NTNU
Norwegian University of
Science and Technology

Ranking and clustering of search results

Analysis of Similarity graph

Ksenia Alexander Shevchuk

Master of Science in Informatics
Submission date: May 2008
Supervisor: Herindrasana Ramampiaro, IDI

Norwegian University of Science and Technology
Department of Computer and Information Science

Table of context

1 Introduction

Managing large databases is one of the main problems of information systems. Today almost everyone uses such huge database as the World Wide Web to find relevant information of any kind.

The main problem is how to find the appropriate document among the millions of others, and that is not only the problem of the World Wide Web but also the problem of such databases as large company's documentation databases and libraries.

It is not difficult to find all the documents which corresponds the request but it is often the problem to get the few most relevant documents from thousands of corresponding ones. That is the problem of ranking.

How to provide a fast ranking algorithm, which is not dependent on the structure of the database—as an alternative to Hyper-link-based ranking algorithms like PageRank, which are dependent on the hyper-link structure of the database—is the topic of this project.

1.1 Overview

The most common search today is web search. Searching using the web search engine like Google is writing some query in the search field, and retrieving the set of documents which corresponds to this query. This is text based search, where the search engine retrieves all the documents which include or not (depends on the type of a query) the query words.

There are three main problems in the process of retrieving: how to preprocess documents of the whole database to provide quick retrieval (typically, by building an inverted index over terms); how to deal with the user query to provide all of the documents that could be interesting for the user; and how to rank the resulting retrieved document set to show the user the most relevant results in the first page of the list of results.

Most often full text search, which evaluates each document as a string of different terms, is preferred. To speed up the search process some common terms built over the text can be used (such as names of different chapters), or some structure can be added to the text (like a trie) Figure 1 shows the example of a trie [1].

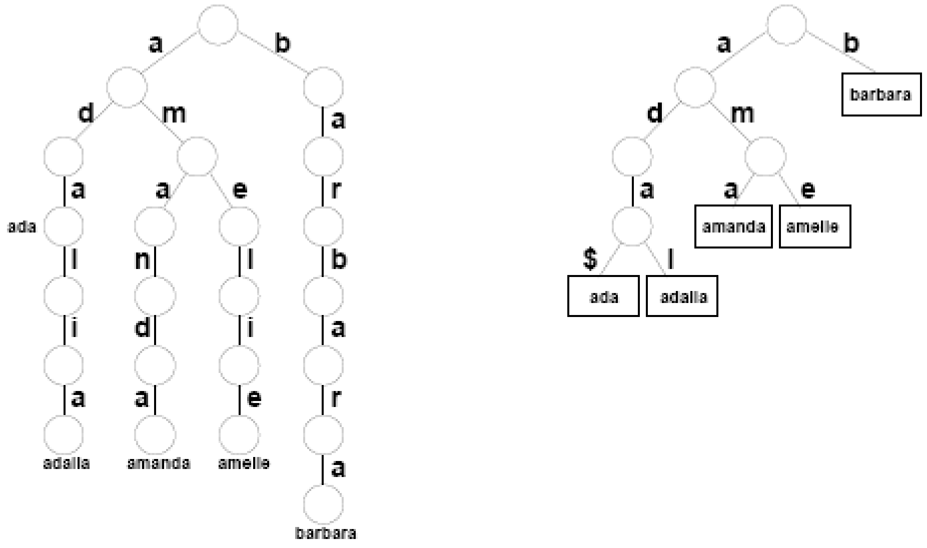


Figure 1. A trie over strings ”ada”, ”adalia”, “amanda”, “amelie” and “barbara”. On the left, basic formulation, on the write, the version used in practice

Dealing with the user query is the problem of which type of the retrieval models to chose. There are three main models: Boolean, Vector and Probabilistic. While the Boolean model retrieves only document that includes all words of the query and does not take in account how often the specific term occurs in the document, the Vector model provides some ranking in respect to occurrence of terms in the document (not required that all of the terms of the query must be found in the document) [for more details see p. 16]

The problem of ranking (the main problem of this project) is resolved in part in the Word Wide Web by evaluating link structure of the document set. The documents of the web include hyper-links to each other and the graph of such links can be built, where nodes of the graph are documents and edges are the hyper-links. The main idea is that the more important documents that have hyperlink/hyperlinks to the particular document, the more important document is. This algorithm provides not only the most relevant documents to the query but also the most important, good quality documents as the first in the results list [for more details p. 19].

1.2 Motivation

All of the Web search engines rank the result list of retrieved documents, but why is it so important to do that? For example, the Word Wide Web is a very huge database of different types of documents (html, pdf, doc, etc.) and different types of information (ex., if one searches for just a car, he can receive both information about different brands of cars and about George Bush’s car), and the number of retrieved documents could be thousands or millions. Thus there is a need to specify as good as possible the subject of interest while searching, and search systems (like Google) should have opportunity to automatically pick out the most relevant search results (rank the result).

Most users of different search engines do not look beyond the first page of results. Therefore, it is important for the ranking function of the search engine to output the desired results within the top few pages. However, users are most interested in pages that are not only relevant, but also authoritative, that is, trusted sources.

In Web search (hyperlink ranking), the focus shifts from relevance to authoritativeness. The task of the ranking function here is to identify and rank highly the authoritative documents within the collection of Web pages.

However, there are many domains for which hyperlinks are only sparsely applied, or even absent. Furthermore the results of hyperlink ranking are not always good enough (there could be some web pages that are created only for hyper linking).

Using the similarity ranking method, the problem of having an artificially created relevance of the document could also arise. The different versions of the same document can make that document relevant in respect to similarity.

But the similarity ranking method has some advantages. For example, not only text databases are needed to be searchable. It should be possible to search in pictures, music and films collections. It should be possible to find a picture which has something specific in it (an elephant), and if one has managed to compute the values of similarity between two objects (ex. pictures) then it is possible to use the method of similarity ranking. But it could be very difficult to identify objects in the picture, or some actor, playing in a film; this problem is not discussed in this work.

1.3 Problem specification

The method of ranking that uses similarity graph analysis does not need documents in the database to be linked with hyperlinks. This method does not deal with authority; instead, the method sorts documents with respect to their relevance by using the similarity graph, which is built up by calculating similarity values for all pairs of documents in the database. The document is more relevant if it is strongly similar to the large number of the documents of the retrieved part (is the most central node of this part of the documents). This method can be used in all domains—even those (ex., images) for which all standard forms of analysis (text and hyperlink analysis) are impossible.

In this work the similarity matrix will be studied. The problem here is to rank the results of a search in the easiest and quickest way, using the similarity matrix. We assume that the eigenvector of the similarity matrix of the retrieved documents gives the rank value to every of those (the higher the value for the document in the eigenvector (EVC value) is, the more relevant the document is). But it is often difficult to calculate the eigenvector (too much time and resources is needed). Here we'll try to confirm that for the high clustered (high linked) similarity matrix the eigenvector rank the documents in the same manner as Link Popularity vector (Link Popularity of the node is the sum of the weights of all the outgoing links o this node) does. So there are the following problems to be solved in this work:

1. Evaluate the clustering of the similarity matrix and confirm that it is high.
2. Compare the ranking results of the eigenvector ranking and the Link Popularity ranking and confirm for the high clustered graph the correlation between those is larger than for the low clustered graph.

1.4 Layout of the thesis

In chapter 2, the main theoretical background will be outlined. Theory behind our problem is discussed in 2.1, the basic concepts of the Information Retrieval Model (Theory of organization of information retrieval systems) are presented in 2.2 and four different ranking hyperlink methods will be discussed in 2.3.

Chapter 3 covers and compares four different definitions of the clustering coefficient for weighted graphs found in the literature.

In the chapter 4 the main idea of how to rank the searching results in the easiest way (4.1), how to evaluate the clustering of the similarity graph (invention of the new clustering coefficient for the weighted graph) (4.2) and how to test the new clustering coefficient (4.3) are discussed.

In the last chapter (chapter 5) two tests are described. The test of the new clustering coefficient (by compare the coefficient of similarity graph, random graph and non weighted graph) (5.1) and the test of the ranking method (5.2).

Discussion and evaluation of the result of the work can be find in chapter 6. The thesis ends with some recommendations for the future work and conclusion.

2 Theoretical background

In this section some theoretical explanation of the problems of this thesis will be introduced and some basic theory of information retrieval will be discussed: models of retrieving the document set from the database based on the user query, and the currently most popular hyperlink analysis algorithms of ranking.

2.1 Theoretical explanation of the problem

As mentioned before the method of ranking, discussed in this thesis uses the similarity graph built for the whole database, but how does the information retrieval system based on the similarity graph work? The process is as follows. The whole database is first preprocessed by building the similarity graph (matrix). When the user writes the query to the database, documents that could be relevant to the query are retrieved (using some information retrieval model discussed below). After that the result document set is ranked with respect to the subgraph of similarity for this set. The most relevant document for the topic will be the one that has the largest centrality value of all documents of the retrieved subset. This is the advantage of the similarity-based ranking algorithm—that we don't need to analyze all the nodes from the whole set, thus we save time and resources (the hyperlink analysis needs to process the whole database to find out the most central results [see p. 19]).

2.1.1 Similarity matrix

How is the similarity matrix built? The similarity graph ("S-graph"— a weighted, symmetric graph), which shows how similar documents in the domain are (all with all) is the graph whose nodes are documents of the domain, and whose links show similarity between the nodes (link weights represent degree of similarity). Those weights (similarity values) vary from 0 to 1: $s_{ij} \in [0,1]; i = 1,2...N; j = 1,2...N$, where N is the number of nodes in the graph. Then the similarity matrix is the *N:N* squared symmetric matrix whose elements are the similarity values (ex. the element on the *i*-th row and *j*-th column is the similarity value between nodes *i* and *j*)

The process of building the similarity matrix is shown in Example 1.

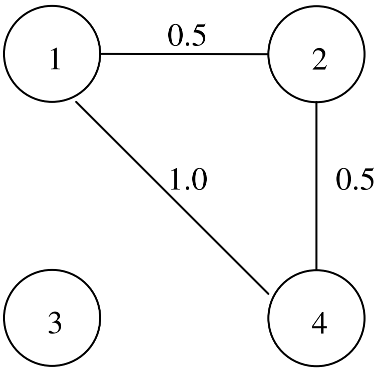
Example 1

We have four text documents:

1. The main problem of this project is the problem of ranking.
2. Our problem, discussed in the paper, is the problem of ranking.
3. My current work includes some discussion about retrieved documents relevance.

4. The main problem of this project is the problem of ranking.

Let’s assume that the similarity value for documents 1 and 2 is “0.5”, for 1 and 3- “0”, for 1 and 4- “1”, for 2 and 3- “0”, for 2 and 4- “0.5” and for 3 and 4-“0”. Now we can build the similarity graph and the similarity symmetric matrix (Figure 2).



a)

$$\begin{pmatrix} 1 & 0.5 & 0 & 1 \\ 0.5 & 1 & 0 & 0.5 \\ 0 & 0 & 1 & 0 \\ 1 & 0.5 & 0 & 1 \end{pmatrix}$$

b)

Figure 2. a)- similarity graph for Example 1, b)- similarity matrix

How the documents are retrieved according to the user query as well as the way of calculating the similarity between every pair of documents is not the subject of this work, but in Section 2.2 the basic theory of these problems is briefly presented.

2.1.2 Eigenvector centrality

After the similarity matrix is built, the search in the database can be performed. There is no use of finding the most central node (the node which is the most similar to every other node) of the whole similarity graph. That document (corresponding to the most central node) will be just the most similar to all other documents, thus the most common. It is more useful to calculate centrality for a topic-focused subset of important nodes. For example, if we find out the subset of documents which has the word ‘flowers’ inside, than the most relevant document for the topic ’flowers’ will likely be the one that has the largest centrality value of all documents of this subset. The relevance of each document (centrality) is the corresponding value in the eigenvector of the similarity matrix (EVC- eigenvector centrality). The following explanation shows the point here.

We start with the similarity matrix of all documents retrieved in respect to some query and we need to find the value of relevance for each of those documents. First we set the same positive value $p_i, i \in [1; N]$ for each of the documents, where N is the number of retrieved documents (nodes in the similarity graph). Next we calculate new values using the equation $p_i^{new} = \sum_{j \neq i} p_j s_{ij}$, where s_{ij} is the similarity value between nodes *i* and *j*. This means that the new value of relevance of the node *i* (p_i^{new}) is calculated as the sum of relevancies of nodes linked to it, normalized by similarity links weights (s_{ij}). Then we normalize the vector of relevancies ($P = (p_1 \quad ... \quad p_N)$) and calculate again new values of relevancies, using the same equation. We must repeat this again and again until this vector will converge, which means that $p_i^{new} / c = p_i$ or $P^{new} / c = P$, where c is the constant of normalization. So we have an eigenvector problem here: $P * \lambda = S * P$, where λ is the largest eigenvalue (because we took only positive relevancies in the start and those only grew with iterations). Thus *P* is the eigenvector of the similarity matrix *S*.

The problem is that the number of the nodes in the retrieved by querying subset can still be very large (more than 1000 nodes), and the eigenvector calculation for such a graph needs much time and resources. In this project we'll try to replace eigenvector calculation with the calculation of link popularities of the nodes.

2.1.3 The main idea of the project. Link popularity

The alternative method to rank the results of search is to calculate just the Link Popularity for each of the result documents. If the eigenvector calculation requests a lot of iterations then the Link popularity is just the one iteration, setting the first relevance values like "1": $p_i = \sum_{j \neq i} 1 * s_{ij}$ .The Link popularity of the node *i* is the sum of all weights of the links of this node: $L_i = \sum_{j=1}^{k_i} w_{i,j}$, where L_i is Link popularity of the node *i*, $w_{i,j} = s_{ij}$ is the weight of the link between nodes *i* and *j,* and k_i is the number of the nodes linked to the node *i*. We can also define $LP = \{L_1, L_2 ... L_i ... L_N\}$ as the vector of link popularities for every node.

The problem which is faced here is to show that ranking with Link Popularity gives roughly the same result as ranking with the eigenvector for the similarity matrix. Here we suppose that if the graph is well linked (which is true for every similarity graph) then the eigenvector will rank the documents in the same manner as the link popularity [see p. 31 for explanation].

But we need to evaluate how well linked a graph is: we need some numeric coefficient which is high for well linked graphs and low for poorly linked graphs. Such a coefficient is called the clustering coefficient.

2.1.4 Clustering coefficient for unweighted graph

For graphs that are unweighted (links weights can be only “1” or “0”) two alternative clustering coefficients have been defined. Here good clustering implies that if one node in the graph is linked to two others then there is a high probability that those two are linked with one another [2]. The clustering coefficient for an unweighted graph is large if there are many triangles in it, which is the case if there are often three linked nodes. In the other words, clustering coefficient is the fraction of the connected three nodes groups that are connected each to each (in triangles), or probability of two nodes to be neighbors if they have the same neighbor.

It can be calculated like the following: $cc = \frac{3 * N_t}{N_{con}}, cc \in [0,1]$, where N_t is the number of triangles in the graph and N_{con} - the number of connected triples of nodes; or alternatively $cc = \frac{6 * N_t}{N_{path}}, cc \in [0,1]$, where N_{path} is the number of paths that have length two.

A triangle and a connected triple are shown on the Figure 3. For the graph in Figure 3 a) the clustering coefficient will be $cc = \frac{3*1}{3} = 1$, while for Figure 3 b) cc = 0.

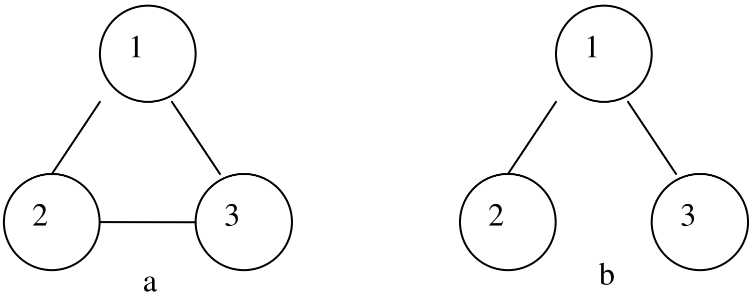


Figure 3. Triangle (a) and connected triple (b)

Figure 4 shows the connection between triples and paths. One connected triple in the graph corresponds to two directed paths with length 2.

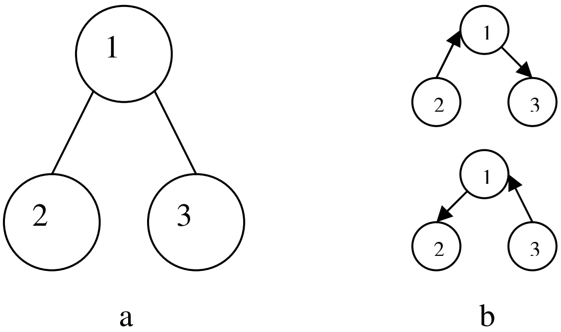


Figure 4. Connected triple and directed path length 2 (a- connected triple, b- corresponding paths)

There is another definition of the clustering coefficient such as $cc_i = \frac{n_{t,i}}{n_{con,i}}, cc_i \in [0,1]$, where cc_i is the clustering coefficient of the node *i*, $n_{t,i}$ - number of triangles linked to the node *i*, and $n_{con,i}$ - the number of connected triples linked to the node i, triples that have the node i in the middle (j-i-k). The clustering coefficient of the whole graph than is calculated as follows. $cc = \frac{1}{N}\sum_N cc_i$, where N is the total number of nodes in the graph.

Now we have two definitions:

Definition 1:

$$cc_1 = \frac{3 * N_t}{N_{con}}, cc_1 \in [0,1] \text{ or } cc_1 = \frac{6 * N_t}{N_{path}}, cc_1 \in [0,1], \text{ where}$$

N_t - the number of triangles in the graph

N_{con} - the number of connected triples

N_{peth} - the number of directed paths that have the length two

Definition 2:

$$cc_2 = \frac{1}{N}\sum_N cc_i \text{ , } cc_i = \frac{n_{t,i}}{n_{con,i}}, cc_i \in [0,1]$$

$n_{t,i}$ - number of triangles which include the node i

$n_{con,i}$ - number of connected triples which include the node i

N- number of nodes in the graph

Example 2 shows that these two definitions gives the different values of clustering.

Example 2

Two different clustering coefficients calculated for the graph shown in Figure 5.

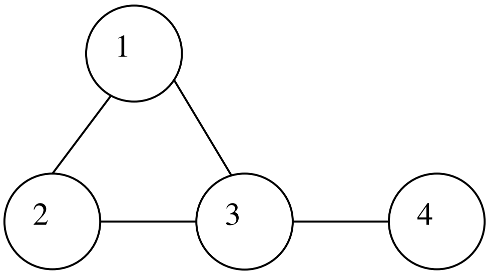


Figure 5. Example graph.

1. $cc_1 = \frac{3 * 1}{5} = 3/5$

2. $cc_2 = \frac{1+1+1/3+0}{4} = 7/12$

These two coefficients are only appropriate for unweighted graphs but in our situation we have similarity graphs that are weighted with weights which vary from 0 to 1, and that's why we need here some other measure of clustering which has the same properties as those coefficients for unweighted graphs. The challenge for me here is to invent such a clustering coefficient based on the above definitions. But which of those two definitions is more appropriate to base on: the one which calculates the coefficient for the whole graph or that one which first calculates the clustering coefficient for every node? The problem of this choice is the subject of the next section.

2.1.5 Comparison of the two coefficients

If we make a plot (axis- cc_1 (CMN) and ordinate- cc_2(CDW)) [see Figure 6] from the results of Table II of Mark Newman's article [2], we can see that these two clustering coefficients are like each other for the low values, while cc_2 is much larger for high values; but it is also obvious that both are not larger than 1 and are like each other in this point.

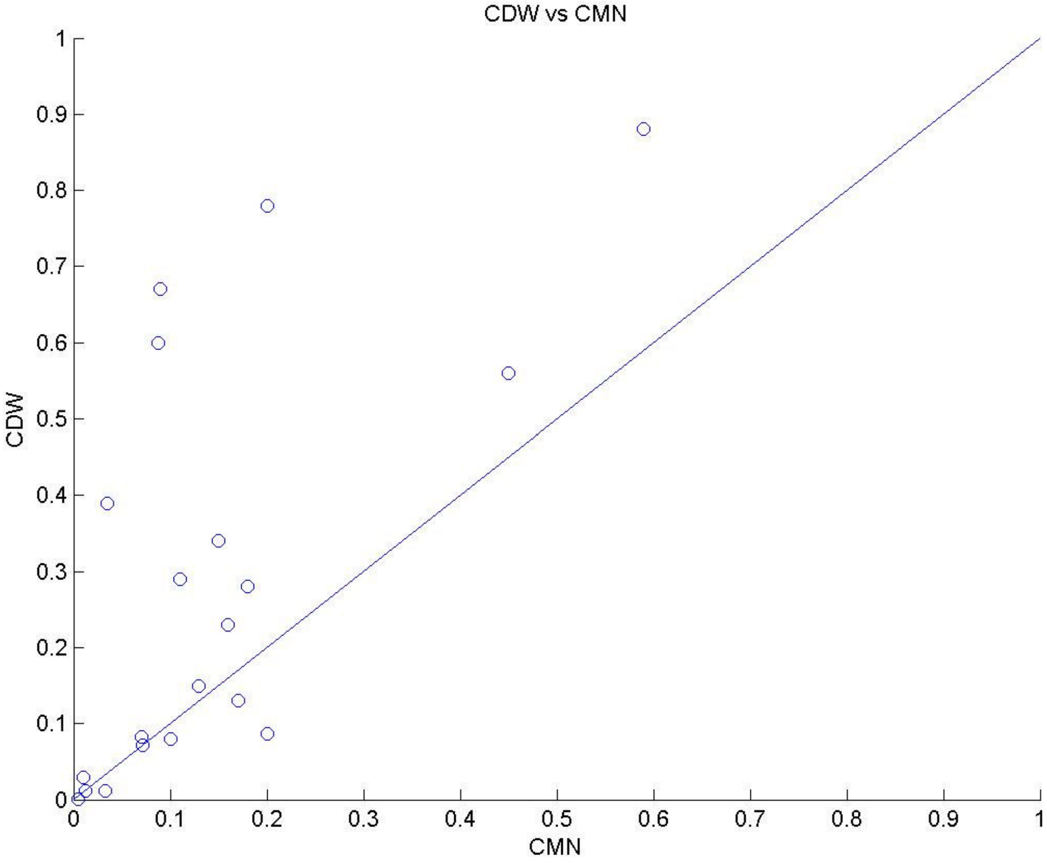


Figure 6. Correlation between cc_1 (CMN) and cc_2(CDW)

We can assume from these data that for most graphs the scatter from the Figure 6 will take the form of a curve shown in the Figure 7.

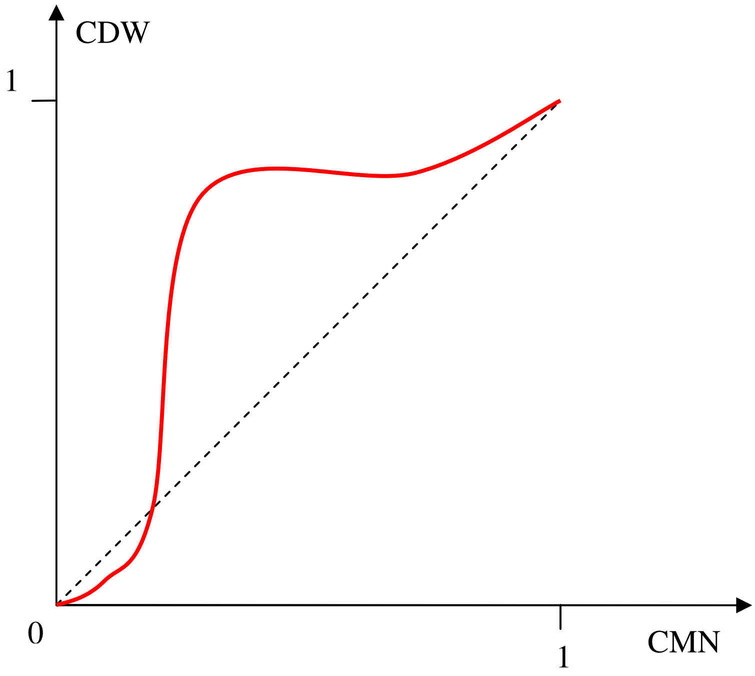


Figure 7. Proposed correlation curve. Axis- cc_1 (CMN) and ordinate- cc_2 (CDW))

Let’s take a look at the two extreme cases of graphs shown in Example 3 and Example 4.

Example 3 (flower)

Here is the graph with the one central node (which has a large number of neighbors (6)) and a large number (6) of nodes linked to this central node (which have a few (2) neighbors).

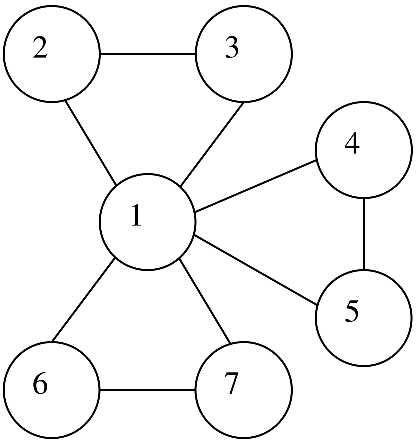


Figure 8. The graph of the Example 3 (flower)

This graph (Figure 8) has the first clustering coefficient very low, but the second very high.

1. $cc_1 = \dfrac{3*3}{6+15} = 3/7$

2. $cc_2 = \frac{6+3/15}{7} = 31/35$

Density of this graph is low: $t = \frac{9}{\binom{7}{2}} = 9/28$ (the number of links divided to the number of all possible links).

Now we'll generalize the Example 3. Let's assume that the number of triangles linked to the middle node is *k*, then the number of nodes is *N*=2*k*+1. The number of connected triples linked to the middle node is $\binom{N-1}{2} = \frac{2k(2k-1)}{2} = 2k^2 - k$ and the number of possible links in the graph is $\binom{N}{2} = \frac{(2k+1)2k}{2} = 2k^2 + k$. So we can write:

1. $cc_1 = \frac{3*N_t}{N_{con}} = \frac{3*k}{2k+2k^2-k} = \frac{3}{2k+1}$

2. $cc_2 = \frac{1}{N}\sum_N cc_i = \frac{2k + k/(2k^2-k)}{2k+1} = \frac{4k^2-2k+1}{4k^2-1}$

3. $t = \frac{N_l}{N_{\max}} = \frac{3k}{2k^2+k} = \frac{3}{2k+1}$, where N_l is the number of links and $N_{\max}$ is the number of all possible links.

The larger *k* is, the higher the second clustering coefficient is, and the lower the first one is.

Example 4 (chain)

The graph here (Figure 9) has just one triangle and a long chain linked to that triangle.

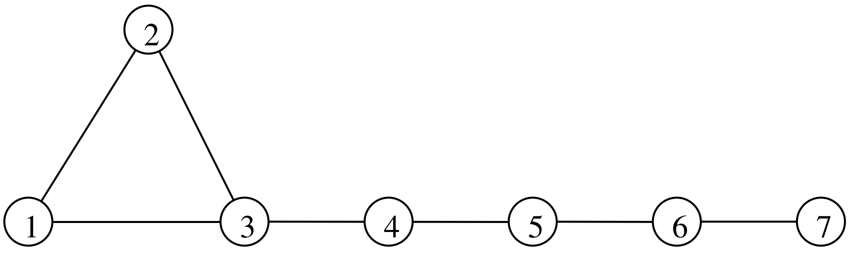


Figure 9. Graph from the Example 4 (chain)

This graph has the both clustering coefficients low and the second is lower than the first one. Density is also low.

1. $cc_1 = \frac{3*1}{3+2+3} = 3/8$

2. $cc_2 = \frac{2+1/3}{7} = 1/3$

3. $t = \frac{7}{\binom{7}{2}} = 1/4$

Now we'll generalize the Example 4 as we did in the previous one. Let's assume that the number of nodes in the chain (node 4, node 5…) is *k*, then the number of nodes is *N*=3+*k*. The number of possible links in the graph is $\binom{N}{2} = \frac{(k+3)(k+4)}{2} = \frac{k^2+7k+12}{2}$. So we can write:

1. $cc_1 = \frac{3*N_t}{N_{con}} = \frac{3*1}{1+1+3+k-1} = \frac{3}{k+4}$
2. $cc_2 = \frac{1}{N}\sum_N cc_i = \frac{1+1+1/3}{k+3} = \frac{7/3}{k+3}$
3. $t = \frac{N_l}{N_{\max}} = \frac{6+2k}{k^2+7k+12}$

The larger *k* is, the lower both of the clustering coefficients are, but the second clustering coefficient is lower than the first one (for all $k > 0$).

