AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY
Faculty of Physics and Applied Computer Science

Master thesis

Michał Ochman

major: **Applied Computer Science**

specialisation: **Databases**

Identification of the selected types of substructures in the opinion networks

Supervisor: **dr hab. Jarosław Kwapień**

Kraków, September 2013

Kraków, September 2013

The subject of the master thesis and the internship by Michał Ochman, student of 5th year major in applied computer science, specialisation in databases

The subject of the master thesis: **Identification of the selected types of substructures in the opinion networks.**

Supervisor:	dr hab. Jarosław Kwapień
Reviewer:	....
A place of the internship:	IFJ PAN, Kraków

Programme of the master thesis and the internship

1. First discussion with the supervisor on realization of the thesis.
2. Collecting and studying the references relevant to the thesis topic(s).
3. The internship:
   - getting to know the idea of complex networks,
   - implementation of software used to fetch documents,
   - preparing network adjacency matrix from downloaded list of words and users,
   - discussion with the supervisor focused on optimizing the matrix obtained,
   - preparation of the internship report.
4. Continuation of calculations concerning the thesis subject.
5. Ordering and first analysis of the calculation results.
6. Final analysis of the results obtained, conclusions – discussion with and final approval by the thesis supervisor.
7. Typesetting the thesis.

Dean's office delivery deadline: September 2013

....	....
(department's head signature)	(supervisor's signature)

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Introduction

In this work we will consider the problem of identifying different types of structures and substructures in opinion networks. The formal definition for the term *opinion network* was not proposed yet, but informally it can be described as a *collaboration network* representing opinions amongst a set of particular agents, e.g. individual users of message boards. A collaborative network is a network that consists of agents that are mostly autonomous and heterogenous, but that cooperate to achieve some kind of a common goal. The interactions of such network are generally supported by computer networks.

1.2 Motivation

Due to the hypothesis that some agents may be dishonest and may, for example, create multiple accounts on message boards to either advertise and/or malign certain brands of products it is desirable, if not necessary, to filter out quasi-opinions posted by such agents. For many message boards, like the one examined during writing of this thesis, honest and unpolluted opinions are key factor in retaining its reputability. Without the possibility to identify and remove unwanted opinions it is easy to decline and lose the user base. However, one has to note that there is no sure and reliable way of doing it automatically or applying it in a generic way across different opinion networks.

Because of the character of the thesis the assumption was to download messages from one of the message boards and create a complex network using obtained data. Then it was proposed to analyse newly established complex network using several already known methods and/or develop new methods if necessary.

1.3 Related work

B. Kujawski, *et al.* [1] have presented us with an empirical study of opinion networks created by users of message boards and news groups. They showed that they organise themselves in groups of scale-free trees which structures depend on the topics discussed.

R. V. Solé, *et al.* [2] have reviewed the state of the art on language networks and their possible relevance to cognitive science. They discuss the syntax of the language and its emergence through the process of learning the language. They present a study to illustrate how this way of research can try to answer questions about language organisation and evolution.

M.-S. Shang, *et al.* [3] were trying to prove that it is important to understand the structure and evolution of web-based user-object networks as they play significant role in e-commerce nowadays. They have studied two large-scale and popular web sites, audioscrobbler.com and del.icio.us, where users are connected with music groups and bookmarks, respectively. They propose a new index, named collaborative similarity, to quantify the diversity of tastes based on the collaborative selection, which I use later in this paper.

1.4 Software used

1.4.1 Data mining

Data was downloaded using custom written web crawler. PHP language was used along with sfWebBrowser plugin from Symfony framework for crawling and PHP Simple HTML DOM Parser for HTML manipulation. Text processing was performed with a help of regular expressions found in both PHP and Python languages. MySQL was used as a storage for relational data.

1.4.2 Calculations

The most data processing was performed employing proprietary scripts in PHP and Python. Python was especially used wherever multi-threaded operations were possible, thanks to its outstanding multiprocessing package. In most cases, 4-8 processing cores were used to process the data, with up to 20 (using 3 machines) in extremely time consuming calculations.

For some of the network computations, NetworkX package for Python was used. NetworkX is a great tool for the creation, manipulation, and study of the structure, dynamics, and functions of complex networks. One of its best features is the use of Python's data structures for graphs, so there is no extra overhead. It allows for arbitrary data to be set on both nodes and edges. It is very easy to write additional procedures to use with this package and many algorithms has already been implemented.

For other computations, Gephi was used, which is an interactive visualisation and exploration platform for all kinds of networks. It allows for easy manipulation of the network and makes it simple to instantly perform operations on it. The tool is very intuitive and makes it possible to analyse very big networks. It contains various metrics that are ready to use as soon as you import your network.

1.4.3 Figures

Figures with networks were created using Gephi. Figures with graphs were created using Adobe Illustrator®. This software was also utilised to convert SVG files from Gephi to EPS files to be used with this paper. All other figures were plotted with gnuplot.

1.5 Structure of this paper

This paper is divided into 5 distinct chapters related to a single group of problems.

First chapter makes an introduction to the problem. It presents the motivation behind the thesis and discusses related work. In addition, a list of software that was used to produce parts of this paper has been listed there.

Second chapter introduces the reader to the very general concepts of *graph theory*. It provides the most important terms and definitions that will be used throughout this work. It also focuses on issues related to *network theory* which is a part of graph theory, but in an area of computer and network science. It has applications in many disciplines like social sciences which are of particular interest to those working with social networks.

Third chapter is dedicated to a field of computer science called *natural language processing*. It concerns the interactions between human languages and computers and approaches the problems of parsing and understanding written messages.

Fourth chapter presents the reader with the methodology proposed in attempt to understand the processes taking place in the network. Several implementations are examined thoroughly with the results presented and interpreted.

Last but not least, the fifth chapter concludes the ideas presented in the paper. It discusses what was and was not achieved during the assignment. It also provides an idea for further research than can be conducted later using the data obtained. Finally, it gives some interesting ideas that are worth considering but were not approached for various reasons.

2 Network theory

2.1 Definitions

2.1.1 Graph

A graph G is a pair of sets (V, E) such that $E \subseteq [V]^2$. V or $V(G)$ is a set of arbitrary objects called *vertices* (singular is *vertex*) or *nodes*, while E or $E(G)$ is a set of vertex pairs, called *edges* or *arcs* on occasion; elements of E are two-element subsets of V [4].

There are two graph types in respect to edge type: *undirected* and *directed*, pictured on figure 2.1a and 2.1b respectively. In the former, edges are ordered pairs of vertices, i.e. $e = (u, v) = (v, u)$. In the latter they are unordered or just sets of two vertices, that is $e = (u, v) \neq (v, u) = d$.

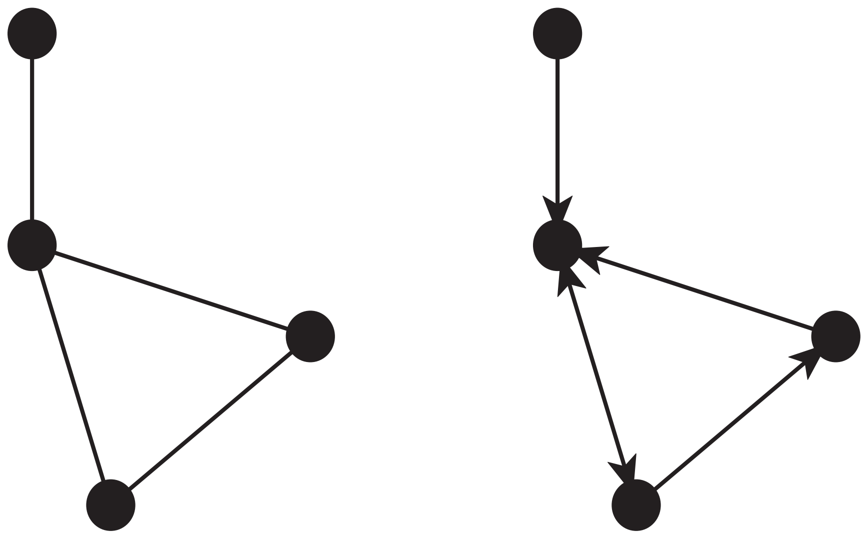
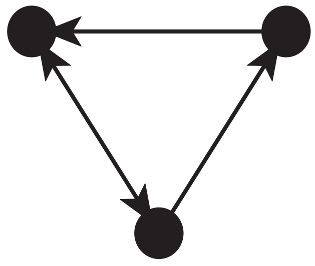

(a) An undirected graph. (b) A directed graph.

Figure 2.1: Types of graphs in respect to orientation.

Then we have *simple graphs*, where there are no edges from a vertex to itself (called loops) and there are no more than one edge between any two distinct vertices (i.e. edges form sets). On the other side we have *multigraphs* (or *pseudographs*), that may both have loops (not everyone agrees with that) and multiple edges (also called parallel edges) between different vertices (thus they form a multiset).


(a) A simple graph.

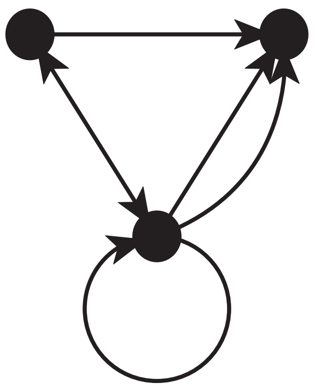

(b) A multigraph.

Figure 2.2: Types of graphs in respect to edge distinctness.

If (u, v) is an edge in an undirected graph then we call u a *neighbour* of v and vice versa. Then we call the number of such neighbours a *degree* of a vertex. In directed graphs however, if $u \to v$ is a directed edge, then we call u the *predecessor* of v, which in turn we call a *successor* of u. The *in-degree* of a vertex is the number of predecessors and the *out-degree* of a node is the number of its successors. It is important to note that given u and v and if $u \to v$ and $v \to u$ then $u \leftrightarrow v$.

A graph $H = (U, D)$ is a *subgraph* of $G = (V, E)$ if $U \subseteq V$ and $D \subseteq E$.

2.1.2 Paths

2.1.2.1 Path

A *path* is a sequence of edges where each successive edge share a vertex and all other edges have no vertices in common.

2.1.2.2 Cycle

A *cycle* is a path which starts and ends at the same vertex and has at least one edge.

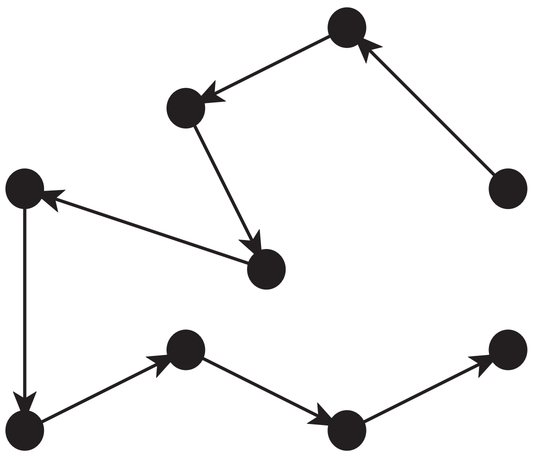


Figure 2.3: A path.

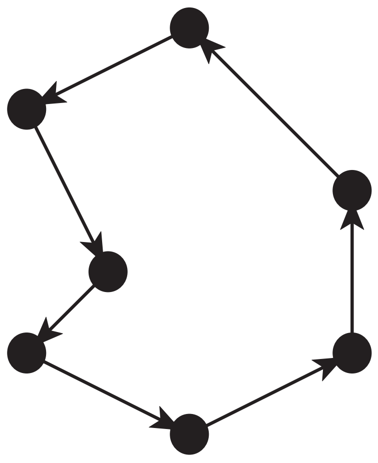


Figure 2.4: A cycle.

2.1.2.3 Tree

A *tree* is a special graph that is a connected acyclic graph. A tree is also a minimal connected graph, meaning that removing any of the edges will make the graph disconnected.

2.1.2.4 Spanning tree

A *spanning tree* of graph G is a subgraph that is a tree and contains all vertices of G. Obviously, no spanning trees exist for disconnected graphs.

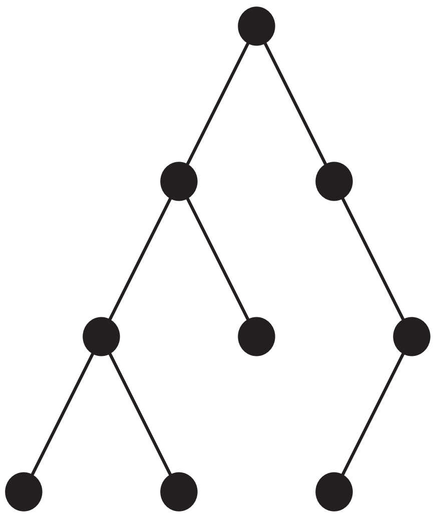


Figure 2.5: A tree.

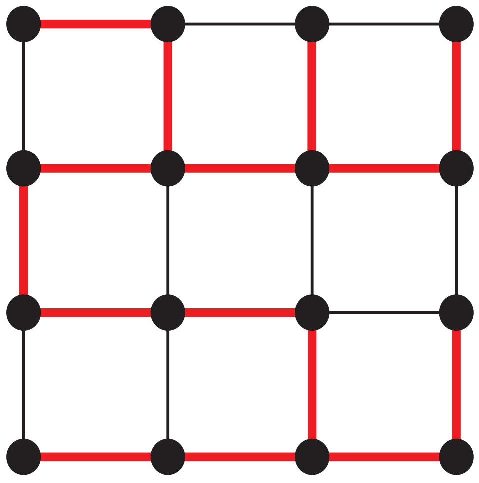


Figure 2.6: Spanning tree (bold edges) of grid graph.

2.1.3 Weighted graph

A *weighted graph* is a graph that associates a label—called *weight*—with every edge in the graph. Weights of edges are usually represented with real numbers. The *weight of a path* in such graph is the sum of the edge weights of the edges forming a path. A non-edge has sometimes a special weight assigned with a value of infinity. Weight can also represent *costs*, then words are used instead of numeric weight. A graph is always assumed to be unweighted unless otherwise has been stated.

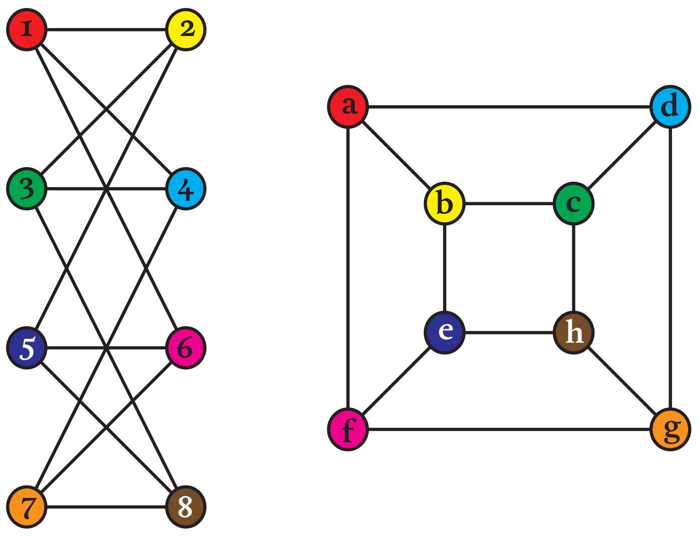
2.1.4 Graph isomorphism

An *isomorphism* of graphs G and H is a bijection between the vertices of G and H

$$f : V(G) \to V(H), \tag{2.1}$$

such that an edge exists between any two vertices u and v of G if and only if an edge exists between $f(u)$ and $f(v)$ in graph H. A correspondence of this kind is commonly called 'edge-preserving bijection,' where isomorphisms are 'structure-preserving bijections' in general notion.

The above definition assumes that graphs are undirected and unweighted. However, a different definitions of isomorphism may be applied to other types of graphs, by adding the requirements to preserve the corresponding additional elements of structure like link directions, weights, *etc.*, with the following exception in respect to graph labelling. If labels in graph are uniquely taken from the integer range $1, \ldots, n$, where n is the number of the vertices of the graph, two labeled graphs will be isomorphic if the corresponding underlying unlabelled graphs are isomorphic.



$f : V(G) \to V(H)$
$f(1) = a$
$f(2) = b$
$f(3) = c$
$f(4) = d$
$f(5) = e$
$f(6) = f$
$f(7) = g$
$f(8) = h$

Figure 2.7: Graphs G (left), H (middle) and an isomorphism between them (right).

The concept of *graph isomorphism* makes it possible to characterise graph properties essential to the structure of the graphs themselves from properties associated with graph representations like graph drawings or data structures for graphs and labelings, *etc.* For example, if a graph has exactly one cycle, then all graphs in its isomorphism class[1] also have exactly one cycle. On the other hand, in the common case when the vertices of a graph are (represented by) the integers $1, 2, \ldots, N$, then the expression

$$\sum_{v \in V(G)} v \cdot \deg v, \tag{2.2}$$

may be different for two isomorphic graphs. But that was mentioned as an exception to the general definition.

2.1.5 Graph properties

In order to focus on the abstract structure of graphs, we define *graph properties* as maintained under all possible *isomorphisms* of a graph. However, there is a distinction referred to this term; specifically, a *property* is usually referred to descriptive characterisations of

[1]A collection of graphs isomorphic to each other.

graphs (i.e. it is a class of graphs), while *invariant* is used for properties expressed quantitatively (i.e. it is a function from graphs to some other set, like $\mathbb{N}$). To avoid naming collisions, from now on I will mention only properties and invariants in the former sense.

2.1.5.1 Graph invariants

Order The *order* of a graph is the number of vertices in a graph and is denoted by $|V(G)|$ or $|G|$.

Size The *size* of a graph is the number of its edges [5] in a graph and is denoted by $|E(G)|$ or $||G||$. In an undirected graph, there are $0 \leq E \leq \binom{V}{2}$ and in directed graph: $0 \leq E \leq V(V-1)$.

Eccentricity The *eccentricity* $\epsilon(v)$ of vertex v is the greatest geodesic distance between v and any other vertex.

Radius The *radius* r of a graph is the minimum eccentricity of any vertex in the graph.

Diameter The longest of the shortest path lengths is the *diameter* d of a graph. It is the maximum eccentricity of any vertex in the graph. In order to find it, one first need to find the shortest path between each pair of vertices; then the greatest length of any of these paths would be the diameter of the graph.

2.1.5.2 Graph properties

Connected Let $G = (V, E)$ be a graph. A non-empty graph G is *connected* if any two of its vertices are linked by a path in G, i.e.

$$\forall_{i,j \in \mathbb{N} \cap i,j < |G|} \forall_{v_i \in V} \exists_{v_j \in V} (v_i, v_j) \in E. \tag{2.3}$$

In other words, we call a graph *connected* if there is a path from any vertex to any other vertex.

A maximal connected subgraph of G is a *component* of G. The empty graph has no components since connected graphs are non-empty.

