific, Singapore, 2012']);
printStr=strcat(printStr,[char(13),str]);
printStr=strcat(printStr,[char(13),char(13),s]);
printStr=strcat(printStr,[char(13),char(13),'Staistically significant positive correlations']);
[pairx,pairy,corrvalues]=find((sigmat>0).*sigmat);
temp1=pairx; temp2=pairy;
pairxs=pairx(temp1<temp2);
pairys=pairy(temp1<temp2);
corrvaluess=corrvalues(temp1<temp2);
PairsAndCorrelations=[uniqwords(pairxs) uniqwords(pairys) num2cell(corrvaluess)];
strr='';
for i=1:size(pairxs,1)
strr=strcat(strr,[char(13),num2str(pairxs(i)),'    ',cell2mat(uniqwords(pairxs(i))),'    <-->    ',num2str(pairys(i)),'
',cell2mat(uniqwords(pairys(i))),'    ',num2str(corrvaluess(i))]);
end
printStr=strcat(printStr,strr);
printStr=strcat(printStr,[char(13),char(13),'Staistically significant negative correlations']);
[pairx,pairy,corrvalues]=find((sigmat<0).*sigmat);
temp1=pairx; temp2=pairy;
pairxs=pairx(temp1<temp2);
pairys=pairy(temp1<temp2);
corrvaluess=corrvalues(temp1<temp2);
```

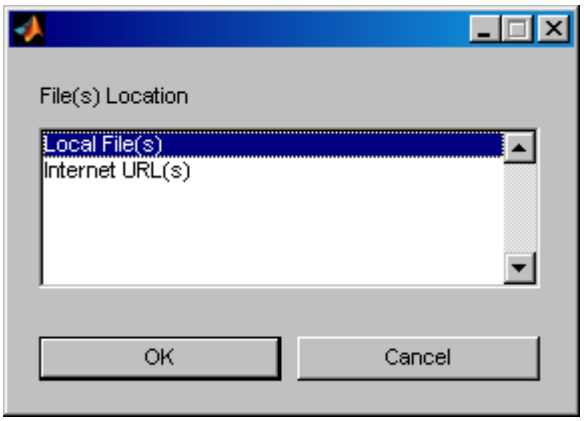
```
PairsAndCorrelations=[uniqwords(pairxs) uniqwords(pairys) num2cell(corrvaluess)];
strr='';
for i=1:size(pairxs,1)
strr=strcat(strr,[char(13),num2str(pairxs(i)),'  ',cell2mat(uniqwords(pairxs(i))),'  <-->  ',num2str(pairys(i)),'
',cell2mat(uniqwords(pairys(i))),'  ',num2str(corrvaluess(i))]);
end
printStr=strcat(printStr,strr);
```






Correlation Measure
Pearson Correlation (Linear Correlation)
Spearman Correlation (Rank Correlation)
Point Correlation (0-1 Correlation)
OK
Cancel


Number of Random Samplings (e.g., 500)
1000
Frequency Threshold ((0,1])
0.9
Significance Level ((0,0.1])
0.001
Proportion of Minimum Frame Size ((0,0.5])
0.001