In these two examples from above we have two different cases: in the first case (Example 3) the first clustering coefficient is lower than the second and in the other case (Example 4) the first clustering coefficient is higher than the second one. The Figure 10 shows these cases. This supports our assumption: that for large values of clustering, the first clustering coefficient will be lower than the second one, and that the opposite will hold for the small values.

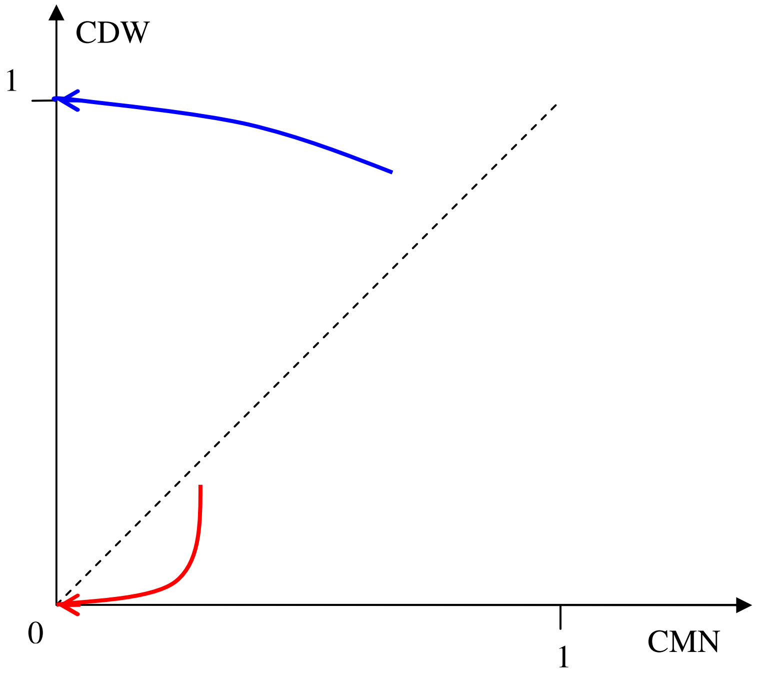


Figure 10. Correlation between clustering coefficients with the growth of nodes number. Blue- Example 3 and red- Example 4

Example 3 is much like the case when a few nodes have a large number of neighbors (high link popularity) and a large number of nodes have a few neighbors (low link popularity). The Figure 11 shows how the number of nodes with a given link popularity depends on the link popularity. Most of point in the Figure 6 which lie above the middle line are examples of such a graph. Such (power-law) graphs can not be seen as a similarity graphs as well as the graph from Example 3.

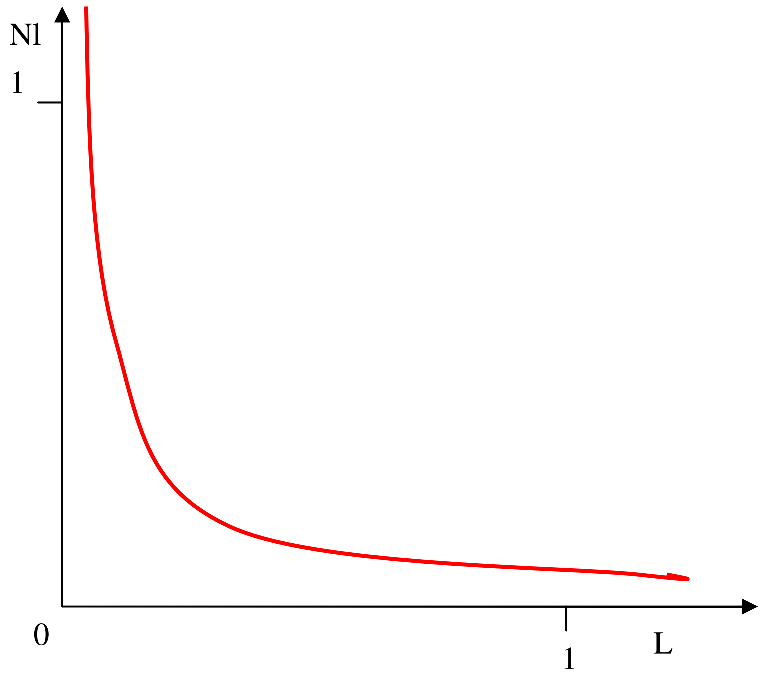


Figure 11. Link popularity (axis) and the number of node (Nl) with such the Link popularity L

Both of the examples above are not typical similarity graphs (graphs with the large clustering coefficient), because in the similarity graph the neighbors of the node (which are strongly linked to the node) are supposed to be linked with each other (if the document is very like two other documents then those are also like each other). But the second clustering coefficient of the Example 3 tells that this graph has high clustering coefficient and that's why it could be evaluated like a good similarity graph. So we can not evaluate such a graph using the second clustering coefficient.

In our case we are most interested in how the whole graph is linked and not how each node is linked. That's why we need to take the triangle as the unit of analysis, rather than the node. We need to evaluate the percentage of triangles in the whole graph and not the percentage of triangles linked to the node. Thus in this project we have chosen the first clustering coefficient of the previous section as a basis for our new definition of clustering coefficient of the weighted graph.

2.2 Basic Retrieval

Now let's discuss some basic of the information retrieval to have some overview around the main theme of this thesis.

The Information retrieval system consists of three main parts: IR (Information Retrieval) model, the logical view of the documents and the user task [3]. IR model is the set of rules of how documents should be chosen from the database based on the user's query. Logical view of the document is the way of how the document is represented in the system to provide retrieval. It can be represented as a full text, with index terms (words that represents semantics of the document) or with the full text and structure (chapters, paragraphs, etc.) In the case of index terms, representation weights (numerical values) are usually used to identify the relevance of each index term for information retrieval. Weight is associated with the pair (document, index term) and qualifies the importance of the index term for describing the document semantic contents. Users query is a set of index terms (words that are needed to be found in the document). Problem here is that a lot of semantics in a document or user request is lost by replacing the text with index terms.

There are three main IR models: Boolean, Vector and Probabilistic. This models uses index terms documents representation for retrieval.

The most simple is the Boolean model. Weights for Index terms can be only 1 or 0. $w_{i,j} \in \{1,0\}$.

Queries are Boolean expressions of keywords.

For example, $q = k_b \vee (k_a \wedge \neg k_c) = (1,0,0) \vee (1,1,0) \vee (1,1,1) \vee (0,1,0) \vee (0,11)$, where *q*- query and k_a, k_b, k_c - keywords of the query. Figure 12 illustrates the sets of documents that contains keywords and query result.

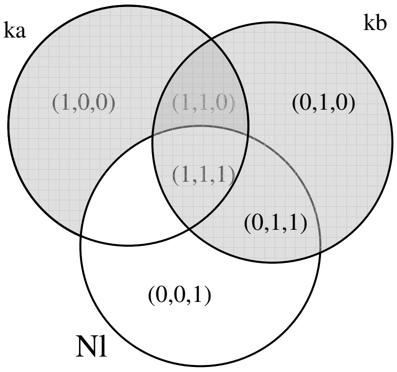


Figure 12. Three sets of the documents containing k_a, k_b and k_c Index terms. With the blue color the query $q = k_b \vee (k_a \wedge \neg k_c) = (1,0,0) \vee (1,1,0) \vee (1,1,1) \vee (0,1,0) \vee (0,11)$ is shown.

Documents are retrieved by counting the similarity between documents and user's query. If *sim*=1 between the document and the query then the document is relevant, if *sim*=0 then it is not relevant. For each of the documents of the collection Index term vector is assigned. It has as many dimensions as Index terms in the whole collection. Documents vector has 1 in the dimensions of existing Index terms in it and 0 in all other dimensions. Similarly the query vector is built up.

sim=1 if the document have 1 in all those dimensions where the query vector has.

sim=0 in all other cases.

In Vector model weights for each of index terms for each of the documents are assigned. These weights are >= 0 and non-binary. Similarity is also non-binary in comparison with the Boolean model and varies from 0 to 1. As in the Boolean model, vectors are assigned to the documents and to the query. Similarity is calculated as the cosine of the angle between two vectors: d_j and $\vec{q}$. $sim(d_j, q) = \frac{\vec{d}_j \bullet \vec{q}}{|\vec{d}_j| \times |\vec{q}|} = \frac{\sum_{i=1}^{t} w_{i,j} \times w_{i,q}}{\sqrt{\sum_{i=1}^{t} w_{i,j}^2} \times \sqrt{\sum_{j=1}^{t} w_{i,q}^2}}$, where $|\vec{d}_j|, |\vec{q}|$ - norms, $w_{i,j}$ - weight of k_i index term in d_j document and $w_{i,q}$ is the weight of k_i index term in q. In Figure 13 similarity is the cosine of *θ*.

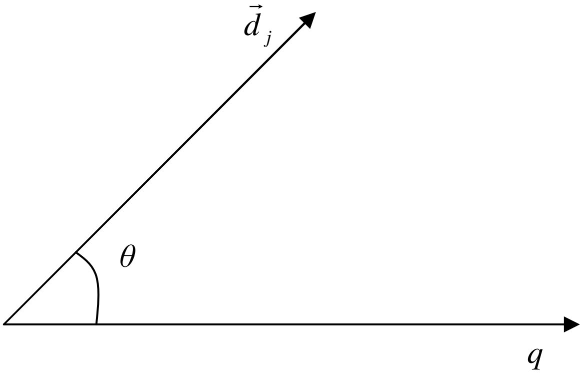


Figure 13. ***sim*($\vec{d}_j$,*q*)=*cos*(*θ*)**

Thus we can rank all retrieved documents and can find those that are most similar to the query.

In the Probabilistic model similarity between the query and the vector is calculated with the help of probabilities of relevance and non-relevance for each of retrieved documents. Weights for index terms here are binary. $w_{i,j} \in \{1,0\}$ $w_{i,q} \in \{1,0\}$.

If *q* is subset of index terms, *R*- set of relevant documents and $\bar{R}$ - set of non-relevant documents, then similarity is calculated as follows. $sim(d_j, q) = \frac{P(R|d_j)}{P(\bar{R}|d_j)}$, where $P(R|d_j)$ - the probability for the document d_j to be relevant to the query *q* and $P(\bar{R}|d_j)$ is the probability for d_j to be non-relevant. This can be transformed as follows. $sim(d_j,q)=\frac{P(d_j|R)\times P(R)}{P(d_j|\bar{R})\times P(\bar{R})}$, where $P(d_j|R)$ - the probability of randomly selecting the document d_j from the set *R* of relevant documents and *P*(*R*) is the probability that a document randomly selected from the collection is relevant. $P(d_j|\bar{R})$ and $P(\bar{R})$-analogous and complementary. This expression can be written for the index terms as follows. $sim(d_j,q)=\frac{(\prod_{i=1}^{n} P(k_i|R))\times(\prod_{i=1}^{n} P(\bar{k}_i|R))\times P(R)}{(\prod_{i=1}^{n} P(k_i|\bar{R}))\times(\prod_{i=1}^{n} P(\bar{k}_i|\bar{R}))\times P(\bar{R})}$, where $P(k_i|R)$ - the probability that the index term k_i exists in a document randomly selected from the set *R*, $P(\bar{k}_i|R)$ - the probability that the index term k_i is not found in the document d_j and analogous for $P(k_i|\bar{R})$ and $P(\bar{k}_i|\bar{R})$. *n* is the number of all index terms in the collection. Out from this the key expression for ranking computation in the probabilistic model is $sim(d_j,q)\sim\sum_{i=1}^{n} w_{i,q}\times w_{i,j}\times\left(\log\frac{P(k_i|R)}{1-P(k_i|R)}+\log\frac{1-P(k_i|\bar{R})}{P(k_i|\bar{R})}\right)$. In this model documents can be retrieved recursively. In the first iteration the following assumption is made: $P(k_i|R)$=0.5 and $P(k_i|\bar{R})=\frac{n}{N}$, where *N* is the total number of the documents in the collection. In the next iteration out from the set of retrieved documents in the first iteration (*V*) and sub sets of *V* (V_i) consisting special index terms (k_i) probability can be calculated as $P(k_i|R)=\frac{V_i+0,5}{V+1}$ and $P(k_i|\bar{R})=\frac{n-V_i|0,5}{N-V+1}$.

If you compare these models, the most cleanly formalized and simple is the Boolean model, but it could give bad recall or bad precision, because of it requires exact matching of query terms to document terms. It doesn't retrieve the documents which are partly appropriate to the query. Thus this model is the weakest classic method.

The advantage of the probabilistic model is that documents are ranked in decreasing order of their probability of being relevant, hence it is better than Boolean model because it provides partial matches (with probability <1). Disadvantage is that this method does not take in account the frequency of occurrence of the term in the document (binary weights). The solution to this problem is to assign non binary weights to index terms which refer in addition to other terms properties to their frequency of occurrence. This provides a vector model. Thus it also retrieves "partly relevant" documents. It seems that it outperforms both the Boolean and the probabilistic model. Of course here the result depends on the weight calculating technique.

All the above methods are defined to find all the documents that could be relevant to the query. But the number of such documents can be enormous, that's why we need to rank them afterwards.

2.3 Hyperlink analysis algorithms of ranking

The most well known methods of ranking are the hyperlink ranking algorithms. The explanation about how does those algorithms work is given below.

First the common variables and constants for all of the hyperlink algorithms are defined [4].

In the beginning the hyperlink graph is constructed based on the set of Web pages. A node is created for every Web page, and a directed edge is placed between two nodes if there is a hyperlink between the corresponding Web pages (if there are multiple links between two pages, only a single edge is placed). No self-loops are allowed. The problem is to assign the weight to each of the nodes to represent its authority. These weights are used to rank the pages.

Let P be the set of nodes, and let N be the size of the set P. Let $G = (P, E)$ denote the hyperlink graph. The input to the link analysis algorithm is the matrix W of the graph G, where $W[i, j\,] = 1$ if there is a link from node i to node j , and zero otherwise. The output of the algorithm is an N-dimensional (N-the number of the documents) vector a, where a_i, the i-th coordinate of the vector a, is the authority weight of node i in the graph.

Here are some other terms that are used in hyperlink analysis. For a node i $B(i)=\{j\!:\!W[j,i]=1\}$ is the set of nodes that point to node i (Backwards links), and $F(i)=\{j\!:\!W[i,j]=1\}$ is the set of nodes that are pointed to by node i (Forward links). Set A is used to denote the set of notes with non-zero authority score, and H to denote the set of notes with non-zero hub score. The undirected authority graph $G_a = (A, E_a)$ is the graph on the A, where we place an edge between two nodes i and j, if $B(i) \cap B(j) \neq \varnothing$. This graph is defined by the matrix $W^T W$ (number of elements in $B(i) \cap B(j)$).

2.3.1 InDegree Algorithm

In this algorithm the popularity (authority) of a page is measured by the number of pages that link to this page (link popularity). It ranks pages according to their in-degree in the graph G. That is, for every node i, $a_i = |B(i)|$.

2.3.2 PageRank Algorithm [5]

The main idea of this algorithm is that a good authority is one that is pointed to by many good authorities. Not all links carry the same weight.

Brin and Page, the inventors of PageRank, began with a simple summation equation, The PageRank authority weight *i*-th node, a_i, is the sum of the authority weights of all nodes pointing into *i*.

$a_i = \sum_{j \in B(i)} \frac{a_j}{|F(j)|}$, where $|F(j)|$ is the number of forward links from node *j*. Notice that the PageRank weight a_j is tempered by the number of recommendations made by *j* ($|F(j)|$). The problem is that the a_j values, the weights of pages pointing to page *i*, are unknown. To resolve this problem, Brin and Page used an iterative procedure.

In the beginning, all pages have equal weights (1/*N*). Next step is to compute a_i using the above equation. This is followed by calculating a_j, etc.

Let $a_i^{(k+1)}$ be the weight of node *i* at iteration $k+1$. Then, $a_i^{(k+1)} = \sum_{j \in B(i)} \frac{a_j^k}{|F(j)|}$ and

$a_i^{(0)} = 1/N, i \in P$

This process is repeated until the PageRank weights will eventually converge to some final stable values. Brin and Page use the notion of a random surfer. Imagine a web surfer who randomly follows the hyperlink structure of the Web. If surfer spends a large proportion of his time on a particular page, then this page is important.

Example 5

Figure 14 illustrates the directed graph of six Web pages and Table 1 shows the results of iterations.

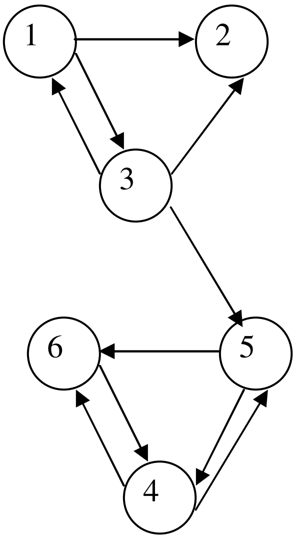


Figure 14. Web page graph of six pages

Iteration0	Iteration1	Iteration2	Rank at Itr.2
$a_1^{(0)} = 1/6$	$a_1^{(1)} = 1/18$	$a_1^{(2)} = 1/36$	5
$a_2^{(0)} = 1/6$	$a_2^{(1)} = 5/36$	$a_2^{(2)} = 1/18$	4
$a_3^{(0)} = 1/6$	$a_3^{(1)} = 1/12$	$a_3^{(2)} = 1/36$	5
$a_4^{(0)} = 1/6$	$a_4^{(1)} = 1/4$	$a_4^{(2)} = 17/72$	1
$a_5^{(0)} = 1/6$	$a_5^{(1)} = 5/36$	$a_5^{(2)} = 11/72$	3
$a_6^{(0)} = 1/6$	$a_6^{(1)} = 1/6$	$a_6^{(2)} = 14/72$	2

Table 1. Three iterations of weighting and rankling results

Let *M* be a normalized form of the matrix *W*, where $M_{ij}^T = \frac{W_{ij}^T}{|F(i)|}$ and $\vec{a}^{(k)}$ is the weights vector (with components $a_i^{(k)}, i \in 1..N$) at the *k*-th iteration. Now $a_i = \sum_{j \in B(i)} \frac{a_j}{|F(j)|}$ equation can be written as $\vec{a}^{(k+1)} = M^T \vec{a}^{(k)}$.

For the graph from Figure 14:

$$M = \begin{pmatrix} 0 & 1/2 & 1/2 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 1/3 & 1/3 & 0 & 0 & 1/3 & 0 \\ 0 & 0 & 0 & 0 & 1/2 & 1/2 \\ 0 & 0 & 0 & 1/2 & 0 & 1/2 \\ 0 & 0 & 0 & 1 & 0 & 0 \end{pmatrix}$$

Here we can mention that the problem of finding the authority weights vector $\vec{a} = (a_1 \quad a_2 \quad ... \quad a_N)$ is the problem of finding the eigenvector of the matrix M^T, which corresponds to the largest eigenvalue ($\lambda_{\max} = 1$): $\lambda_{\max}\vec{a} = M^T\vec{a}$

Each of the iterations involves one vector-matrix multiplication, which generally requires $O(N^2)$ computations. *M* is a very sparse matrix (a large proportion of its elements are 0) because most web pages links to only a few of other pages. Sparse matrices are good for several reasons. First, they require minimal storage, since sparse storage schemes, which store only the nonzero elements of the matrix and their locations exist. Second, vector-matrix multiplication involving a sparse matrix requires much less than the $O(N^2)$ computations. It requires $O(N)$ (if the total number of nonzero links is *O*(*N*)) The iterative process is a simple linear stationary process.

There could be some problems with such method of weights calculation. For example, there is the problem of rank sinks (dangling nodes). Sink is the situation when it is not possible to come out of some part of the graph following the links. The random surfer gets caught whenever he enters a dangling node (sink). Figure 15 shows the simple example of such situation.

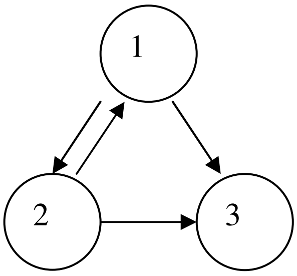


Figure 15. Node three is a sink node.

In the graph from Figure 14 nodes with numbers four, five and six make also a sink. The problem with sinks is that if you use $a^{(k+1)} = M^T a^{(k)}$ equation for calculating the weights, then a lot of nodes (those not in sinks) will have zero weight in the converged eigenvector.

To fix this, Brin and Page define their first adjustment, which we call the stochasticity adjustment. They allow the "random Web surfer" to hop (again at random) from any page to any other, with probability ε/*N* . Then the probability of following an out-link from *j* is $(1-\varepsilon)(1/F(j))$. In terms of matrices, this amounts to adding an all-to-all 'random surfer matrix' (or 'teleportation matrix') $\hat{R}$, and weighting the sum to retain probability conservation. The PageRank eigenvalue equation then becomes $a = \left((1-\varepsilon)M^T + \varepsilon\hat{R}\right)a$. The all-to-all matrix $\hat{R}$ (with $\hat{R}_{ij} = 1/N$) makes the effective network which is represented by the sum $\left((1-\varepsilon)M^T + \varepsilon\hat{R}\right)$ a single SCC (strongly connected component—a graph for which all nodes are reachable from all other nodes). Hence there are no sinks in this effective network, and thus no sink problem: all entries in *a* are positive definite [6].

There is also a problem with cycles (an example is shown in Figure 16). The algorithm doesn't converge at all in this situation.

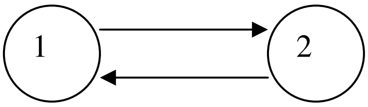


Figure 16. Cycle

Here the node 1 only points to the node 2 and vice versa, creating an infinite loop or cycle. Iterates in this situation will not converge no matter how long the process is run. The cyclic graph has zero eigenvalue gaps (for example the graph in Figure 16 has eigenvalues ±1). However a bigger graph can have cycles (and usually does), but still converge. What is needed is that the cycle lengths have a greatest common factor 1. An example of such a graph is shown in Figure 17.

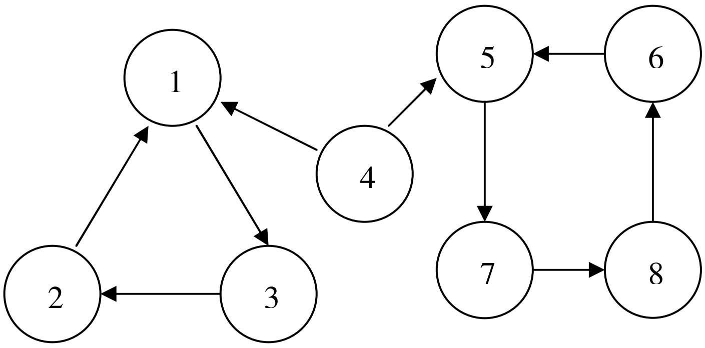


Figure 17. Graph with cycles which will converge

These problems are solved by Brin and Page by modification of $a^{(k+1)} = M^T a^{(k)}$ equation according to the Markov theory. This equation looks like the power method (the method of finding the eigenvalue and eigenvector of matrix) applied to a Markov chain (the process in which the further step is not dependent on the previous, there can be different further steps which have different probability to happen) with transition probability matrix M^T. Nodes with no forward links, create 0 rows in the matrix. All the other rows create stochastic rows (a right stochastic matrix is a square matrix each of whose rows consists of nonnegative real numbers with each row summing to 1 [7]). Thus, M^T is called sub stochastic.

With Markov theory we can make adjustments to $a^{(k+1)} = M^T a^{(k)}$ equation by modifying M^T matrix that insure convergence properties, because for any starting vector, the power method applied to a Markov matrix *P* converges to a unique positive vector called the stationary vector as long as *P* is aperiodic.

2.3.3 HITS (Hyperlink Induced Topic Distillation) Algorithm

The rule in HITS algorithm is that a good authority is one that is pointed to by many good hubs.
In this algorithm a two-level weight propagation scheme is proposed. Every page has two identities: hub and an authority weights. Hub weight is the sum of the authority weights of the nodes that are pointed to by the hub (quality of the page as a pointer to useful resources,) and the authority weight is the sum of the hub weights that point to this authority.
The following iterative algorithm is used for computing the hub and authority weights. First all authority and hub weights are set to 1. In subsequent iterations, the authority weights are updated, and then the hub weights are updated. After each iteration a normalization step is applied, so that the vectors become unit vectors in some norm. The algorithm iterates until the vectors converge.

Let *h* denote the *N*-dimensional vector of the hub weights, where h_i, the *i*-th coordinate of vector *h*, is the hub weight of node *i*. a_i is correspondingly the *N*-dimensional vector of the authority weights.

Algorithm:

Repeat until the weight converges.

```
    For $i \in H$ ; $h_i = 1/c_h \sum_{j \in F(i)} a_j$
    For $i \in A$; $a_i = 1/c_a \sum_{j \in B(i)} h_j$
    Normalize
```

c_h and c_a are undetermined constants (are chosen in the manner to give a vector of unit length as a result) without which a consistent solution is in general impossible. If this process converge we'll get those vectors of hubs and of authorities as eigenvector equation:

$$h = 1/c_h * W * a = 1/c_h * W * 1/c_a * W^T * h = 1/(c_a * c_h) * W * W^T * h$$

$$a = 1/c_a * W^T * h = 1/c_a * W^T * 1/c_h * W * a = 1/(c_a * c_h) * W^T * W * a$$

From these two equation we can see that *h* and *a* are eigenvectors of matrices $H = WW^T$ and $A = W^T W$, which are symmetric.

2.3.4 Salsa (Stochastic Approach for Link-Structure Analysis) Algorithm

This algorithm combines ideas from both HITS and PageRank. Here hub and authority scores are taken from HITS, and normalization (hence stochastic) from PageRank. The Salsa algorithm performs a random walk, alternating between the hub and authority directions. The random walk starts from some authority node selected uniformly at random, then it proceeds by alternating between backward and forward steps. When at a node on the authority direction the algorithm selects one of the incoming links uniformly at random and moves to a hub node. When at a node on the hub direction, the algorithm selects one of the outgoing links uniformly at random and moves to an authority. The authority weights are defined to be the stationary distribution of this random walk. Formally, the Markov Chain of the random walk has transition probabilities.

In this algorithm *a* and *h* are eigenvectors of matrices $A = M^T W_{norm}$ and $H = W_{norm} M^T$, were the matrix W_{norm} is row normalized matrix W ($W_{ij,norm} = \frac{W_{ij}}{|B(i)|}$)

3 State of the art. Existing definitions of clustering coefficient for the weighted graph

Here some existing definitions of the clustering coefficient of the weighted graph are reviewed. All of these definitions are based on the second definition for the unweighted graph introduced in the introduction [see p. 9] (one node clustering).

First some terms useful for the following definitions are introduced.

For each graph the adjacency matrix *A* can be associated, whose elements a_{ij} take the value 1 if an edge connects nodes *i* and *j* and 0 otherwise (*i*,*j*= 1,2,…,*N*, where *N* is the number of nodes in the graph).

The strength of the node *i*: s_i is the sum of weights of all the links linked to the node *i* (Link popularity): $s_i = \sum_{j \in G(i)} w_{ij}$, where *G*(*i*) is the neighborhood of the node *i*. Degree k_i of the node *i* is the number of links that are linked to this node.

Then the first definition of the clustering coefficient [8] for the node is $c_1^w(i) = \frac{1}{s_i(k_i - 1)} \sum_{j,h} \frac{(w_{ij} + w_{ih})}{2} a_{ij} a_{ih} a_{jh}, c_1^w \in [0;1]$. Here we take into account weights of only two edges linked to the node *i* (w_{ij} and w_{ih}). We have a normalization factor $s_i(k_i - 1)$ which ensures that $0 \le c_1^w(i) \le 1$ and that $c_1^w(i)$ recovers the topological clustering coefficient in the case $w_{ij} = const; i, j = 1,2,...,N$ .Thus this definition combines both topological properties and weights distribution. But this definition considers only two of the three weights of triangles of the node.

Another definition [9] is $c_2^w(i) = \frac{1}{k_i(k_i - 1)} \sum_{j,h} (\tilde{w}_{ij}, \tilde{w}_{ih}, \tilde{w}_{jh})^{1/3}, c_2^w \in [0;1]$, $\tilde{w}_{ij} = w_{ij} / w_{\max}$, where $w_{\max}$ is the maximum weight over the whole graph. This definition also fulfils the requirement of being the topological clustering coefficient in the binary case.