A graph that is *disconnected* consists of several *connected components*. Two vertices are considered to be in the same connected component if there is a path between them.

G is called k*-connected* (for $k \in \mathbb{N}$) if $k < |G|$ and $G - X$ is connected for every $X \subseteq V$ with $k > |X|$. Every non-empty graph is 0-connected and 1-connected graphs are precisely non-trivial connected graphs. The greatest integer k such that G is k-connected is the *connectivity* $\kappa(G)$ of G. Hence graph is disconnected if and only if $\kappa(G) = 0$. The simplest 2-connected graphs are the cycles; all the others can be inductively constructed by adding paths to cycles.

Cyclic Graph is cyclic when it consists of a single cycle.

Acyclic A graph is called acyclic if there are no subgraphs that are cycles.

Bipartite We call a graph a *bipartite* or a *bigraph* if a graph has vertices that can be divided into two disjoint sets U and V such that the edges only exist between independent vertices in U and V and never between vertices of the same set [4]. Equivalent definition is that a bipartite graph is a graph that does not contain any cycles with odd lengths [6]. An example of bipartite graph is presented on figure 4.13 in section 4.4.4.

The two sets U and V may be thought of as a colouring of the graph with two colours: if one colours all nodes in U blue, and all nodes in V green, each edge has endpoints of differing colours, as is required in the graph colouring problem [6, 7]. In contrast, such a colouring is impossible in the case of a non-bipartite graph, such as a triangle: after one node is coloured blue and another green, the third vertex of the triangle is connected to vertices of both colours, preventing it from being assigned either colour.

A bigraph with partition sets U and V with E denoting edges of the graph is often written as $G = (U, V, E)$. If such graph is disconnected, it may have more than one bipartition; [8]. In this case, the (U, V, E) notation is helpful in specifying one particular bipartition that may be of importance in an application.

Bipartite graphs very often arise naturally when dealing with real-world examples. For instance, a graph of movie actors and movies, with an edge between an actor and a movie if the actor has starred in that movie. It is a classic example of an affiliation network, which is a type of bigraph used in analysis of social networks [9].

2.2 Graphs representation

Graphs are usually visualised as *embeddings*. An embedding of a graph $G(V, E)$ is representation of G on Σ plane in which all $v \in V$ are associated to points on Σ and each of $e \in E$ are arcs or straight line segments between points corresponding to vertices at both ends of e. If no two arcs ever intersect, the embedding of a graph is called *planar*, otherwise it is called a *non-planar*. Because the same graph can have many embeddings that are different, it is important not to confuse the particular embedding with a graph itself. In particular, planar graphs can have non-planar embeddings too.

There are many other ways of graph visualisation, but apart from the above and below, they are not useful for analysis of the data that is a subject of this thesis.

Graphs can be explicitly represented by either *adjacency matrices* or *adjacency lists*.

The adjacency matrix of a graph G is a $|V| \times |V|$ matrix of indicator variables. Each entry in this matrix indicates whether a particular edge between indexes is or is not in graph G:

$$A[i,j] = [(i,j) \in E]. \tag{2.4}$$

For undirected graph, similar to the edges, the adjacency matrix is always symmetric: $A[i,j] = A[j,i]$. In simple graphs, diagonal entries in A are all zeros since loops are not allowed. The most valuable property of adjacency matrix is that we can decide in $\Theta(1)$ time whether two arbitrary vertices are connected by an edge just by looking at the appropriate elements of the matrix. We also can scan for a list of all vertex neighbours by looking at the corresponding row (or column), which will require $\Theta(V)$ time. The main disadvantage is that this matrix will always take $\Theta(V^2)$ space regardless of how many edges the graph really have. This makes adjacency matrices efficient for *dense* graphs.

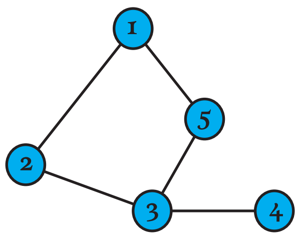


Figure 2.8: Graph G.

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

Figure 2.9: Adjacency matrix.

$1 \to 2 \to 5$

$2 \to 1 \to 3$

$3 \to 2 \to 4 \to 5$

$4 \to 3$

$5 \to 1 \to 3$

Table 2.1: Adjacency list.

On the other hand, adjacency lists are good when dealing with *sparse* graphs (graphs with relatively small number of edges). Adjacency list is an array of linked lists (one per vertex). For undirected graphs each edge (u, v) is stored twice: once in u's neighbourhood and once in v's neighbour list. For directed graph each edge is stored only once. Either way, adjacency list will take $O(V + E)$ space; listing all the neighbours of v vertex takes $O(1 + \deg v)$ time (by scanning the neighbour list). We can determine whether (u, v) is an edge by scanning neighbours of u in $O(1 + \deg u)$ time.

2.3 Types of networks

The simplest type of network is a simple graph: a set of vertices joined by undirected, weighted edges. The complexity of a network may come in various forms. For example, there may be more than one type of vertices in a network (like bigraph) or even different types of edges. Additionally both vertices and edges may have a variety of arbitrary properties associated with them. For instance, in a network with vertices representing people, these vertices may represent people of different sex or nationality, age or many other things (without the network being bipartite). Edges may represent family bonds and friendships, but they also can represent things like geographical proximity or similar income. Edges may carry weights that could tell how well these people know each other. They may be directed or undirected. In the former situation they may for example represent an employment hierarchy in a company or telephone calls between the individuals.

Graphs may be partitioned naturally in several ways. A number of examples of this types of networks may be provided that have a form of bipartite graphs. So-called *affiliation networks* in which people are joined together by common membership of groups can take this form, the two types of vertices with one representing the people and the other the groups.

Graphs may evolve over time. Vertices and edges may appear or disappear. The arbitrary values associated with those vertices and edges may change. There are many other levels of sophistication one can add. The study of networks is by no means a complete science yet, and many of the possibilities have yet to be explored in depth.

2.4 Real world networks

This section will try to describe different types of networks as we know them and their structure. Recent work on the mathematics of networks has been driven largely by observations of the properties of actual networks and attempts to model them. The section has been divided into four loosely formed categories of networks: social networks, information networks, technological networks and biological networks, which will be briefly introduced.

2.4.1 Social networks

A *social network* is a structure made up of set of people (or more generally abstract actors) or groups of people with some pattern of contacts or other sort of interactions between them [9]. Examples of such interactions include friendships between individuals or business relationships between companies. The social sciences have the longest history of the substantial quantitative study of real-world networks of the academic disciplines [9]. The most relevant early works on the subject are Jacob Moreno's work in the 1920s and 30s on friendship patterns within small groups [10] or the mathematical models of Anatol Rapoport [11], who was possibly one of the first theorists who stressed the importance of the degree distribution in networks of all kinds.

A set of experiments by Milgram, the famous 'small-world' experiments [12] are of a much importance. Not a single real network were reconstructed during these experiments, but they tell us a lot about the network structure in general. These experiments examined the distribution of path lengths in networks of acquaintances by asking participants to pass a letter to one of their first-name acquaintances in an order to get it to an individual who was an assigned target of the experiment. Even though most of the letters were lost, about a quarter of them reached the target and on average were carried through the hands of only about six people. This was the origin of the—popular now—concept of so-called 'six degrees of separation,' although the phrase itself never appeared in Milgram's papers.

2.4.2 Information networks

A second kind of network is what is called an *information network* (or *knowledge network*). The classic example of an information network is the network of citations between academic papers [13]. Most articles include citations of previous work on related topics. This forms a network where the vertices are articles. Directed edges are drawn from articles that cite to articles that are cited. Citation networks are acyclic, because papers can only cite papers that have already been written—it is not possible to cite work that has not been written yet.

The World Wide Web, which is a network of Web pages linked together by hyperlinks is another example worth mentioning. It is important not to confuse The Web with the Internet, which is a physical network of computers linked together by physical data connections. The Web is cyclic unlike the citation network.

2.4.3 Technological networks

The third, somewhat classic, class of networks is *technological networks*. They are man-made networks created to distribute some kind of commodity or resource (e.g. electricity or information). The electric power grid is a good example. It is a network of high-voltage transmission lines that spans a large area, as opposed to the local low-voltage AC power delivery lines that span only individual neighbourhoods. The telephone network or delivery networks such as post-offices also fall into this general category. Another important technological network is the Internet.

2.4.4 Biological networks

Probably the most known instance of a *biological network* is the network of metabolic pathways, which represents the metabolic substrates and products having directed edges joining them when a known metabolic reaction exists that acts on a given substrate and produces a given product.

Neural networks also belong to the class of biological networks. The topology of real neural networks is very difficult to measure. However, it has already been done successfully, the best known example being the reconstruction of the 282-neuron neural network of the *Nematode Caenorhabditis Elegans* by White *et al.* [14].

2.5 Small-world networks

A *small-world network* is a type of graph in which most vertices are not joined by edges with one another, but they can be reached from each other using a short path. Precisely, a small-world network is defined as a network where a typical distance l between two randomly chosen vertices grows proportionally to the logarithm of the number of vertices $|V|$ in the network [15]

$$l \propto \log|V|. \tag{2.5}$$

For instance, this results in the small world phenomenon of strangers being linked by a mutual acquaintance in social networks. Many empirical graphs exhibit small-world network characteristics. Networks that are well-modelled by small-world networks include social networks, the connectivity of the Internet and gene networks.

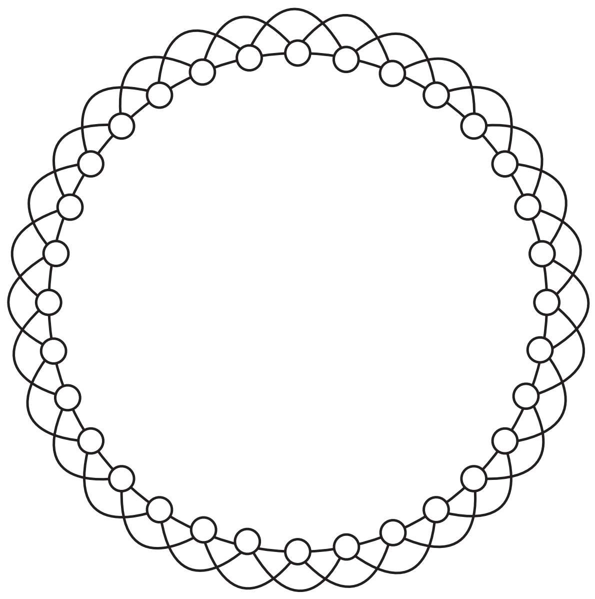


Figure 2.10: A lattice.

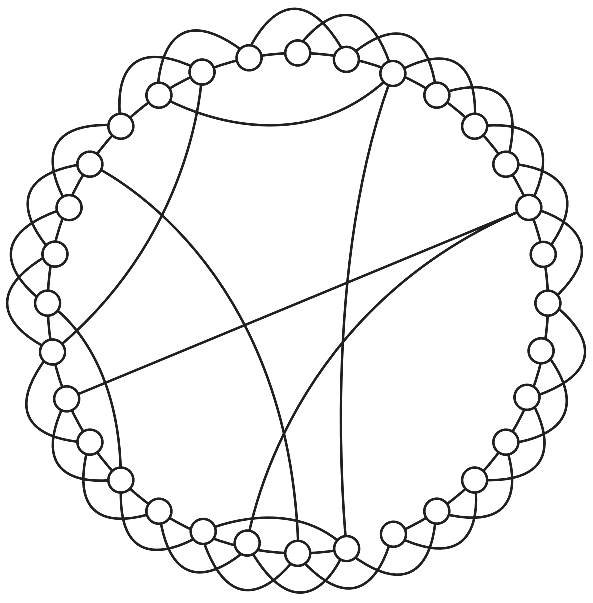


Figure 2.11: A small world model.

A certain category of small-world networks were identified as a class of random graphs by Duncan Watts and Steven Strogatz [15]. They noted that graphs could be classified according to two independent structural features, namely the clustering coefficient, and an average shortest path length. Purely random graphs, built according to the Erdős–Rényi (ER) model, exhibit a small average shortest path length (varying typically as the logarithm of the number of nodes) along with a small clustering coefficient [16]. Watts and Strogatz measured that in fact many real-world networks have a small average shortest path length, but also a clustering coefficient significantly higher than expected by random chance. Watts and Strogatz then proposed a novel graph model, currently named the Watts and Strogatz model, with a small average shortest path length, and a large clustering coefficient. The crossover in the Watts-Strogatz model between a *large world* (such as a lattice, see figure 2.10) and a small world (see figure 2.11) was first described by Barthelemy and Amaral in 1999 [17].

2.6 Network properties

2.6.1 Average path length

The concept of *average path length* in network topology is that it is the average number of steps along the shortest paths for all possible pairs of vertices in the network. In short, it is a good measure of the efficiency of the network and tells a lot about how quickly the information may be passed over the network. It is one of the three most robust measures of network topology, together with network's clustering coefficient and degree distribution of vertices. The average path length will for example tell how many people one would need to communicate through (on average) to contact a complete stranger. Average path length should not be confused with network diameter, which is the longest geodesic—the longest shortest path between any two nodes in the network.

With the help of an average path length one can easily tell whether a network is efficient or not. Shorter average path lengths are generally more desirable. However, the average path length only tells what the path length between two nodes will most likely be, as the network itself might have some very remotely connected nodes and many nodes which are neighbours of each other at the same time.

Considering an unweighted graph G with the set of vertices V, let $d(v_1, v_2)$, where $v_1, v_2 \in V$ denote the shortest distance between vertices v_1 and v_2. Assuming that $d(v_1, v_2) = 0$ if $v_1 = v_2$ or if v_2 is unreachable from v_1. We define the average path length l_G as:

$$l_G = \frac{1}{|V|(|V|-1)} \sum_{i,j} d(v_i, v_j), \tag{2.6}$$

where $|V|$ is the number of vertices in G.

In a real world networks like the World Wide Web, a short average path length aids transferring the information quickly and efficiently and also reduces costs. Similarly, by judging average path length of a metabolic network one can tell the efficiency of mass transfer in it. Also, it is possible to minimise a power grid network losses if one can lower the average path length of the network.

Most real world networks have an average path length which is very short. It leads to a conclusion that a small worlds may exist in such networks and that everyone is connected to everyone else through a very short path. A result being that today, most models of real networks are created with this condition in mind. Several models had been proposed (random network, Watts-Strogatz model, Barabási-Albert model), with all having one thing in common—they all predicted very short average path length [18].

The average path length depends on the system size but does not change drastically with it. Small world network theory predicts that the average path length changes proportionally to $\log|V|$, where $|V|$ is the number of nodes in the network.

2.6.2 Transitivity and clustering coefficient

In networks it is quite common to find that if vertex A is connected to vertex B and vertex B is connected to vertex C, then the probability that vertex A is connected to vertex C is higher than the probability that vertex A is connected to any other randomly chosen vertex. For example, in social networks, the friends of your friends are more likely to also be your friends than a random person that is not a friend of your friend. This deviation from the behaviour of random graphs can be seen in a network property called *transitivity* (also known as *clustering*, although this term has another meaning which may be confusing). From topological point of view, transitivity means that there are more sets of three vertices, each of which is connected to the other two—called triangles. It may be quantified by defining a *clustering coefficient*

$$C = \frac{3 \times \text{number of triangles in the network}}{\text{number of connected triples of vertices}} = \frac{\text{number of closed triplets}}{\text{number of connected triples}}, \qquad (2.7)$$

where a *connected triple* means a single vertex with edges running to an unordered pair of others.

In graph theory, a clustering coefficient is a measure of degree to which nodes in a graph tend to cluster together. Evidence suggests that in most real-world networks, and in particular social networks, nodes tend to create tightly knit groups characterised by a relatively high density of ties; this likelihood tends to be greater than the average probability of a tie randomly established between two nodes [15, 19].

Two versions of this measure exist: the global and the local. The global version was designed to give an overall indication of the clustering in the network, whereas the local gives an indication of the embeddedness of single nodes.

2.6.2.1 Global clustering coefficient

The *global clustering coefficient* is based on node triplets. A *node triplet* is made out of three nodes that are joined by either two or three undirected edges, called open and closed triplet respectively. A triangle is built using three closed triplets, one centred on each of the nodes. We define the global clustering coefficient as the number of closed triplets divided by the total number of triplets. Luce and Perry were the first to attempt to measure this quantity [20]. It can be applied to both undirected and directed networks and it gives an indication of the clustering in the whole network. The measure is also called transitivity [9]. Watts and Strogatz proposed a following clustering coefficient definition: 'Suppose that a vertex v has k_v neighbours; then at most $k_v(k_v-1)/2$ edges can exist between them (this occurs when every neighbour of v is connected to every other neighbour of v). Let C_v denote the fraction of these allowable edges that actually exist. Define C as the average of C_v over all v.' [15]

2.6.2.2 Local clustering coefficient

The *local clustering coefficient* of a vertex in a graph assess how close its neighbours are to being a complete graph. The measure was first introduced by Watts and Strogatz to determine whether a graph is a small-world network [15].

The neighbourhood N_i for a vertex v_i is defined as its immediately connected neighbours

$$N_i = \{v_j : e_{ij} \in E \wedge e_{ji} \in E\}, \tag{2.8}$$

where e_{ij} is an edge connecting vertices v_i and v_j.

We define k_i as the number of vertices $|N_i|$ in the neighbourhood N_i of a vertex.

The local clustering coefficient C_i of a vertex v_i is given by the number of connections within the vertex neighbourhood over the number of connections that could possibly exist between them. Edges e_{ij} and e_{ji} in undirected graphs are considered identical, therefore if a vertex v_i has k_i neighbours, $\frac{k_i(k_i-1)}{2}$ edges could exist among the vertices within the neighbourhood. Hence, we define the local clustering coefficient for undirected graphs as

$$C_i = \frac{2\|\{e_{jk} : v_j, v_k \in N_i, e_{jk} \in E\}\|}{k_i(k_i-1)}. \tag{2.9}$$

The clustering coefficient can also be defined as

$$C_i = \frac{\lambda_G(v)}{\tau_G(v)}, \tag{2.10}$$

where $\lambda_G(v)$ is the number of triangles of $v \in V(G)$ (the number of subgraphs of G with 3 edges and 3 vertices, one of which is v) and $\tau_G(v)$ is the number of triples on $v \in G$ (the number of subgraphs with 2 edges and 3 vertices, one of which is v and such that v is incident to both edges).

To show that these two definitions are the same we show that

$$\tau_G(v) = C(k_i, 2) = \frac{1}{2}k_i(k_i-1). \tag{2.11}$$

The value of the clustering coefficient is within 0 and 1 range with the former if no vertex that is connected to v_i connects to any other vertex that is connected to v_i and the latter if every neighbour connected to v_i is also connected to every other vertex within the neighbourhood.

2.6.3 Betweenness centrality

Betweenness centrality measures a centrality of a vertex in a network. It is equal to the number of shortest paths from all vertices to all others that pass through that node. Betweenness centrality is one of the most useful measures. It quantifies both the load and importance of a node. Development of this measure is attributed to sociologist Linton Freeman [21].