Proportion of Maximum Frame Size ((0,0.5])
0.1
Proportion of Words for Finding Word Chains ((0,1])
0.1
OK
Cancel


File(s) Location
Local File(s)
Internet URL(s)
OK
Cancel

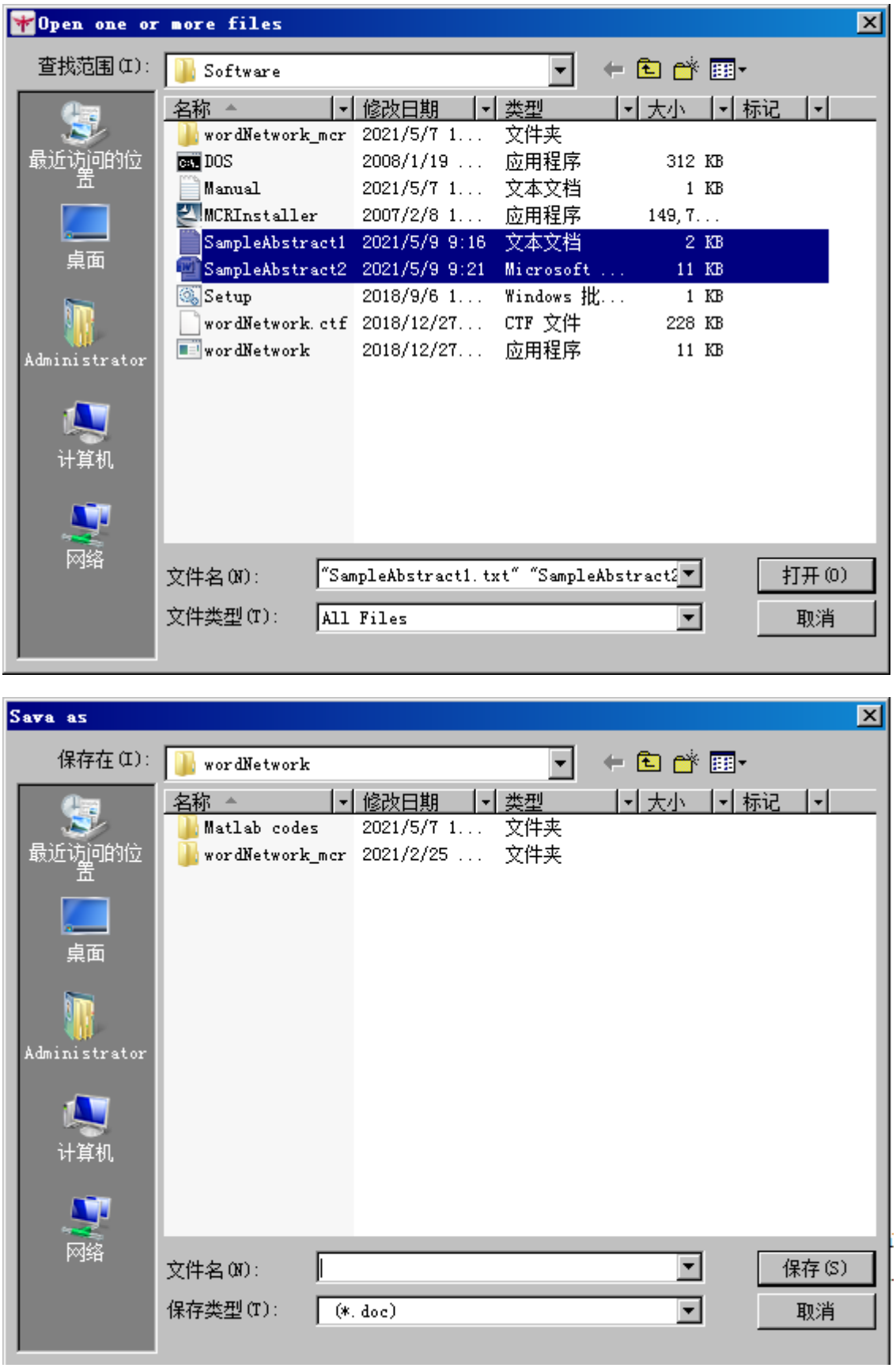


Fig. 1 Graphical interface of the software, wordNetwork.

3 Application Example

Let's consider the two files, each containing an English abstract, from two references (Zhang et al., 2014; Jiang and Zhang, 2015b). In the software, choose the language as English, and correlation measure is point correlation. Number of randomizations, frequency threshold, significance level, proportion of minimum frame size, and proportion of maximum frame size are 1000, 0.9, 0.001, 0.001 and 0.1 respectively. There are totally 145 unique words in the combined abstract. Table 1 shows partial results of word occurrence frequencies of the comnbined abstract. In total of 67 words are considered as important words (Cummulative occurrence frequency is greater than 90%).

The results show that there are statistically 435 significant positive correlations and 336 statistically significant negative correlations (Table 2). A word network with 435 links of positive correlations (Fig. 2) and a word network with 771 links of both positive and negative correlations (Fig. 3) are thus constructed.

Network analysis show that webs / web / food / parasitism / CSM / robustness are crucial words in the combined abstract (Table 3), i.e., these words are central topics or focuses of the two articles.

Table 1 Unique words, word occurrence frequencies and word proportions against total unique words, with occurrence frequency greater than 90% (67 words).

ID	Word	Freq.	Prop.	ID	Word	Freq.	Prop.	ID	Word	Freq.	Prop.