One more definition is [10] $c_3^w(i) = \frac{\sum_{j,h} p_{ij} p_{ih} p_{jh}}{\sum_{j,h} p_{ij} p_{ih}}, c_3^w \in [0;1]$, where p_{ij} is a kind of probability that the weight is like 1: $p_{ij} = \frac{w_{ij} - w_{\min}}{w_{\max} - w_{\min}}$, which is the linear normalization of weight: $p \in [0;1]$. It can be also used as $c_3^w(i) = \frac{\sum_{j,h} \hat{w}_{ij} \hat{w}_{ih} \hat{w}_{jh}}{\sum_{j,h} \hat{w}_{ij} \hat{w}_{ih}}, c_3^w \in [0;1]$, were $\hat{w}_{ij} = w_{ij} / w_{\max}$ This clustering coefficient is the average number of triangles divided by the average number of the connected triples.

The last clustering coefficient, introduced by Holme is defined as $c_4^w(i) = \frac{\sum_{j,h} w_{ij} w_{ih} w_{jh}}{w_{\max} \sum_{j,h} w_{ij} w_{ih}}$.

Table 2 shows the properties of all coefficients above [11].

Feature	c_1^w	c_2^w	c_3^w	c_4^w
Recovering of the topological clustering coefficient in the binary case	X	X	X	
$0 \le c^w(i) \le 1$	X	X	X	
Uses global $w_{\max}$ in normalization		X	X	X
Takes into account all weights in triangle		X		X
Invariant to weight permutation for one triangle		X		
Takes into account weights of edges not participating in any triangle	X		X	X

Table 2. Comparison of different clustering coefficients

All of the weighted clustering coefficients except the fourth will be like the clustering coefficient for an unweighted graph ($c(i) = \frac{2t_i}{k_i(k_i - 1)}$, where t_i is the number of triangles connected to the node *i*). The last coefficient will be like $c(i) = \frac{2t_i}{k_i^2}$ which comes to be like the unweighted clustering coefficient only when $k_i >> 1$.

$0 \le c^w(i) \le 1$ is true for all weighted coefficients except c_4^w which never reaches unity for the reason mentioned above. Let us consider the limiting values in more detail. For all coefficients, $c^w = 0$ signifies the absence of triangles. A necessary condition for $c_1^w = 1$, $c_2^w = 1$ and $c_3^w = 1$ is that edges exist between all neighbours of vertex *i*. However, each coefficient sets a different requirement for the weights. When $c_1^w = 1$ is irrespective of edge weights. Contrary to this, $c_2^w = 1$ requires that the weights of all edges $w_{ij} = w_{jk} = w_{\max}$, all weights in each triangle are equal to the maximum weight in the network. Finally, $c_3^w = 1$ if each "outer" edge $w_{jk} = w_{\max}$, irrespective of the weights w_{ij} of edges that are connected to the node *i*.

Global $w_{\max}$ is used in normalization in all versions except c_1^w, where only the local strength s_i matters. This particular choice means that within the same network, two vertices whose neighborhood topology and relative weight configuration are similar can have the same values of

c_1^w even if all the weights in the neighborhood of one vertex are small and those in the neighborhood of the other are large.

Weights of all edges of triangles in which *i* participates are taken into account in c_2^w and c_4^w definitions. However, c_1^w takes into account only the weights of the edges connected to *i*. If the clustering coefficient for the unweighted graph of the same topology is like 1 ($c = 1$) and all weights of the graph ($w_{jk,j \neq k}$) are equal, then c_3^w is insensitive to the weights w_{ij}.

Invariance to permutation of weights within a single triangle is only present in c_2^w, showing that it deals with the triangles as an entity.

Weights of edges not participating in triangles are taken into account in all definitions except c_2^w, where such edges only enter through the vertex degree *k*.

All the above differences are shown in Table 3.

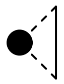
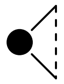
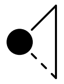
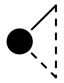
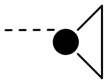
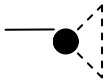
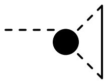
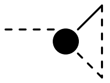
The node	c_1^w	c_2^w	c_3^w
	1	≈0	1
	1	≈0	≈0
	1	≈0	1
	1	≈0	≈0
	≈1/2	1/3	≈1
	≈0	≈0	≈0
	1/3	≈0	1/3
	≈1/2	≈0	≈0

Table 3. Values of the weighted clustering coefficients for different weight configurations when vertex *i* (solid circle) participates in a single triangle. $w_{\max}$ is considered to be unity (1). Here ----are links with weights <<1 and ___are links with weights=1

Table 3 displays the value of the clustering coefficient for a vertex participating in one triangle with varying weight configurations, including vanishingly small weights ($w_{\max}$ is considered to be unity). c_4^w is omitted as it is closely related to c_3^w but is normalized in a way which can be viewed as incorrect. Next, we compare the behavior of the different coefficients in two different empirical networks [11].

First network is the International Trade Network (ITN). The ITN is constructed from trade records between the world's countries during the year 2000, such that vertices denote countries, edges trade relationships, and edge weights trade volumes. The weights are calculated using the dollar volumes of exports and imports between countries ($\exp_{ij}$-export value from *i* to *j* and imp_{ij}- import value from *i* to *j*). We have chosen the edge weights w_{ij} as a measure of the total trade volume such that $w_{ij} = 1/2*(\exp_{ij} + \exp_{ji} + imp_{ij} + imp_{ji})$. The network constructed in this manner has N = 187 countries connected with E = 10252 edges, i.e., it has a relatively high edge density of 52 %.

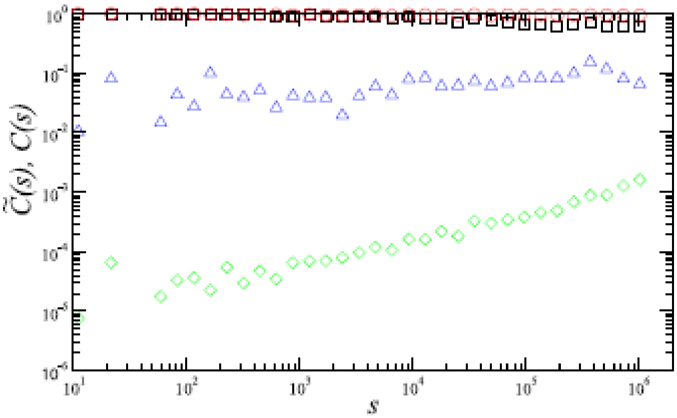


Figure 18. Clustering coefficients computed for the international trade network (ITN) as function of vertex strength s: unweighted *c* (black □) and weighted c_1^w (red ◦), c_2^w (green◇), and c_3^w (blue Δ).

Figure 18 shows the different weighted clustering coefficients as function of vertex strength s. The unweighted clustering coefficient c is also displayed. Due to the large number of edges, c remains high for all s. For low s, c_1^w follows c very closely, whereas for high values of s, c_1^w gets values somewhat higher than c, which can be attributed to high-trade-volume countries engaging in mutual high-volume trade. This effect is far more pronounced in c_2^w, which displays a power-law like increasing trend $c_2^w \propto s^{\beta}$ with β ≈ 0.4, spanning several decades. c_3^w is seen to remain rather insensitive to the weights. Note that the overall level of c_2^w and c_3^w is much lower than that of c and c_2^w due to weight normalization by the global $w_{\max}$ and to a broad distribution of weights.

The next example is Scientific Collaboration Network (SCN): The SCN is constructed from scientists who have jointly authored manuscripts submitted to the condensed matter physics e-print archive (http://www.arxiv.org) from 1995 to 2005. In this network, vertices correspond to scientists and edges to co-authorships of papers. The edge weights have been defined such that $w_{ij} = \sum_p (\delta_{ip}\delta_{jp})/(n_p - 1)$, where the index p runs over all the papers, $\delta_{ip} = 1$ if scientist i is an author of paper p and 0 otherwise, and n_p is the number of authors of paper p. This network has N = 40422 nodes and an average degree of $k \approx 8.7$.

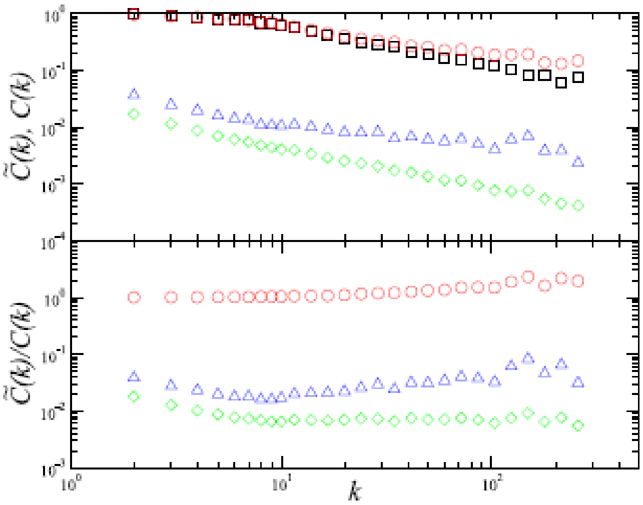


Figure 19. Clustering coefficients computed for the scientific collaboration network (SCN) as function of vertex degree k: unweighted c (black □) and weighted c_1^w (red ◦), c_2^w (green◇), and c_3^w (blue Δ). The lower panel displays the ratio of the weighted coefficients to the unweighted c.

Figure 19 displays different clustering coefficients as a function of degree (upper panel) as well as the ratio of the weighted clustering coefficients to the unweighted coefficient (lower panel). c_1^w remains rather close to c for $k < 10$ but for $k > 10$ their ratio is somewhat increased, indicating that the weights of edges that do not participate in triangles are relatively low and/or the weights of edges participating in several triangles are relatively high. In contrast, the shape of c_2^w differs from c for $k < 10$. The ratio is the largest for low-degree vertices, becoming approximately constant at $k \approx 10$. A possible reason for this is that young scientists (e.g. graduate students) tend to participate in repeated collaborations involving a relatively small number of authors, giving rise to high intensity triangles. c_3^w appears to capture the low-k behavior of c_2^w as well as the high-k-behavior of c_1^w.

It is clear from the above considerations that there is no ultimate formulation for a weighted clustering coefficient. Instead, we have seen that the different definitions capture different aspects of the problem at hand. For unweighted networks, it is straightforward to measure how many edges out of possible ones exist in the neighborhood of a vertex; yet the questions of how to measure the amount of weight located in this neighbourhood and what to compare this with, are far from obvious. In a way, c_1^w and c_2^w can be seen as limiting cases: c_1^w compares the weights associated with triangles to the average weight of edges connected to the focal vertex, while c_2^w disregards the strength of the focal node and measures triangle weights only in relation to the maximum edge weight in the network. c_3^w can be viewed as an interpolation between these two, albeit being a somewhat uncontrollable one as is evident from the examples in Table 3. Given these observations, we feel that there is no single general-purpose measure for characterizing clustering in weighted complex networks.

4 Clustering of the weighted graph and replacing of the eigenvector with the Link popularity

4.1 The main idea

The assumption here is that if the similarity graph is highly clustered than we can use the link popularity vector (LP) for ranking of results and not the eigenvector. In the other words, if the node of the similarity graph has a high eigenvector centrality (EVC) than it also has a high L (link popularity).

The point here is that the similarity graph is a good clustered graph. As discussed before, the clustering coefficient [see p. 9] measures how well nodes in the graph are linked with another. Good clustering for the graph means that if the node is linked to two other nodes than there is a large probability that those two are also linked. It should be true for the similarity graph because it is obvious that if a document is very like two other documents then those two are like each other.

We should remember that the above discussion is true for the links that have high link strengths. Because we have a similarity graph here, if the node is strongly linked with the two other nodes than those two are with high probability strongly linked with each other.

EVC of the node in its turn is high if the node is well connected to the nodes which are well connected. High L of the node means that the node is good connected to its neighbors. But thus if we have high clustering of the graph then if the node is well connected to its neighbors, the nodes of its neighborhood are also well connected (because there is a large probability that every pair of nodes of the neighborhood are linked). And the well connected neighborhood of the well connected node means the high EVC. And opposite: if the node has a high EVC that means first of all that this node has a good connected neighborhood. Thus, by high clustering, the neighbors of the node's neighbors are also neighbors of the node. Out from this we have a high Link popularity of the node.

This point is tested in the further section [see p. 41].

The main problem here is to measure the clustering coefficient of the weighted graph. We discussed the two different definitions of the clustering coefficient for the unweighted graph and some definitions of clustering coefficient for the weighted graph. Those weighted clustering coefficients are not able to measure clustering characteristics taking an account both topology and weights of links in the graph, and thus we need to invent a new definition (definitions) of clustering coefficient for the weighted graph which should have the same properties as those for the unweighted graph.

4.2 Clustering coefficient for the weighted graph

The clustering coefficient for the weighted graph is defined here based on the first definition [the reasons are given on p. 11] of the clustering coefficient for an unweighted graph: $cc = \frac{3 * N_t}{N_{con}}, cc \in [0,1]$.

Here we make two different definitions but the first of those is used afterwards only for similarity graph structure analyzing.

The first definition is the same as that one for the non weighted graph but for the reduced graph. First all weak links are removed and after that the clustering coefficient is calculated as if there are weights "1" on the strong links: $cc_t = \left. \frac{3 * N_t}{N_{con}} \right| w_{ij} > t; i, j = 1,2...N; cc \in [0,1]$. Here t is a parameter which varies from 0 to 1.

The next definition is much more useful and it uses products of weights in triangles and connected triples. If $P_1 = \sum_{\Delta} w_{ab} \bullet w_{bc} \bullet w_{ac}$ is the sum of triangle products over the whole graph, where a, b and c are nodes of the same triangle and w_{ab}, w_{bc}, w_{ac} are the corresponding link weights, and if $P_2 = \sum_{\Lambda} w_{AB} \bullet w_{BC}$ is the sum of connected triple products over the whole graph, where A, B and C are nodes of the same connected triple and w_{AB}, w_{BC} are the corresponding link weights, then the clustering coefficient is defined as $cc_w = \frac{3 * P_1}{\overline{w} * P_2}$, where $\overline{w}$ is the average weight over all links in the graph: $\overline{w} = \frac{\sum_{i=1}^{N} \sum_{j=i+1}^{N} w_{ij}}{N_{link}}$, with N_{link} - the number of the non zero links in the graph.

From this definition we can confirm that this clustering coefficient doesn't vary from 0 to 1 as it was for an unweighted graph. It varies from 0 to ∞.

Example 6

For example, if we have a graph which has only one triangle and the long chain of nodes linked to this triangle, and links of triangle has the largest weight (1), other links are very weak (has weight $0 < \varepsilon << 1$), for example, $\varepsilon = 10^{-3}$ and the number of links with the weight ε is large, for example, $N_{\varepsilon} = 10^8$, than clustering coefficient will be around 30. Figure 20 shows such a graph.

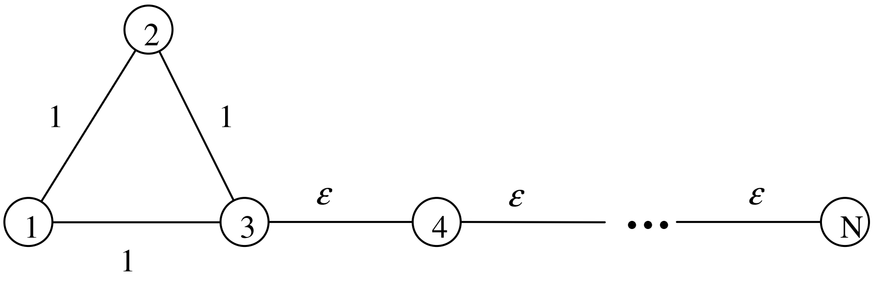


Figure 20. Example graph

For this graph $P_1 = 1$, $P_2 = 3 + 2\varepsilon + (N_\varepsilon - 1)\varepsilon^2$ and $\overline{w} = \dfrac{3 + N_\varepsilon \varepsilon}{3 + N_\varepsilon}$. Then we can write that

$$cc_w = \frac{3}{\dfrac{3 + N_\varepsilon \varepsilon}{3 + N_\varepsilon}(3 + 2\varepsilon + (N_\varepsilon - 1)\varepsilon^2)} \xrightarrow[\varepsilon<<1]{N_\varepsilon \varepsilon>>1} \frac{3}{\varepsilon(3 + N_\varepsilon \varepsilon^2)} = \frac{3}{3*10^{-3} + 10^{-1}} = \frac{3}{0{,}0003 + 0{,}1} = 29{,}91$$

4.3 Evaluation of the clustering coefficient

How we can evaluate our new definition of the clustering coefficient [p. 32]? If we assume that the clustering coefficient is high for the similarity graph, then we can calculate our coefficient for that graph and check if it is high. But how shall we know if it is high? We need something to compare with. We can calculate the same clustering coefficient for the graph with same topology, but with the random weights on the links (random graph), which should have the clustering coefficient much lower than the similarity graph. We can also compare our coefficient for the similarity graph with the coefficient for the graph with the same topology, but all the links weights like “1” (corresponding coefficient for unweighted graph) We suppose that the clustering coefficient for the unweighted graph is also lower than the one for the similarity graph, and clustering coefficients for the random and non weighted graphs are almost like each other. Figure 21 shows the example of the graphs.

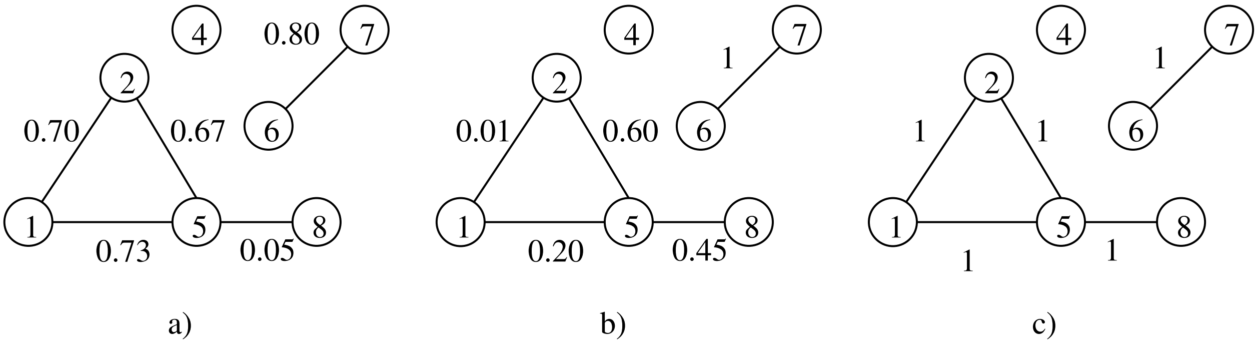


Figure 21. a)- similarity graph, b)- random graph and c)- non weighted graph.

Now we'll try to confirm the above assumption—that clustering coefficients for random (*ccr*) and non weighted (*cc*1) graphs are almost like each other.

First let's take a look at the random graph. The average weight of all the links of such a graph is $\overline{w} = 0{,}5$. Then we can show that our clustering coefficient (*ccw*) for the graph with all links weight equal to 0,5 is exactly equal to the clustering coefficient for an unweighted graph.

$$ccw(0.5) = \frac{3 * \sum\limits_{\Delta} w_{ab} w_{bc} w_{ac}}{\overline{w} \sum\limits_{\Lambda} w_{AB} w_{BC}} = \frac{3 * \sum\limits_{\Delta} 0{,}5 * 0{,}5 * 0{,}5}{0{,}5 \sum\limits_{\Lambda} 0{,}5 * 0{,}5} = \frac{3 * N_{\Delta}}{N_{\Lambda}} = cc1$$

We can say that every random value of weight for *ccr* looks like $w_i = \overline{w} + \varepsilon_i, i = 1,2...n$, were n is the number of links and ε_i can be positive or negative with the same probability because of the randomness. So the sum of all ε_i is almost "0": $\sum\limits_{i=1}^{n} \varepsilon_i \approx 0$.Now we can write an expression for the random clustering coefficient as

$$ccr = \frac{3 * \sum\limits_{\Delta} (\overline{w} + \varepsilon_1)(\overline{w} + \varepsilon_2)(\overline{w} + \varepsilon_3)}{\overline{w} \sum\limits_{\Lambda} (\overline{w} + \varepsilon_4)(\overline{w} + \varepsilon_5)} =$$

$$= \frac{3 * \sum\limits_{\Delta} (\overline{w}^3 + (\varepsilon_1 + \varepsilon_2 + \varepsilon_3)\overline{w}^2 + (\varepsilon_1\varepsilon_2 + \varepsilon_2\varepsilon_3 + \varepsilon_1\varepsilon_3)\overline{w} + \varepsilon_1\varepsilon_2\varepsilon_3)}{\overline{w} \sum\limits_{\Lambda} (\overline{w}^2 + (\varepsilon_4 + \varepsilon_5)\overline{w} + \varepsilon_4\varepsilon_5)} =$$

$$= \frac{3 * \left(\sum\limits_{\Delta} \overline{w}^3 + \overline{w}^2 \sum\limits_{\Delta} (\varepsilon_1 + \varepsilon_2 + \varepsilon_3) + \overline{w} \sum\limits_{\Delta} (\varepsilon_1\varepsilon_2 + \varepsilon_2\varepsilon_3 + \varepsilon_1\varepsilon_3) + \sum\limits_{\Delta} \varepsilon_1\varepsilon_2\varepsilon_3 \right)}{\overline{w} * \left(\sum\limits_{\Lambda} \overline{w}^2 + \overline{w} \sum\limits_{\Lambda} (\varepsilon_4 + \varepsilon_5) + \sum\limits_{\Lambda} \varepsilon_4\varepsilon_5 \right)} \approx \frac{3 * \sum\limits_{\Delta} \overline{w}^3}{\overline{w} * \sum\limits_{\Lambda} \overline{w}^2} =$$

$$= \frac{3 * \sum\limits_{\Delta} 0{,}5 * 0{,}5 * 0{,}5}{0{,}5 \sum\limits_{\Lambda} 0{,}5 * 0{,}5} = cc1$$

Our testing results from the next chapter confirm that the above reasoning is correct.

5 Testing and results

In this section the following will be tested:

1. New clustering coefficient for weighted graphs. Here we shall compare the result for similarity graphs with results for graphs with the same structure, but with random weights on the links, and with results for the unweighted graphs with the same structure.
2. If the Link Popularity for similarity graphs correlates with the EVC, and is this correlation larger than for the random graphs.

5.1 Description of Clustering coefficient testing

Here we used two different data sets: one was taken from the Wikipedia free encyclopaedia, and the other was just 10 similarity graphs built from hit lists which were obtained from search engine results. MATLAB software was used to do all the calculations of clustering.

In the first case we took, from the similarity graph with 50000 nodes, 20 sub graphs which have 101 nodes each, and 95 sub graphs with 501 nodes each(with the help of MATLAB). For each of these graphs we calculated three different clustering coefficients (*ccw*- clustering coefficient for the similarity graph, *cc*1- clustering coefficient for the unweighted graph and *ccr*- clustering coefficient for the random graph). The two tables of results for 101 nodes and for 501 nodes graphs are shown in Appendix 1. The source code with some comments you can find in Appendix 2. The Figure 22 and Figure 23 compare the clustering coefficients *ccw*, *cc*1 and *ccr*. Here the results are sorted with respect to *cc*1.

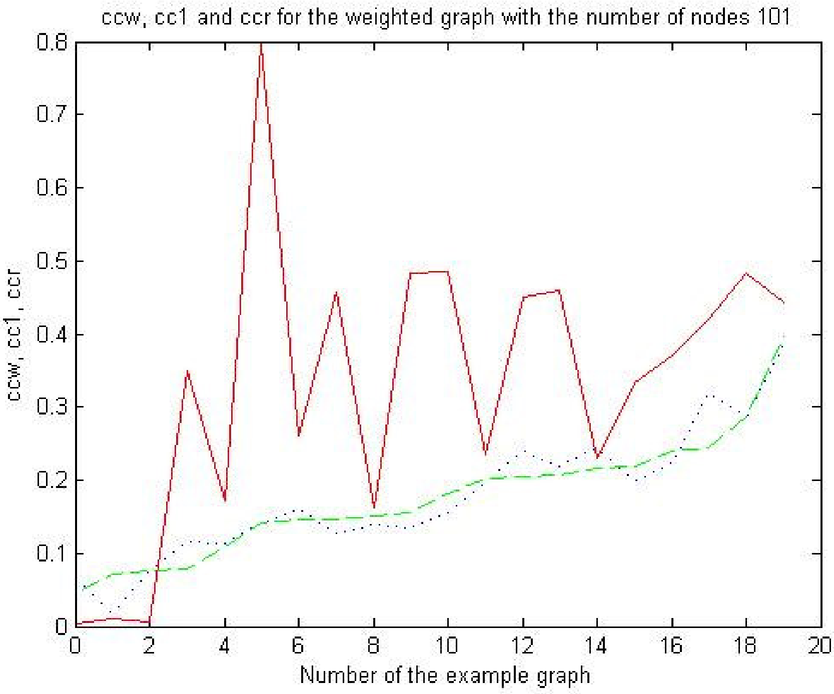


Figure 22. Comparison of *ccw* (red _), *cc*1 (green --) and *ccr* (blue ..) for 101 nodes graphs from the Table 7.

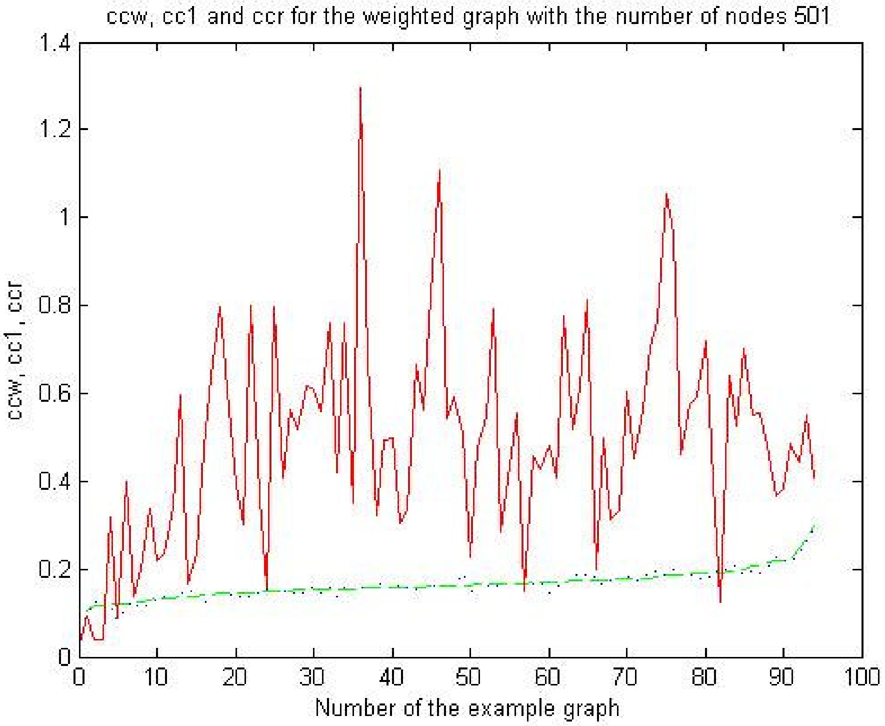


Figure 23. Comparison of *ccw* (red _), *cc*1 (green --) and *ccr* (blue ..) for 501 nodes graphs from the Table 8.

Our assumption made in the previous section is confirmed here: the average value for *ccw* for both cases (101-node graphs and 501-node graphs) is lager than the average values for *cc*1 and *ccr*. There are only 5 exceptions for 101-node graphs, and 7 exceptions for 501-node graphs, which have *ccw* <*cc*1 or/and *ccw*<*ccr*. But as the Figure 24 and the Figure 25 shows all of those exceptions are not good examples for the similarity graphs because they have a very low clustering (small number of triangles in comparison with connected triples) and density. For example, from Table 4 (calculation of the first definition of clustering coefficient of the previous section (with threshold)) one can see that, in the comparison with the other graphs, those exceptions have very low weights on the links. Table 4 shows the results for the first definition clustering coefficient with different thresholds: from 0,1 to 0,9 for 101 nodes graphs. If we delete all low-weighted links (with weights smaller than 0,1) and after that get these clustering coefficient like ”0” it means that there are no links in triangles with weights large than 0,1- no strong triangles, which is not possible in the good similarity graphs. Thus those graphs can’t be evaluated as similarity graphs. Those graphs occur due to the problem of taking the random sub-graphs from the good similarity graph.