The betweenness centrality of a node v is defined as

$$g(v) = \sum_{s \neq v \neq t} \frac{\sigma_{st}(v)}{\sigma_{st}}, \tag{2.12}$$

where σ_{st} is the total number of shortest paths from node s to node t and $\sigma_{st}(v)$ is the number of those paths that pass through v.

2.6.4 Degree distribution

The *degree distribution* is the probability distribution of degrees of vertices over the whole network. It is defined to be the fraction of nodes in the network with degree k

$$P(k) = \frac{|V|_k}{|V|}, \tag{2.13}$$

where $|V|$ is the number of vertices in a network and $|V|_k$ is the number of vertices in a network having a degree k.

The degree distribution is very important in studying both real networks (such as the Internet and social networks) and theoretical networks. Most networks in the real world have degree distributions that are highly right-skewed, meaning that a large majority of nodes have low degree but a small number, known as *hubs*, have high degree. Some networks, notably the Internet, the World Wide Web, and some social networks are found to have degree distributions that approximately follow a power law

$$P(k) \sim k^{-\gamma}, \tag{2.14}$$

where γ is a constant called power law exponent. Networks like that are called scale-free networks and have attracted particular attention for their structural and dynamical properties.

2.6.4.1 Scale-free networks

A scale-free network is a network whose degree distribution follows a power law, at least asymptotically, as defined in equation 2.14. The power law exponent γ is a parameter whose value is typically in the range $2 < \gamma < 3$ for real world networks, although occasionally it may lie outside these bounds [22, 23].

Many networks are believed to be scale-free, including the World Wide Web and social networks, although the matter is still under discussion by scientific community as more sophisticated data analysis techniques become available [24].

2.6.4.2 Maximum degree

The maximum degree k_{max} of a vertex in a network generally depends on the size of the network itself. For some calculations on networks the value of this maximum degree matters. Given a particular degree distribution, the probability of there being exactly m vertices of degree k and no vertices of higher degree is $\binom{n}{m} p_k^m (1–P_k)^{n–m}$, where P_k is the cumulative probability distribution. Hence, the probability h_k that the highest degree on the graph is k is

$$h_k = \sum_{m=1}^{n} \binom{n}{m} p_k^m (1 - P_k)^{n-m} = (p_k + 1 - P_k)^n - (1 - P_k)^n, \tag{2.15}$$

and the expected value of the highest degree is $k_{\text{max}} = \sum_k k h_k$.

3 Natural language processing

The history of *natural language processing* (*NLP*) starts in the 1950 with an article titled *Computing Machinery and Intelligence* by Alan Turing, the father of modern computing. He proposed what is now called the Turing test as a criterion of intelligence [25]. This field of linguistics, computer science and artificial intelligence is focused on interactions between computers and human languages. Of many challenges NLP has to tackle, the most important concern the understanding of natural language—that is, enabling computers to derive meaning from the input like human language.

3.1 Text segmentation

A process of breaking down the text into meaningful units like words or sentences is called *text segmentation*. The term can be applied to either intellectual actions performed by humans while they read the text or artificial processes computer implement, which are the subject of natural language processing.

The problem seems trivial in English, because words have explicit boundaries—spaces—but it most certainly is not trivial in some other written languages where such markers are sometimes ambiguous or are simply absent. However, the space 'character' may not be sufficient also in English, the most apparent example include common contradictions like *don't* $\rightarrow$ *do not*. Another problem is that the users of the language may neither be the native speakers exclusively nor they can use completely grammatically correct spelling and punctuation. Despite the fact that in English, the space is a good approximation of a word delimiter, incorrect use of punctuation makes the problem more difficult; there are many examples such as hyphenated words, emoticons or larger constructs like URIs.

The equivalent of the space character is not found in languages like Chinese or Japanese, where sentences are delimited but words are not and in Vietnamese, where syllables, instead of words, are delimited. All these sophistications make the process very difficult in many cases, but fortunately we are working with English text exclusively in the next chapter.

The Unicode Consortium has published a Standard Annex [26] on Text Segmentation, exploring the issues of segmentation in multiscript texts.

3.2 Word stemming

Stemming is the process of linguistic normalisation, which goal is to reduce inflected (or sometimes derived) words to their *stem*, *base* or *root form*, or more generally a common form of a written word. Stemming programs are commonly referred to as stemming algorithms or stemmers. A stemming algorithm may for example reduce word *cats* to its root word *cat*. It does not mean that the step need to be identical to the morphological root of the word. For example, the word *argues* could be reduced to the stem *argu*, which is not a word.

It is usually satisfactory to reduce related words to the same stem (even if this stem is not in itself a valid root). Algorithms for stemming have been studied in computer science since 1968 [27]. Treating the words with the same stem as synonyms are used by many search engines as a kind of query broadening, a process called conflation [28].

Several types of stemming algorithms exist. Their main differences lay in how well they perform, how accurate they are and what kind of obstacles they may overcome. The most used stemmers are described in following sections.

3.2.1 Lookup algorithms

Lookup algorithms are stemmers that are employing lookup table to find the inflected form of a word. This approach is very simple, fast and easily handles exceptions. The problem is that it is memory or disk space consuming as all inflected forms must be found in the table, and that can make the table size very large. Also new or unfamiliar words are not handled, even if they are perfectly regular, e.g. HD-800s → HD-800 (Sennheiser HD-800 headphones). For languages with simple morphology, like English, the sizes of tables are moderate. For languages like Polish, which are highly inflected tables may contain hundreds of potential inflected forms for each root.

3.2.2 Suffix-stripping algorithms

Suffix-stripping algorithms, in contrast to lookup algorithms, do not rely on a lookup tables, but instead they use a generally smaller list of *rules*, which provide solutions for the algorithm.

Algorithms of this type were introduced by Martin Porter in 1980 [29]. Stemmers like that were very widely used since then and became a *de facto* standard for working with English texts.

Given an input word form they then take these rules and try to find the root form of input word. For example the rules for words ending in *ed*, *ing* or *ly* may just say to remove these endings.

Thanks to the concept of suffix stripping it is much easier to maintain a list of rules like that than to use brute force algorithms (assuming the maintainer knows the challenges of linguistics and morphology and know how to encode suffix stripping rules).

The performance of suffix stripping algorithms if poor when dealing with exceptions, like *ran* and *run*. The solutions produced by suffix stripping algorithms are limited to those parts of speech that have well known suffixes, with not many exceptions. On the other hand, a well formulated set of rules does not exist for all parts of speech so it may be a problem.

3.2.3 Lemmatisation algorithms

Lemmatisation algorithms attempt to improve upon the challenge of exceptions and missing rules for suffix-stripping algorithms. The process involves determining the part of speech first and then applying different normalisation rules for each. The reason why this determination is conducted first is that the stemming rules may have to change depending on a word's part of speech in some languages. These algorithms are based on a simple idea stating that if we are able to obtain more information about the word, then we are able to more accurately apply normalisation rules (which are, more or less, suffix stripping rules).

The main disadvantage of lemmatisation is that it is highly dependent on correct identification of the lexical category the word falls in. Even though the normalisation rules seem to overlap for certain parts of speech, incorrect identification of the lexical category negates the added benefit of this solution over suffix-stripping algorithms.

3.2.4 Stochastic algorithms

Using a probability to identify the root form of a word is a domain of so-called *stochastic algorithms*. They are trained using a table of relations between an inflected and a root form to develop a probabilistic model. The model they create is usually expressed in the form of some linguistic rules, which are complex, similar to those in suffix-stripping. The stemming itself is then is executed by inputting an inflected form to the model learned by algorithms. Then the word is processed in accordance to the internal rules of the model and the root form of a word is produced. This is again similar to suffix stripping, except that the model decides whether to apply the most appropriate rule or just return the word inputted on the grounds of the smallest probability of the output being incorrect.

3.3 Word classification

3.3.1 Stop words

The most common words are called *stop words*. They are usually filtered out prior to, or after, processing of text. They usually have little meaning. There is no single list of stop words that can be used across all tasks. Any group of words may form a list of stop words to serve a particular purpose. Table 3.1 shows 10 most common English words by sorted by their rank [30].

Rank	Word
1	the
2	be
3	to
4	of
5	and
6	a
7	in
8	that
9	have
10	I

Table 3.1: The most common words in English.

3.3.2 Function words (closed class)

Function words, also called *grammatical words* or *structure-class words,* are words that does not have semantic content of their own or they have ambiguous meaning. They are instead defined in terms of their use or of the function they serve. For example, they may express grammatical relationships with other words within a sentence. They may specify the attitude or mood of the speaker as well. Another purpose of these words is to hold sentences together.

Function words might be pronouns, determiners and prepositions. They also may be auxiliary verbs, conjunctions, grammatical articles or particles, all of which belong to the group of closed-class words which means that it is very uncommon to have new function words created in the course of speech. Interjections may sometimes be considered as function words with an exception that they do not belong to the closed class words.

Because function words are used to glue sentences together, it is impossible to isolate them from another words in the sentence as they may indicate the speaker's mental model as to what is being said.

3.3.3 Content words (open class)

Content words, often called *lexical words* or an *open class words,* are words such as nouns, most verbs (although, not it all languages), adjectives and adverbs that carry the content of the meaning of a sentence. They usually refer to some object, an action or some other meaning that may not be linguistic. The meaning of an open class words is that new words can be and, in fact, are added to the lexicon easily. The processes through new words are added are for example compounding, derivation or borrowing (from other languages). Words of this class carry the main articulation of sentences. They are usually inflected, which means they may vary in form, especially in languages that are highly inflected.

3.4 Zipf's law

Zipf's law is an empirical law proposed by American linguist George Kingsley Zipf [31, 32] although the French stenographer Jean-Baptiste Estoup appears to have noticed the regularity before Zipf [33]. The law is defined using mathematical statistics. It refers to the fact that many types of data studied in the physical and social sciences can be approximated with a specific distribution, now called *Zipfian,* which is one of a family of related discrete power law probability distributions.

Zipf's law states that given some corpus[1] of natural language statements, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, *etc.* For example, in the Brown Corpus of American English text, the word *the* is the most frequently occurring word, and by itself accounts for nearly 7% of all word occurrences (69,971 out of slightly over 1 million). True to Zipf's Law, the second-place word *of* accounts for slightly over 3.5% of words (36,411 occurrences), followed by *and* (28,852). Only 135 vocabulary items are needed to account for half the Brown Corpus.

Empirically, a data set can be tested to see if Zipf's law applies by running the regression

$$\log R = a - b \log n, \tag{3.1}$$

where R is the rank of the datum, n is its value and a and b are constants. Zipf's law applies when $b = 1$.

[1]A corpus (pl. corpora) or text corpus is a large and structured set of texts. They are used to do statistical analysis and hypothesis testing, checking occurrences or validating linguistic rules within a specific language territory.

Zipf's law is most easily observed by plotting the data on a log-log graph, with the axes being rank order and frequency. For example, the word *the* (as described above) would appear at $x = \log 1$, $y = \log 69971$. The data conform to Zipf's law to the extent that the plot is linear.

Formally, Zipf's law predicts that out of a population of N elements, the frequency of elements of rank k,$f(k; s, N)$, is:

$$f(k; s, N) = \frac{\frac{1}{k^s}}{\sum\limits_{n=1}^{N} \frac{1}{n^s}}, \tag{3.2}$$

where s is the value of the exponent characterising the distribution.

Zipf's law holds if the number of occurrences of each element are independent and identically distributed random variables with power law distribution $p(f) = \alpha f^{-1-1/s}$ [34].

In the example of the frequency of words in the English language, N is the number of words in the English language and, if we use the classic version of Zipf's law, the exponent s is 1. $f(k; s, N)$ will then be the fraction of the time the kth most common word occurs.

The law may also be written:

$$f(k; s, N) = \frac{1}{k^s H_{N,s}}, \tag{3.3}$$

where $H_{N,s}$ is the Nth generalized harmonic number. The simplest case of Zipf's law is a $1/f$ *function*. Given a set of Zipfian distributed frequencies, sorted from most common to least common, the second most common frequency will occur $1/2$ as often as the first. The third most common frequency will occur $1/3$ as often as the first. The nth most common frequency will occur $1/n$ as often as the first. However, this cannot hold exactly, because items must occur an integer number of times; there cannot be 2.5 occurrences of a word. Nevertheless, over fairly wide ranges, and to a fairly good approximation, many natural phenomena obey Zipf's law.

3.5 Heaps' law

Heaps' law, also called *Herdan's law*, is an empirical law describing the number of distinct words in a document as a function of the length of the document. That type of relation is sometimes called a *type-token relation*. The law can be formulated as

$$V_R(n) = Kn^{\beta}, \tag{3.4}$$

where V_R is the number of distinct words in a text document of size n. K and β are free parameters determined empirically. With English text corpora, K is usually between 10

and 100, and β is usually found in a range of $[0.4; 0.6]$. Heaps' law means that as more text is read, discovery of new, not previously found, words is less apparent.

The law was originally discovered by Gustav Herdan in 1960 [35], but is frequently attributed to Harold Stanley Heaps. Under mild assumptions, the Heaps' law is an asymptotic equivalent of Zipf's law concerning the frequencies of individual words within a text [36]. This is a consequence of the fact that, in general, the type-token relation of a homogenous text can be derived from the distribution of its types [37].

4 Methodology and Implementation

4.1 Data retrieval

The purpose of this thesis was to analyse the discussions that may appear on message boards (or Internet forums) or news groups, hence a set of data like that had to be obtained. I had a freedom of choice, so I decided to take a look at the messages posted on an Internet forum called *head-fi* (http://www.head-fi.org/f/). It is a forum gathering users wanting to talk about audio equipment such as headphones (mainly), amplifiers or speakers. I was already familiar with this message board so I knew—*more or less*—what to expect from the community taking part in conversations on this forum. I also knew the *physical* topological structure so it was easier for me to write the software used to *crawl* the site and save the messages in my database.

Initially, I have anticipated to download every single message posted on this forum from the beginning of its existence and that was neither hidden nor removed—i.e. available to browse—at the time of downloading the data. I looked at the most recent *identifier* (or ID) of a post and knew that the total number of messages posted on the forum was nearing 8.5 million, providing that the posts were not removed during its existence. Section 4.1.2 describes procedures used to browse the forum.

4.1.1 Data structure

Because the messages on the Internet forums are traditionally saved in a relational dabatase I have decided to reflect the structure of this particular forum in my own—also relational—database, but with a simpler, more fit to my needs, structure.

I chose to split the board into three logical parts: *forums*, *topics* (or *threads*) and *posts* (or *messages*) that were also following the order in which the message board was being crawled. Because posts are written by *users* (or *members*), I have also decided to further *normalise* the database by creating a separate table for them. The structure of the database at a time was then as presented in tables 4.1, 4.2, 4.3 and 4.4, containing all the information I could—and wanted—to save.

Field	Description
id	Unique identifier; referenced in topics table as forum_id.
title	The title of the forum as text.

Table 4.1: Forums table structure.

Field	Description
id	Unique identifier; referenced in posts table as topic_id.
forum_id	Reference of a forum, the topic is in.
title	The title of the topic as text.

Table 4.2: Topics table structure.

Field	Description
id	Unique identifier; referenced in posts table as user_id.
name	The name of the user as text.

Table 4.3: Users table structure.

Field	Description
id	Unique identifier.
in_reply_to	A comma separated list of IDs of the posts that were quoted in this post.
topic_id	ID of the topic the post is in.
user_id	ID of the user that is the author of this post.
date	Date and time the message was posted.
body	Post contents as text.

Table 4.4: Posts table structure.

A database table with posts features a *in_reply_to* field which I thought would be good to have to specifically bind messages that quote others. However, it was never used later.

4.1.2 Web crawling

Starting out from the Home Page I had to download the list of forums first. Algorithm 4.1 shows the procedure needed to do that. After running it I had 25 sub-forums available to crawl deeper into the message board.

Algorithm 4.1 Crawl forums list.

```
1: procedure CrawlForumsList(url)
2:     html ← FetchUrl(url)
3:     anchors ← ForumTitleAnchors(html)
4:     for all anchor in anchors do
5:         forumId ← IdFromAnchor(anchor)
6:         forumTitle ← TitleFromAnchor(anchor)
7:         WriteToDB("Forums", forumId, forumTitle)
8:     end for
9: end procedure
```

Then I started with *Headphones (full-size)* forum (the first one on the list) and I began crawling this forum and fetching the topics. This task was a little more complicated, because the topics were not listed all on one page. Instead, they were paginated[1]: 50 topics per page. I have decided not to use *brute force* techniques to crawl the topics list, but to approach the problem more intelligently and simulate web browser along with HTML parser that allowed me to find and follow hyperlinks labelled as 'Next page'. Procedure required to accomplish this task is featured in the listing of algorithm 4.2.

Algorithm 4.2 Crawl forums.

```
1: procedure CrawlForums(forumUrlFormat)
2:     forums = ReadFromDB("Forums") ▷ Set of (forumId, forumTitle) tuples from DB
3:     for all (forumId, forumTitle) in forums do
4:         nextUrl ← PrepareUrl(forumUrlFormat, forumId)
5:         repeat
6:             html ← FetchUrl(nextUrl)
7:             anchors ← TopicTitleAnchors(html)
8:             for all anchor in anchors do
9:                 topicId ← IdFromAnchor(anchor)
10:                topicTitle ← TitleFromAnchor(anchor)
11:                WriteToDB("Topics", topicId, topicTitle, forumId)
12:            end for
13:            nextUrl ← NextUrl(html)             ▷ Taken from 'Next' hyperlink
14:        until nextUrl is not ∅
15:    end for
16: end procedure
```

[1]*Pagination* is a process of dividing the content into discrete chunks, called pages. It is used mostly to limit the number of results displayed to the user in order to both reduce the amount of data the server is required to send and the client is required to receive.

One problem arose during the fetching of post threads: *bumping*. Bumping is a phenomenon occurring on virtually every message board that have time assigned to each posted message. When a member of the forum posts in a thread it will jump to the top of the list since it is the latest updated thread. That meant that after all pages have been crawled, I had to repeatedly crawl the beginning of the list until I found no new topics on the first page of the sub-forum to save to database.

Last step involved browsing the threads in order to save individual messages. It was very similar to the previous one because posts were also paginated. However, this time, bumping was not a problem, since messages inside threads were ordered by time they were posted in ascending order (first-come, first-served), so I only need to crawl each topic once. The process is shown as an algorithm 4.3 below.