1	food	13	0.089655	24	always	1	0.0068966	47	increase	1	0.0068966
2	web	10	0.068966	25	analyze	1	0.0068966	48	increased	1	0.0068966
3	webs	7	0.048276	26	analyzed	1	0.0068966	49	indices	1	0.0068966
4	parasites	6	0.041379	27	basal	1	0.0068966	50	intervality	1	0.0068966
5	csm	5	0.034483	28	based	1	0.0068966	51	links	1	0.0068966
6	parasitism	4	0.027586	29	carpinteria	1	0.0068966	52	living	1	0.0068966
7	predator	4	0.027586	30	clusion	1	0.0068966	53	lower	1	0.0068966
8	prey	4	0.027586	31	clustering	1	0.0068966	54	lso	1	0.0068966
9	sub	4	0.027586	32	coefficient	1	0.0068966	55	marsh	1	0.0068966
10	full	3	0.02069	33	collected	1	0.0068966	56	median	1	0.0068966
11	results	3	0.02069	34	degree	1	0.0068966	57	not	1	0.0068966
12	species	3	0.02069	35	demonstrated	1	0.0068966	58	order	1	0.0068966
13	arthropod	2	0.013793	36	did	1	0.0068966	59	organizing	1	0.0068966
14	cascade	2	0.013793	37	dominated	1	0.0068966	60	pajek	1	0.0068966
15	fitting	2	0.013793	38	effect	1	0.0068966	61	parasite	1	0.0068966
16	generality	2	0.013793	39	effectiveness	1	0.0068966	62	performed	1	0.0068966
17	mean	2	0.013793	40	effects	1	0.0068966	63	predators	1	0.0068966
18	model	2	0.013793	41	existence	1	0.0068966	64	present	1	0.0068966
19	robustness	2	0.013793	42	free	1	0.0068966	65	previously	1	0.0068966
20	showed	2	0.013793	43	goodness	1	0.0068966	66	preys	1	0.0068966
21	study	2	0.013793	44	greater	1	0.0068966	67	reduce	1	0.0068966
22	vulnerability	2	0.013793	45	host	1	0.0068966				
23	affect	1	0.0068966	46	important	1	0.0068966				

Table 2 Positive and negative between-word correlations (point correlations) (23 pairs of words are listed respectively).

Positive Between-word Correlations					Negative Between-word Correlations				
ID	Word	ID	Word	Value	ID	Word	ID	Word	Value
1	food	2	web	0.3901	2	web	3	webs	-0.23639
1	food	3	webs	0.28262	2	web	6	parasitism	-0.14992
2	web	4	parasites	0.37873	5	csm	6	parasitism	-0.16152
1	food	5	csm	0.33534	1	food	7	predator	-0.18237
2	web	5	csm	0.1321	6	parasitism	7	predator	-0.20921
3	webs	5	csm	0.13658	1	food	8	prey	-0.24431
1	food	6	parasitism	0.10613	6	parasitism	8	prey	-0.2117
3	webs	6	parasitism	0.26107	1	food	9	sub	-0.25864
2	web	7	predator	0.14684	6	parasitism	9	sub	-0.18063
4	parasites	7	predator	0.28044	3	webs	10	full	-0.13707
2	web	8	prey	0.15188	6	parasitism	10	full	-0.24837
4	parasites	8	prey	0.31159	2	web	11	results	-0.19067
7	predator	8	prey	0.85827	9	sub	11	results	-0.12253
2	web	9	sub	0.22639	1	food	12	species	-0.12821
4	parasites	9	sub	0.46058	3	webs	12	species	-0.30122
7	predator	9	sub	0.62502	5	csm	12	species	-0.18698
8	prey	9	sub	0.76662	10	full	12	species	-0.11197
1	food	10	full	0.3043	11	results	12	species	-0.21137
2	web	10	full	0.37945	4	parasites	13	arthropod	-0.16818
4	parasites	10	full	0.22125	6	parasitism	13	arthropod	-0.16089
5	csm	10	full	0.34257	9	sub	13	arthropod	-0.11179
3	webs	11	results	0.41121	12	species	13	arthropod	-0.12978
5	csm	11	results	0.29702	3	webs	14	cascade	-0.13349

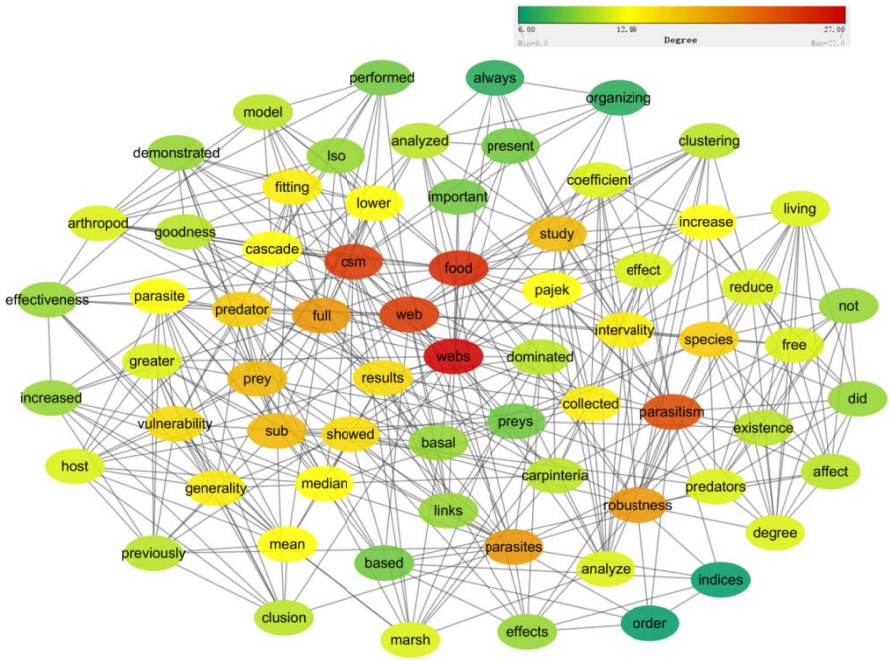


Fig. 2 The word network constructed from word pairs with significant positive point correlations.