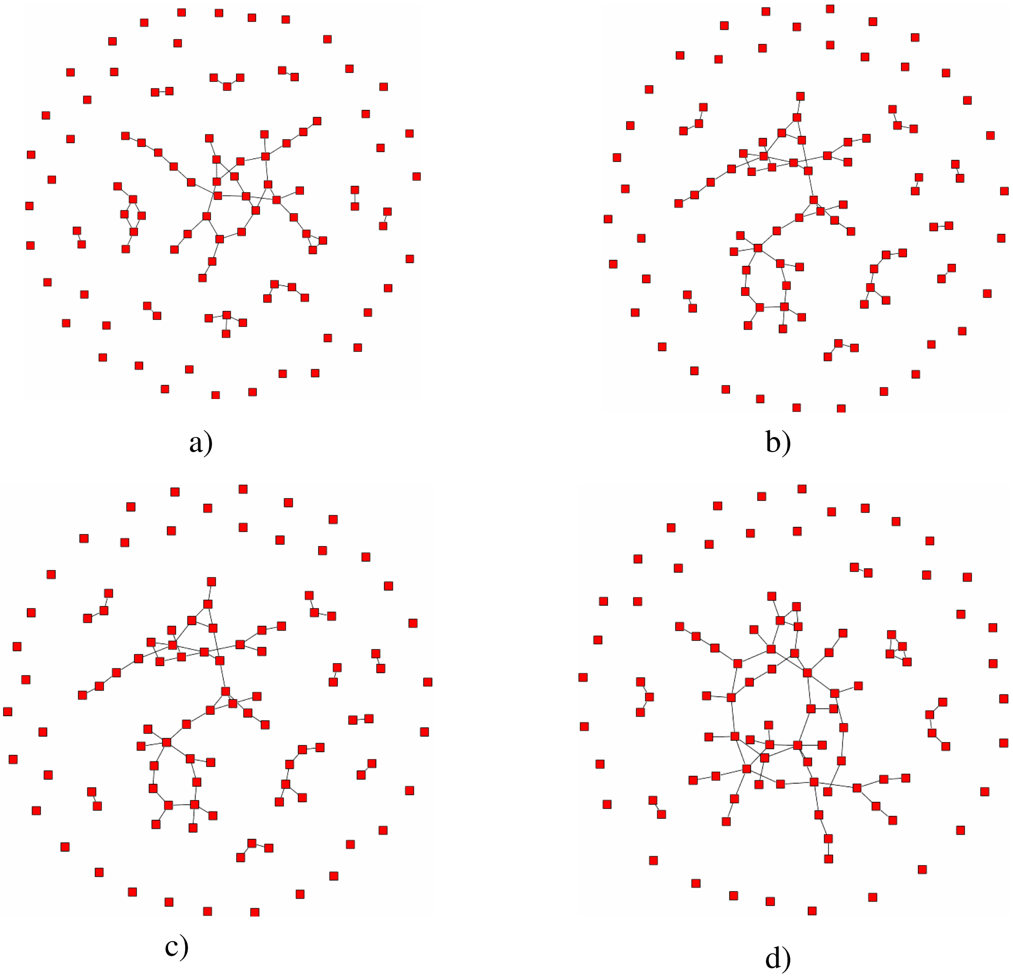


Figure 24. a)- graph from index[100;200]; b)- index [200;300]; c= index[400;500] and d)- index[700;800]

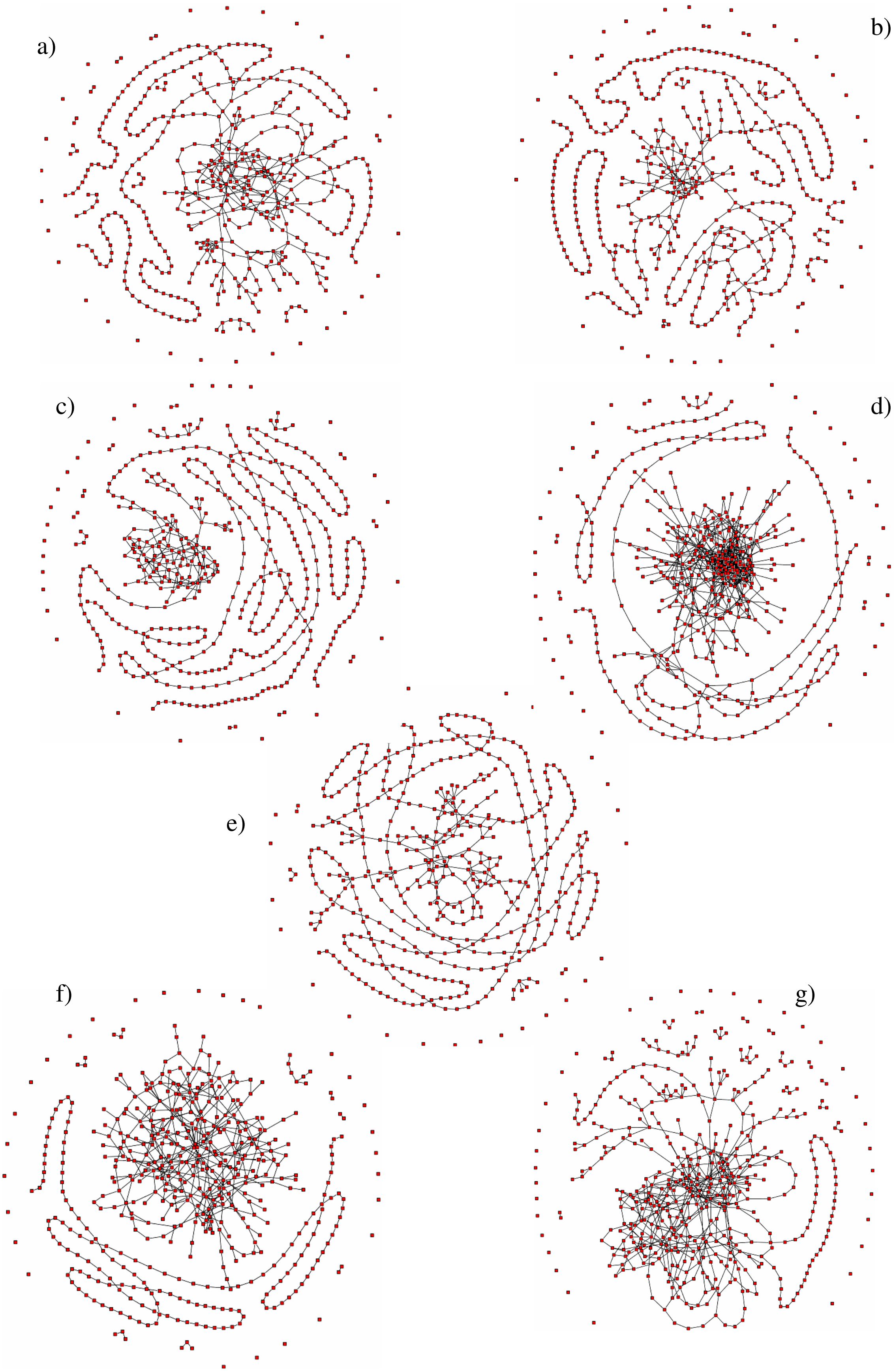
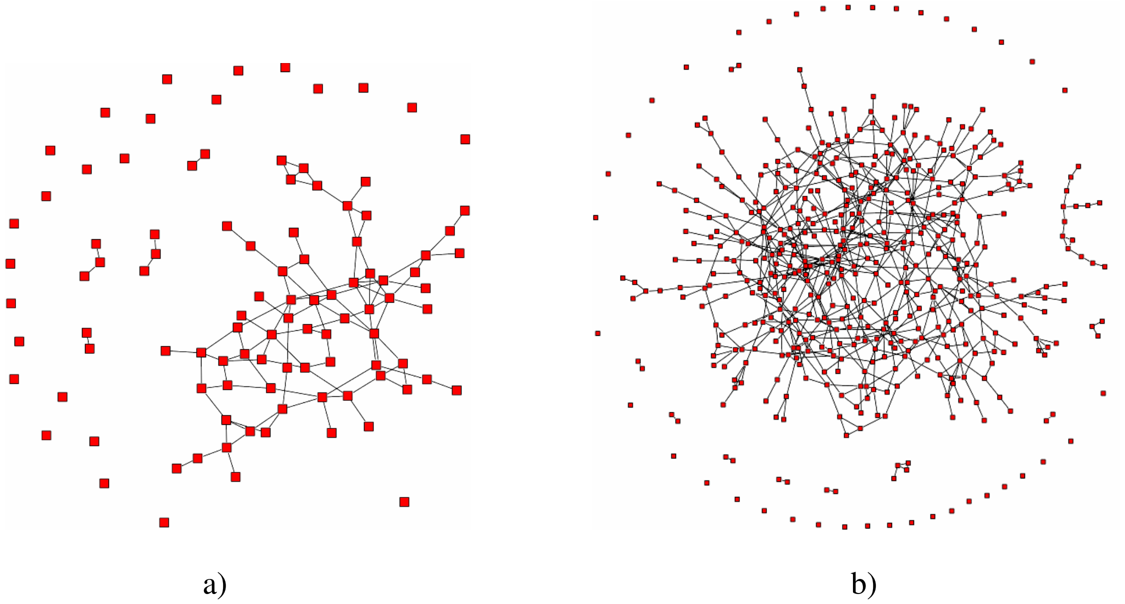


Figure 25. a)- index[3500;4000]; b)- index[4000;4500]; c)- index[4500;5000]; d)- index[5000;5500]; e)- index[6500;7000]; f)- index[28500;29000]; g)- index[34500;35000]

cct(0,1)	cct(0,2)	cct(0,3)	cct(0,4)	cct(0,5)	cct(0,6)	cct(0,7)	cct(0,8)	cct(0,9)	index
0,193548	0,5	0	0	0	0	0	0	0	[1;100]
0	0	0	0	0	0	0	0	0	[100;200]
0	0	0	0	0	0	0	0	0	[200;300]
0,315287	0,204545	0	0	0	0	0	0	0	[300;400]
0,148148	0,157895	0	0	0	0	0	0	0	[400;500]
0,228571	0,295775	0	0	0	0	0	0	0	[500;600]
0,3	0,545455	0	0	0	0	0	0	0	[600;700]
0	0	0	0	0	0	0	0	0	[700;800]
0,268293	0,4	0	0	0	0	0	0	0	[800;900]
0,256708	0,284314	0,275641	0,428571	0	0	0	0	0	[900;1000]
0,205298	0,207653	0,228311	0,176471	1	0	0	0	0	[1000;1100]
0,280936	0,326733	0,338983	0	0	0	0	0	0	[1100;1200]
0,399388	0,402921	0,41475	0,342268	0	0	0	0	0	[1200;1300]
0,157895	0,15808	0,16	0,095238	0	0	0	0	0	[1300;1400]
0,134328	0,145161	0,157895	0,230769	0,375	0	0	0	0	[1400;1500]
0,131474	0,121622	0,104478	0,098901	0,375	0	0	0	0	[1500;1600]
0,244514	0,303797	0,666667	0,75	0	0	0	0	0	[1600;1700]
0,22619	0,28125	0,375	0,75	1	1	1	1	1	[1700;1800]
0,24	0,214286	0	0	0	0	0	0	0	[1800;1900]
0,111111	0,096774	0,193548	0,2	0,333333	0,6	0	0	0	[1900;2000]

Table 4. The first definition for clustering coefficient.

For comparison, Figure 26 shows some of the better graphs with higher clustering. In this picture you can see that the number of triangles in comparison with the number of connected triples is larger than in the previous figures. But one must remember that it is not enough for high clustering to have a large number of triangles. Those triangles should have the high weight also. So such a graph as Figure 25 d) can be acceptable as a similarity graph if the link weights of the "tails" are very small (which is not true in our case) and if the link weights of the triangles are large.


a)

b)

Figure 26. a)- index[1900;2000] and b)- index[48500;49000]

Now let’s take a look at the 10 full similarity graphs (not slices from the large similarity graph as in the previous testing example).

The results analogous to those from Table 8 and Table 9 are shown in Table 5.

N	n	t	tw	cc1	ccw	ccr	the query
288	41058	0,99347	0,096708	0,99929	5,808	0,9998	"belona"
487	117830	0,99565	0,16302	0,99926	2,0901	0,99927	"fotball"
479	114480	1	0,10056	1	1,7295	1	"fredrikstad"
466	105640	0,97502	0,087417	0,98745	3,3828	0,98733	"jens stoltenberg"
472	110100	0,99046	0,083118	0,99581	1,2381	0,99576	"mette marit"
406	82215	1	0,3065	1	1,5552	0,99985	"schibsted"
457	102860	0,98717	0,13778	0,99449	1,5656	0,9943	"selvangivelse"
493	121280	1	0,19707	1	1,1847	1,0001	"telenor"
457	102860	0,98717	0,13778	0,99449	1,5656	0,99465	"toyota"
424	89555	0,99865	0,11509	0,99903	1,4061	0,99918	"wikipedia"
Average				0,996982	2,15257	0,997024	

Table 5. Clustering coefficient for fool similarity graphs

And the results like Figure 22 Figure 23 will be like those in Figure 27.

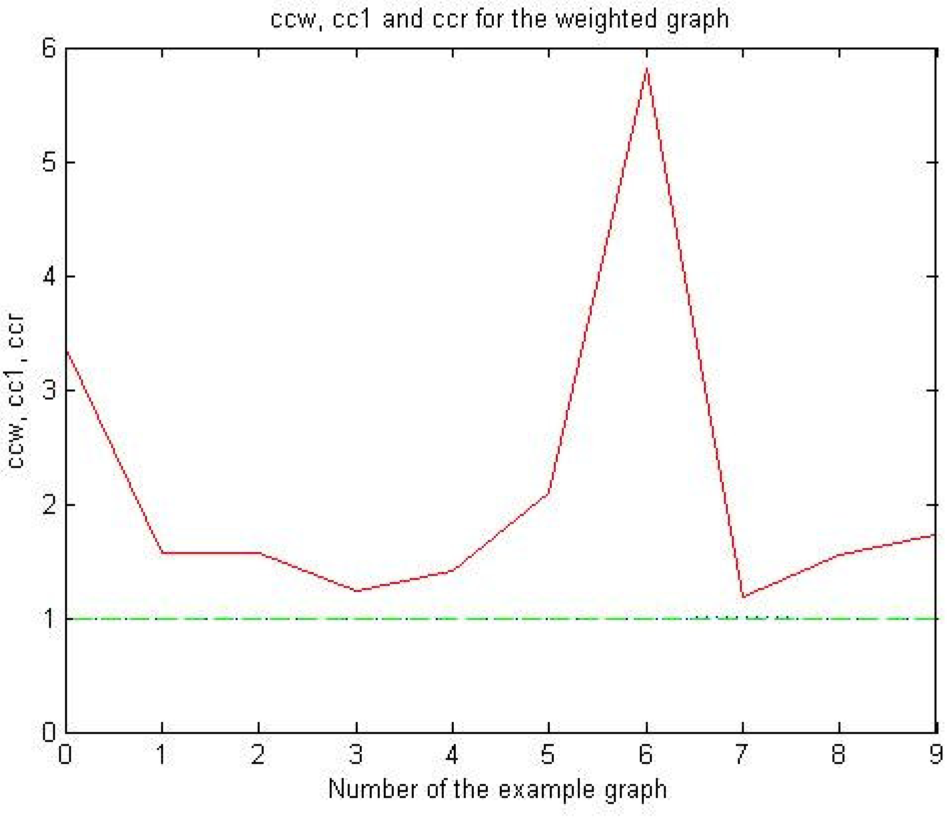


Figure 27. Comparison of *ccw* (red _), *cc*1 (green --) and *ccr* (blue ..) for 501 nodes graphs from the Table 5

Here we can see that the difference between average *ccw* and *cc*1/*ccr* is even larger than in the previous testing, and also that *ccw* is always higher than *cc*1/*ccr*. So we believe that the exceptions from the previous testing were due to the similarity test graphs being not good enough (poorly focused). But why are the first and the fourth values of *ccw* in Table 5 so much larger than “1”?

Let's make a histogram of links strength for all the graphs used in Table 5 (Appendix 3. Histograms of link strengths for similarity graphs). Here we can see that the larger the difference between high values of link strength and low values of link strength is, the higher *ccw* is. This is analogous to the situation in Example 6, where we have 3 links with weight "1" and a large number of links with weight 0.001.

If we now take a look at the histograms for some of those 'slice' graphs from the previous testing (Appendix 4. Histograms for the 'slice' graphs.) we can note that, if the graph has a high *ccw,* then it has a histogram like the three first ones in the Appendix 4: most of the links have some special link strength. The explanation for this is that, if most of the link strengths are like each other, then there is a large probability that the ccw is also large [from our definition on p. 32].

5.2 EVC and Link popularity correlation testing

For the first part of the testing we took the largest connected component for each of the 11 graphs (good similarity examples) and drew a scatter graph (Link popularity- x and EVC- y) for both similarity and random graphs. A connected component of a graph is a subgraph in which there is a path from every node to every other node.

Here we need to take the largest connected component (component with maximum number of nodes) for every graph, because nodes of the other small components have a "0" EVC value. Table 6 shows some results. The source code you can find in the Appendix 6.

In Table 6 the *ccw* of the component, the correlation coefficient between EVC and LP, and the weighted and unweighted densities of the component are shown. Here the correlation coefficient (Cor) of two vectors X and Y is calculated as following: $Cor(X,Y) = C(X,Y) / \sqrt{C(X,X)C(Y,Y)}$, $C(X,Y) = E * ((X - \mu_x)(Y - \mu_y))$, E is the expected value, and $\mu_x = EX$

Index	N of the component	ccw of the component	Correlation coefficient	t	tw
[500;600]	40	0,310115746	0,962296	0,091026	0,016227
[500;600]rand	40	0,227517651	0,928454	0,091026	0,046485
[700;800]	50	0,000393004	0,791615	0,044898	0,004653
[700;800]rand	50	0,029001219	0,507246	0,044898	0,022221
[1000;1100]	89	0,233880974	0,934325	0,067416	0,021048
[1000;1100]rand	89	0,214769852	0,914391	0,067416	0,034555
[1200;1300]	93	0,443101543	0,986663	0,122253	0,045135
[1200;1300]rand	93	0,392977451	0,980712	0,122253	0,063107
[1300;1400]	93	0,163231938	0,878905	0,04792	0,016194
[1300;1400]rand	93	0,114210458	0,822305	0,04792	0,023692
[1500;1600]	80	0,172180569	0,880754	0,038291	0,012217
[1500;1600]rand	80	0,106511738	0,773038	0,038291	0,017408
[1600;1700]	9	1,095762779	0,94615	0,277778	0,05125
[1600;1700]rand	9	0,439093782	0,852147	0,277778	0,128175
[2000;2500]	494	0,458717008	0,866009	0,025162	0,005001
[2000;2500]rand	494	0,182818871	0,829829	0,025162	0,012529
[6000;6500]	341	0,52998867	0,535485	0,009971	0,001707
[6000;6500]rand	341	0,125245006	0,61576	0,009971	0,004926
[7500;8000]	458	0,283769289	0,735926	0,010243	0,001088
[7500;8000]rand	458	0,111682506	0,706898	0,010243	0,004965
[8000;8500]	454	0,373868501	0,843412	0,010785	0,001357
[8000;8500]rand	454	0,138695349	0,773868	0,010785	0,005335

Table 6. Correlation between Link popularity and EVC

Here we can see that the most part of similarity graphs have larger clustering coefficient than random graphs and larger correlation coefficient. There are only two exceptions here: one (blue) with the *ccw* of the random graph large than *ccw* of the similarity graph (but the clustering coefficient is so low that we can't evaluate this graph as a similarity graph) and one (red) with the correlation coefficient of the similarity graph lower than correlation coefficient for the random graph (but here the both density values are very low, which is also not a good property for the similarity graph). Further details are shown in Appendix 7. Correlations of LP and EVC.

If we make a scatter plot between EVC and LP for our 10 almost complete similarity graphs, we get, for all of the random graphs (built with the same topology as the similarity graphs), the correlation coefficient almost exactly "1" (*Cor*=0,9997). The correlation coefficients for similarity graphs are also high, but lower than those for random graphs. The results are shown in

Table 7. The scatter plot for a random graph is shown in Figure 28, and all of the scatter plots for the similarity graphs can be found in Appendix 5..

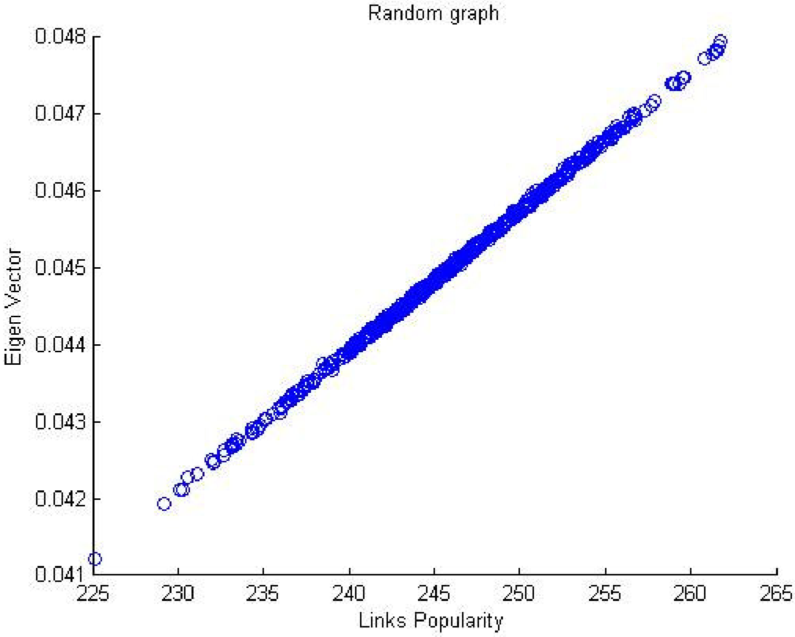


Figure 28. Scatter plot of LP and EVC for the random graph buil on similarity graph of "telenor" query

Our explanation of this result is as follows. If we have a large number of nodes, then the weight distribution on the links will be almost normal The Link popularity for every node will then be $LP \approx \overline{w} * N = 0.5 * N$ (for the Figure 28 $LP \approx 0.5 * 493 = 246.5$); and as Figure 28 shows, LP varies very little, in the interval (225; 265) (if you'll compare with the same figure for the similarity graph (Appendix 5) it varies in the interval (20; 180)). Because all the nodes have the same distribution of the link weights connected to them, and the same number of links, the equation $p_i^{new} = \sum_{j \neq i} p_j s_{ij}$ [see p. 7] will converge with only one iteration, and the EVC value will be almost the same for all the nodes. This is shown in the Figure 28: EVC varies in the interval (0,041; 0,048). In comparison, for the true similarity graphs EVC varies in the interval (0,01; 0,08) (Appendix 5). Thus *ccw* for the similarity graph in this case is always higher than *ccw* for the random graph; but our assumption that the higher *ccw* always gives the higher correlation coefficient between LP and EVC does not hold in this case.

	belona	fotball	fredrikstad	jens stoltenberg	mette marit	schibsted	selvangivelse	telenor	toyota	wikipedia
C	0.9375	0.9914	0.9836	0.9078	0.9985	0.9788	0.9862	0.9988	0.9862	0.9898
Cr	0.9997	0.9996	0.9996	0.9997	0.9997	0.9996	0.9997	0.9997	0.9997	0.9997

Table 7. Correlation coefficients for similarity graphs of the different queries (belona), C- for the similarity graphs and Cr- for the random graphs.

We can also mention that the scatter plots from Appendix 5 are related to the histograms from Appendix 3. For example, those graphs that have two sets of link strengths ("belona", "football", "jens stoltenberg") have also two sets of Link popularity scores and EVC scores.

6 Discussion and evaluation

In this work the new clustering coefficient for the weighted graph was invented. This coefficient (ccw) shows how good clustered (good linked) the graph is. As the testing results of the previous section show, this coefficient is good to use for evaluating weighted graphs because it is higher for the real similarity graph than for the random (which we assume to be badly linked than the similarity graph) and the unweighted graph of the same topology. But this coefficient doesn't have the property of the clustering coefficient for the unweighted of being always not higher than "1" and not lower than "0". It varies from "0" to "∞". It seems also that the clustering coefficient (*ccw*) gives information about the distribution of link weights. If the clustering coefficient is close to "1" than there is more like normal distribution of weights (Appendix 3 ("mette marit")), but if the clustering coefficient is significantly higher than "1" then there is a large number of low weighted links and some high weighted links, but little or nothing in between (Appendix 3 ("belona")).

In the first testing case of the previous section we had that the correlation coefficient between the eigenvector of the similarity graph and the link popularity vector of the graph was higher for the similarity graphs than for random graphs. It was also true that the higher the clustering coefficient is, the higher the correlation between LP and EVC is. But in the other testing case (almost complete similarity graphs) this trend was opposite: the correlation coefficient between LP and EVC was higher for the random (low clustering) graphs than for similarity (high clustering) graphs. This was not expected; and we found that this correlation depends on the density of the graph. (For complete random-weighted graphs, the EVC scores and the link popularity scores are almost the same for every node (see p.41)). In the second testing case we had link density almost like "1" in all of the graphs—so that these graphs were complete or almost complete.

However, the correlation coefficient between the LP and EVC for the similarity graphs is high in both testing cases, so as we supposed in chapter 1 (1.3 Problem specification) we can use the Link Popularity vector instead of the eigenvector for ranking search results.

7 Conclusion and Future work

During this work, a clustering coefficient for weighted symmetric graphs was invented and tested. It can be used in the future for evaluation of clustering in weighted graphs. But how the coefficient depends on the distribution of weights, and which value of the clustering coefficient is the "best" (shows the highest clustering of the graph) needs to be better defined.

The LP (link popularity) vector can be used to rank the search results (using similarity graph) instead of EVC (eigenvector) vector. But how the density of the graph affects this correlation coefficient, and how the clustering coefficient is related to the correlation coefficient, are questions for future work.

References

[1] R. Baeza-Yates, G. Navarro. Text Searching: Theory and Practice. In: Formal Languages and Applications, Berlin, Springer, pages 565-597 (2004)

[2] M. E. J. Newman. The structure and function of complex networks. In: SIAM Review, Vol. 45, No. 2, pages 167-256 (2003)

[3] R. Baeza-Yates, B. Ribeiro-Neto, Modern Information Retrieval. ACM Press Series/Addison Wesley, New York, (1999)

[4] A. Borodin, G. O. Roberts, J. S. Rosenthal, P. Tsaparas. Link analysis ranking: Algorithms, theory and experiments. In: ACM Transactions on Internet Technology (TOIT), Vol. 5, No. 1, pages 231 - 297 (2005)

[5] A. N. Langville, C. D. Meyer. Google's PageRank and Beyond. The science of search engine ranking. Award for Best Professional (AAP)/Scholarly Book in Computer & Information Science. (2006)

[6] J. Bjelland, G. Canright, K. Engø-Monsen. Link analysis and Web search, Oslo, Telenor R&I (2008)

[7] Wikipedia free encyclopaedia. http://en.wikipedia.org/wiki/Stochastic_matrix (2008)

[8] M. Barthelemy, A. Barrat, R. Pastor-Satorras, A. Vespignani. Characterization and modelling of weighted networks. In: Phisica A 346, pages 34-43 (2005)

[9] J.-P. Onnela, J. Saramåki, J. Kertesz, K. Kaski. Intensity and coherence of motifs in weighted complex networks. In: Phys. Rev. E 71 (2007)

[10] S. E. Ahnert, D. Garlaschelli, T. M. A. Fink, G. Caldarelli. An ensemble approach to the analysis of weighted networks. In: Phys. Rev. E 76 (2007)

[11] J. Saramåki, M. Kivelå, J.-P. Onnela, K. Kaski, J. Kertesz. Generalizations of the clustering coefficient to weighted complex networks. In: Phys. Rev. E 75 (2007)

[12] Library at http://www.mathworks.com/matlabcentral/fileexchange/loadFile.do?objectId=10922&objectType=FILE (2008)

[13] Tulip hompage. http://tulip.labri.fr/ (2008)

Appendix 1. Result tables for three clustering coefficients (cc1, ccw and ccr)

In the tables N is the number of nodes in the graph, n is the number of links in the graph, t is the density of the graph (which is $t = \frac{n}{N(N-1)/2}$), tw is the weighted density ($t = \frac{\sum_{i=1}^{n} w_i}{N(N-1)/2}$) and index shows the numbers of nodes which were taken from the whole 50000 nodes graph.