Algorithm 4.3 Crawl topics.

```
 1: procedure CrawlTopics(topicUrlFormat)
 2:     topics = ReadFromDB("Topics")      ▷ Set of (topicId, topicTitle) tuples from DB
 3:     for all (topicId, topicTitle) in topics do
 4:         nextUrl ← PrepareURL(topicUrlFormat, topicId)
 5:         repeat
 6:             html ← FetchUrl(nextUrl)
 7:             posts ← Posts(html)
 8:             for all post in posts do
 9:                 postId ← IdFromPost(post)
10:                 postBody ← BodyFromPost(post)
11:                 postDate ← DateFromPost(post)
12:                 userId ← UserFromPost(post)
13:                 quotes ← QuotedFromPost(post)
14:                 WriteToDB("Posts", postId, postBody, postDate, userId, quotes, topi-
    cId)
15:             end for
16:             nextUrl ← NextUrl(html)                ▷ Taken from 'Next' button
17:         until nextUrl is not ∅
18:     end for
19: end procedure
```

Executing the three crawling procedures mentioned above took a long time for a single sub-forum so I have decided to stop further browsing. I had a lot of data to analyse so it was not crucial to continue. Table 4.5 displays the number of entities in the database I have extracted.

Entity	Total count
Forums	25
Topics	111,047
Posts	1,941,970
Users	52,605

Table 4.5: Total count of extracted entities.

At this point, I had one type of nodes that would form my bipartite network: users. Now it was time to get the other type—words—but they had to be extracted first, which I describe in the following section.

4.1.3 Word extraction

Words had to be extracted from the post message, excluding any quotations from earlier posts as they would pollute the actual word usage by users. The process was divided into three parts: *text segmentation*, *word classification* and *word stemming*.

4.1.3.1 Text segmentation

Fortunately, the problem of hyphenation, where words are split by syllables for presentation purposes (in justified text, especially in articles and books) and then conjoined with hyphens is non-existent in Internet forums due to the fact that advanced text rendering (like hyphenation) is not yet supported in widely used web browsers.

After several iterations of trial and error I have finally decided to segment words using a regular expression. The final pattern was: `/\s+|-{2,}|(?!-)\p{P}+/`. It consists of three subpatterns that try to segment words in three ways. The breakdown of this pattern can be found in table 4.6. The '`|`' (also called *pipe*) character denotes an `OR` statement inside this regular expression meaning that any one of the subpatterns provided cause the word to split.

Pattern	Description
\s+	One or more whitespace characters.
-{2,}	Two or more hyphens.
(?!-)\p{P}+	One or more punctuation characters not preceded by a hyphen, without matching said hyphen. It will split when two or more hyphens are encountered or at least one other punctuation character.

Table 4.6: Breakdown of splitting regular expression pattern.

I thought it will be interesting to check whether the number of new distinct words that appeared during segmentation conform to the Heaps' law, which formulated as $V_R(n) = Kn^\beta$ has parameter K typically between 10 and 100, and parameter β typically between 0.4 and 0.6 in english texts. In figure 4.1 the red crosses represent the actual numbers (on axis OX) of words after the number of posts (on axis OY) where blue line is the function $V_R(n)$ that has been fitted to that set of numbers with the values of parameters $K = 83.3 \pm 1.1(1.313\%)$ and $\beta = 0.5984 \pm 0.0009(0.1557\%)$ respectively.

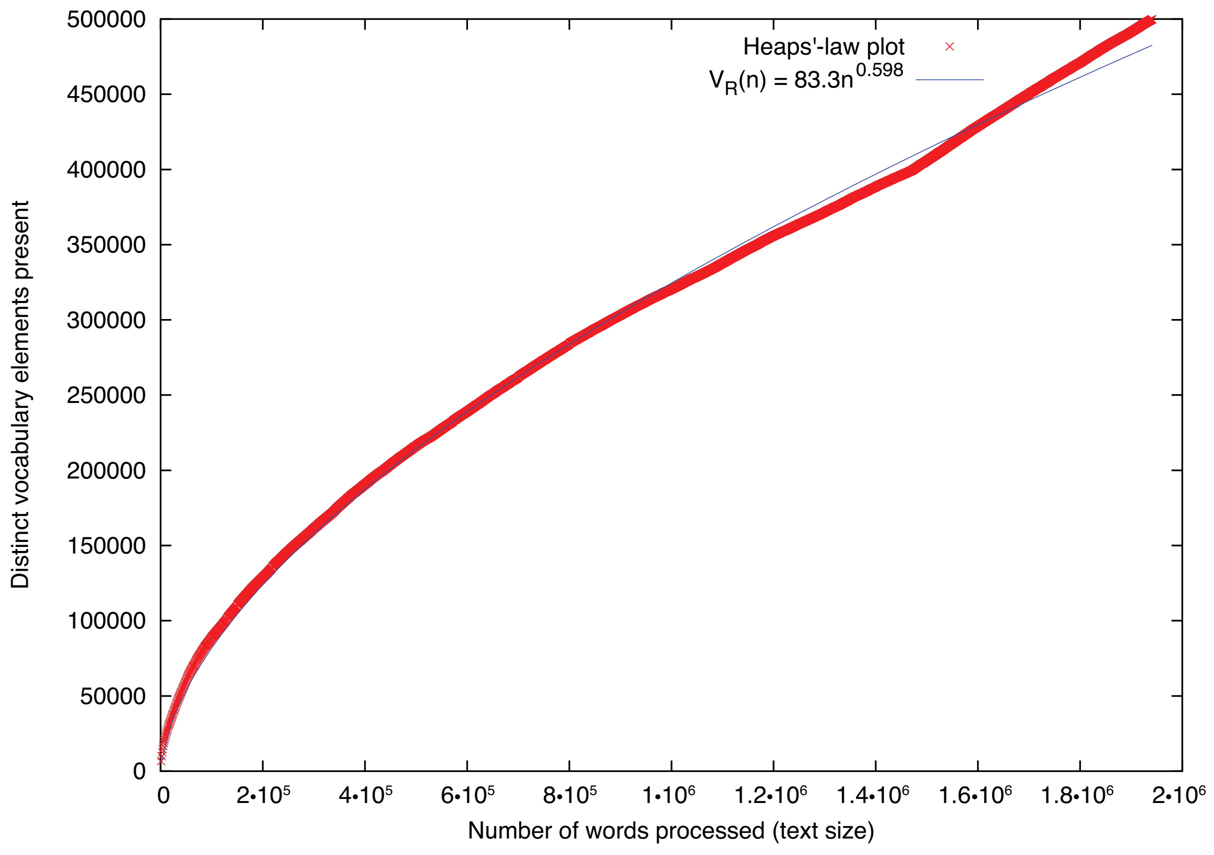


Figure 4.1: Heaps'-law plot in linear scale.

This is a perfect indication that the messages posted by the users of the head-fi forum I am analysing conform to Heaps' law. Figure 4.2 shows the same in logarythmic scale.

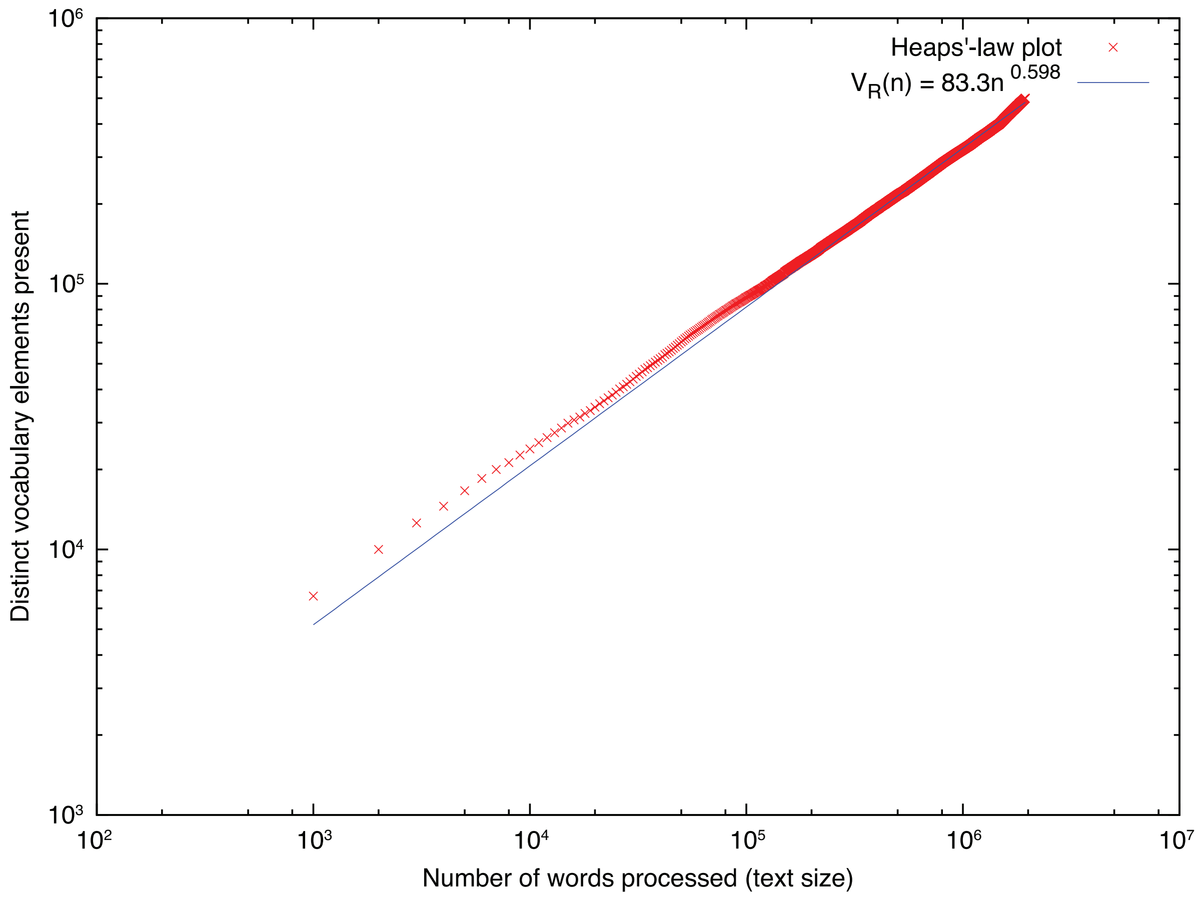


Figure 4.2: Heaps'-law plot in logarithmic scale.

4.1.3.2 Word classification

The three main groups of words I have selected to use are *products*, *opinions* and *prices*. Words that did not fit into one of these were not classified, i.e. left as just regular words.

Products To create a list of keywords identifying products (and brands) I had to browse a lot of music equipment related websites and create a list by parsing menus, categories and other text on these pages. I have also used price comparison services which were a great help since I could browse to *headphones* category and download a list of products of which prices they compared.

A list of brands and products was then manually checked for correctness and the entries that seemed too generic, like *audio*, *sound* or *bass* were removed. After all that I ended having 4,500 keywords related to products.

Opinions Opinions were divided into *positive* and *negative*. Examples of both sub-groups can be found in tables 4.7a and 4.7b.

Positive opinion
love
portable
convincing
recommend
nice
superb
neat
comfortable
beautiful
incredible

(a) Selected examples of positive opinions.

Negative opinion
joke
laugh
distracting
lacking
incompetent
bad
boring
suspicious
horrid
terrible

(b) Selected examples of negative opinions.

Table 4.7: Selected examples of opinions.

Prices Prices were very easy to classify. As the forum is used mostly by *Americans*[2] and *Canadians* the price is usually denoted as \$$x$, USD$x$ or CADx, where \$, USD and CAD are a currency marks and x is the actual price (in various formats, but that does not matter).

4.1.3.3 Word stemming

After words have been split and classified, it turned out that there exist some discrepancies in 'conventions' used by users to discuss the products.

The first part of the process involved normalisation of found words. This had to be done due to problems in finding *the* right word splitting pattern. Some words were prefixed or suffixed by additional characters that were not correct.

Because the forum is using English language exclusively and the results of stemming non-keyword words are not critical I have decided to use non-aggressive suffix-stripping algorithm. I was not interested in finding correct root of a word, because a generic stem would be sufficient. Stems were then interpolated into root words that were being used the most.

What was not covered here were typographical errors that could not easily be corrected in an automated way and the effort of manual intervention would not be justified and using automatic correction according to English dictionary could be too risky.

Separate problem was finding stems for products. Many headphone models differ only by a prefix or suffix letter(s). Additionally, users tend to refer to many models appending a *s* letter to the model name. For example, one user wrote 'I ruined my grado SR80s.'

[2]Inhabitants of the United States of America

while discussing *Grado SR80* headphone model. The problem is that some models have this 'extra' *s* letter appended to the model name itself, example being *Denon AH-NC732s* headphones. This meant that while stemming keywords related to products U had to be *extremely* cautious and careful—any error here would skew the results later.

Fortunately, I had a solid base of products gathered from classification step that made this process somewhat easier, as I could almost automatically remove unnecesary *s* by marking the products that did not have it in their name. The product keyword number was reduced to 3,668 (23% reduction) thanks to that process.

4.1.3.4 Results

The resulting word set I could begin to work with is presented in table 4.8. Keywords represent 4.38% of all words.

Products	**Positive opinions**	**Negative opinions**	**Prices**	**Function words**	**Regular words**	**Total words**
$3,668$	$2,111$	$4,360$	$10,259$	$1,085$	$468,532$	$490,015$[3]

Table 4.8: Breakdown of the number of words and the category they belong to.

The results were saved into separate database table for further reference and usage. Its structure is shown in table 4.9. I have decided to call this table a *relationship* table, because it indicates clear associations between the words written by users as posts.

Field	**Description**
user_id	Unique user reference; found in users table as id.
word_id	Unique word reference; found in words table as id.
post_id	Unique post reference; found in posts table as id.
used	Date the word has been used; equal to post data.
meta	Metadata indicating the type of keyword if necessary.

Table 4.9: Relationships table structure.

[3]The number certainly looks very high but there are over 2 billion words contained in Oxford English Corpus [30]

4.2 Conventional analysis

I have decided to conduct a conventional analysis to check whether we can find some regularities or anomalies that will help identifying groups of users using non-complex network methods.

4.2.1 Time oriented distribution of posts

First, I decided to have a look at the distribution of messages in topics during each year, month and hour during the day. The forum started in June 2001 and has had successively more messages posted per year for 6 consecutive years. Then it suddenly stopped and noted a significant, 32.36% drop in message count in 2008. In this year, the United States of America, as well as pretty much the rest of the World, has suffered from major financial crisis. I never had access to the information about the *page views*[4] of this particular message board, but I suspect that it might have been one of the reasons why an interest of the forum specialised in somewhat expensive equipment has decreased during that time. Figure 4.3 shows the number of messages posted during each year of the data I have gathered (from June 21, 2006 to April 10, 2012).

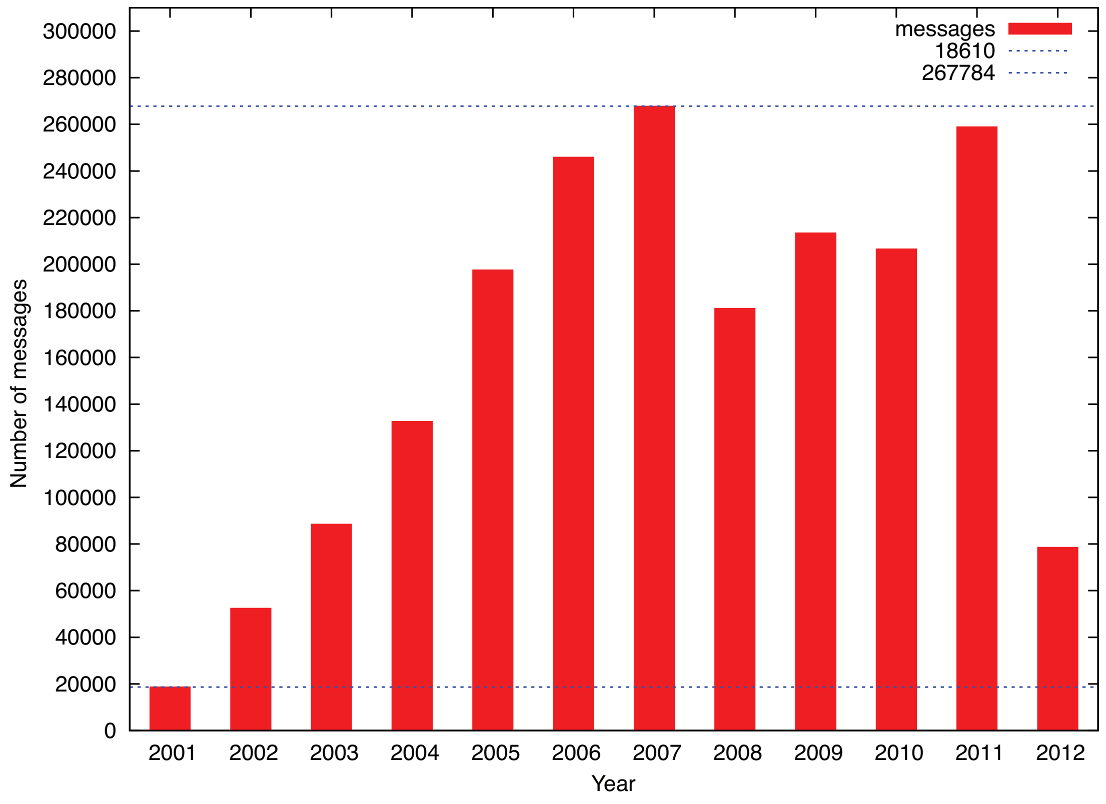
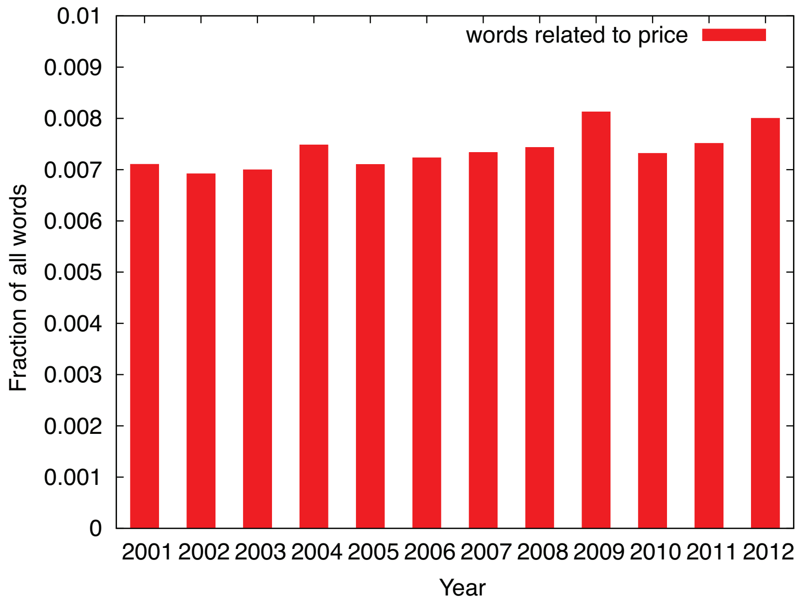


Figure 4.3: Message number distribution by year.