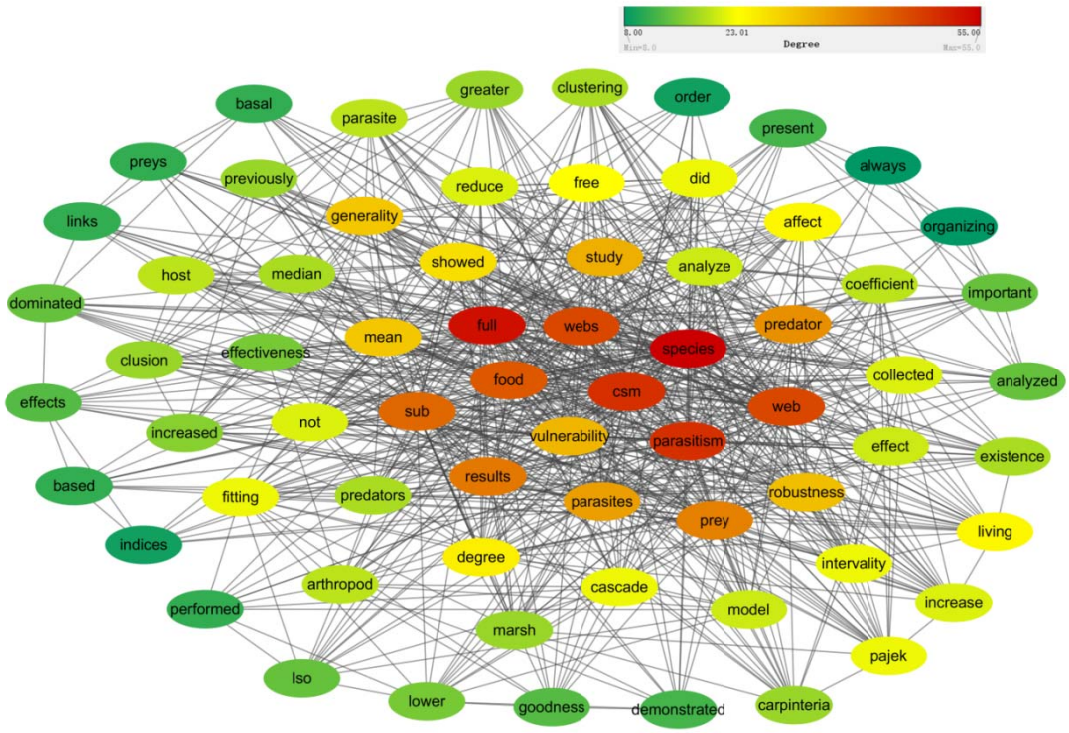


Fig. 3 The word network constructed from word pairs with significant positive and negative point correlations.

Table 3 Centrality measures of the important words.