N	n	t	tw	cc1	ccw	ccr	index
100	69	0,01395	0,00195	0,1472	0,4567	0,1284	[1;100]
101	55	0,0109	0,00155	0,0441	0,0032	0,0674	[100;200]
101	59	0,0117	0,00195	0,0723	0,0097	0,0184	[200;300]
101	112	0,0222	0,0032	0,2877	0,4839	0,2865	[300;400]
101	78	0,01545	0,0019	0,2163	0,2311	0,2436	[400;500]
101	89	0,0176	0,00275	0,2184	0,334	0,1983	[500;600]
101	54	0,0107	0,0016	0,2455	0,4199	0,3199	[600;700]
101	67	0,01325	0,0014	0,0763	0,0051	0,0763	[700;800]
101	80	0,01585	0,002	0,1809	0,4847	0,1567	[800;900]
101	204	0,0404	0,0117	0,2409	0,3707	0,2231	[900;1000]
101	265	0,0525	0,01605	0,2022	0,2345	0,1981	[1000;1100]
101	123	0,02435	0,0056	0,2042	0,451	0,2394	[1100;1200]
101	523	0,10355	0,03755	0,3954	0,4431	0,385	[1200;1300]
101	206	0,0408	0,00895	0,1503	0,1637	0,1395	[1300;1400]
101	85	0,01685	0,004	0,0795	0,3489	0,1169	[1400;1500]
101	124	0,02455	0,00685	0,1091	0,1723	0,1146	[1500;1600]
101	124	0,02455	0,00455	0,2077	0,4594	0,2187	[1600;1700]
101	123	0,02435	0,0053	0,1425	0,7969	0,1404	[1700;1800]
101	76	0,01505	0,0024	0,1458	0,2598	0,1607	[1800;1900]
101	104	0,0206	0,00455	0,1561	0,4833	0,1341	[1900;2000]
Average				0,17612	0,330595	0,1783	

Table 8. Clustering coefficient for 101 nodes graphs

N	n	t	tw	cc1	ccw	ccr	index
501	3065	0,02445	0,00555	0,1877	0,4587	0,1874	[2000;2500]
501	777	0,0062	0,00195	0,1744	0,1969	0,1814	[2500;3000]
501	686	0,00548	0,002584	0,14946	0,158765	0,154723	[3000;3500]
501	572	0,00457	0,002318	0,12082	0,087682	0,084218	[3500;4000]
501	529	0,00422	0,002686	0,1185	0,037947	0,118054	[4000;4500]
501	527	0,00421	0,003206	0,08511	0,022804	0,065669	[4500;5000]
501	894	0,00714	0,002249	0,19218	0,123907	0,19318	[5000;5500]
501	1074	0,00857	0,001852	0,14651	0,300957	0,139348	[5500;6000]
501	705	0,00563	0,001676	0,1382	0,165122	0,147377	[6000;6500]
501	524	0,00418	0,00318	0,11794	0,038495	0,126195	[6500;7000]
501	636	0,00508	0,001801	0,12377	0,200421	0,11033	[7000;7500]
501	1090	0,0087	0,001005	0,11904	0,319126	0,114721	[7500;8000]
501	1120	0,00894	0,001216	0,15544	0,350656	0,152527	[8000;8500]
501	904	0,00722	0,001369	0,14519	0,39172	0,133987	[8500;9000]
501	810	0,00647	0,001124	0,15784	0,304261	0,160854	[9000;9500]
501	801	0,0064	0,001165	0,15602	0,493368	0,166162	[9500;10000]
501	1179	0,00941	0,001126	0,17866	0,562911	0,17728	[10000;10500]
501	1067	0,00852	0,001218	0,15718	0,495992	0,152236	[10500;11000]
501	1274	0,01017	0,002001	0,17391	0,814473	0,184845	[11000;11500]
501	1063	0,00849	0,00116	0,13468	0,340036	0,131701	[11500;12000]
501	1125	0,00898	0,00169	0,18882	0,716779	0,174497	[12000;12500]
501	1207	0,00964	0,001725	0,16601	0,553342	0,160839	[12500;13000]
501	1051	0,00839	0,001173	0,15147	0,518364	0,137774	[13000;13500]
501	1159	0,00925	0,001833	0,15833	0,331788	0,156	[13500;14000]
501	1013	0,00809	0,001735	0,17002	0,406979	0,159665	[14000;14500]
501	1092	0,00872	0,001549	0,14707	0,801552	0,134988	[14500;15000]
501	971	0,00775	0,000985	0,15586	1,29571	0,156738	[15000;15500]
501	1051	0,00839	0,002468	0,13821	0,236818	0,145349	[15500;16000]
501	1124	0,00897	0,001388	0,16219	0,542103	0,158394	[16000;16500]
501	1131	0,00903	0,002479	0,19137	0,400309	0,196012	[16500;17000]
501	947	0,00756	0,001117	0,15	0,794527	0,149012	[17000;17500]
501	1220	0,00974	0,001763	0,15023	0,406813	0,148377	[17500;18000]
501	1212	0,00968	0,001821	0,12817	0,336995	0,123798	[18000;18500]
501	882	0,00704	0,001151	0,1423	0,638735	0,141008	[18500;19000]
501	1128	0,00901	0,001539	0,18856	0,592483	0,179411	[19000;19500]
501	971	0,00775	0,001103	0,14373	0,796134	0,144897	[19500;20000]
501	864	0,0069	0,001508	0,16114	0,823072	0,162207	[20000;20500]
501	965	0,0077	0,001389	0,15416	0,760639	0,158728	[20500;21000]
501	985	0,00786	0,001444	0,19987	0,702719	0,184956	[21000;21500]
501	1048	0,00837	0,00132	0,16376	0,474865	0,165303	[21500;22000]
501	963	0,00769	0,001694	0,13451	0,23656	0,136838	[22000;22500]
501	954	0,00762	0,001425	0,17492	0,499399	0,160631	[22500;23000]
501	974	0,00778	0,001435	0,20788	0,555642	0,190973	[23000;23500]
501	1135	0,00906	0,002672	0,221	0,381789	0,218923	[23500;24000]
501	960	0,00766	0,002164	0,24761	0,443693	0,237617	[24000;24500]
501	981	0,00783	0,002514	0,29669	0,405307	0,291925	[24500;25000]
501	1106	0,00883	0,002366	0,22168	0,483318	0,215775	[25000;25500]
501	1012	0,00808	0,001639	0,17511	0,313019	0,1737	[25500;26000]
501	1053	0,00841	0,001456	0,15195	0,559949	0,143513	[26000;26500]
501	1303	0,0104	0,002395	0,1685	0,426062	0,169681	[26500;27000]
501	1038	0,00829	0,002024	0,17713	0,331658	0,179077	[27000;27500]
501	1042	0,00832	0,002569	0,16318	0,226068	0,143864	[27500;28000]

N	n	t	tw	cc1	ccw	ccr	index
501	1071	0,00855	0,002738	0,22016	0,366151	0,229161	[28000;28500]
501	653	0,00521	0,001987	0,1662	0,149186	0,159599	[28500;29000]
501	952	0,0076	0,001538	0,16884	0,482454	0,147109	[29000;29500]
501	1017	0,00812	0,001484	0,15947	0,560944	0,151692	[29500;30000]
501	1015	0,0081	0,001396	0,19615	0,525879	0,206249	[30000;30500]
501	971	0,00775	0,001467	0,20498	0,549256	0,199566	[30500;31000]
501	1057	0,00844	0,001218	0,16385	0,546128	0,162632	[31000;31500]
501	945	0,00754	0,001182	0,15184	0,607291	0,161003	[31500;32000]
501	701	0,0056	0,001854	0,156	0,322801	0,16631	[32000;32500]
501	839	0,0067	0,001396	0,1215	0,399616	0,114021	[32500;33000]
501	884	0,00706	0,001219	0,17968	0,705455	0,191614	[33000;33500]
501	1071	0,00855	0,001742	0,21443	0,48156	0,204559	[33500;34000]
501	1108	0,00885	0,001681	0,19586	0,639113	0,204419	[34000;34500]
501	681	0,00544	0,003489	0,10551	0,09374	0,093311	[34500;35000]
501	929	0,00742	0,001862	0,16281	0,593517	0,16301	[35000;35500]
501	772	0,00616	0,001455	0,18837	0,574518	0,18712	[35500;36000]
501	870	0,00695	0,001178	0,15279	0,757621	0,156489	[36000;36500]
501	919	0,00734	0,001324	0,16299	0,509101	0,191466	[36500;37000]
501	1093	0,00873	0,001574	0,15366	0,418827	0,139058	[37000;37500]
501	863	0,00689	0,002019	0,26795	0,549482	0,26346	[37500;38000]
501	1037	0,00828	0,001318	0,15064	0,564194	0,14786	[38000;38500]
501	954	0,00762	0,001556	0,1584	0,664596	0,154632	[38500;39000]
501	745	0,00595	0,002018	0,16508	0,282668	0,157175	[39000;39500]
501	912	0,00728	0,001167	0,18761	0,962863	0,19971	[39500;40000]
501	846	0,00675	0,001494	0,17783	0,605513	0,181785	[40000;40500]
501	820	0,00655	0,00138	0,17042	0,775921	0,161524	[40500;41000]
501	916	0,00731	0,001947	0,14809	0,363551	0,151156	[41000;41500]
501	946	0,00755	0,001113	0,14034	0,472739	0,122379	[41500;42000]
501	867	0,00692	0,000896	0,15184	0,617221	0,150279	[42000;42500]
501	1008	0,00805	0,000932	0,14481	0,609448	0,148538	[42500;43000]
501	1249	0,00997	0,001475	0,17211	0,519738	0,177122	[43000;43500]
501	844	0,00674	0,000886	0,1355	0,596559	0,14027	[43500;44000]
501	781	0,00624	0,001232	0,18281	0,763873	0,19394	[44000;44500]
501	873	0,00697	0,000944	0,16431	0,790968	0,1637	[44500;45000]
501	915	0,00731	0,001127	0,1863	1,053668	0,186993	[45000;45500]
501	1087	0,00868	0,003071	0,17788	0,45184	0,174185	[45500;46000]
501	795	0,00635	0,000999	0,15592	0,672109	0,150983	[46000;46500]
501	691	0,00552	0,000854	0,17302	0,614588	0,189682	[46500;47000]
501	989	0,0079	0,001521	0,16826	0,4561	0,166683	[47000;47500]
501	548	0,00438	0,002391	0,12232	0,136275	0,130124	[47500;48000]
501	639	0,0051	0,001321	0,1339	0,22064	0,130531	[48000;48500]
501	687	0,00549	0,000792	0,16135	1,105902	0,161391	[48500;49000]
501	798	0,00637	0,001114	0,16547	0,437163	0,172673	[49000;49500]
Average				0,16496	0,488718	0,16303	

Table 9. Clustering coefficient for the 501 nodes graphs

Appendix 2. Source MATLAB code for clustering coefficient calculation

1) A few commands for taking the sub matrix of the matrix

```
load sim.mat %loading the large 50000 nodes matrix
mat=sim
sim1000_1100=mat(1000:1100,1000:1100)% taking the 100 nodes large sub matrix
save sim1000_1100.mat sim1000_1100 %saving the sub matrix to the mat file
```

2) The function for ccw, cct , cc1 and cct(tt) calculation

```
function [T] = clust_1(r,c,non,tt)%the function which returns the vector T
%whose elements are cc1, ccr and cct with the threshold like tt
%the input parameters are r-vector of row numbers of non zero element of
%the matrix, c-vector of column numbers of non zero element of the matrix
%and non- vector of non zero elements
n1=length(c);% the number of non zero elements
n2=n1-1;
trip=[0];%the vector of connected triples,
%each element is the product of the two weights of the links in the triple
triangles=[0];%the vector of the triangles
%each element is the product of three weights of the links in the triangle
k=1;
kt=1;
for i=1:n2 %for all the nodes
    i1=i+1;
    while (i1<=n1) && (c (i)==c(i1))% if nodes i and i1 are linked to the
        %same node
        non1=non (i)*non(i1);%then calculate the product of the link weights
        trip(k)=non1;% and add the triple to the triple vector
        for i2=1:n1 %check this triple if it is triangle
            x1=(c(i2)==r(i)) && (r(i2)==r(i1)); %if nodes i and i1 are linked
            t=false;
            if x1
                non2=non1*non(i2);% calculate the product of the three links
                non2=non2*10000000000;
                non2=round (non2);
                non2=non2/10000000000;
                i5=kt-1;
                if t==false
                    triangles(kt)=non2;% add the triangle to triangle vector
                    kt=kt+1;
                end
            end
        end
        k=k+1;
        i1=i1+1;
    end
end
n3=length(trip);% the number of connected triples
sumtr=0;
for i=1:n3
    sumtr=sumtr+trip(i);% sum of all the connected triples
end
n3=length(triangles);% the number of triangles
sumtriang=0;
for i=1:n3
    sumtriang=sumtriang+triangles(i);% sum of all the triangles
end
```

```
s=0;
for i=1:n1
    s=s+non(i);
end
%trip
%sumtr
%triangles
%sumtriang
clustOnes=length(triangles)/length(trip);% calculating of cc1
s=s/n1;
clustW=sumtriang/(s*sumtr);% calculating of ccw
for i=1:n1% deleting of all the weights that are smaller than tt
    if non (i)<tt
        c(i)=0;
        r(i)=0;
        non(i)=0;
    end
end
cc=nonzeros(c);
rr=nonzeros(r);
nonn=nonzeros(non);
s1=0;
for i=1:length(nonn)
    s1=s1+nonn(i);
end
s1=s1/length(nonn);
for i=1:length(cc)
    FFmatT(rr(i),cc(i))=nonn(i);% forming the redused matrix
end
%FFmatT
%spy(FFmatT)
n1=length(cc);
n2=n1-1;
k=1;
kt=1;
sumtr1=0;
triangles1=[0];
for i=1:n2 % calculation of cct
    i1=i+1;
    while (i1<=n1) && (cc (i)==cc(i1))
        sumtr1=sumtr1+1;
        non1=nonn(i)*nonn(i1);
        for i2=1:n1
            x1=(cc(i2)==rr(i)) && (rr(i2)==rr(i1));
            t=false;
            if x1
                non2=non1*nonn(i2);
                non2=non2*10000000000;
                non2=round (non2);
                non2=non2/10000000000;
                i5=kt-1;
                if t==false
                    triangles1(kt)=non2;
                    kt=kt+1;
                end
            end
        end
        k=k+1;
        i1=i1+1;
    end
end
%sumtr1
sumtriang1=length(triangles1);
if sumtriang1==1 && triangles1(1)==0
```

```
    sumtriang1=0;
end
%sumtriang1
clustT=sumtriang1/sumtr1;
T=[clustOnes clustW clustT]% resulting vector
```

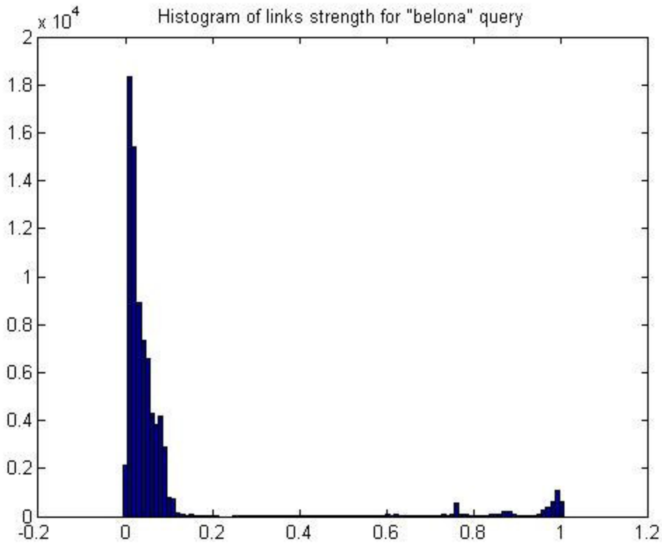
3) The function which calls the previous function for both random and original graph and calculates two densities.

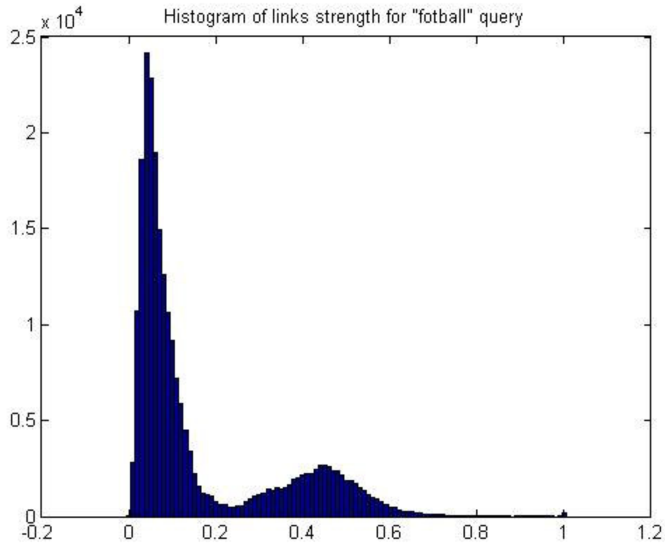
```
function [r] = clustChek(mat,place,tt)% the function which calls the previous
function,
% for original and for random graphs, calculates also two densities,
% and writes the result to the xls file.
% mat-the sim. matrix, place- the coordinates in the xls file were to write
% and tt- threshold.
Fmat=spconvert (mat);% convert the matrix to the sparse form
[r1,c1,non1]=find(Fmat);%take column, row and value vectors of the non zero
elements
n=length(non1);
for i=1:length(non1)% make the random graph
    FmatR(r1(i),c1(i))=rand;
end
FmatR(n,n)=0;
FFmatR= (FmatR+FmatR');% make the random matrix symmetric
nonR=nonzeros(FFmatR);%take the vector of non zero elements from the random
graph
N=length(Fmat);%number of elements
n=length(non1)/2;%number of links
Nmax=N*(N-1)/2;%the maximum possible links
Tett=n/Nmax;% density
sW=0;
for i=1:length(non1)%sum all weights
    sW=sW+non1(i);
end
TettW=sW/(2*Nmax);%weighted density
T=clust_1(r1,c1,non1,tt);% call the previous function for original graph
cT=T(3);% read cct
T1=clust_1(r1,c1,nonR,tt);% call the previous function for the random graph
RR=T1(2);
Result=[N n Tett TettW T(1) T(2) RR cT];% make the vector of results
r=xlswrite('Result.xls', Result,'RR1',place);% write the result vector to the
file
end
```

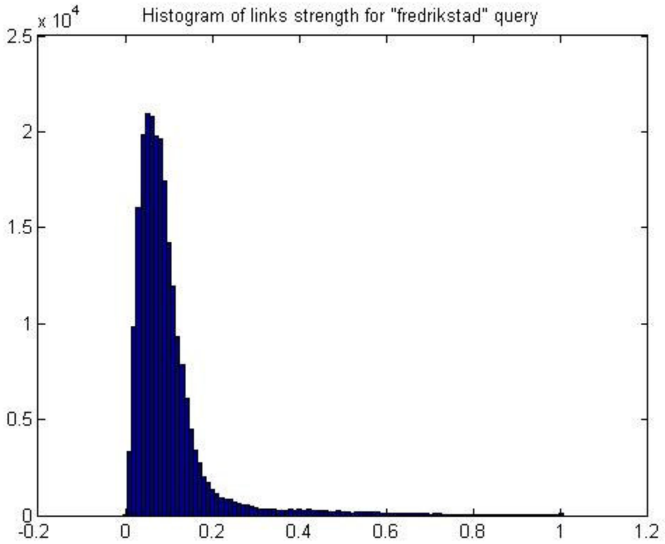
4) The three commands for each mat file, which stores the matrix.

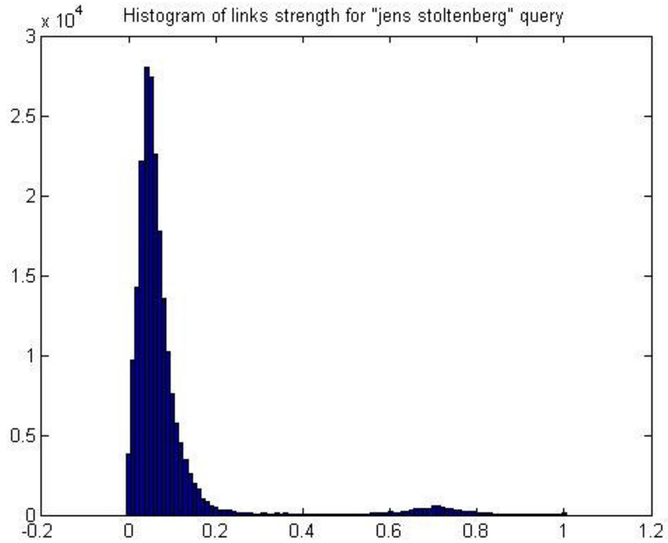
```
load sim22000_22500.mat; % read the matrix from the mat file
mat=sim22000_22500;% make the matrix variable mat
e=clust(mat,'A73',0.1);% call the previous function
```

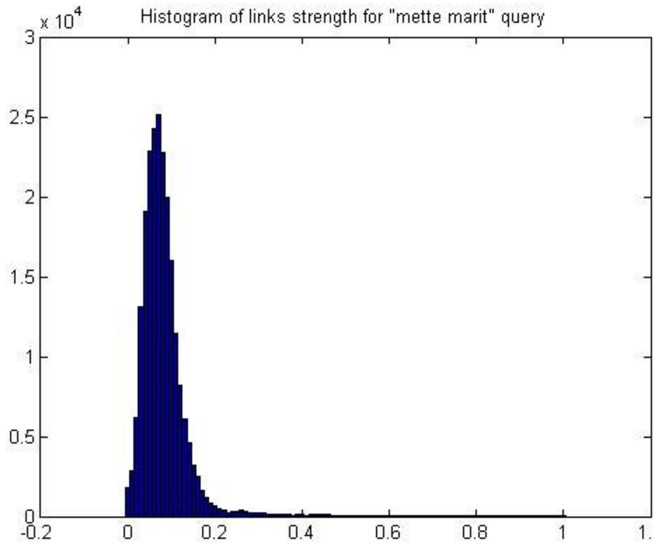
Appendix 3. Histograms of link strengths for similarity graphs

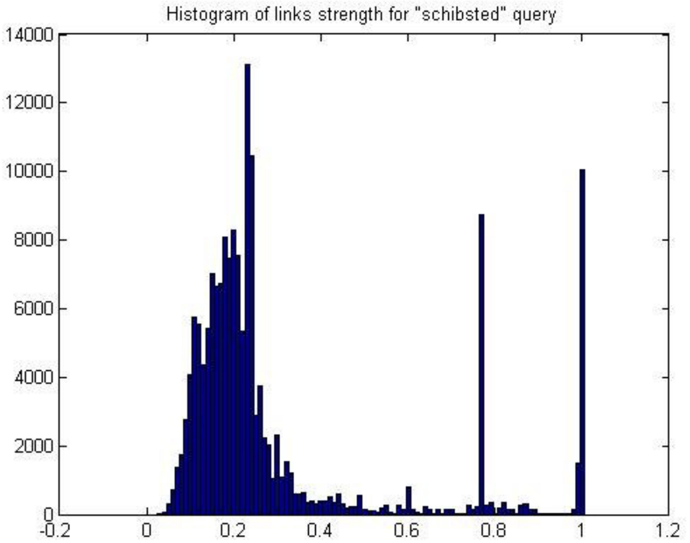


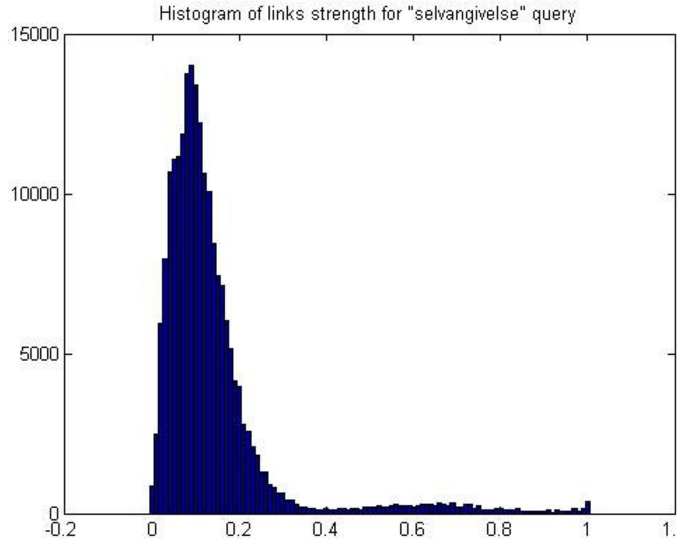




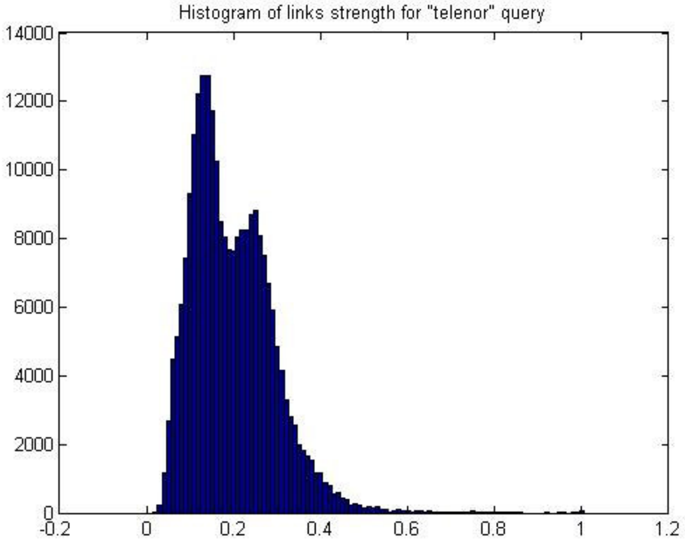




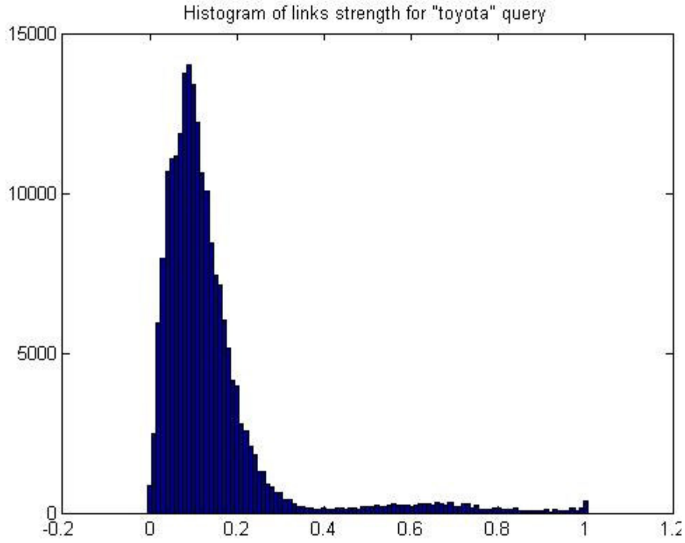




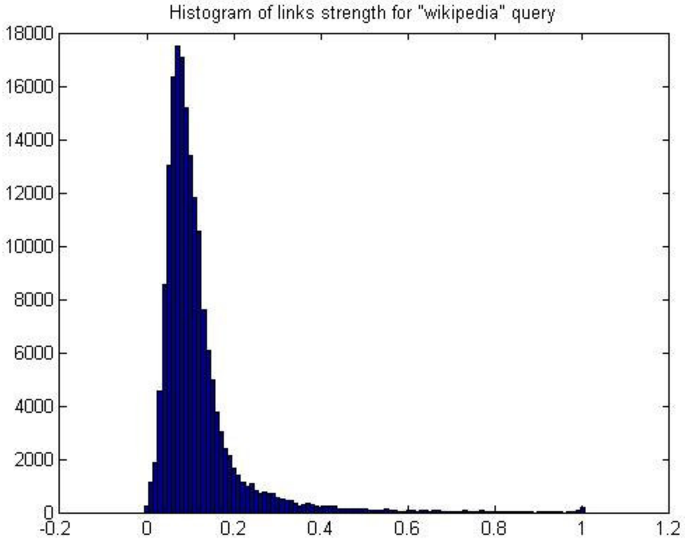
Histogram of links strength for "selvangivelse" query



Histogram of links strength for "telenor" query

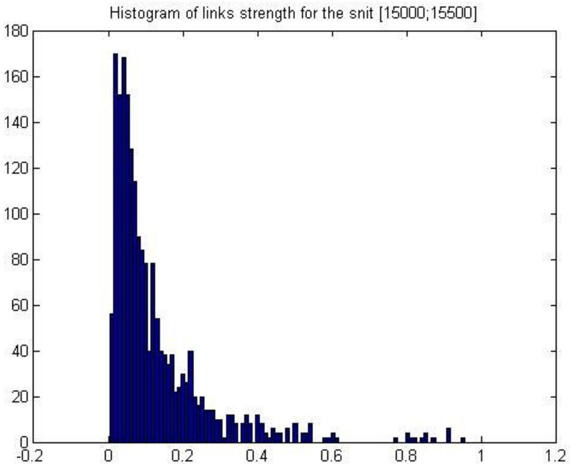


Histogram of links strength for "toyota" query

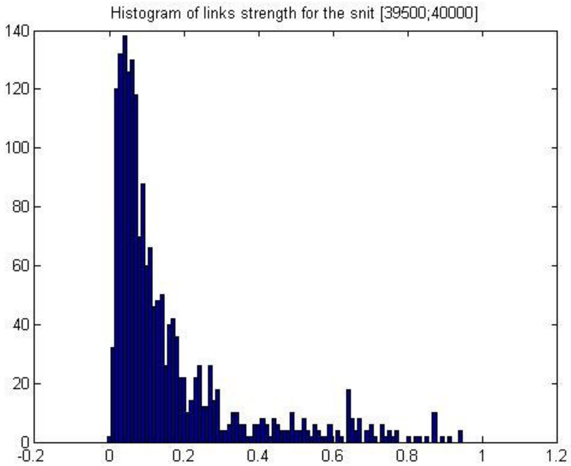


Histogram of links strength for "wikipedia" query

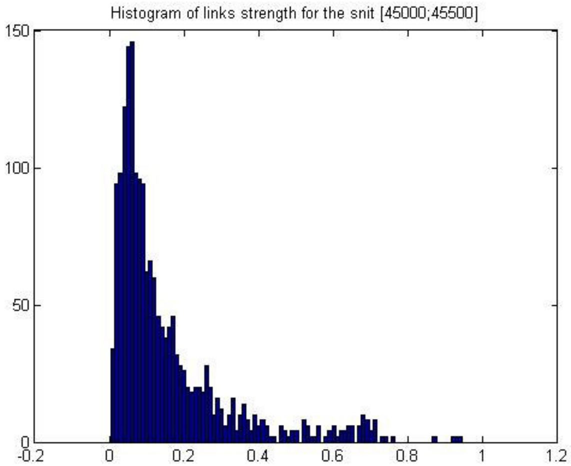
Appendix 4. Histograms for the 'slice' graphs.