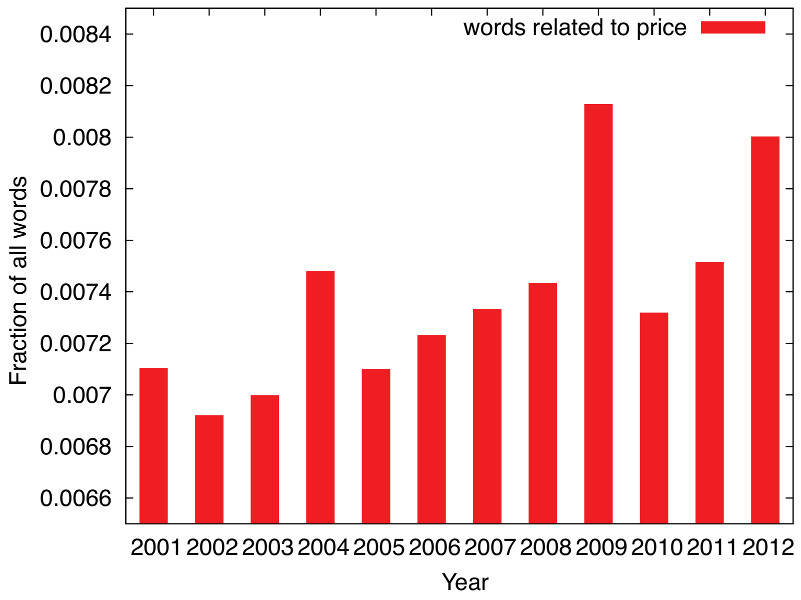
Out of the curiosity I wanted to find out if I can confirm my suspicions by plotting the

[4]Individual page impressions happening when users browse the website

number of words related to prices against the total number of words. Figures 4.4a and 4.4a exhibit this ratio: the former, shown in full scale, and the latter 'zoomed in' to exaggerate the peak usage that has happened during the year 2009. It may or may not be related to the Great Recession of 2009[5] following the financial crisis of 2008. I will not follow on this topic any longer since this thesis is not about interpretation of the data from an economic standpoint.



(a) Full scale.



(b) Scale zoomed in.

Figure 4.4: Ratio of the usage of words related to prices to the total number of words.

[5]C. Rampell from New York Times (2009-03-11). *'Great Recession': A Brief Etymology.* (available at http://economix.blogs.nytimes.com/2009/03/11/great-recession-a-brief-etymology/)
D. Wessel from The Wall Street Journal (2010-04-08). *Did 'Great Recession' Live Up to the Name?* (available at http://online.wsj.com/article/SB10001424052702303591204575169693166352882.html)

I do not think that it is possible to identify any groups of users here, other than those that registered early or not. Of course we could also try to extract the users that have signed up and did not post for some period of time or any other idea that can come to one's mind, but that information would not be very useful in my opinion.

Next, on figure 4.5 we can see the distribution of messages grouped by month during which they were posted. One can clearly tell 4 peaks on January, February, March and December. I would account it to the fact that there is a 'Christmas fever' period in December, where people are usually shopping for more gifts (so-called Christmas-gift-shopping patterns) [38, 39]. The explanation for the several months following December could be fairly trivial, since these months are ideal for discussing recently obtained Christmas gifts. What would be interesting from this thesis standpoint is that we can at last identify a group of users that may be a target for potential marketing campains: people wanting to buy something.

Days, or weeks immediately preceding and succeeding Christmas (or any other period of increased shopping) also seem to be the perfect time for malicious actions like advertorials[6] or sponsored reviews.

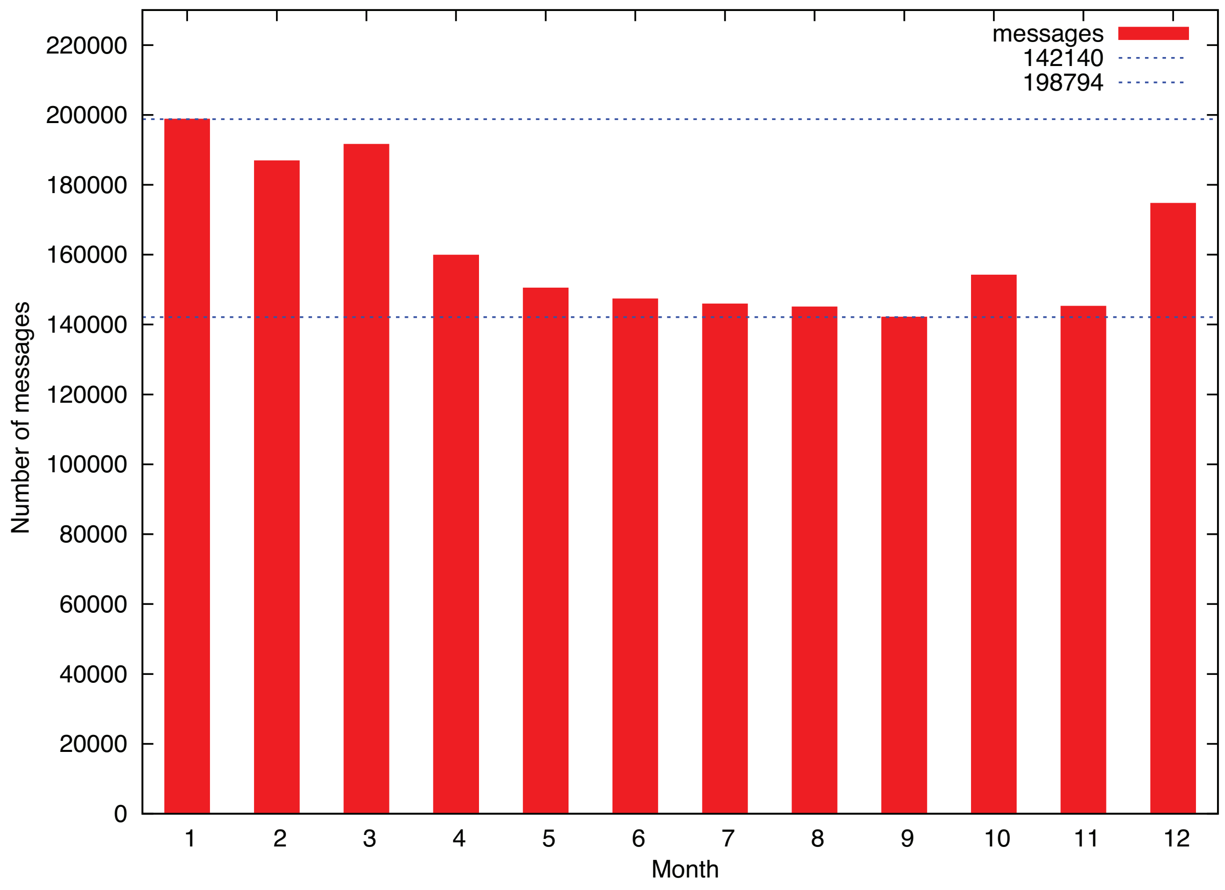


Figure 4.5: Message number distribution by month.

The last figure in this section, figure 4.6 shows the distribution of messages posted

[6]An advertisment in a form of an editorial.

throughout the day by the hour they were posted. The results do not seem to differ from the regular charts we see when analysing Internet traffic [40]. The significantly lower number of messages posted during night hours is most likely related to the fact that, in general, people seem to sleep during this time of day [41]. The number of messages in the night is huge anyway (it is around a half of the top number of messages during the day) which might indicate that the users of the forum are either very active or the community is international (i.e. located in different timezones). Otherwise, the night fluctuations would drift towards 0.

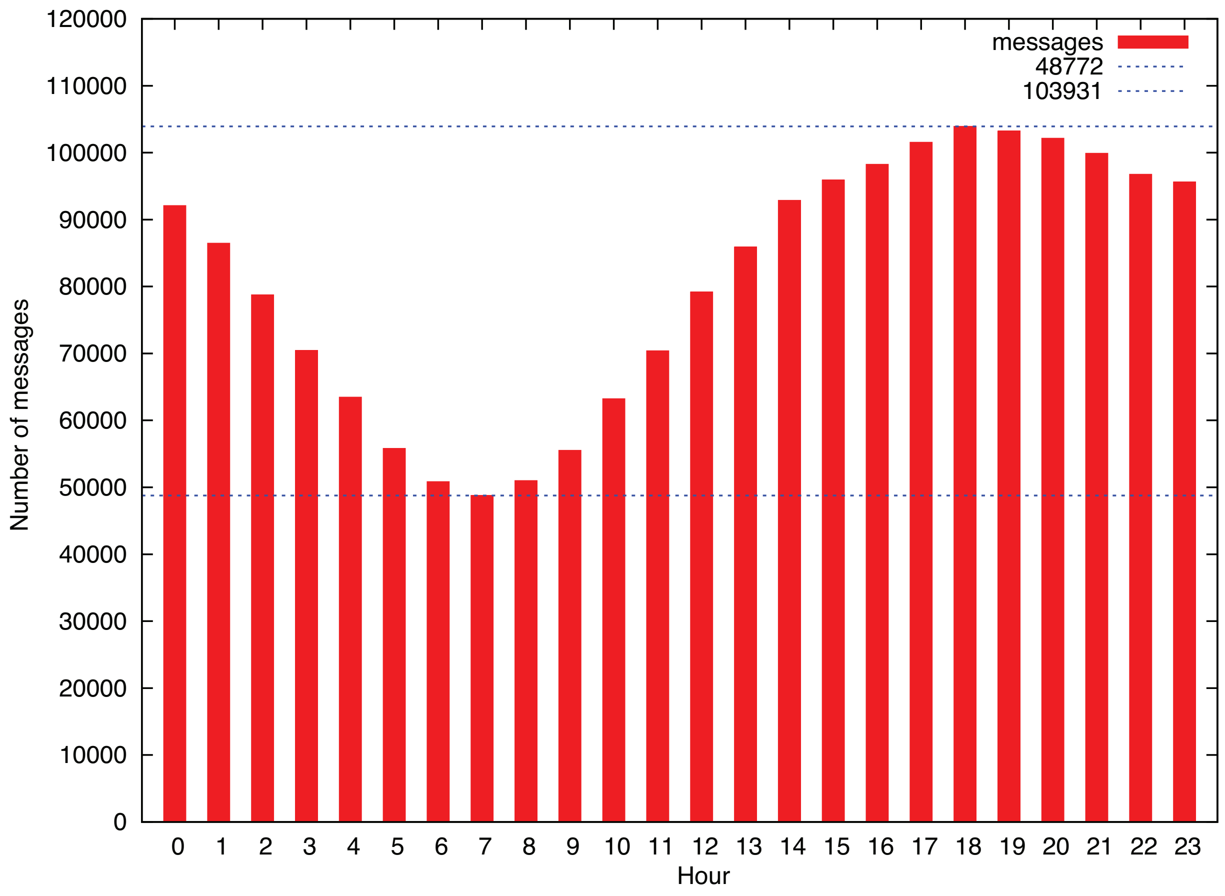


Figure 4.6: Message number distribution by hour.

I would not try to identify any particular groups of users based on the hours they post their messages at. This could be done if we had access to the timezone information of each user. However, if the members of the message board was posting from the same timezone this task would be trivial. It would be sufficient to calculate a distribution of the time the messages were posted for each user and then group them by having similar distribution.

4.3 Static network analysis

The bipartite network was created using the users and the words filtered out of stop words and function words. Word list was then additionally limited with Zipf's law in mind: nodes representing 100 of the most occurring words (not being keywords) and 10% of the words occurring the least were removed from the network (these were with 11 or less occurrences). Table 4.10 shows examples of words that were removed. It is understandable that they would only noise the results.

The most used
guys
question
response
lol
btw

(a) Some of the most used.

The least used
wehre
phenominon
inmate
pacino
more

(b) Some of the least used.

Table 4.10: Selected examples of the most and the least used non-keyword words.

4.3.1 Network simplification

Network built using above data was made out of $72,363$ nodes, of which $51,938(71.77\%)$ being users and $20,435(28.23\%)$ being words, connected by $5,497,840$ edges. This was a massive network in a need of a major simplification. The number of edges was especially important to reduce. I have chosen an edge pruning network simplification, introduced by Zhou *et al.* [42]. They have proposed a generic framework of network simplification by sharing new methods for pruning the edges while keeping the graph maximally connected. It is important to note that this method is lossy, which means we lose some information from the graph. The method is intended to be used with weighted graphs. In our network, the weights represent the probability that the user have used the word, i.e. the number of times he used this particular word divided by the number of total words he or she wrote. Then, the connectivity would be defined as the maximum probability.

Additional definitions are required to understand the problem. Let path P be a set of edges $P = (u_1, u_2), (u_2, u_3), \ldots (u_{k-1}, u_k) \in E$. I use the notation $P(u_1 \to u_k)$ to represent the path between u_1 and u_k (u_1 and u_k being endvertices of P).

The problem is parameterised with a *path quality function* $q(P) \to \mathbb{R}^+$. The form of this function is dependent on the type of the graph. In our graph it is the probability that a path exists. We can then assume that the value of any path quality function in this graph is positive, and that a larger value of q indicates better quality.

Given two nodes u and v in a weighted graph, they might be linked by a direct edge or a path, or none when the graph is disconnected. A simple way of quantifying if and how strongly they are connected is to examine the quality of the best path between them [43]. Thus, the connectivity between two nodes $u, v \in E$ is defined as

$$C(u, v; E) = \begin{cases} \max_{P \subset E : P(u \to v)} q(P) & \text{if such } P \text{ exists,} \\ -\infty & \text{otherwise.} \end{cases} \tag{4.1}$$

A natural measure for the *connectivity* of this graph is then the average connectivity over all pairs of nodes

$$C(V, E) = \frac{2}{|V|(|V| - 1)} \sum_{u,v \in E, u \neq v} C(u, v; E). \tag{4.2}$$

This network was connected so there is no need to calculate the connectivity for each disconnected component and we know that $C(V, E) > 0$.

The goal would to produce a graph $H(V, F)$ that is simplification of graph G and $F \subset E$ such that the *ratio of connectivity* is maximised. A ratio of connectivity is defined as

$$rk(V, E, E_R) = \frac{C(V, E\ E_R)}{C(V, E)}, \tag{4.3}$$

where $E_R \subset R$ is a set of edges to be removed from the graph and $C(V, E\ E_R)$ is the connectivity of the graph after edges E_R has been removed. The connectivity of the graph obviously cannot be increased while removing edges, thus $rk = 1$ means that the connectivity was left unchanged, $0 < rk < 1$ means that the removal resulted in some loss of connectivity, while $rk = -\infty$ means that the graph has been disconnected.

Algorithm 4.4 Naive approach.

```
1: procedure PruneNaively($G(V, E)$)
2:     Sort edges $E$ by weights in ascending order.
3:     $F \leftarrow E$
4:     $n \leftarrow (|E| - (|V| - 1))$
5:     $i \leftarrow 1$
6:     $j \leftarrow 1$
7:     while $i \leq n$ do
8:         if $C(u, v; F \setminus \{e_j\}$ is not $-\infty$ then
9:             $F \leftarrow F \setminus e_j$
10:            $i \leftarrow i + 1$
11:        end if
12:        $j \leftarrow j + 1$
13:    end while
14:    Return $H = (V, F)$
15: end procedure
```

I did not have enough time to conduct *optimal* edge pruning so I have decided to conduct pruning randomly using *naive approach* and see its effects, keeping in mind that rk should be between 0 and 1. Algorithm 4.4 shows the approach I have selected. It sorts edges by their weights in ascending order first and then iteratively removes edges from the top of the lists when this operation would not cause the graph to be cut. The most important reason why I favoured this approach over the *brute force* (that is not described here) algorithm is that it consists of less steps and is much less time consuming (in fact, it is the quickest) [42]. I could not use *path simplification* or *combinational approach* because it would remove too much information important in my opinion.

Figure 4.7 shows the effect of the naive approach of pruning. The network has 115 nodes and 134 edges. A total of 20 edges were removed, 19 of which with a weight of 1 and 1 of weight 2. For that kind of network the results are very good: removal of $\approx 15\%$ edges resulted in only $\approx 4\%$ loss of connectivity. Unfortunately, the average path length has increased from 3.35 to 5.53 which is illustrated by the node neighbourhood heat map that was calculated using distance from the node with the greatest degree (the biggest one), with edge weights in mind—the less intense the colour, the more distant the node is.

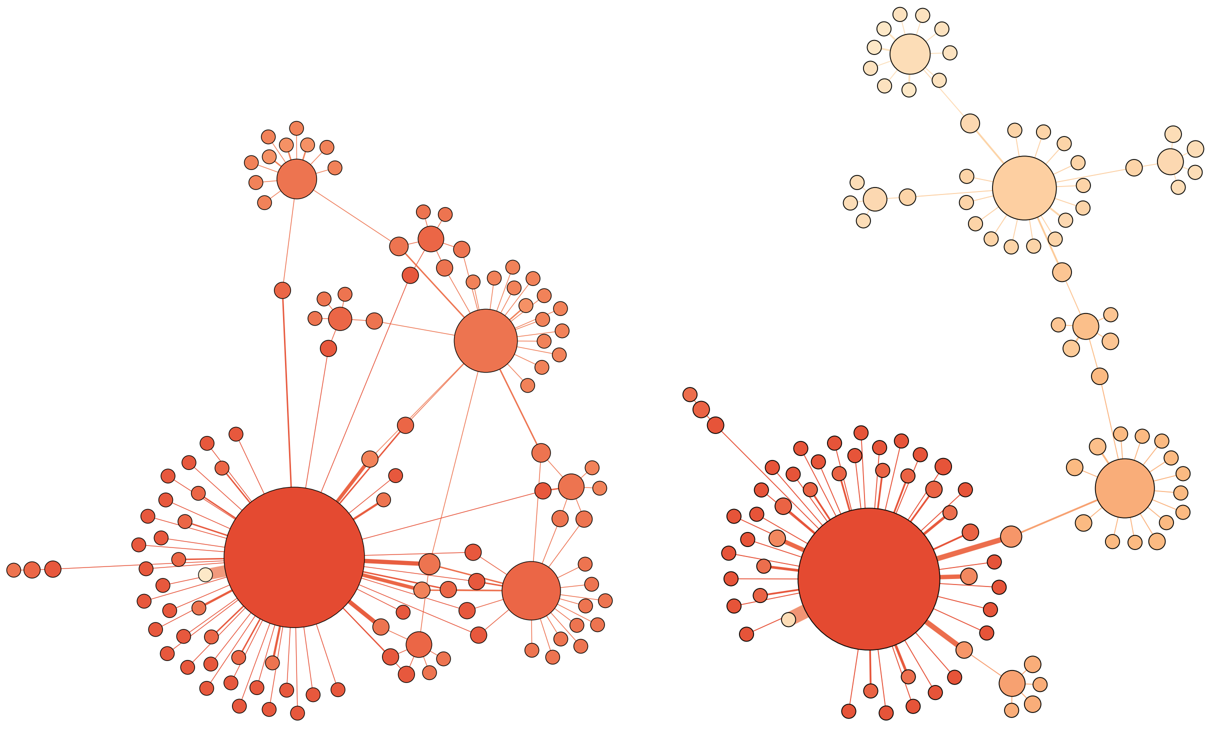

(a) Before pruning. (b) After pruning.