Degree centrality		Closeness centrality		Betweenness centrality	
Word	Value	Word	Value	Word	Value
webs	27	food	0.0093458	web	0.10945
food	25	web	0.0092593	parasitism	0.10674
web	24	webs	0.0091743	webs	0.09498
csm	24	parasitism	0.0084746	predators	0.090909
parasitism	23	csm	0.0084034	sub	0.090457
parasites	19	robustness	0.0083333	robustness	0.088648
full	19	study	0.0083333	csm	0.072818
robustness	19	parasites	0.0082645	showed	0.068747
prey	17	showed	0.0082645	parasites	0.066938
sub	17	full	0.0081967	intervality	0.063772
study	17	results	0.0081967	preys	0.060606
predator	16	dominated	0.0081301	results	0.057892
species	16	collected	0.008	pajek	0.057892
results	15	prey	0.0079365	median	0.056988
showed	15	sub	0.0079365	food	0.055179
vulnerability	15	predator	0.007874	lower	0.053822
fitting	14	effects	0.0078125	study	0.05337
generality	14	basal	0.0077519	marsh	0.051108
collected	14	analyzed	0.0076923	prey	0.049299
intervality	14	carpinteria	0.0076923	full	0.048847
cascade	13	intervality	0.0076336	effects	0.048394
mean	13	pajek	0.0076336	previously	0.04749

increase	13	species	0.0075758	reduce	0.047038
lower	13	analyze	0.0075758	analyzed	0.045228
median	13	effect	0.0075758	species	0.044324
pajek	13	increase	0.0075758	parasite	0.043872
parasite	13	links	0.0075758	links	0.043419
arthropod	12	cascade	0.0075188	effectiveness	0.042515
analyze	12	important	0.0075188	present	0.040706
coefficient	12	lower	0.0075188	based	0.039349
degree	12	present	0.0075188	collected	0.039349
effect	12	coefficient	0.0074627	vulnerability	0.037992
free	12	arthropod	0.0074074	dominated	0.037992
greater	12	clustering	0.0073529	increased	0.03754
host	12	goodness	0.0073529	not	0.037087
living	12	marsh	0.0073529	arthropod	0.036183
marsh	12	median	0.0073529	lso	0.03573
predators	12	predators	0.0073529	goodness	0.035278
reduce	12	based	0.0072993	increase	0.035278
model	11	effectiveness	0.0072993	cascade	0.034826
affect	11	greater	0.0072993	predator	0.034374
analyzed	11	parasite	0.0072993	basal	0.033469
carpinteria	11	clusion	0.0072464	fitting	0.033017
clusion	11	previously	0.0072464	mean	0.033017
clustering	11	demonstrated	0.0071942	performed	0.033017
dominated	11	fitting	0.0071429	coefficient	0.032564
existence	11	model	0.0071429	living	0.032564
goodness	11	preys	0.0071429	model	0.032112
previously	11	host	0.0070922	order	0.03166
basal	10	lso	0.0070922	greater	0.030303
demonstrated	10	generality	0.0070423	generality	0.029851
did	10	existence	0.0070423	affect	0.029851
effectiveness	10	vulnerability	0.0069444	always	0.029851
effects	10	indices	0.0068966	analyze	0.029851
increased	10	order	0.0068966	carpinteria	0.029851
links	10	always	0.0068493	clusion	0.029851
lso	10	organizing	0.0068493	clustering	0.029851
not	10	mean	0.0068027	degree	0.029851
based	9	degree	0.0068027	demonstrated	0.029851
important	9	performed	0.0068027	did	0.029851
performed	9	affect	0.0067568	effect	0.029851
present	9	increased	0.0066667	existence	0.029851
preys	9	free	0.0064935	free	0.029851
always	7	living	0.0064935	host	0.029851
organizing	7	did	0.0064103	important	0.029851
indices	6	not	0.0064103	indices	0.029851
order	6	reduce	0.00625	organizing	0.029851

The shortest (minimum) word tree (the tree with minimum total links) is calculated, as shown in Table 4 and Fig. 4. There are totally 66 links in the tree.

Table 4 The word tree constructed from word pairs with significant positive point correlations.

Between word	and word	Between word	and word	Between word	and word
food	web	food	lso	webs	host
food	webs	food	pajek	webs	marsh
food	csm	Food	present	webs	organizing
food	parasitism	Web	parasites	webs	parasite
food	full	Web	predator	webs	previously
food	arthropod	Web	prey	parasites	existence
food	cascade	Web	sub	parasites	increased
food	model	Web	species	parasites	indices
food	robustness	Web	basal	parasites	order