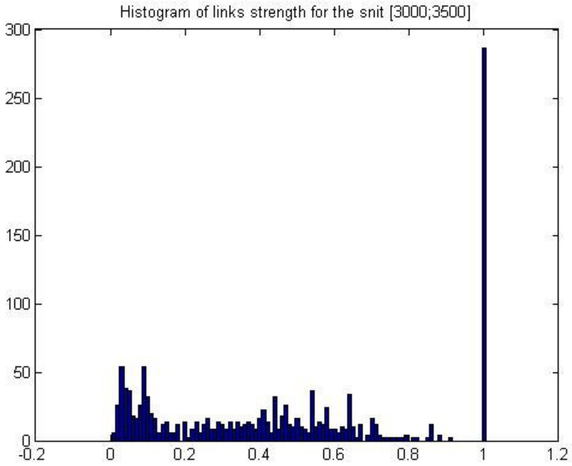
ccw=1,29571



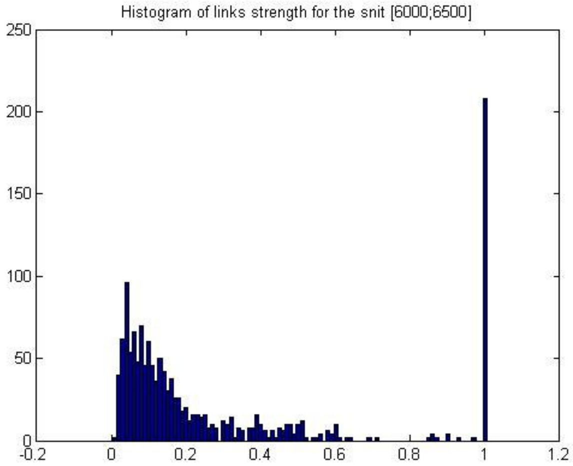
ccw=0,962863



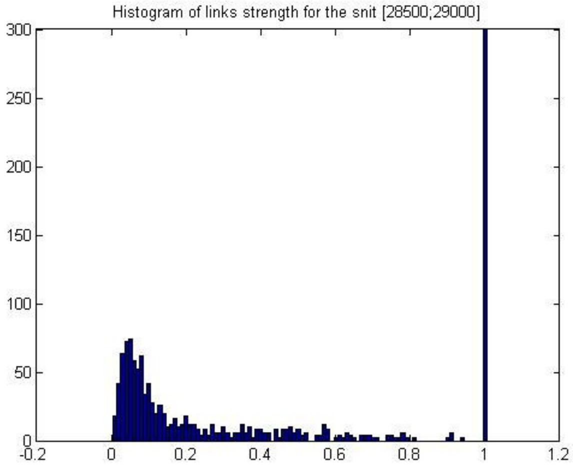
ccw=1,053668



ccw=0,087682

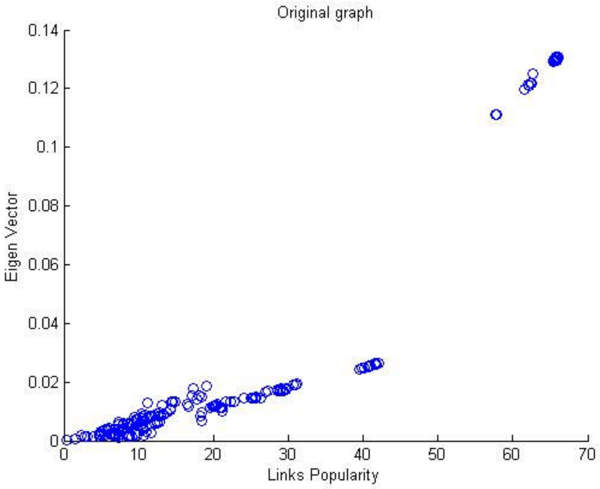


ccw=0,165122

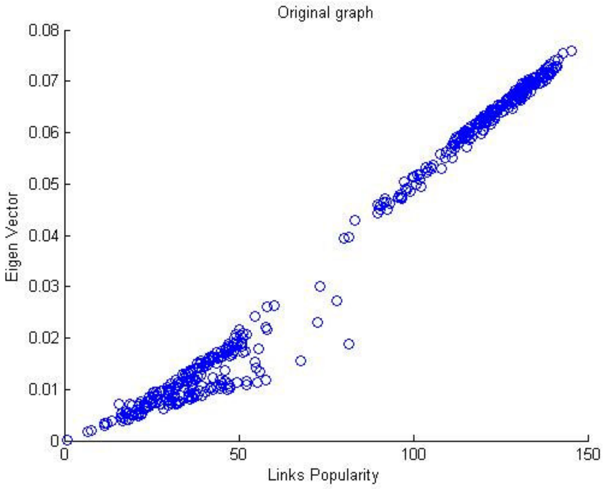


ccw=0,149186

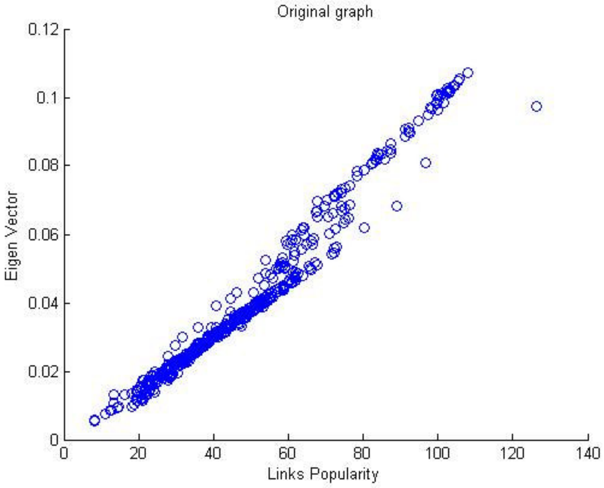
Appendix 5. Scatter plots of LP and EVC for the similarity graphs



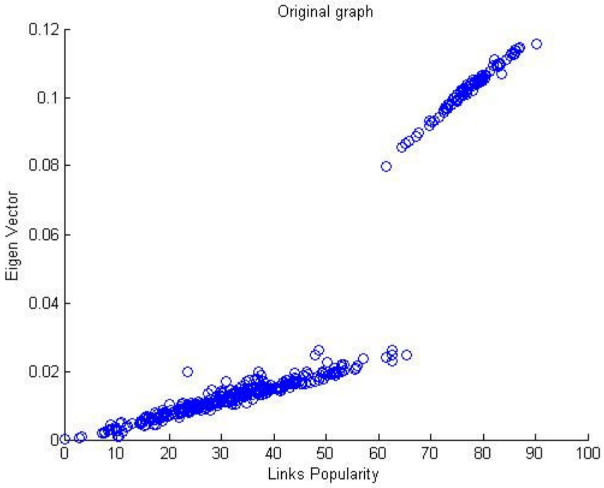
"belona"



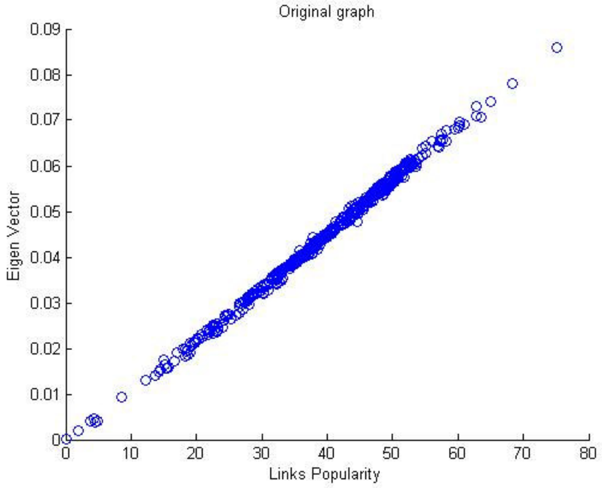
"fotball"



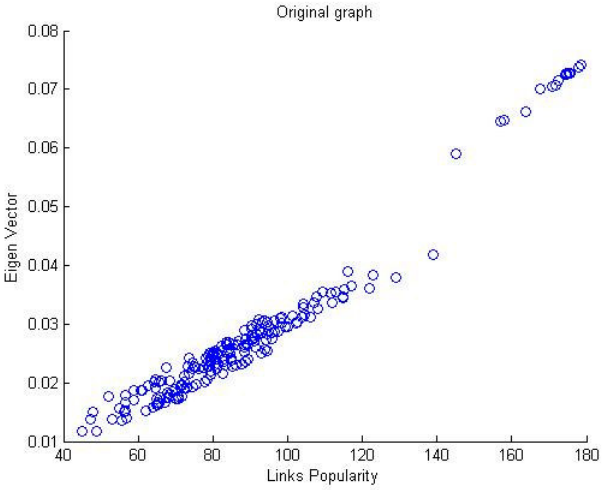
"fredrikstad"



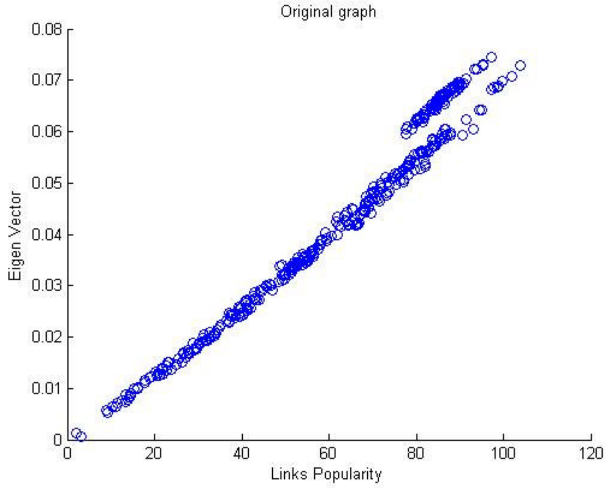
"jens stoltenberg"



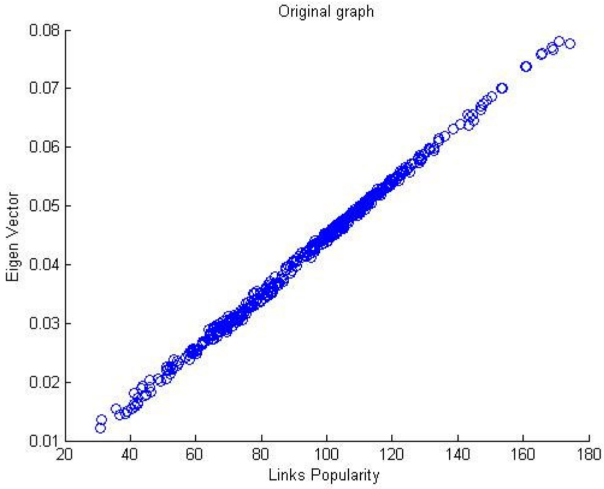
"mette marit"



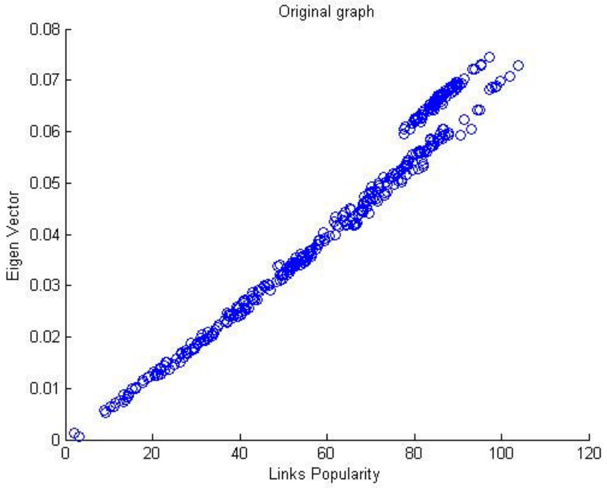
"schibsted"



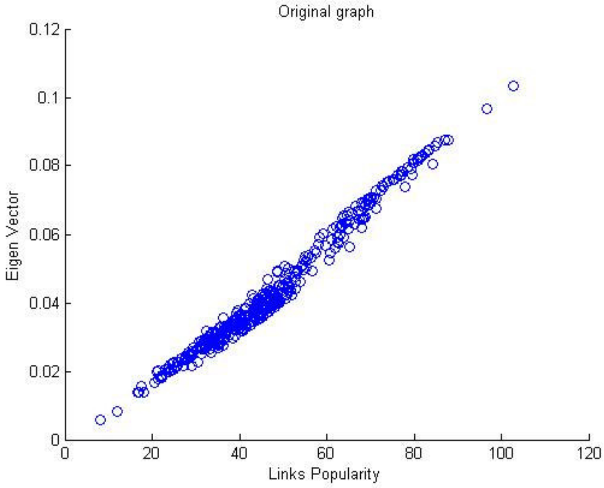
"selvangivelse"



"telenor"



"toyota"



"wikipedia"

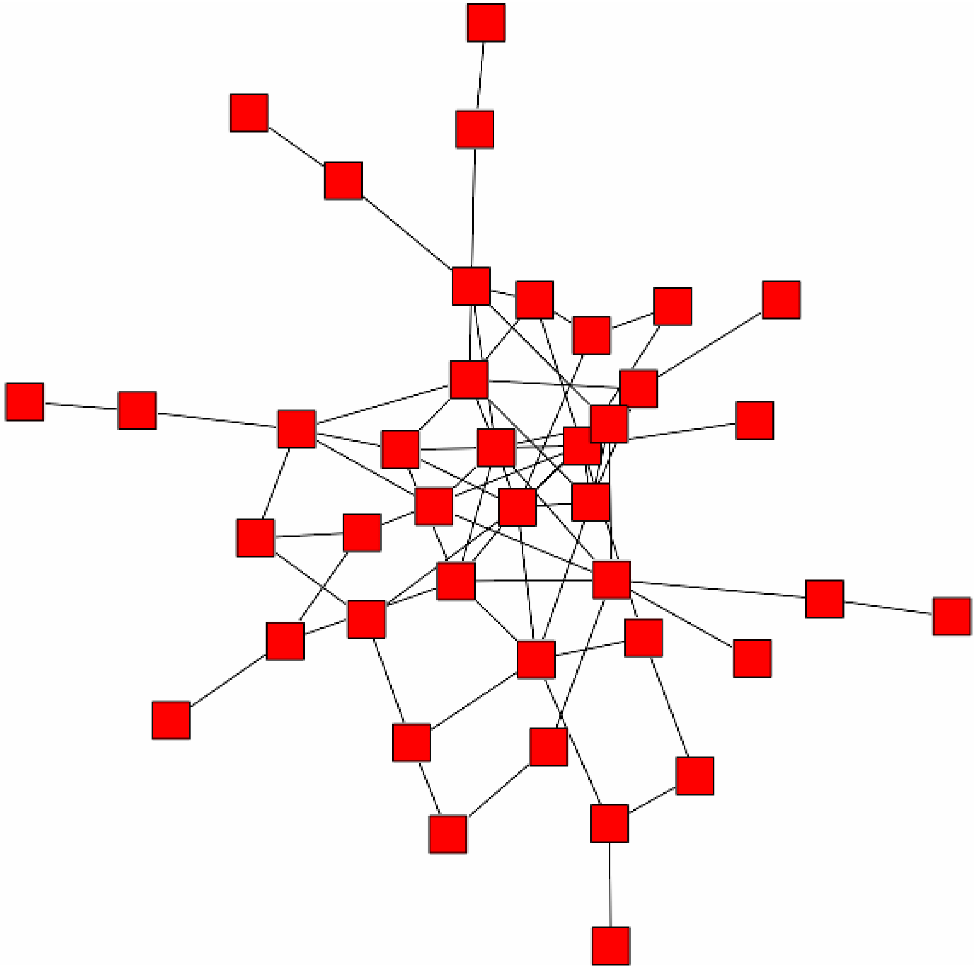
Appendix 6. Source code for LP and EVC calculation

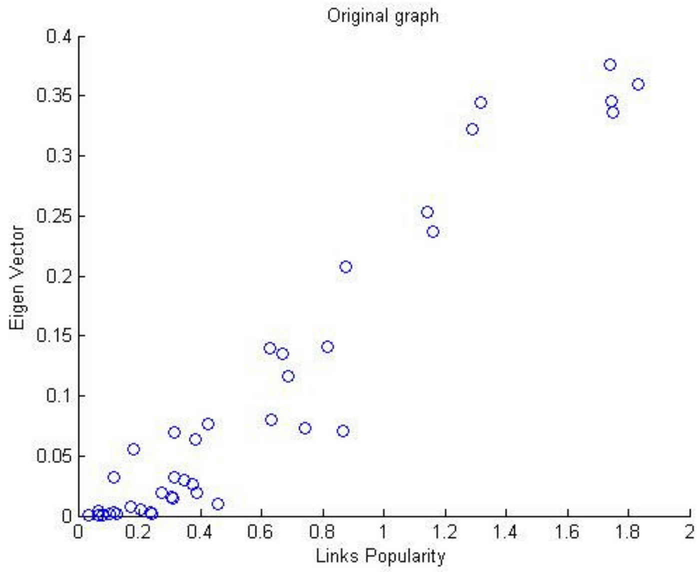
```
function [LP,V,clustW,Cor,N,matrix,Tett,TettW,n]=comp(mat)%function that
%calculates Link popularity vector(LP), EVC(V) and correlation coefficient
%(Cor) between EVC and LP of the largest component of the matrix
Fmat=spconvert (mat);
n =length (Fmat);
[r1,c1,non1]=find(Fmat);
Fmat (n,n)=0;
FFmat= (Fmat+Fmat');
A=sparse(FFmat);
[scc scc_sizes]=components(A)[12];%find out the components, scc- vector
%of numbers of components (if the node i is in a component j then
%scc(i)==j) and scc_sizes- number of nodes in the component
max=1;
for i=1:length(scc_sizes)% find the largest component
    if scc_sizes(i)>max
       max=scc(i);
    end
end
nod=find(scc==max);%find the numbers of nodes of the largest component
clear matrix;
for i=1:length(nod)%make the similarity matrix of the component
    for j=1:length(nod)
        matrix(i,j)=FFmat(nod(i),nod(j))/2;
    end
end
[rr,cc,nonn]=find(matrix);
for i=1:length(nonn)% make the random component matrix
    matrixR(rr(i),cc(i))=rand;
end
[rr,cc,nonnR]=find(matrixR);
N=length(matrix);%number of nodes
n=length(nonn)/2;%number of the links
Nmax=N*(N-1)/2; %maximum possible links
Tett=n/Nmax;% density of the component
sW=0;
for i=1:n
    sW=sW+nonn(i);
end
TettW=sW/Nmax;% weighted density of the component
sWR=0;
for i=1:n
    sWR=sWR+nonnR(i);
end
TettWR=sWR/Nmax;% weighted density of the random component
clustW=clust_1LP(rr,cc,nonn)% clustering coefficient of the component
clustWR=clust_1LP(rr,cc,nonnR);%ccw of the random component
[LP,V]=LP_V(matrix);
[LPR,VR]=LP_V(matrixR);
V=abs(V);
scatter(LP,V);% make the graphical representation: x-LP, y-EVC
title('Original graph')
xlabel('Links Popularity');
ylabel('Eigen Vector');
Cor=corrcoef(LP,V)% correlation coefficient
CorR=corrcoef(LPR,VR);
end
```

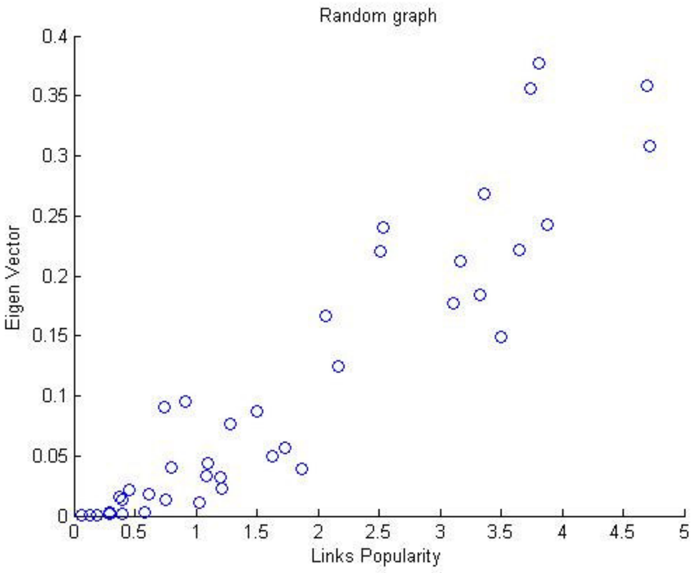
Appendix 7. Correlations of LP and EVC for ‘slice’ graphs

Here we used Tulip software to get pictures of the graphs [13].

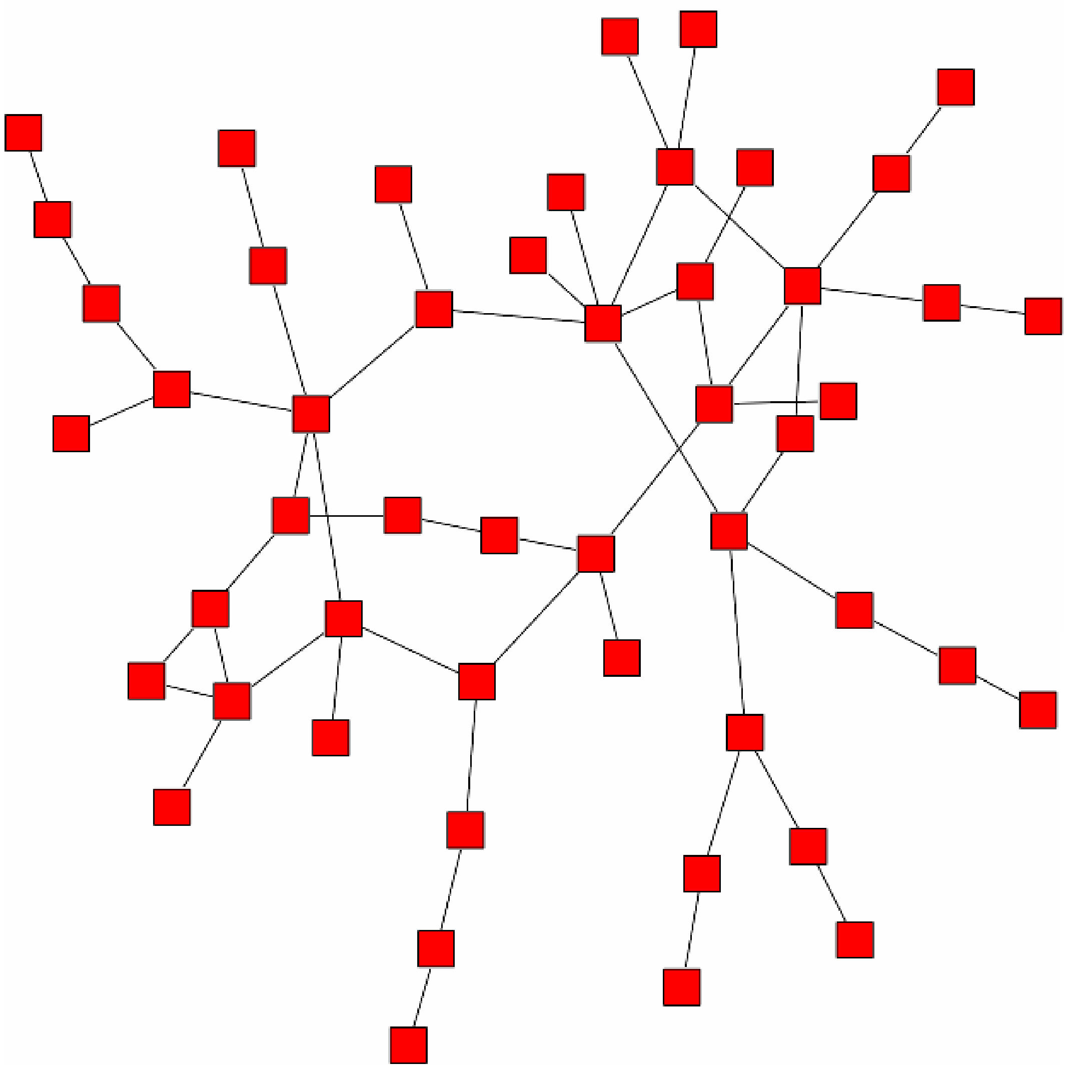
index	N of component	ccw of component	correlation	t	tw	n
[500;600]	40	0,310115746	0,962296	0,091026	0,016227	71
[500;600]rand	40	0,227517651	0,928454	0,091026	0,046485	71