Figure 4.7: The effect of pruning of example network.

The results of pruning of the whole network are very surprising. It was possible to remove $2,613,126$ edges, which makes a 47.53% of the network! The whole process took around 5 minutes and 12 seconds on my setup: one core of Intel i7 (3740QM) processor and 16GB of RAM. What is even more surprising is the fact that the original connectivity was retained. It is very hard to say why it is the case, but with a high probability I would blame the word extraction process, especially the word splitting pattern which after even so many iterations was apparently not perfect enough.

4.3.2 Network properties

The network was then evaluated for its basic properties. They are displayed in table 4.11. N, M and E denote the number of users, words and edges connecting them, respectively. $\langle k \rangle$ and $\langle d \rangle$ are the average user degree and average word degree. The network is made up of one connected component. γ_k and γ_d are the exponents of the power-law degrees of user degree distribution and word degree distribution respectively. l_G is an average path length, C is the global clustering coefficient and $\langle C_i \rangle$ is the average local clustering coefficient.

N	**M**	**E**	$\langle k \rangle$	γ_k	$\langle d \rangle$	γ_d	l_G	C	$\langle C_i \rangle$
$51,938$	$20,435$	$2,884,714$	55.54	1.63	141.23	1.04	2.68	0.0438	0.0121

Table 4.11: Network properties.

Due to the fact that the network is bipartite, we have to check two degree distributions: one for users and one for words they have written. both to follow the power-law

$$\begin{aligned} p(k) &\sim k^{-\gamma_k}, \\ p(d) &\sim d^{-\gamma_d}, \end{aligned} \tag{4.4}$$

with the values of the exponents γ_k and γ_d shown in table 4.11.

Figure 4.8 shows degree distribution of user nodes. One of the first things that is noticeable while looking at this figure is that the degree distribution for degrees in a range $[1 : 30]$ is approximately linear and flat. The reason is that there are a lot of users that sign up to this forum with a particular purpose in mind, such as to ask a question or get a recommendation. They create a single topic for their needs and never discuss outside of it and then never return to the forum after they posted their last message. In fact, users having degree $k \leq 30$ and posting only within a single topic represent 69.35% of all the users having such degree. This *in my opinion* explains why the degree distribution $p(k)$ is skewed as much in the degree range $[1 : 30]$. If we take the other part of the range, i.e. $[31 : 3950]$, we see that users like that represent only 17.78% of the users from that degree range. What is also interesting is rather large number of *hubs*, i.e. users with degree significantly higher than average. Total number of nodes with degree $k \geq 1000$ is 1.24%, which makes one user of 80 an important node.

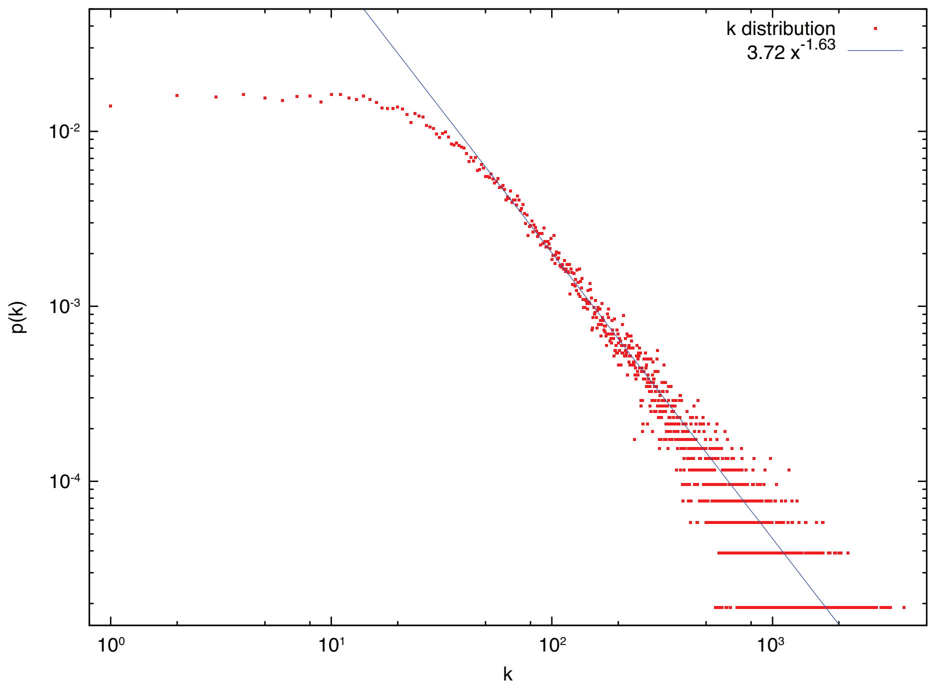


Figure 4.8: Degree distribution of users.

The degree distribution of word nodes is shown on figure 4.9. There are no surprises here other than the value of the power-law exponent γ_d which is very low, especially where we find the value of this exponent in most real-world scale-free networks to be between 2 and 3. I think it is quite safe to assume that if we had no typographical errors in the network and we didn't have to conduct the stemming process, the value would be higher.

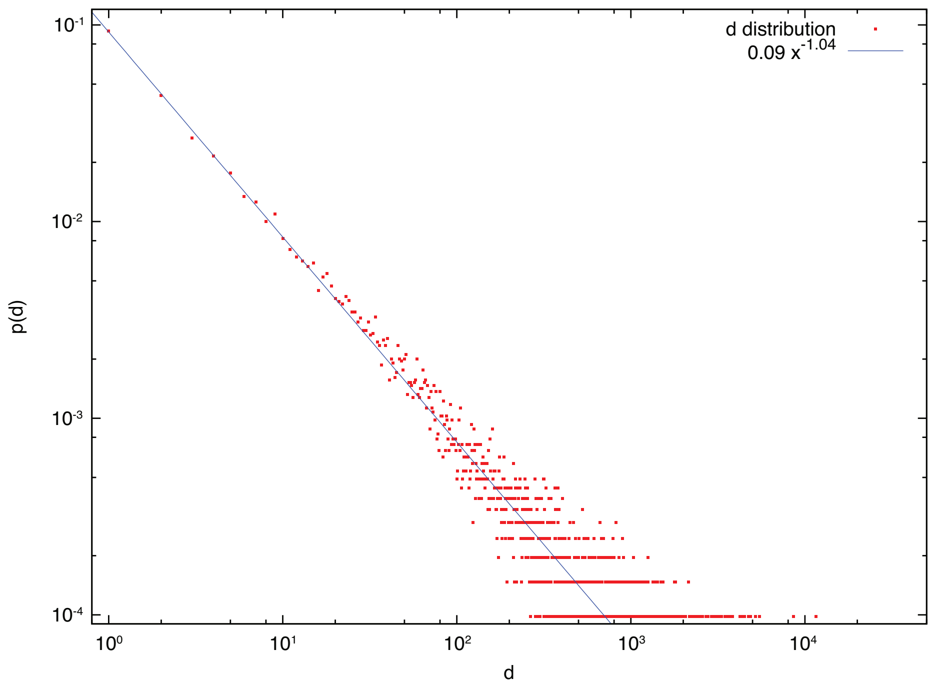


Figure 4.9: Degree distribution of words.

4.4 Dynamic network analysis

4.4.1 Discretisation of data

Because the messages are posted on this forum virtually every minute the information about them might be considered as being continuous in time. In order to study the evolution of the network in time, there is a need to define discrete time intervals that will split the network in parts. Networks that are dynamic are called longitudinal networks.

The time interval using which the data had to be split had to be determined empirically with a network to be kept in a size that would be representational. The choice was between a *week-long*, *two weeks*-long and a *month*-long interval. Eventually, a 30 *day*-long time interval was selected, main reason being the fact that the date span of downloaded messages was almost 12 years long which created 141 data buckets. Additionally, message threads are often more than a week long, so we did want to avoid the situation where most threads are split into several buckets. It also allowed different kind of distributions stand out more because they were created with more data.

The biggest downside was that the network sizes were still very big, thus all computations that had to be performed on a whole network took more time.

4.4.2 Estimating users' opinions about products

In order to estimate the opinions different users have about different products I had to prepare a script traversing the posts saved in my database and saving a value representing opinion. I had to create a separate table in my database for that, which structure is shown in table 4.12.

Field	Description
user_id	Unique user reference; found in users table as id.
word_id	Unique word reference; found in words table as id.
opinion	A real value representing opinion.
bucket	Bucket number from which the opinion value was calculated.

Table 4.12: Opinions table structure.

The script was based on algorithm 4.5. I have run it against every bucket, first gathering the list of users and keywords in that bucket. The algorithm was ran for every user and keyword. It fetched all messages written by the user in a bucket it was ran for. Then for every post it has checked whether the keyword was used within the post and returned position of each occurrence of that word. The position was established to be the nth 'token' in the message, where 'tokens' were obtained in the exact same way as words in section 4.1.3.1. Then the algorithm extracted all words marked as opinions from the message body and for each of these opinions it has checked their position as it did with keywords. Next, it has checked whether the opinion was *positive* or *negative* and applied a measure on a keyword based on squared distance between itself and the opinion biased positively or negatively.

Algorithm 4.5 Estimating user's opinion about product.

```
 1: procedure EstimateOpinion(user, keyword, bucket)
 2:     estimate ← 0
 3:     posts ← Posts(user, bucket)
 4:     for all post in posts do
 5:         keywordPositions ← KeywordPositions(keyword, post)
 6:         for all keywordPosition in keywordPositions do
 7:             opinions ← Opinions(post)
 8:             for all opinion in opinions do
 9:                 opinionSign ← OpinionSign(opinion)
10:                 opinionPosition ← OpinionPosition(opinion, post)
11:                 distance ← |keywordPosition - opinionPosition|
12:                 estimate ← estimate + (opinionSign / distance$^2$)
13:             end for
14:         end for
15:     end for
16:     return estimate
17: end procedure
```

4.4.2.1 Measure

Let e_o be an opinion estimate obtained using a method described above. Its value should be found in a range described by the following sum

$$\frac{1}{1^2} + \frac{1}{2^2} + \frac{1}{3^2} + \frac{1}{4^2} + \ldots = \sum_{n=1}^{\infty} \frac{1}{n^2} = \frac{\pi^2}{6}, \tag{4.5}$$

and as each keyword can have opinions written before and after itself, we have two multiply the sum in equation 4.5 by a factor of 2. Hence,

$$-\frac{\pi^2}{3} \leq e_o \leq \frac{\pi^2}{3}. \tag{4.6}$$

Application of squared distance as a base for the measure favours opinions that are very close to keywords. This was extremely important especially for long posts so that keywords mentioned at the end of the message did not get influenced by an opinion mentioned at the beginning of it. With this approach it is expected for the most opinions to be found in a range of $[1/4; 1/4]$—most opinions would be generally no further away than 2 words from a keyword.

Because measuring data in such way is not perfect and may create results contradictory to the real data I have decided to introduce a *neutral* opinion using the middle range of the numbers. Hence, opinions were assigned in a following way:

- *positive:* $L \leq e_o \leq \frac{\pi^2}{3}$,
- *neutral:* $-L < e_o < L$,
- *negative:* $-\frac{\pi^2}{3} \leq e_o \leq L$,

where $L = 0.01$ is an empirically found constant allowing for fairly even division of opinion types. Table 4.13 shows the number of opinions found, divided by the type of the opinion.

Positive	**Neutral**	**Negative**	**Total**
$4,219,548$ (42.99%)	$3,398,282$ (34.62%)	$2,197,055$ (22.39%)	$9,814,885$ (100%)

Table 4.13: Number of opinions about products found, by the opinion type.

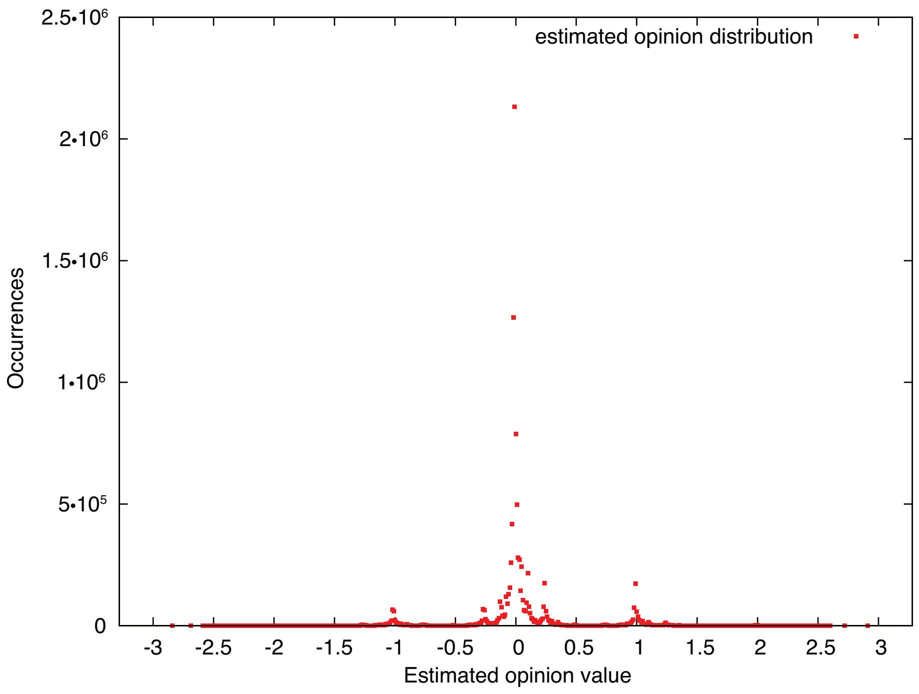


Figure 4.10: Distribution of opinions.

The maximum and minimum value of the estimated opinion were found to be approximately 2.924 and -2.825 respectively, so they lay within the limits shown in equation

4.6. Figure 4.10 shows distribution of estimated opinion values. As expected, most of the opinions fall in the $[{}^1/_4; {}^1/_4]$ range.

There are also visibly more opinions on the positive side of the scale than on its negative counterpart. The most dense is the neutral region but it is the consequence of having so much opinions in only 0.02 wide scale. 'Harmonic'-like peaks are conspicuous at -1, $-{}^1/_4$, ${}^1/_4$ and 1, which one can see closer on figure 4.11 using a logarithmic scale for Y axis. Additionally we can see peaks at -2 and 2 which were not distinguishable on a linear scale.

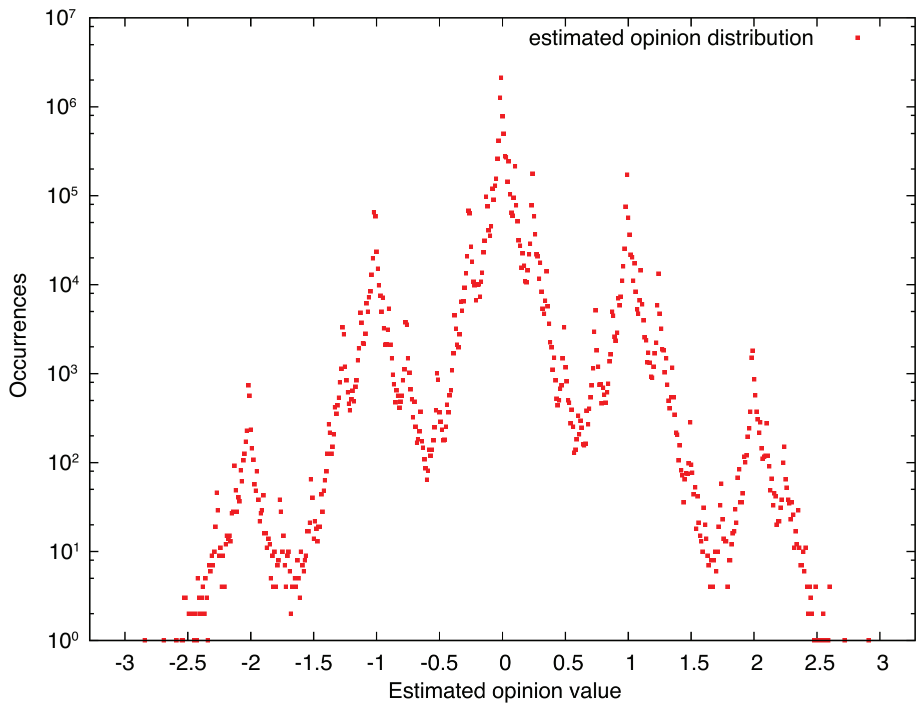


Figure 4.11: Distribution of opinions with Y axis in logarithmic scale.

The visible peaks are a clear evidence that the majority of messages with opinions are short enough for only several opinions influencing a keyword. Otherwise, the distribution would have had a more smooth, cone-like shape. The perfect cone-like shape would be achieved if posts were consisting of one keyword only with a lot of opinions written on both sides of it. Also, the advantage of positive opinions is more evident using this scale.

4.4.2.2 Problems

Checking *opinion sign* is not as easy as determining whether the opinion is positive or negative. The problem is that there exist modifier words that completely change the meaning of word. Examples of words like that include *not* or *don't*.

In order to mitigate this problem and to bias the opinions correctly it was necessary to check for existence of such modifier words in a proximity of opinion words.

Following expressions illustrate the problem: '*not* **good**,' '*not* sufficient **good**,' '*not* at all **good**.' One can clearly see that it is not necessary to check just one word before an opinion.

I have decided to cover all three cases like that even though there may be more edge cases. Hence, modifier words were only checked as far as two words from an opinion and were not taken into consideration if there were another opinion or keyword in between them, for example in messages like '*not* **bad**, **good** even,' where not stopping after one opinion have been found would cause the modifier to act on both opinions instead.

4.4.3 Average path length

Figure 4.12 shows changes of average path length calculated for each bucket of data. The general trend shows that the average length of paths in the network has been decreasing. This may be due to the fact that most of the users did not leave the forum and continued to write messages while retaining their connections. Nevertheless the path lengths are much higher, in a range between 3.27 and 3.55, than the one calculated for the whole network which was calculated at 2.68. This is clearly because of the whole network being significantly larger (having more edges), thus having higher probability for connections to exist.