food	study	Web	based	parasites	predators
food	analyze	Web	lower	csm	fitting
food	analyzed	Web	performed	csm	median
food	clustering	Web	preys	parasitism	affect
food	coefficient	webs	results	parasitism	degree
food	collected	webs	showed	parasitism	did
food	dominated	webs	always	parasitism	not
food	effect	webs	carpinteria	predator	generality
food	effectiveness	webs	clusion	predator	vulnerability
food	important	webs	demonstrated	full	mean
food	increase	webs	effects	species	free
food	intervality	webs	goodness	species	living
food	links	webs	greater	species	reduce

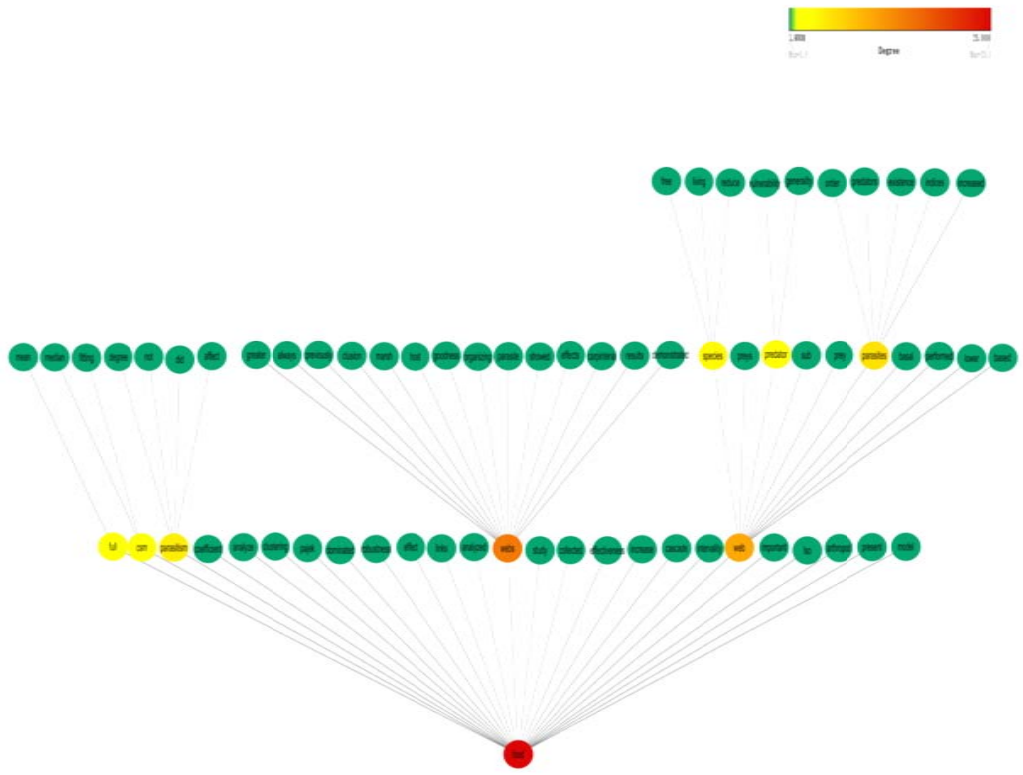


Fig. 4 The word tree constructed from word pairs with significant positive point correlations.

Word chains between two words (a chain with minimum total links) for the first 10% words in 67 words are calculated as follows:

Chain between food and web: food->web
Chain between food and webs: food->webs
Chain between food and parasites: food->web->parasites
Chain between food and csm: food->csm
Chain between food and parasitism: food->parasitism
Chain between food and predator: food->web->predator
Chain between web and webs: web->food->webs
Chain between web and parasites: web->parasites
Chain between web and csm: web->csm
Chain between web and parasitism: web->food->parasitism
Chain between web and predator: web->predator
Chain between webs and parasites: webs->analyzed->parasites
Chain between webs and csm: webs->csm
Chain between webs and parasitism: webs->parasitism
Chain between webs and predator: webs->analyzed->predator
Chain between parasites and csm: parasites->web->csm
Chain between parasites and parasitism: parasites->effects->parasitism
Chain between parasites and predator: parasites->predator
Chain between csm and parasitism: csm->food->parasitism
Chain between csm and predator: csm->web->predator
Chain between parasitism and predator: parasitism->food->web->predator

Some of the word modules are listed bellow:

Standalone link ID 1 (Link weight=0.20896) (food,study)

Links in module ID 2 (Link weight=0.13433) (food,analyze) (food,effect) (food,pajek) (study,analyze) (study,effect) (study,pajek)

Standalone link ID 3 (Link weight=0.1791) (web,full)

Links in module ID 4 (Link weight=0.074627) (web,dominated) (web,links) (full,dominated) (full,links)

Links in module ID 5 (Link weight=0.1791) (webs,csm) (webs,parasitism)

Links in module ID 6 (Link weight=0.059701) (parasites,predator) (parasites,prey) (parasites,sub) (predator,basal) (predator,based) (predator,preys) (prey,basal) (prey,based) (prey,preys) (sub,basal) (sub,based) (sub,preys)

Standalone link ID 7 (Link weight=0.10448) (parasites,based)

Standalone link ID 8 (Link weight=0.20896) (predator,prey)

Links in module ID 9 (Link weight=0.16418) (predator,sub) (prey,sub)

Standalone link ID 10 (Link weight=0.14925) (results,showed)

Links in module ID 11 (Link weight=0.089552) (results,carpinteria) (results,collected) (results,marsh) (showed,carpinteria) (showed,collected) (showed,marsh)

Links in module ID 12 (Link weight=0.074627) (results,median) (showed,median) (carpinteria,median) (collected,median) (marsh,median)

Links in module ID 13 (Link weight=0.14925) (species,affect) (species,free) (species,living) (affect,degree) (affect,free) (affect,living) (degree,free) (degree,living)

Standalone link ID 14 (Link weight=0.16418) (species,degree)

Links in module ID 15 (Link weight=0.1194) (species,did) (species,existence) (species,not) (species,predators) (affect,did) (affect,existence) (affect,not) (affect,predators) (degree,did) (degree,existence) (degree,not) (degree,predators) (did,existence) (did,free) (did,living) (did,predators) (existence,free) (existence,living) (existence,not) (free,not) (free,predators) (living,not) (living,predators) (not,predators)