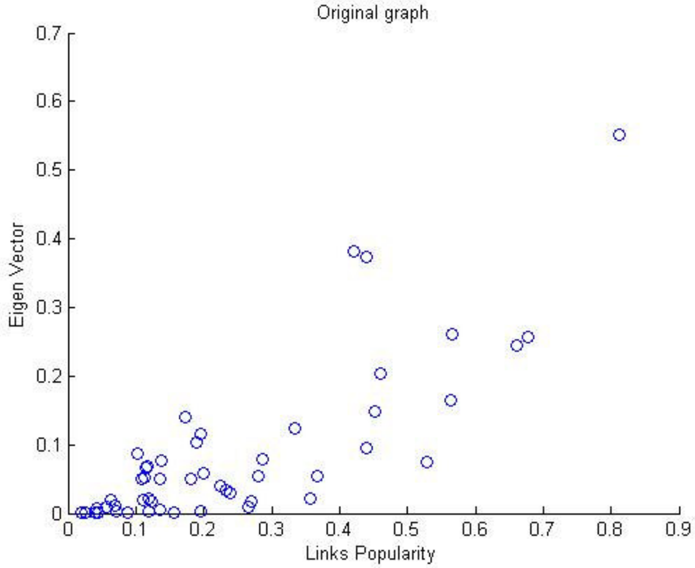


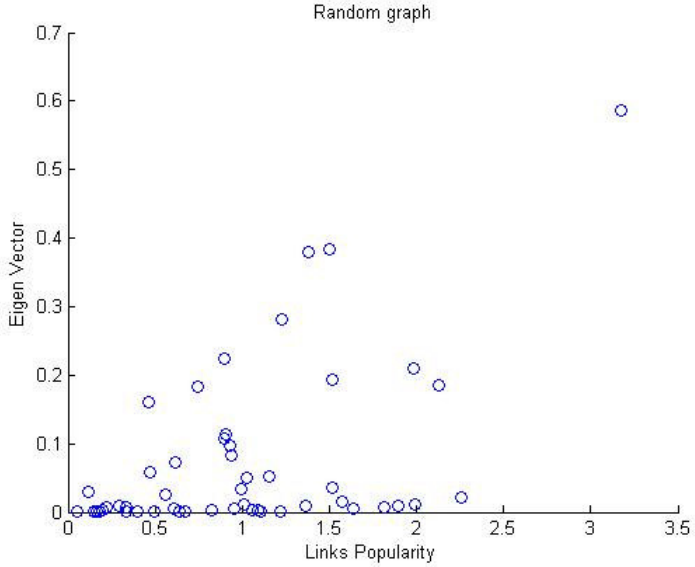




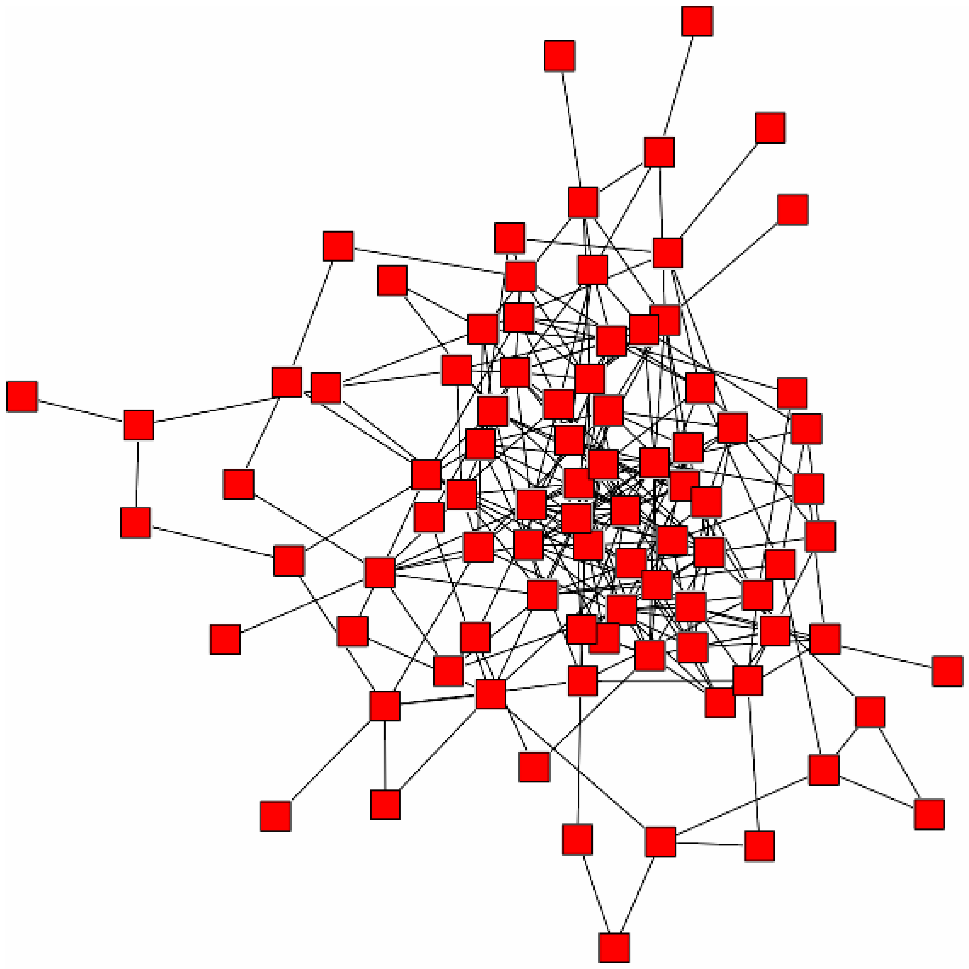
index	N of component	ccw of component	correlation	t	tw	n
[700;800]	50	0,000393004	0,791615	0,044898	0,004653	55
[700;800]rand	50	0,029001219	0,507246	0,044898	0,022221	55

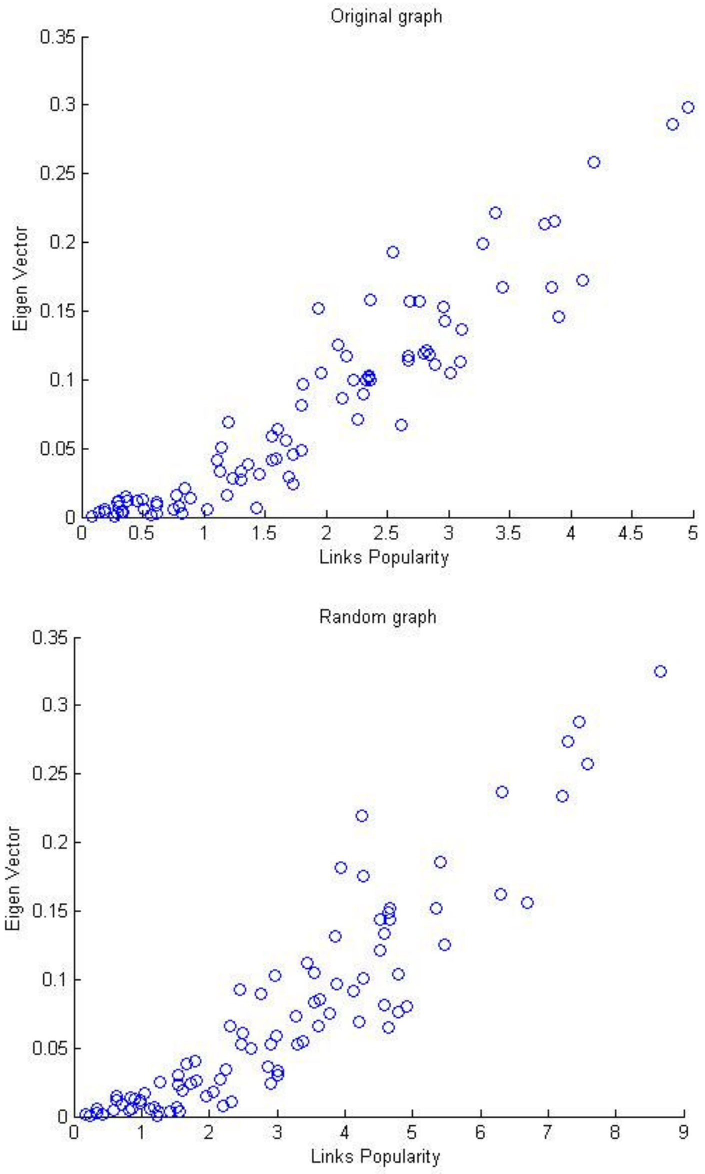




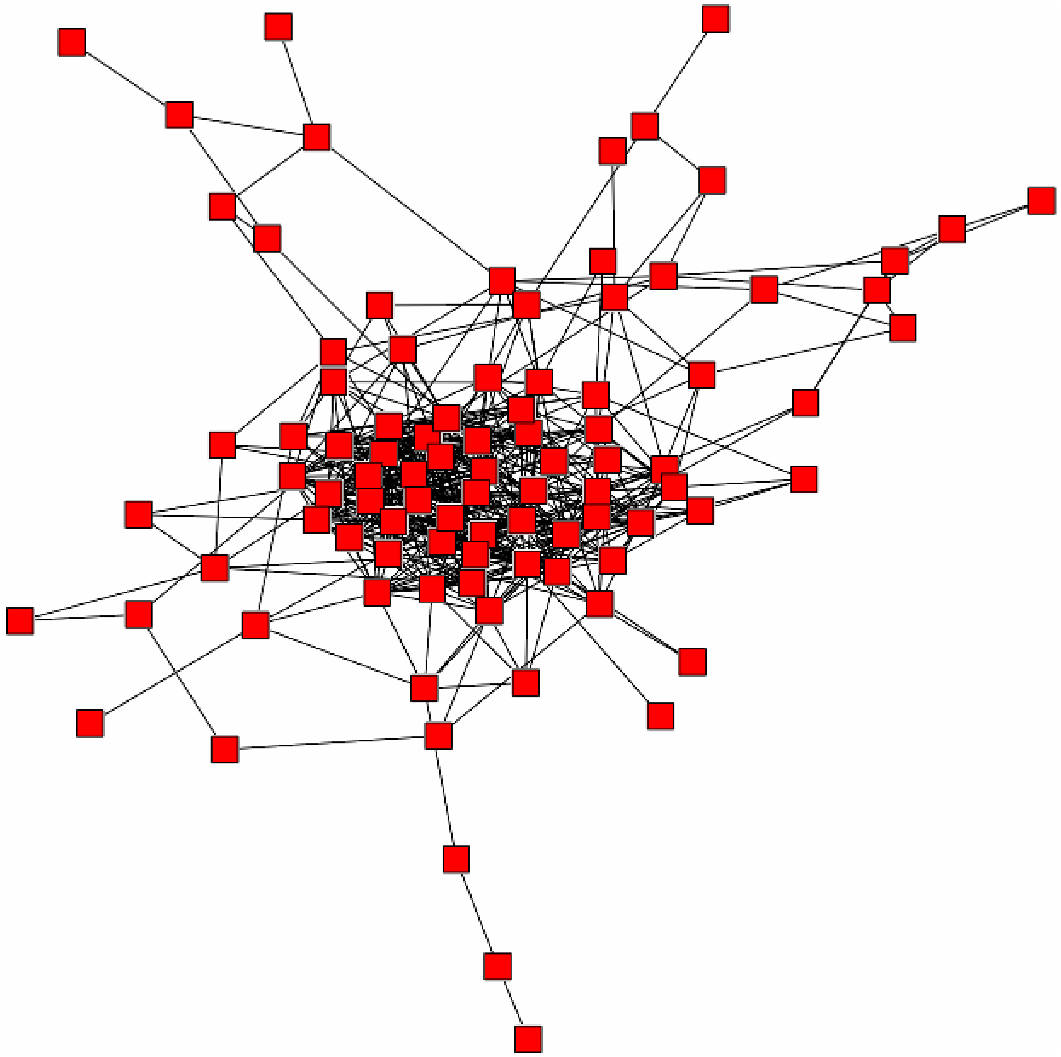


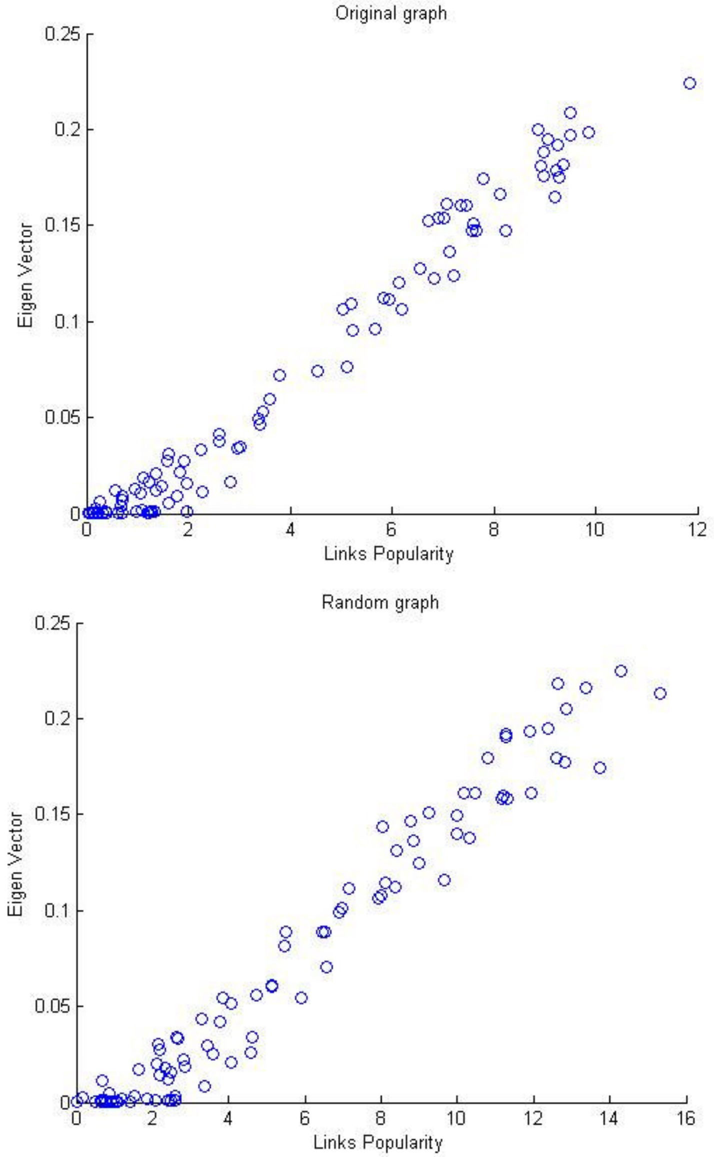
index	N of component	ccw of component	correlation	t	tw	n
[1000;1100]	89	0,233880974	0,934325	0,067416	0,021048	264
[1000;1100]rand	89	0,214769852	0,914391	0,067416	0,034555	264



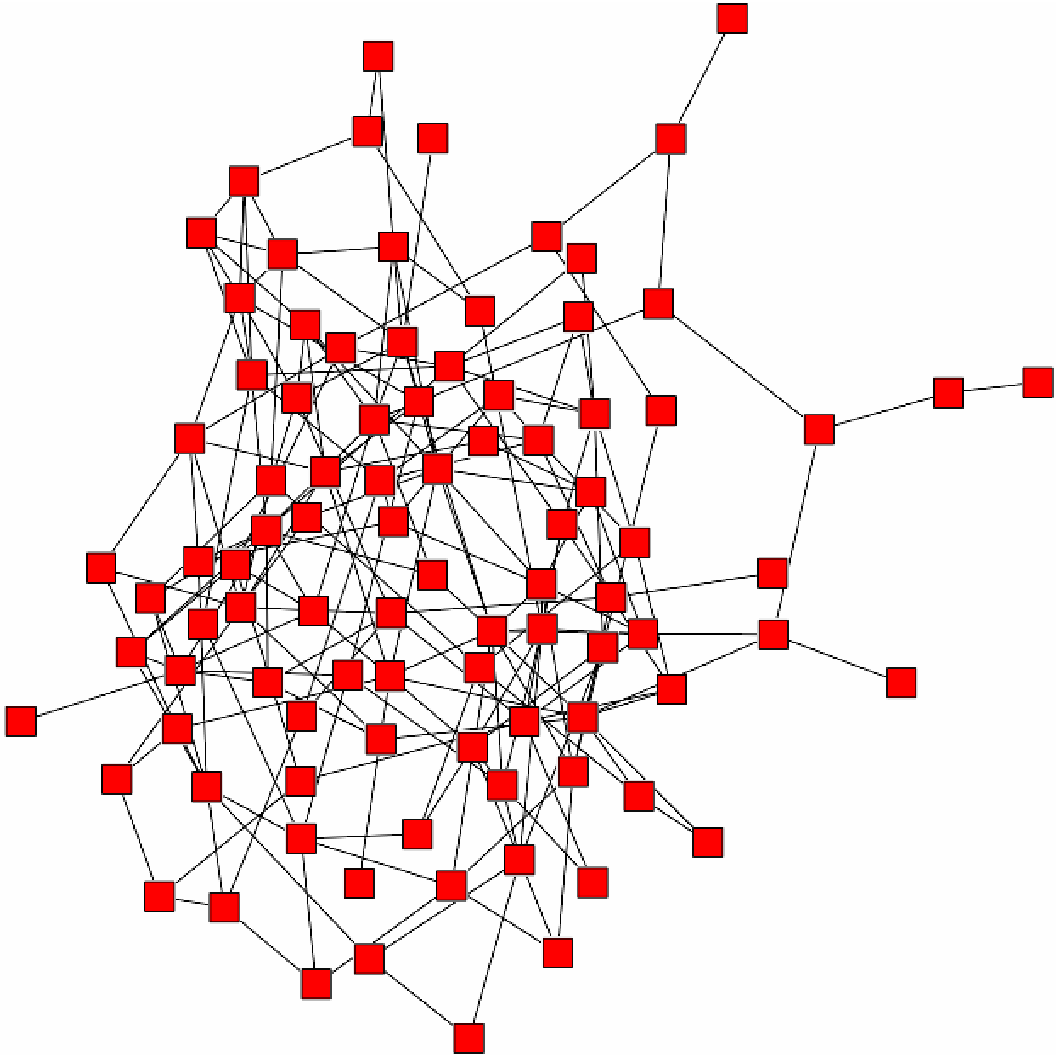


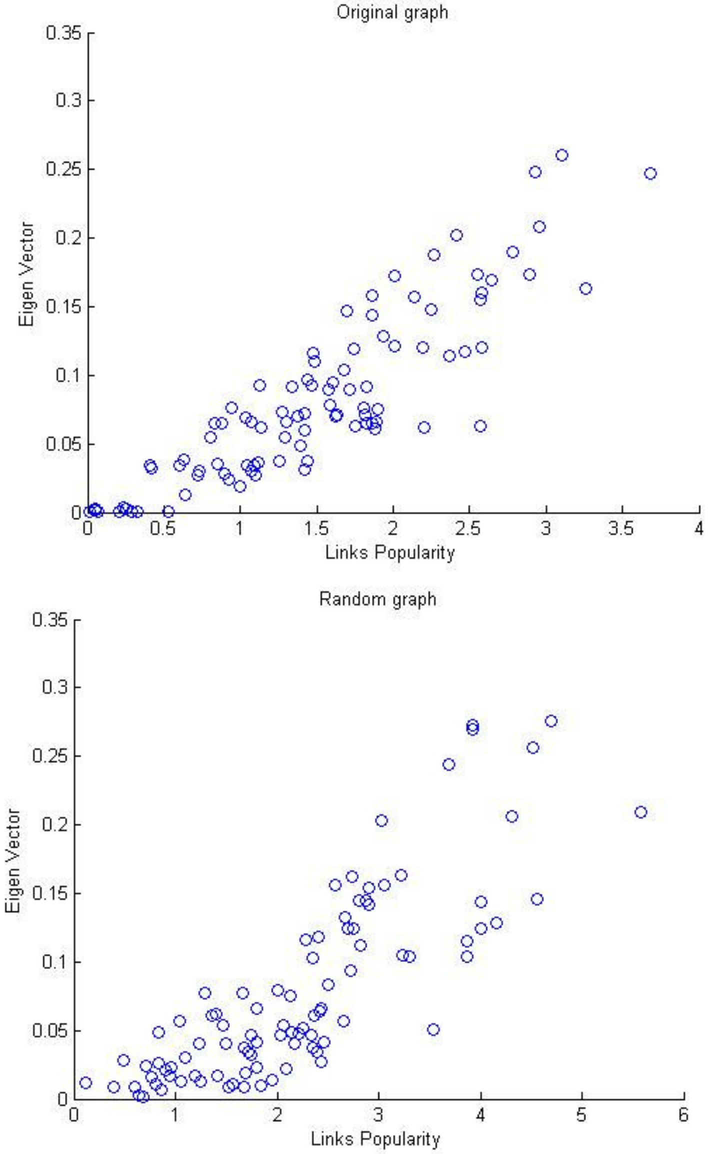
index	N of component	ccw of component	correlation	t	tw	n
[1200;1300]	93	0,443101543	0,986663	0,122253	0,045135	523
[1200;1300]rand	93	0,392977451	0,980712	0,122253	0,063107	523



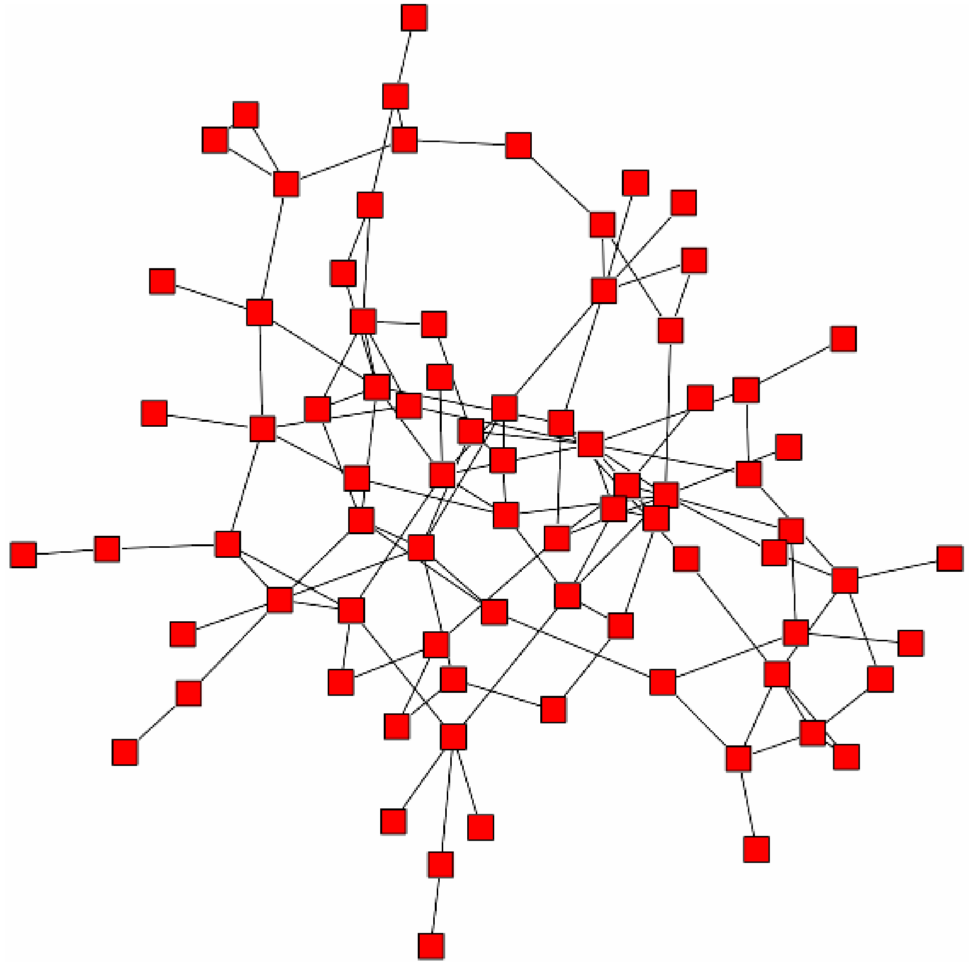


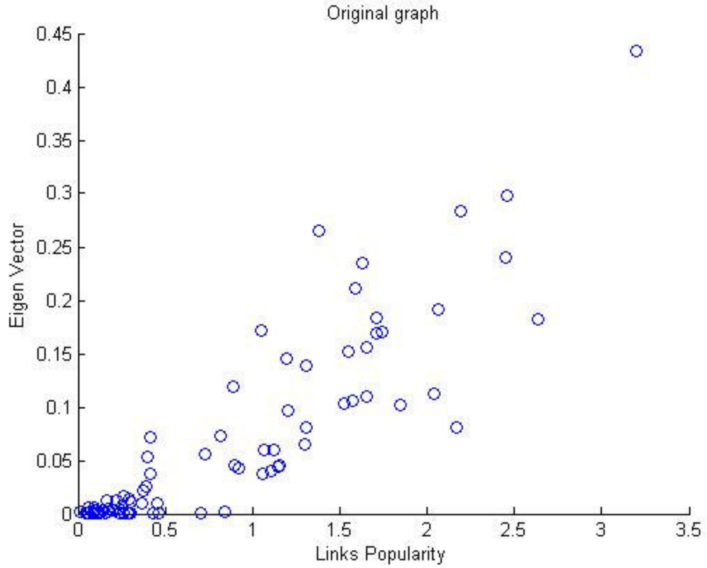
index	N of component	ccw of component	correlation	t	tw	n
[1300;1400]	93	0,163231938	0,878905	0,04792	0,016194	205
[1300;1400]rand	93	0,114210458	0,822305	0,04792	0,023692	205

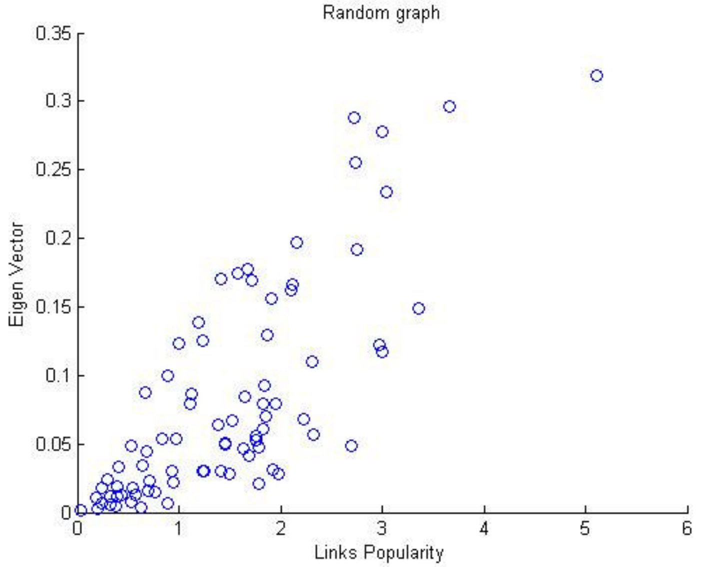




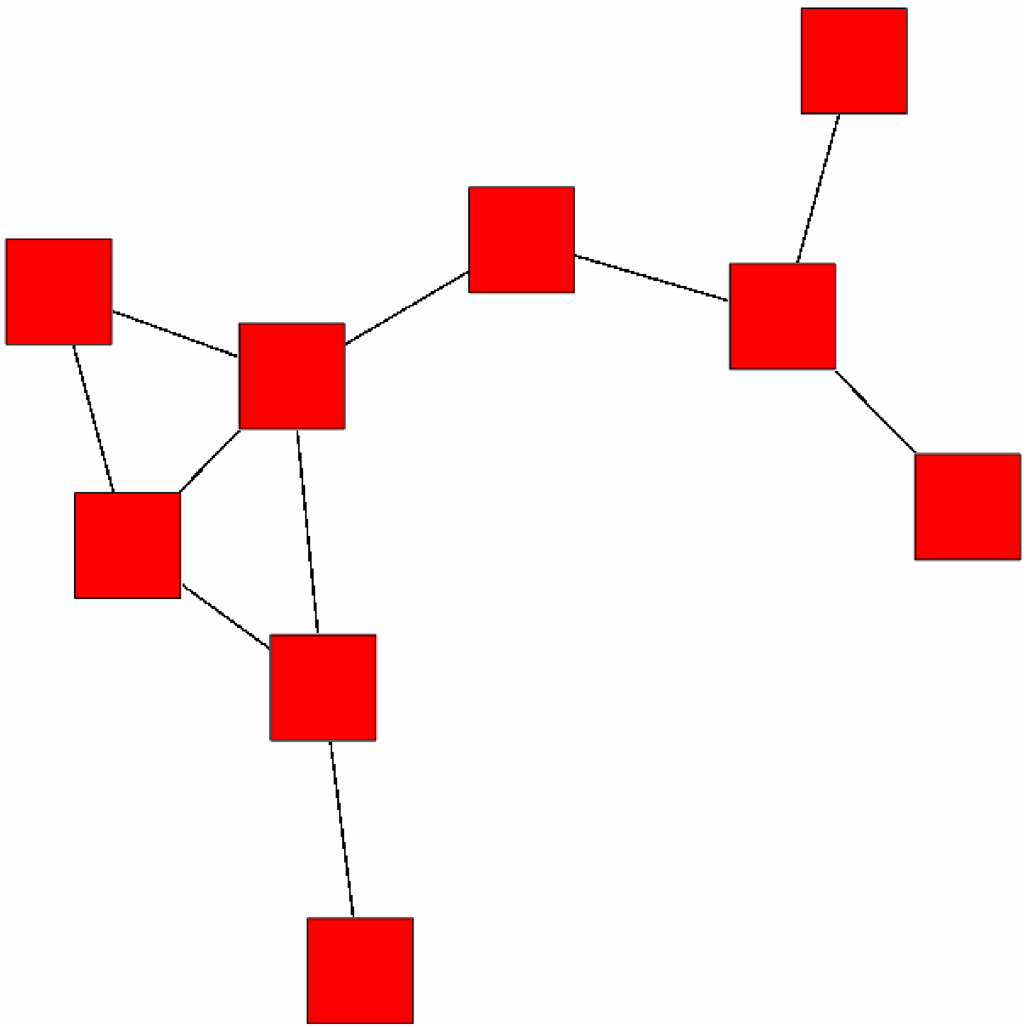
index	N of component	ccw of component	correlation	t	tw	n
[1500;1600]	80	0,172180569	0,880754	0,038291	0,012217	121
[1500;1600]rand	80	0,106511738	0,773038	0,038291	0,017408	121