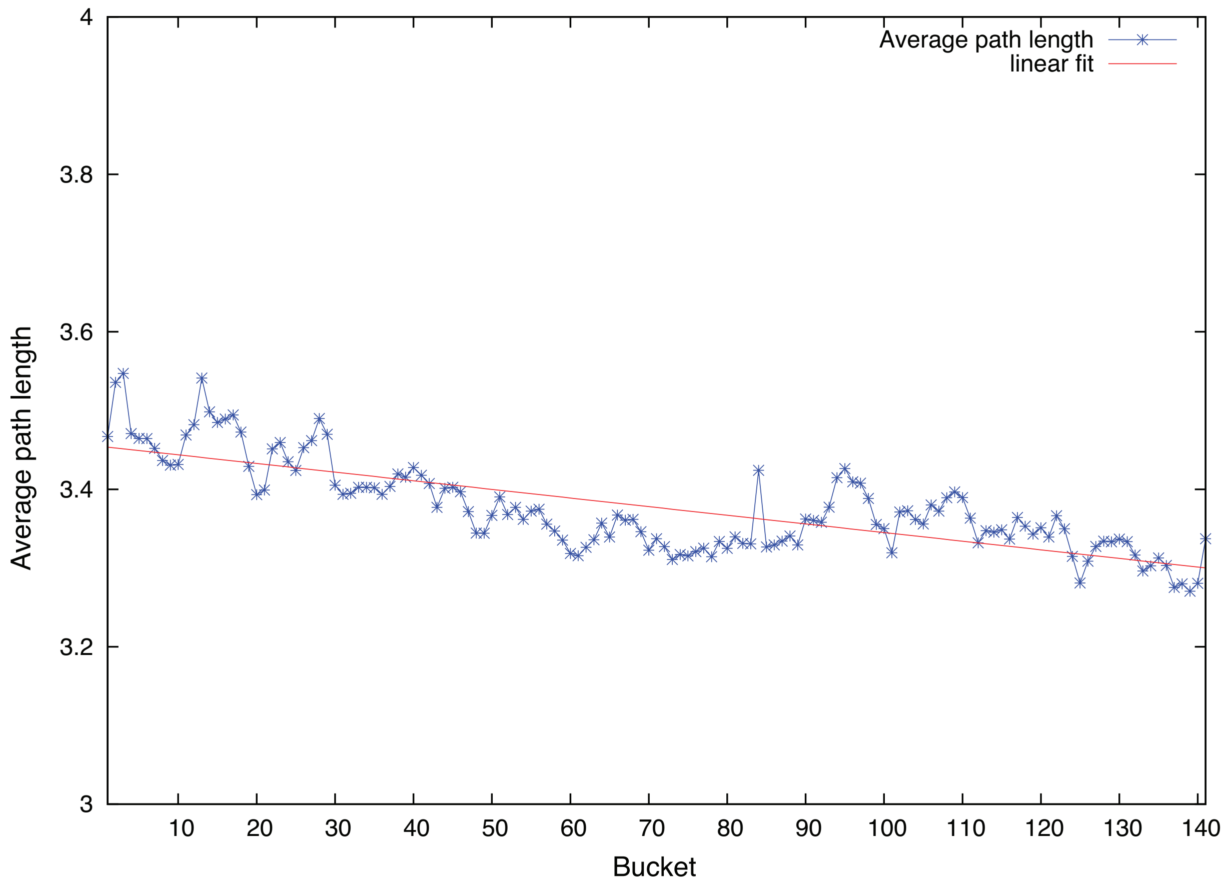


Figure 4.12: Changes of average path length over time.

4.4.4 Collaborative Similarity

Ming-Sheng Shang *et al.* have proposed a new index called *collaborative similarity*, to quantify the diversity of tastes based on the collaborative selection [3]. This newly proposed index sounds promising for the data under analysis in this thesis. It may be a good idea to explore and look at the *user-message network* from the other angle and if we can find that the user selection is highly clustered—i.e. not that much diverse—then we will be able to easily identify main groups of users.

4.4.4.1 The concept

Figure 4.13 illustrates a small bipartite network that consists of six users and eight objects. The degree of user i, denoted by k_i, is defined as the number of objects connected to i. Analogously, the degree of object α, denoted by d_α, is the number of users connected to α. For example, as shown in figure 4.13, $k_i = d_\alpha = 3$. The density function, $p(k)$, is the probability that a randomly selected user is of degree k, while the cumulative function, $P(k)$, denotes the probability that a randomly selected user is of degree no less than k.

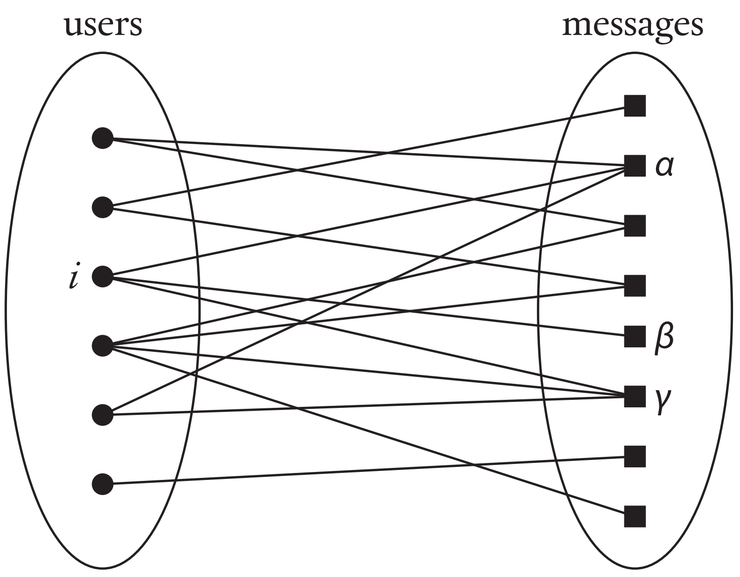


Figure 4.13: Small user-message bipartite network.

The *nearest neighbors' degree* for user i, denoted by $d_{nn}(i)$, is defined as the average degree over all the objects connected to i. For example, as shown in figure 4.13,

$$d_{nn}(i) = \frac{d_\alpha + d_\beta + d_\gamma}{3} = \frac{7}{3}. \tag{4.7}$$

The degree-dependent nearest neighbors' degree, $d_{nn}(k)$ is the average nearest neighbors' degree over all the users of degree k, that is,

$$d_{nn}(k) = \langle d_{nn}(i) \rangle_{k_i = k}. \tag{4.8}$$

Corresponding definitions for objects, say $p(d)$, $P(d)$, $k_{nn}(\alpha)$ and $k_{nn}(d)$, are similar and thus omitted here.

The traditional clustering coefficient [15] cannot be used to quantify the clustering pattern of a bipartite network since it always give a zero value. Lind *et al.* [44] proposed a variant counting the rectangular relations instead of triadic clustering, which can be applied to general bipartite networks. However, this letter aims at a special class of bipartite networks, and thus we propose a new index to characterise the clustering selections[7] resulted from the collaborative interests of users. A standard measure of object similarity according to the collaborative selection is the *Jaccard similarity* [45].

$$s_{\alpha\beta} = \frac{\Gamma_\alpha \cap \Gamma_\beta}{\Gamma_\alpha \cup \Gamma_\beta}, \tag{4.9}$$

where Γ_α and Γ_β are the sets of neighbouring nodes of α and β, respectively. Obviously, $s_{\alpha\beta} = s_{\beta\alpha}$ and $0 \leq s_{\alpha\beta} \leq 1$ for any α and β. For example, as shown in figure 4.13, $s_{\alpha\beta} = s_{\beta\gamma} = 1/3$ and $s_{\alpha\gamma} = 1/2$. The collaborative similarity of user i is then defined as the average similarity between i's selected objects:

$$C_u(i) = \frac{1}{k_i(k_i - 1)} \sum_{\alpha \neq \beta} s_{\alpha\beta}, \tag{4.10}$$

where α and β run over all i's neighbouring objects. For example, as shown in figure 4.13, the collaborative similarity of user i is $C_u(i) = 7/18$. According to the definition, a user whose collections are very similar to each other will have high collaborative similarity. For example, a user who only watches science fiction movies is probably of higher collaborative similarity than the one who has very diverse interests of movies. The user collaborative similarity of the whole network is defined as

$$C_u = \frac{1}{N'} \sum_i C_u(i), \tag{4.11}$$

where i runs over all users with degrees larger than 1 and N' denotes the number of these users. The degree-dependent collaborative similarity, $C_u(k)$, is defined as the average collaborative similarity over all the k-degree users. Corresponding definitions for objects are as following:

$$C_o(\alpha) = \frac{1}{d_\alpha(d_\alpha - 1)} \sum_{i \neq j} s_{ij}, \tag{4.12}$$

where $s_{ij} = \left| \frac{\Gamma_i \cap \Gamma_j}{\Gamma_i \cup \Gamma_j} \right|$ is the Jaccard similarity between users i and j and:

$$C_o = \frac{1}{M'} \sum_\alpha C_o(\alpha), \tag{4.13}$$

[7]Here the term 'clustering' describes the fact that a user's selections are usually very similar to each other, and may belong to a few clusters or communities according to the standard clustering analysis or community detection.

where M' denotes the number of objects with degrees larger than 1. $C_o(d)$ is the average collaborative similarity over all the d-degree objects.

4.4.4.2 Results

Bucket	N	M	E	$\langle k \rangle$	$\langle d \rangle$	C_u	$\bar{s}_m$	C_m	$\bar{s}_u$
1	193	2616	29560	78.02	5.76	0.1608	0.1046	0.1278	0.1453
11	417	3647	55201	68.54	7.84	0.1376	0.0955	0.1182	0.1148
21	609	3787	73964	62.79	10.10	0.1253	0.0936	0.1167	0.1028
31	718	4027	89545	63.74	11.37	0.1235	0.0926	0.1166	0.0986
41	942	4247	93741	55.60	12.33	0.1070	0.0860	0.1047	0.0857
51	1318	4622	118121	51.71	14.75	0.0975	0.0812	0.0995	0.0768
61	1966	5232	191560	54.71	20.56	0.0988	0.0849	0.1033	0.0747
71	1701	5095	89352	52.53	17.54	0.0975	0.0800	0.0996	0.0741
81	1905	5090	91795	48.19	18.03	0.0932	0.0787	0.0985	0.0711
91	1680	4774	77094	45.89	16.15	0.0900	0.0775	0.0935	0.0736
101	1817	5544	105478	58.05	19.03	0.1096	0.0867	0.1103	0.0799
111	1743	4989	81130	46.55	16.26	0.0918	0.0777	0.0961	0.0716
121	1706	4900	81357	47.69	16.60	0.0968	0.0784	0.1000	0.0755
131	1994	5562	104444	52.38	18.78	0.0986	0.0796	0.1011	0.0732
141	2080	5250	94994	45.67	18.09	0.0915	0.0757	0.0962	0.0712

Table 4.14: The basic properties of every 10th data bucket.

Table 4.14 summarises basic statistics of selected data buckets. N, M and E denote the number of users, messages and edges connecting them, respectively. $\langle k \rangle$ and $\langle d \rangle$ are the average user degree and average message degree. C_u and C_m are the average collaborative similarity for users and messages, and for comparison, $\bar{s}_m$ and $\bar{s}_u$ are the average similarities over all message pairs and over all user pairs, respectively. The user selection is considered to be poorly clustered (i.e., more diverse) because the condition for that is $C_u \gg \bar{s}_m$ whereas C_u is the same order of magnitude as $\bar{s}_m$. Unfortunately, this basically means that it will not be easy to identify groups of users that may have formed in this message board and I had to abandon this somewhat interesting index. Figure 4.14 show the way C_u and $\bar{s}_m$ were changing over time. In no bucket, C_u was much greater than $\bar{s}_m$ and even in bucket 84 they were pretty close to each other: 0.0746 vs 0.0683 respectively. This does not however disqualify that method completely. The results only mean that it will not be easy to find any anomalies. There are 52605 users in the database so I could just run over a $C_u(i)$ of every user and look for suspicious changes. However strenuous it sounds, it may be worth it. What we are looking for are users whose $C_u(i)$ is significantly different than it was before, incidating possible *marketing campaign*.

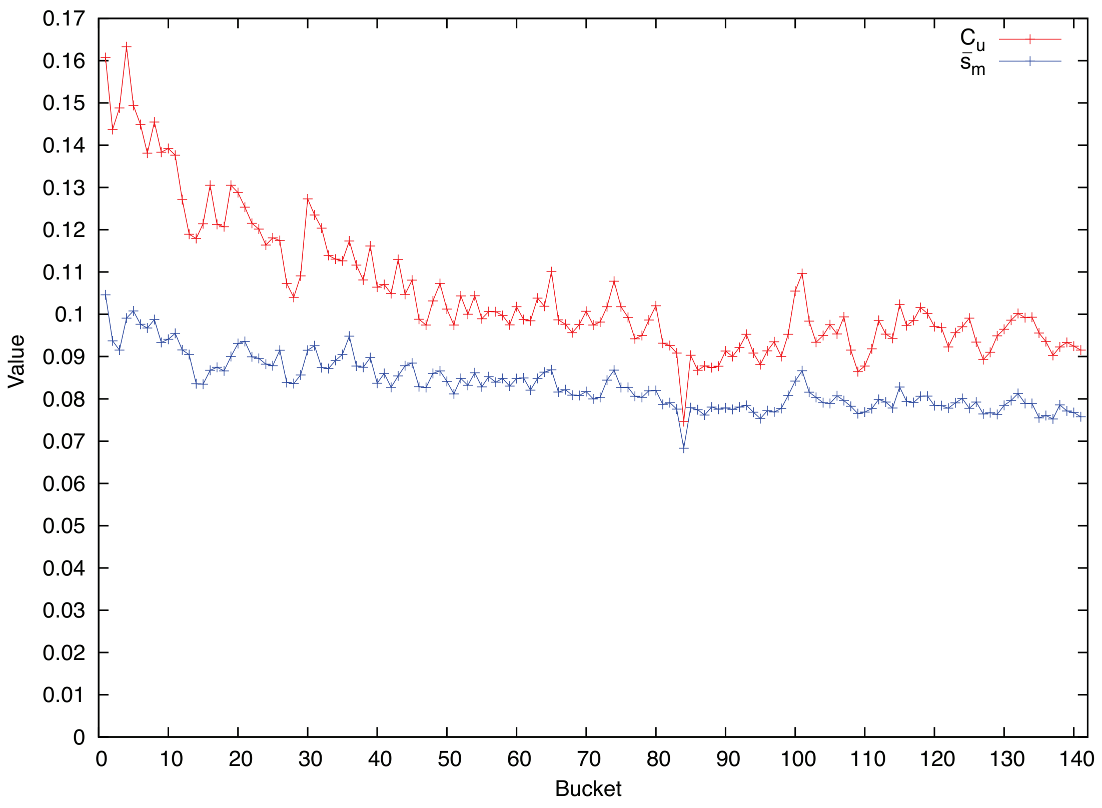
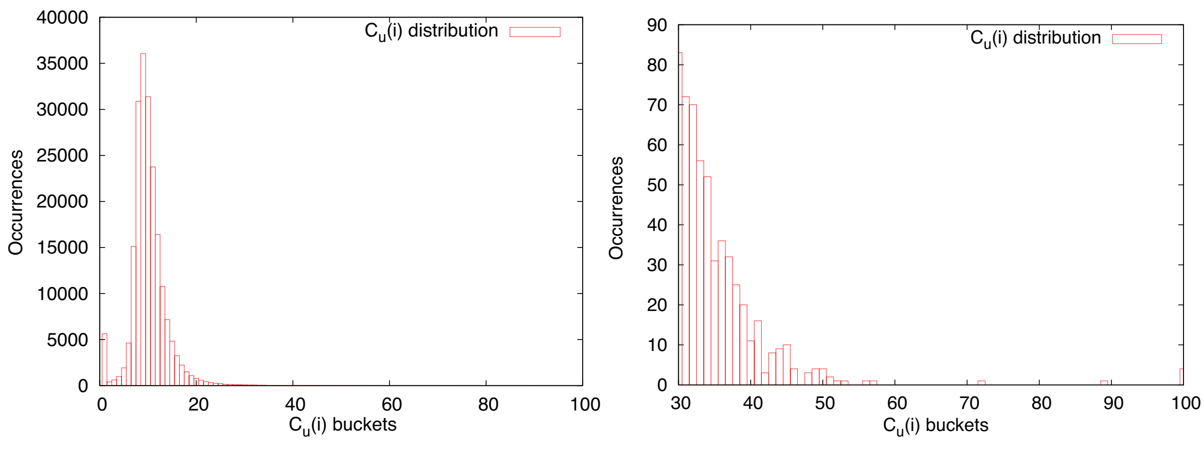


Figure 4.14: Changes of C_u and $\bar{s}_m$ over time.

Figure 4.15a represents distribution of $C_u(i)$ across all users. Where $C_u(i)$ was in a range $[0, 1]$ it was stretched to $[0, 100]$ to create 101 distinct buckets so that the distribution may have been calculated. Large number of nodes in '0' bucket constitutes with rather big amount of users with the degree less than or equal to 1, where $C_u(i)$ is calculated as 0 for such degree. Figure 4.15b shows the tail of the distribution from figure 4.15a as it is not visible due to the scale used.



(a) Full scale.

(b) The tail.

Figure 4.15: Distribution of $C_u(i)$ across all users.

Figure 4.16 shows changes of $C_u(i)$ for user $i = 12195$. One can clearly tell that the collaborative similarity for that user suddenly peaks three times as it was before. The next step would involve looking at the individual messages posted by that user during the time the peak occurred (bucket number 107).

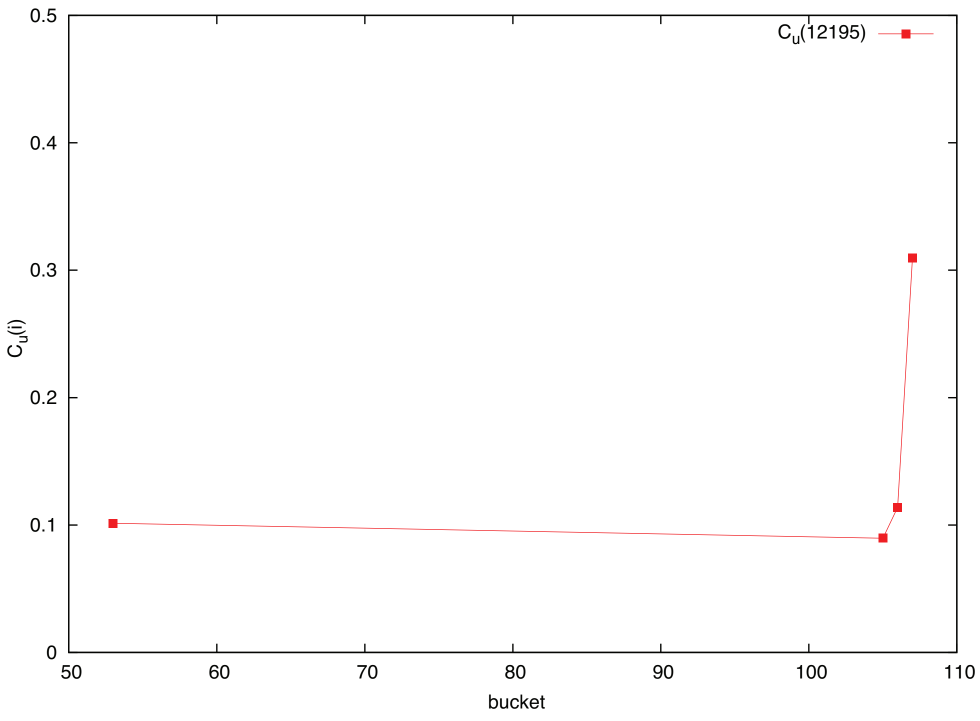


Figure 4.16: Changes of $C_u(i)$ for $i = 12195$.