Links in module ID 16 (Link weight=0.074627) (arthropod,demonstrated) (arthropod,goodness)

Links in module ID 17 (Link weight=0.10448) (cascade,fitting) (cascade,lower) (fitting,model) (fitting,lso) (fitting,performed) (model,lower) (lower,lso) (lower,performed)

Standalone link ID 18 (Link weight=0.14925) (cascade,model)

Links in module ID 19 (Link weight=0.13433) (cascade,lso) (model,lso)

Links in module ID 20 (Link weight=0.1194) (cascade,performed) (model,performed) (lso,performed)

Standalone link ID 21 (Link weight=0.14925) (fitting,lower)

Links in module ID 22 (Link weight=0.1791) (generality,mean) (generality,vulnerability) (mean,vulnerability)

Links in module ID 23 (Link weight=0.10448) (generality,clusion) (generality,greater) (generality,host) (generality,increased) (generality,parasite) (generality,previously) (mean,clusion) (mean,greater) (mean,host) (mean,increased) (mean,parasite) (mean,previously) (vulnerability,clusion) (vulnerability,greater) (vulnerability,host) (vulnerability,increased) (vulnerability,parasite) (vulnerability,previously) (clusion,increased) (greater,increased) (increased,previously)

Links in module ID 24 (Link weight=0.059701) (robustness,clustering) (robustness,coefficient)

(robustness,increase) (robustness,intervality) (clustering,reduce) (coefficient,reduce) (increase,reduce) (intervality,reduce)

Standalone link ID 25 (Link weight=0.13433) (robustness,reduce)

Links in module ID 26 (Link weight=0.074627) (always,analyzed) (always,important) (always,present) (analyzed,organizing) (important,organizing) (organizing,present)

Standalone link ID 27 (Link weight=0.089552) (always,organizing)

Links in module ID 28 (Link weight=0.16418) (analyze,effect) (analyze,pajek) (effect,pajek)

Links in module ID 29 (Link weight=0.10448) (analyzed,important) (analyzed,present)

Standalone link ID 30 (Link weight=0.089552) (basal,preys)

Standalone link ID 31 (Link weight=0.14925) (carpinteria,collected)

Links in module ID 32 (Link weight=0.13433) (carpinteria,marsh) (collected,marsh)

Links in module ID 33 (Link weight=0.14925) (clusion,greater) (clusion,previously) (greater,previously)

Links in module ID 34 (Link weight=0.1194) (clusion,host) (clusion,parasite) (greater,host) (greater,parasite) (host,previously) (parasite,previously)

Standalone link ID 35 (Link weight=0.14925) (clustering,coefficient)

Links in module ID 36 (Link weight=0.1194) (clustering,increase) (clustering,intervality) (coefficient,increase) (coefficient,intervality)

Standalone link ID 37 (Link weight=0.13433) (demonstrated,goodness)

Standalone link ID 38 (Link weight=0.13433) (did,not)

Standalone link ID 39 (Link weight=0.10448) (dominated,links)

Links in module ID 40 (Link weight=0.074627) (effects,indices) (effects,order) (indices,order)

4 Discussion

The method in present study is substantially a tool of big data analytics. The results from the tool are integrated exploration of data. With certain fault tolerance, big data analytics allows limited missing and local errors in data sets and results. An integrated rather than exact analysis is thus more of significance.

The software can be improved in these aspects: (1) reduce the time cost of running the software; (2) include more file types; (3) search (e.g., search of Google, Yahoo, Web of Science, Scopus, KEGG, Pathway

Central, Database of Interacting Proteins, PubMed, GenBank, etc., by using a set of key words; IAEES, 2021) and read internet resources more easily; (4) include or optimize more reasonable stop words and splitting words; (5) include more correlation measures (e.g., Jaccard coefficient, etc. Zhang, 2016d, 2018); (6) include more metwork analysis methods (e.g., link prediction, flow analysis, network dynamics, etc. Zhang, 2016d, 2018); (7) include more languages.