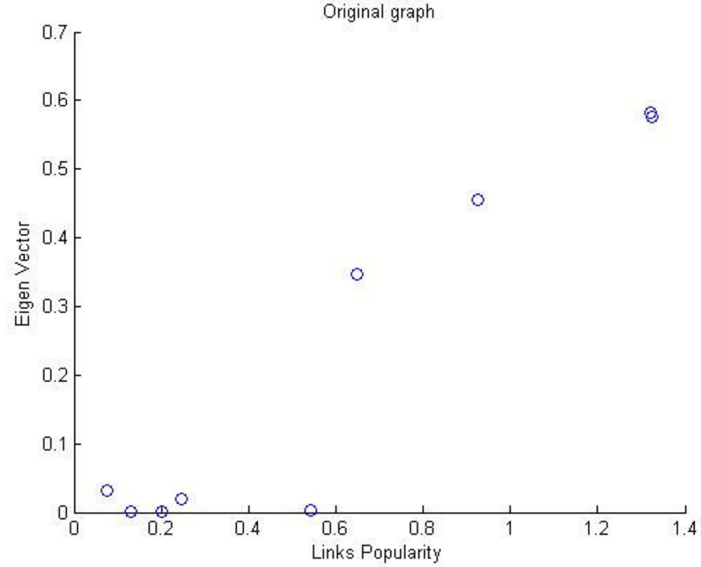


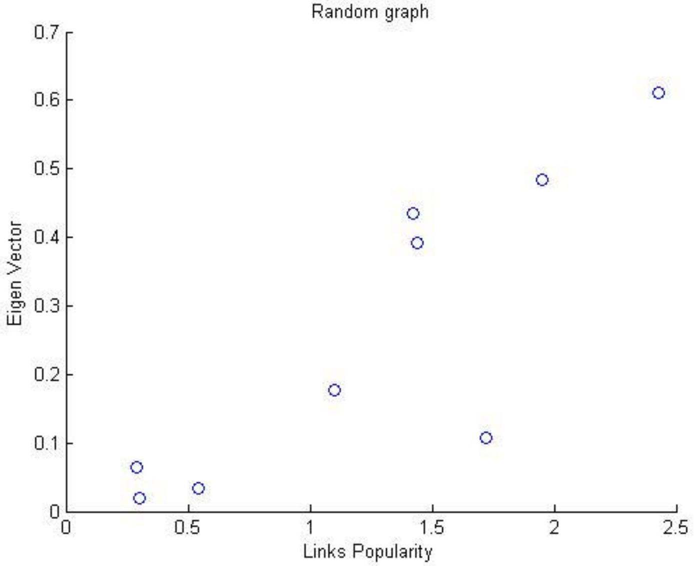




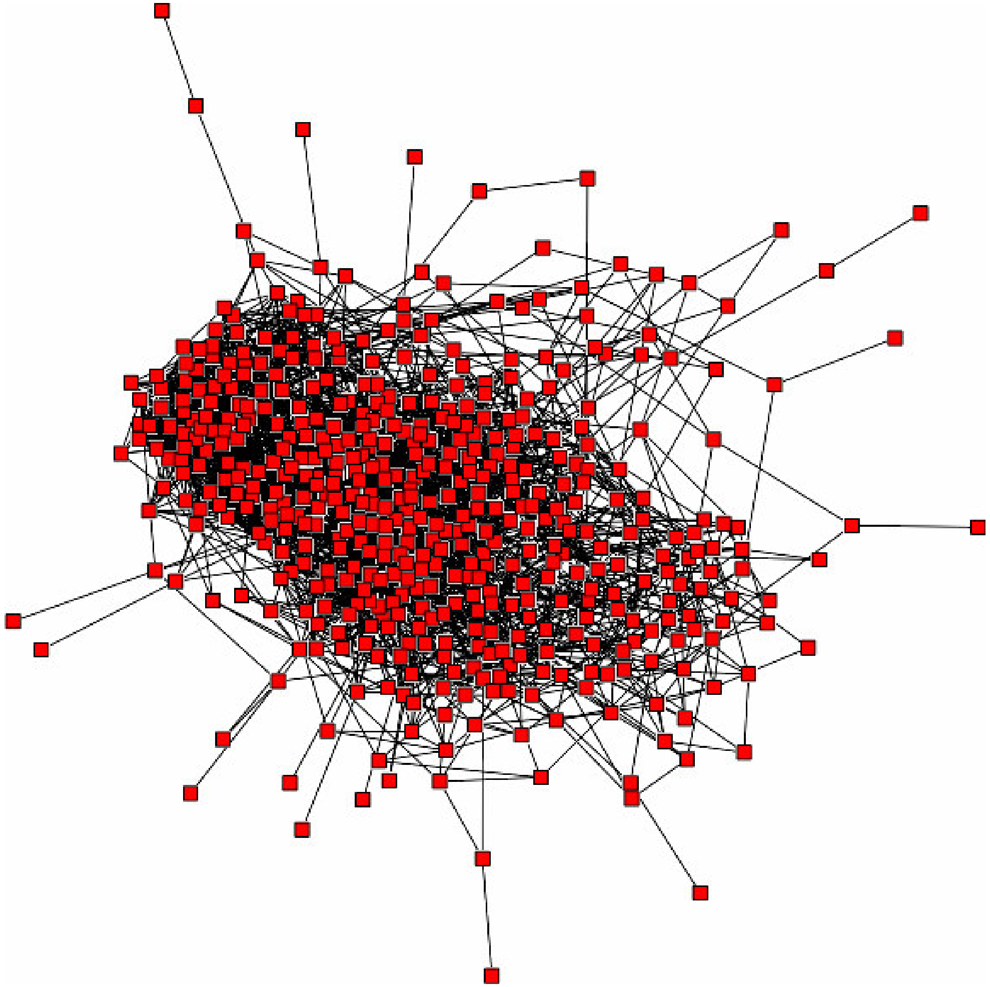
index	N of component	ccw of component	correlation	t	tw	n
[1600;1700]	9	1,095762779	0,94615	0,277778	0,05125	10
[1600;1700]rand	9	0,439093782	0,852147	0,277778	0,128175	10

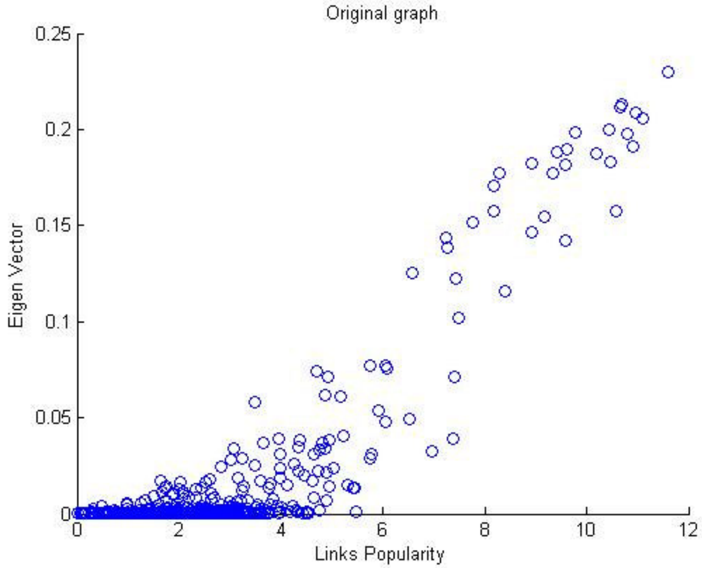


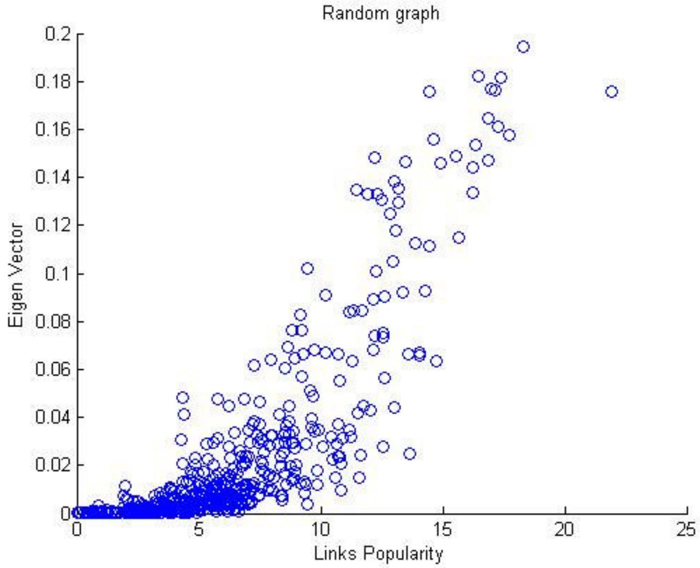




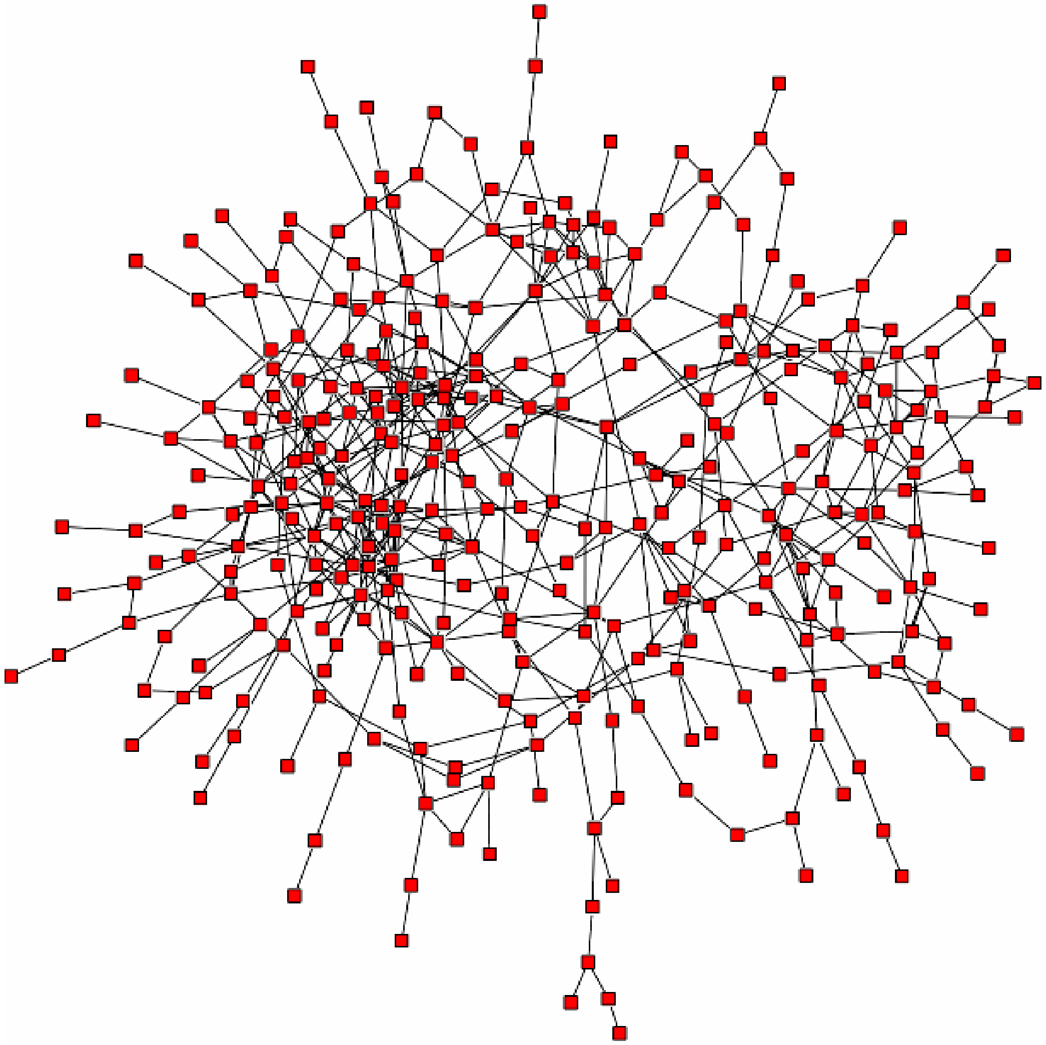
index	N of component	ccw of component	correlation	t	tw	n
[2000;2500]	494	0,458717008	0,866009	0,025162	0,005001	3064
[2000;2500]rand	494	0,182818871	0,829829	0,025162	0,012529	3064

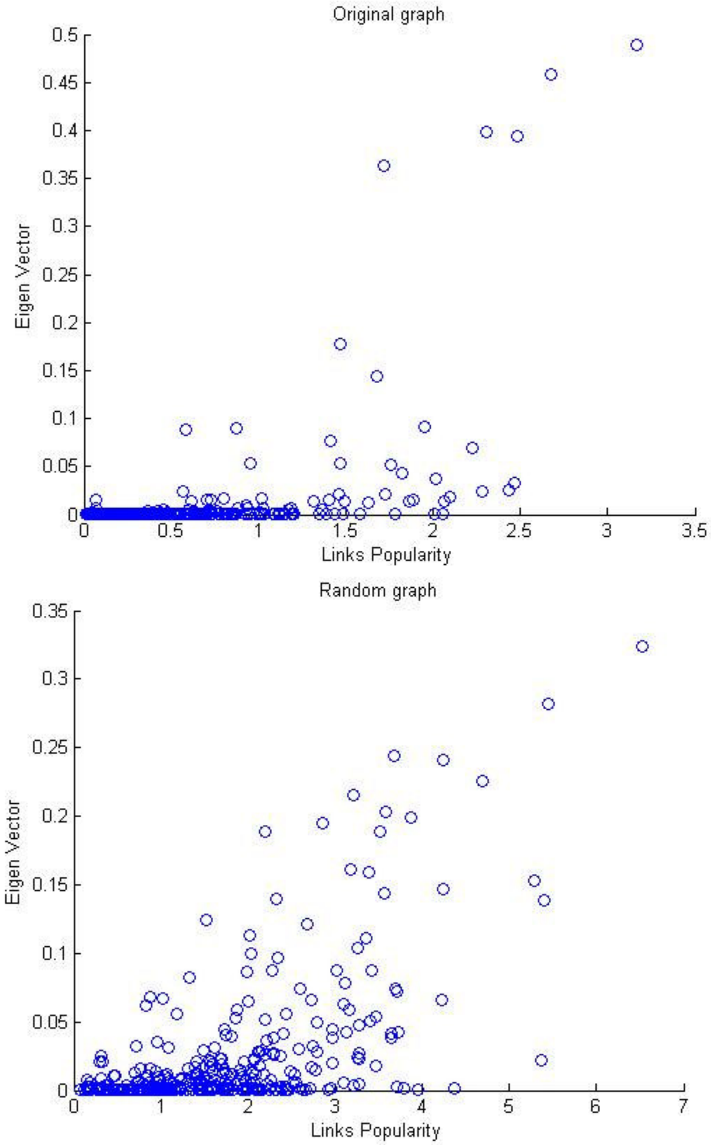




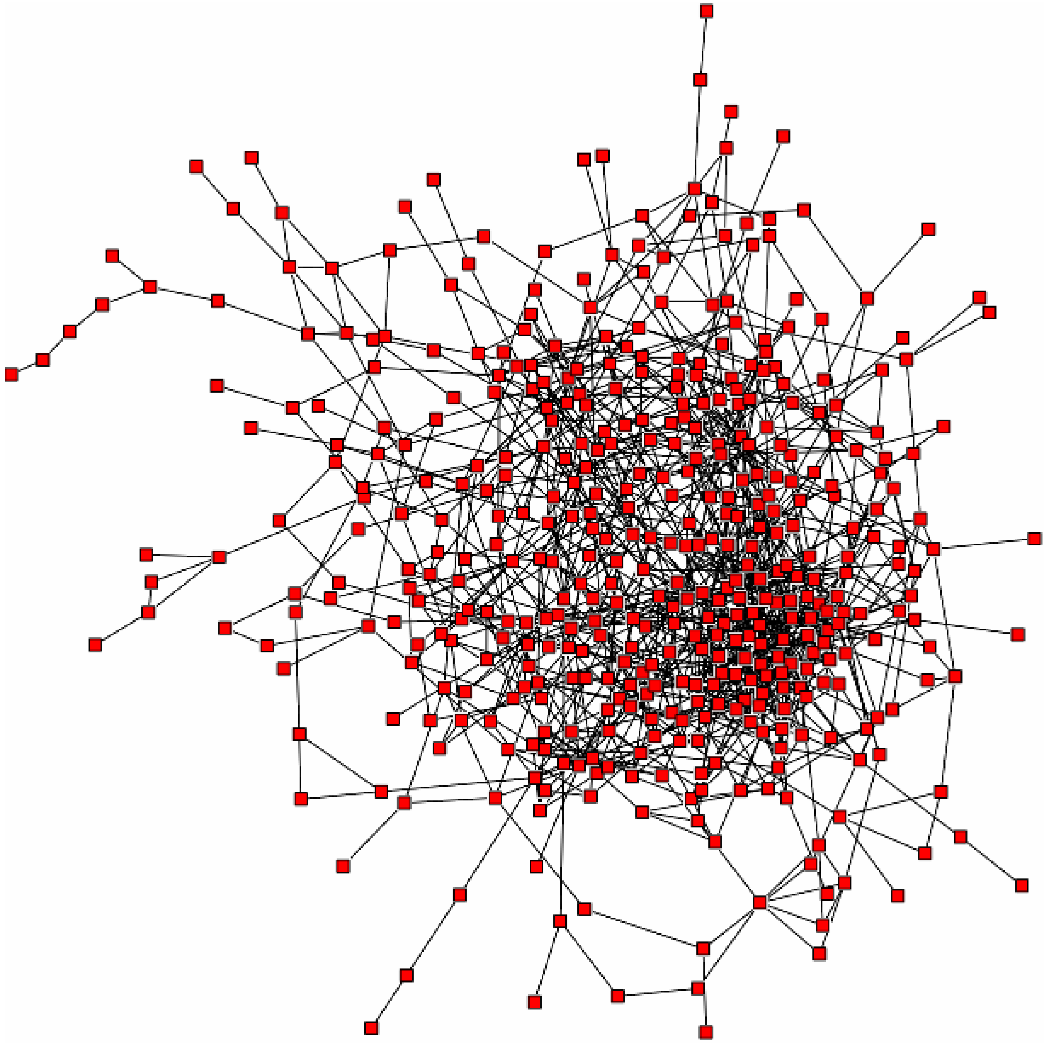


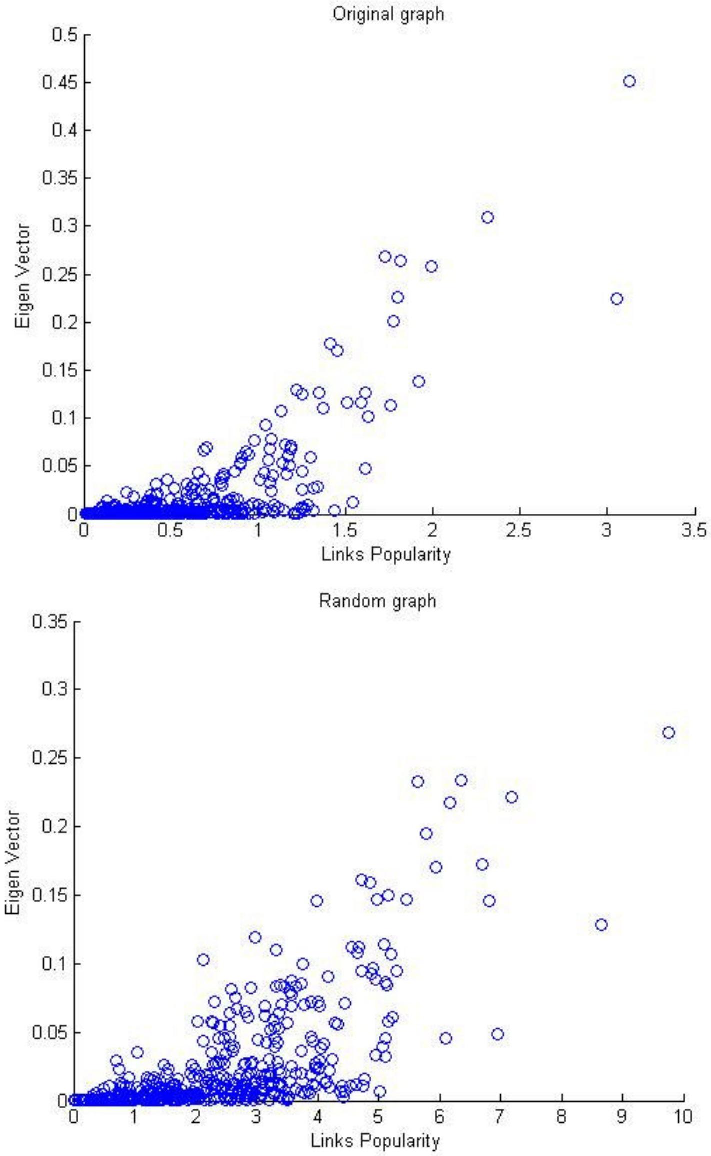
index	N of component	ccw of component	correlation	t	tw	n
[6000;6500]	341	0,52998867	0,535485	0,009971	0,001707	578
[6000;6500]rand	341	0,125245006	0,61576	0,009971	0,004926	578



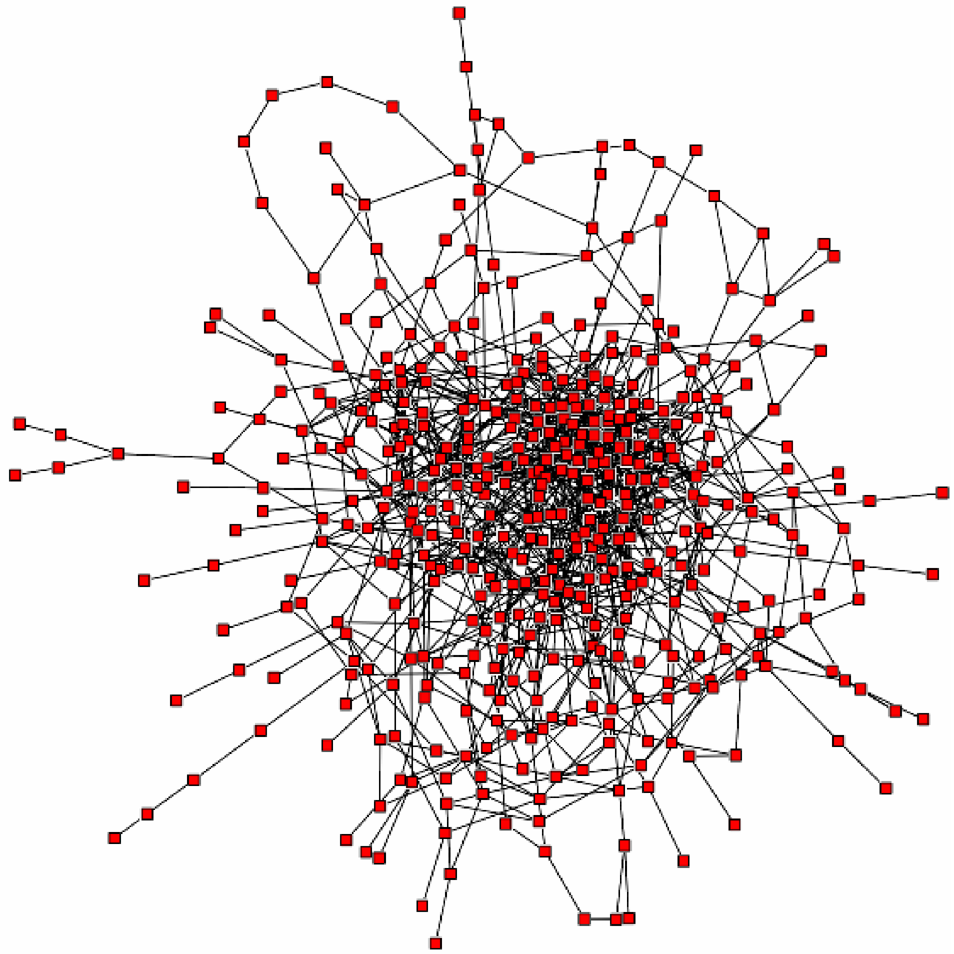


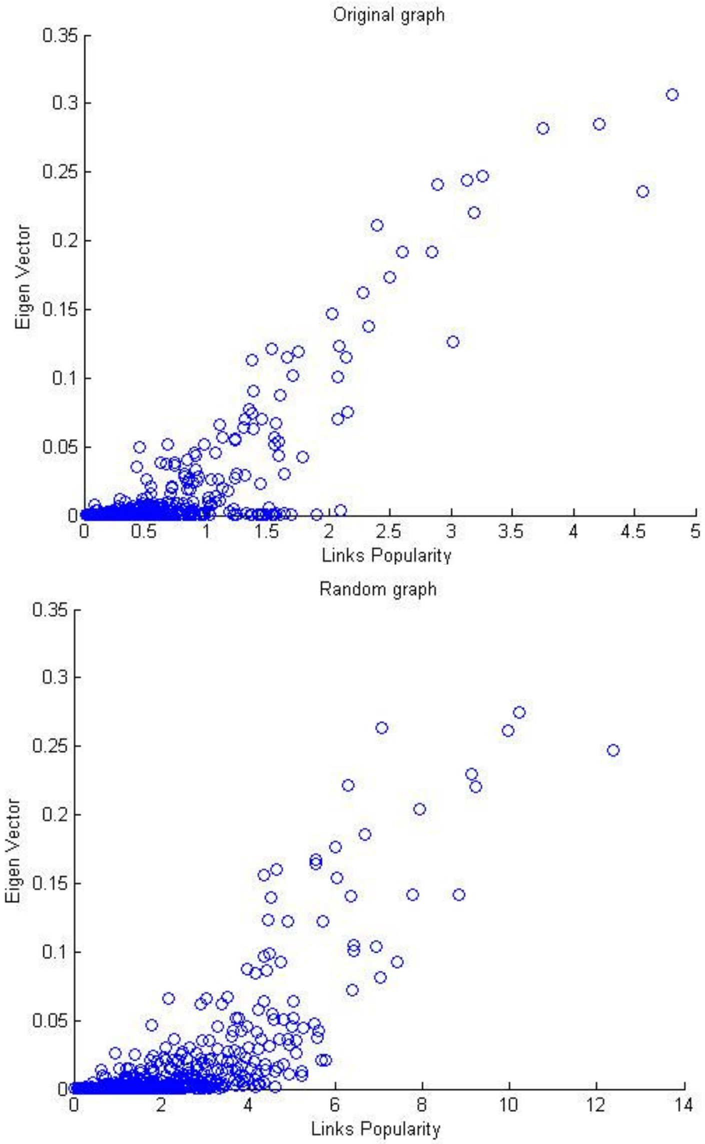
index	N of component	ccw of component	correlation	t	tw	n
[7500;8000]	458	0,283769289	0,735926	0,010243	0,001088	1072
[7500;8000]rand	458	0,111682506	0,706898	0,010243	0,004965	1072





index	N of component	ccw of component	correlation	t	tw	n
00;8500]	454	0,373868501	0,843412	0,010785	0,001357	1109
[8000;8500]rand	454	0,138695349	0,773868	0,010785	0,005335	1109




Original graph
Eigen Vector
Links Popularity
Random graph
Eigen Vector
Links Popularity