Acknowledgment

This work is supported by The National Key Research and Development Program of China (2017YFD0201204). I am thankful to the help of my postgraduate SH Xin for her drawing of network figures.

References

Chan SB, et al. 1982. Graph Theory and Its Applications. Science Press, Beijing, China

Floyd RW. 1962. Algorithm 97: shortest path. Communications of the ACM, 5(6): 345

IAEES. 2021. Databases. http://www.iaees.org/publications/journals/nb/databases.asp

Jiang LQ, Zhang WJ. 2015a. Determination of keystone species in CSM food web: A topological analysis of network structure. Network Biology, 5(1): 13-33

Jiang LQ, Zhang WJ. 2015b. Effects of parasitism on robustness of food webs. Selforganizology, 2(2): 21-34

Jiang LQ, Zhang WJ, Li X. 2015. Some topological properties of arthropod food webs in paddy fields of South China. Network Biology, 5(3): 95-112

Li JR, Zhang WJ. 2013. Identification of crucial metabolites/reactions in tumor signaling networks. Network Biology, 3(4): 121-132

Minty GJ. 1965. A simple algorithm for listing all the trees of a graph. IEEE Transactions on Circuit Theory, CT-12(1): 120

Nuwagaba S, Hui C. 2015. The architecture of antagonistic networks: Node degree distribution, compartmentalization and nestedness. Computational Ecology and Software, 5(4): 317-327

Qi YH, Liu GH, Zhang WJ. 2018. Analysis of word occurrence frequency and word association in English text file: A big data analytics method. Network Biology, 8(3): 126-136

Shams B, Khansari M. 2014. Using network properties to evaluate targeted immunization algorithms. Network Biology, 4(3): 74-94

Xin SH, Zhang WJ. 2020. Construction and analysis of the protein–protein interaction network for the olfactory system of the silkworm Bombyx mori. Archives of Insect Biochemistry and Physiology, 105(3): e21737

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. 2012a. Computational Ecology: Graphs, Networks and Agent-based Modeling. World Scientific, Singapore

Zhang WJ. 2012b. How to construct the statistic network? An association network of herbaceous plants constructed from field sampling. Network Biology, 2(2): 57-68

Zhang WJ. 2012c. Several mathematical methods for identifying crucial nodes in networks. Network Biology, 2(4): 121-126

Zhang WJ. 2014. Interspecific associations and community structure: A local survey and analysis in a grass

community. Selforganizology, 1(2): 89-129

Zhang WJ, Wang R, Zhang DL, et al. 2014. Interspecific associations of weed species around rice fields in Pearl River Delta, China: A regional survey. Selforganizology, 1(3-4): 143-205

Zhang WJ, Jiang LQ, Chen WJ. 2014. Effect of parasitism on food webs: Topological analysis and goodness test of cascade model. Network Biology, 4(4): 170-178

Zhang WJ. 2015. Calculation and statistic test of partial correlation of general correlation measures. Selforganizology, 2(4): 65-77

Zhang WJ. 2016a. A Matlab program for finding shortest paths in the network: Application in the tumor pathway. Network Pharmacology, 1(1): 42-53

Zhang WJ. 2016b. A method for identifying hierarchical sub-networks / modules and weighting network links based on their similarity in sub-network / module affiliation. Network Pharmacology, 1(2): 54-65

Zhang WJ. 2016c. Finding trees in the network: Some Matlab programs and application in tumor pathways. Network Pharmacology, 1(2): 66-73

Zhang WJ. 2016d. Selforganizology: The Science of Self-Organization. World Scientific, Singapore

Zhang WJ. 2017a. Finding the shortest tree in the network: A Matlab program and application in tumor pathway. Network Pharmacology, 2(1): 13-16

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: (IV) Classification and network analysis of medicinal functions of Chinese herbal medicines. Network Pharmacology, 2(3): 82-104

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

Zhang GL, Zhang WJ. 2019. Protein–protein interaction network analysis of insecticide resistance molecular mechanism in Drosophila melanogaster. Archives of Insect Biochemistry and Physiology, 100(1): e21523

Zhang WJ, Li X. 2015a. General correlation and partial correlation analysis in finding interactions: with Spearman rank correlation and proportion correlation as correlation measures. Network Biology, 5(4): 163-168

Zhang WJ, Li X. 2015b. Linear correlation analysis in finding interactions: Half of predicted interactions are undeterministic and one-third of candidate direct interactions are missed. Selforganizology, 2(3): 39-45

Zhang WJ, Qi YH. 2020. Matlab algorithm to generate adjacency matrix from connection pairs that nodes are represented by strings. Selforganizology, 7(3-4): 8-14

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