After reading the posts, it looks like one of the forum users started a 'headphone brand elimination game!' topic on July 27, 2009 that was active until August 25, 2009 (buckets 106-107). The rules of the game were simple:

1. The topic started with brand names written down along with initial 10 health points.
2. Users were supposed to post a message containing one brand they want to hurt or heal, adding or removing one health point respectively.
3. Health points did not stack above 20.
4. A brand would lose—thus removed from the list effectively—once it was left with 0 health points.
5. Once four brands were left on the battlefield, their health was restored according to some other rules and the battle followed until one winner has left standing.

The battle struggled between brands *Sennheiser* and *STAX* for a longer period, with the former eventually winning 20-0. I think that the analysis of this topic alone would bring some interesting results. The only problem is that, because of this topic, I had to completely skip the results for buckets 106 and 107 due to the fact that the results would be distorted.

4.4.4.3 $C_u(i)$ fluctuation patterns found

Figure 4.17 shows various types of patters of $C_u(i)$ fluctuations that seem to repeat throughout individual users. I will not try to interpret these as it would require further, thorough examination which I did not perform due to time constraints.

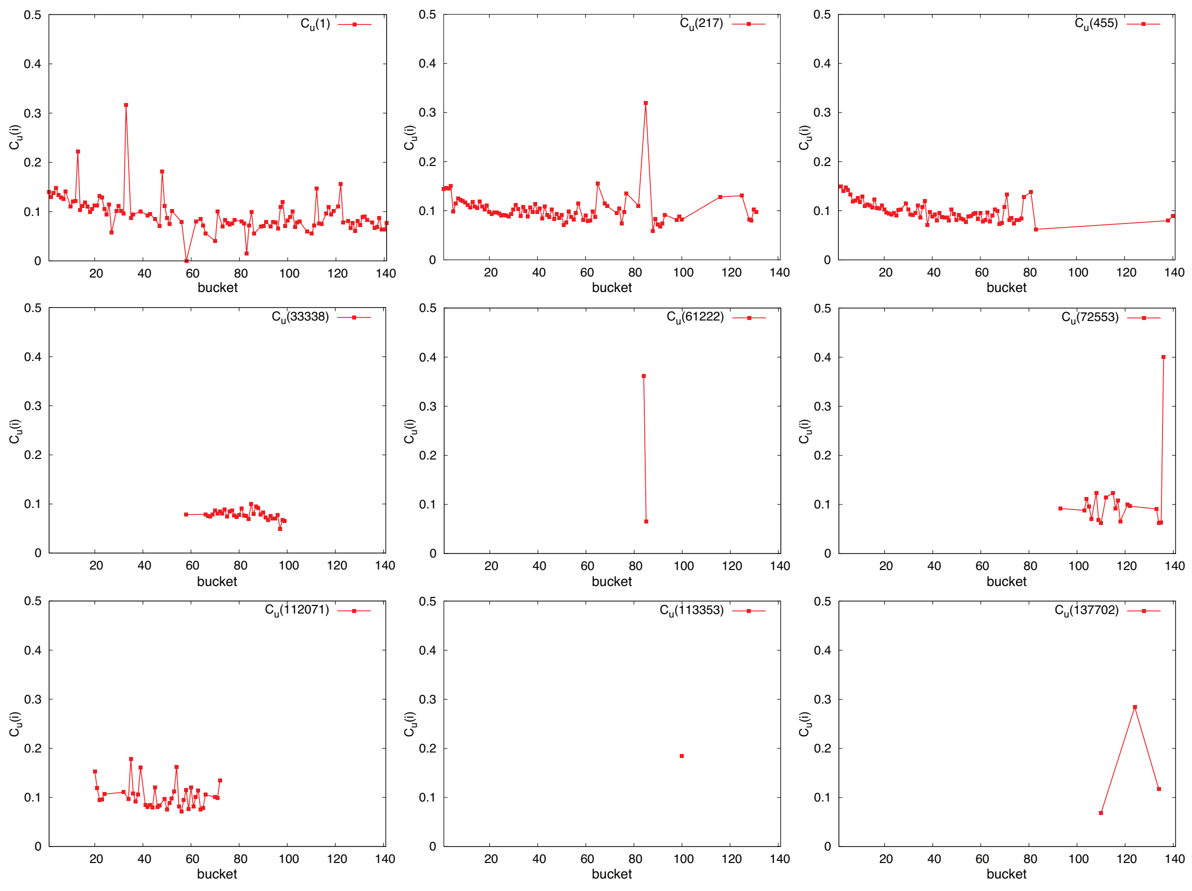


Figure 4.17: Patterns of $C_u(i)$ fluctuations.

4.4.5 Motifs

4.4.5.1 Introduction

One of the methods that I was sure will bring up some groups of users was the motif discovery within the network. It was found that real world network often feature recurrent sub-graphs or patterns. *Motifs* are sub-graphs that repeat themselves in the network or even in networks of the same type. Indeed, motifs are of notable importance largely because they may reflect functional properties. They have recently gathered much attention as a useful concept to uncover structural design principles of complex networks [46].

Network motifs were at first defined systematically in *Escherichia coli*, where they were detected as patterns that occurred in the transcription network much more often than would be expected in random networks [47]. The exact same motifs have been later found in other organisms, ranging from other bacteria [48, 49] and yeast [47, 50], to plants [51] and animals [52]. Motif discovery have been the most successful in biological networks, most likely because they were the most studied at the time—especially the transctiption networks. I will not explain many details about the motifs, because this is a complicated matter and already has been discussed and reviewed thoroughly [53].

4.4.5.2 Motif discovery

Although network motifs may provide a deep insight into the network's functional abilities, their detection is computationally challenging. Because of that, I have decided to only concentrate on the most basic, triangle motifs, that I have discovered using not very sophisticated algorithm conjoining two users sharing similar opinions about particular brands, hoping that this approach will lead to some interesting opportunities. This also allowed me to map the bipartite network I was working on so far to a regular network built on a set of users exchanging opinions only. Because I have already joined users with brands they were discussing based on the opinion being positive, negative or neutral, the only step required to obtain these triangles was to link users with matching opinions. Data required for motif discovery has been obtained in section 4.4.2.

First, the algorithm builds a bipartite network with users and opinions about the brands. Edges (solid lines) represent the opinion the user has about the product. It is not possible to have more than one edge between the nodes, because all opinions about the brand users may have are averaged and only this average number is taken into account during network creation. When the network is ready, algorithm finds user nodes having edges to the same brand nodes and links them together (dashed lines). The final step involves deletion of initial edges along with brand nodes resulting in non-bipartite network consisting of user nodes only. Figure 4.18 illustrates steps taken by the algorithm. Graphs in figures 4.18a

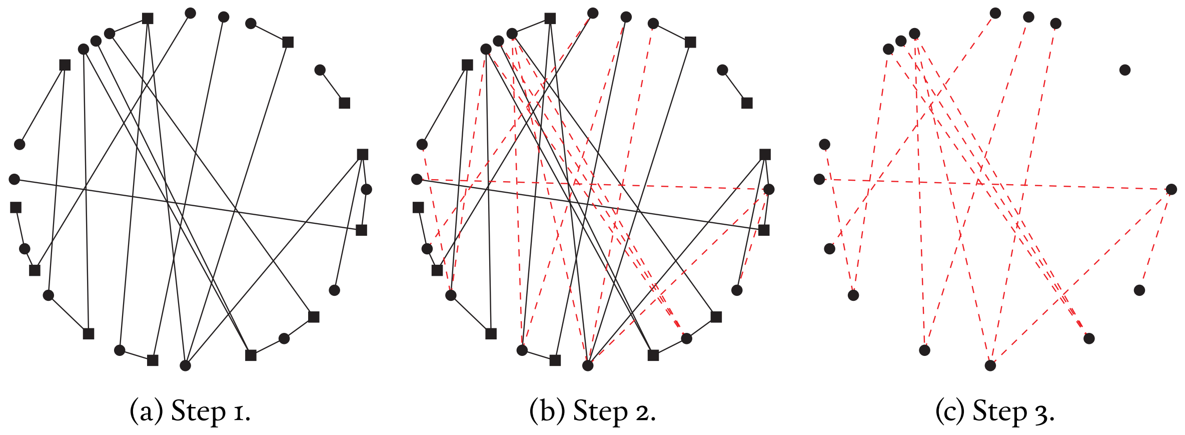

(a) Step 1. (b) Step 2. (c) Step 3.

Figure 4.18: Algorithms steps.

and 4.18b are bipartite and in figure 4.18c the graph is a non-bipartite graph projected from bipartite network. Circled nodes represent users and squared nodes represent words (brand names). Solid edges show which words were used by which users and dashed edges serve as an opinion shared between two users about a brand.

4.4.5.3 Substructures derived from motif analysis

Figure 4.19 shows a fragment of network created with the above method from bucket 106 which was excluded before in section 4.1.3.4. The network is a multigraph with nodes representing the users and edges representing an opinion about a brand shared between two users.

The network was constructed using opinions about 10 brands presented in table 4.15 ordered by the number of times they were used in that bucket. Even though *Stax* won the game mentioned in section 4.1.3.4 it was not discussed between the users in an essential manner. The number of opinions about that brand was so low that it did not matter in the whole network. The size of each node was determined from it's betweenness centrality to stress the importance of certain nodes in this network despite the fact that degrees are fairly equal and distributed uniformly in a range $[100 : 216]$.

Three brands were discussed in a particularly frequent manner between the users, thus the network consisted mostly of opinions about them. Hence, the classes of users presented in figure 4.19 are made of three brands discussed the most: *Grado, Sennheiser* and *Denon*. Grado was discussed the most (see table 4.15), but only 18.93% of shared opinions in

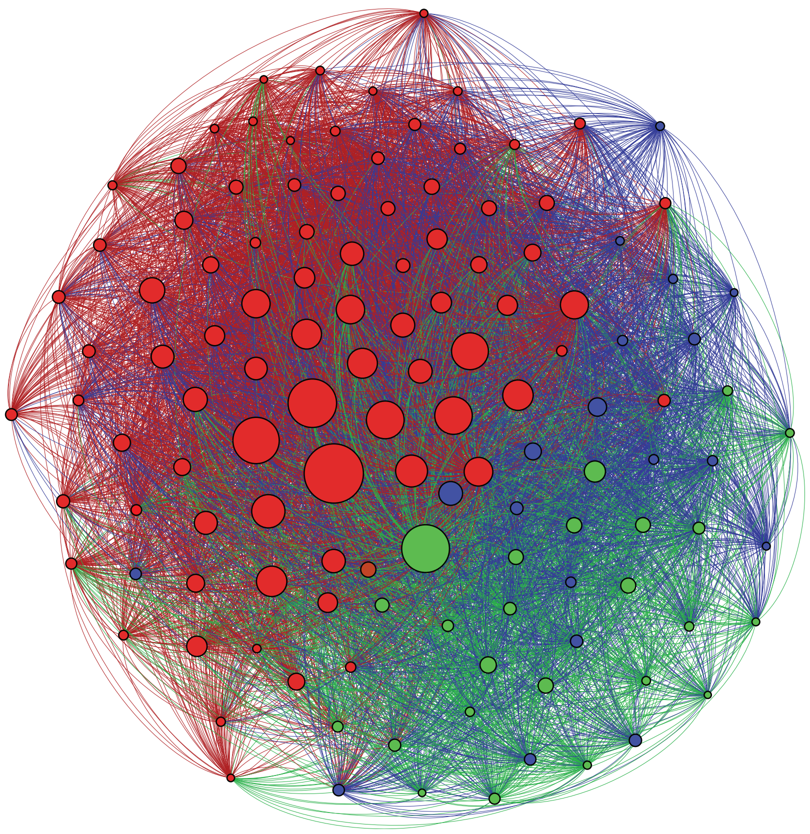


Figure 4.19: Network obtained with motif discovery.

the network are about this brand. This means that the opinions about Grado (green edges) are inconsistent, so users sharing opinion about Sennheiser or Denon does not share opinion about Grado in general.

Position	Brand	Times used
1	Grado	$1,948$
2	Sennheiser	$1,408$
3	Denon	881
4	AKG	804
5	Stax	560
6	Beyerdynamic	423
7	Sony	411
8	Yamaha	364
9	Bose	212
10	Panasonic	132

Table 4.15: Brands used to create network shown in figure 4.19.

While Denon was the least discussed from 'the great three' it certainly indicates that users' opinions are reciprocal with over half (53.56%) of the network edges (red coloured) joining users. Opinions about Sennheiser (blue links) form the other 27.51% of the network. Vertices were coloured with the colour of edges that were linked to that vertex the most.

Particularly interesting is the case of users that either do not discuss some brands (more likely) or have no common opinions about that brand (much less likely) with other users. This means that a substructure may exist where users are focused on several brands only and do not discuss others at all. The conclusions here might be too far-reaching, but this idea might be interesting to tackle.

The average path length is 1.143 which is extremely short and shows how efficient the network is. The average clustering coefficient for this network was calculated as 0.891. The graph is nearly complete with a density being equal to 0.857 meaning that almost every user in this network share a similar opinion with another one. The network was meant to be maximally dense, so I would say that the missing 0.143 density constitute a contradictory opinions.

5 Summary

This paper has shown the steps required to perform an analysis of opinion networks. It described the methodology necessary for obtaining the data as well as shown a way the data can be normalised prior to study and has examined the problems that may arise during this process.

The thesis, boldly assumed, that it will be possible to find examples of surreptitious messages glorifying or condemning certain products. Unfortunately such examples were never found manually and whole analysis was performed on a blind assumption that they really exist in hope of finding them. If such messages were found *a priori* of the study then the methodology could have been adjusted appropriately. It is very likely that this forum has very good moderation which has been making sure that malicious messages are removed. It would be hard to believe that certain businesses do not try to exploit a message board like this since it is a very tasty morsel. Nevertheless, it was very inspiring to conduct such analysis to see what can we find out about the community by exploring the properties of the network it communicates in.

A new measure has been proposed in section 4.4.2 that may help estimating the opinions some agents may have or may say they have. The measure can be used in various applications, for instance as a basis of a recommendation system. It may be used as a popularity estimate among groups of users of message boards.

One of the biggest challenges was coping with such a huge amount of data. Most computations were conducted using 4-8 processing cores with up to 20 used in extreme cases. Any manual intervention was laborious and every mistake in calculations was a great waste of time.

Usage of the complex networks theory as a tool for understanding data as such presented here may be easily justified. It is sufficient to look at only three important measures: average path length, degree distributions and network clustering coefficient in order to get a general idea of the network. On the other hand, there is no and probably there will never be a single methodology for exploration of different kinds of networks. Choosing correct approach is one of the most difficult tasks before anyone trying to understand the mechanisms governing in the networks. As a side note, one has to be very careful in drawing conclusions from the study.

5.1 Potential future work

In this part I would like to describe other promising approaches I did not have time to tackle or I did not perform due to various reasons.

First experiment that I would like to tackle if I had more time would be to analyse different type of fluctuations of $C_u(i)$ observed in section 4.4.4.3. It would be very interesting to check if similar shapes of $C_u(i)$ changes reflect similar behaviour of users.

I would also like to study the popularity of products based on the keywords appearing in *long discussions* or in very old, bumped discussions. I am also curious what kind of user groups I could identify that are discussing the most and the least popular topics.

During the course of the assignment I have come to a conclusion that instead of trying to find malicious posts, I can use some of the metrics and come up with a methodology allowing for some kind of marketing campaign.

This would be the complete counter-assumption of what was proposed in the thesis. The marketing campaign could be focused on users like the recommendation system. It could be based on the changes of their product preferences, as well as their nearest neighbour's preferences. Then it could contextually offer new products to the users that system marks as highly interested. Another investigation that could be performed would be to examine the opinion persistence of hubs over time and how the 'followers' of these hubs react to those changes.

In my opinion it would be very suspicious to post comprehensive answers (that can be trusty) to various questions posted on the forum immediately after they were asked. Hence, response time distribution could be designated to find the optimal time of operation for marketing purposes.

In spite of the concept being interesting, the actual implementation might not work well as it may seem, referring to a very interesting opinion of one of the users of this forum: 'I like reading others' opinions. But my buying decision ultimately depends on my personal judgement. If I don't like something I bought, I would just return it. In most cases, I rarely returned items that I bought as I tend to do a lot of researches before I decided to buy the item.'

Bibliography

[1] B. Kujawski, J. Hołyst, G. J. Rodgers, *Growing trees in internet news groups and forums*. Physical Review E **76**, 036103, 2007.

[2] R. V. Solé, B. C. Murtra, S. Valverde, L. Steels, *Language Networks: their structure, function and evolution*. Complexity **15** (6), pp 20-26, 2010.

[3] M.-S. Shang, L. Lu, Y.-C. Zhang, T. Zhou, *Empirical analysis of web-based user-object bipartite networks*. A Letters Journal Exploring the Frontiers of Physics, EPL **90**, 48006, 2010.

[4] R. Diestel *Graph Theory – Graduate Texts in Mathematics*. Springer-Diestel, 4th Edition, corrected reprint, 2012.

[5] J. M. Harris *Combinatorics and Graph Theory*. Springer-Verlag, 2000.

[6] A. S. Asratian, T. M. J. Denley, R. Häggkvist, *Bipartite Graphs and their Applications*. Cambridge Tracts in Mathematics **131**, Cambridge University Press, 1998.

[7] E. R. Scheinerman, *Mathematics: A Discrete Introduction*. 3rd Edition, Cengage Learning, 2012.

[8] G. Chartrand, P. Zhang, *Chromatic Graph Theory*. Discrete Mathematics And Its Applications **53**, CRC Press, 2008.

[9] S. Wasserman, K. Faust, *Social Network Analysis: Methods and Applications*. Structural Analysis in the Social Sciences **8**, Cambridge University Press, 1994.

[10] J. L. Moreno, *Who Shall Survive: A New Approach to the Problem of Human Interrelations*. Nervous and Mental Disease Publishing Co., 1934.

[11] A. Rapoport, *Contribution to the theory of random and biased nets*, Bulletin of Mathematical Biophysics **19**, pp 257–77, 1957.

[12] S. Milgram, *The small world problem*, Psychology Today **2**, pp 60–67, 1967.

[13] L. Egghe, R. Rousseau, *Introduction to Informetrics*. Elsevier, Amsterdam, 1990.

[14] J. G. White, E. Southgate, J. N. Thompson, S. Brenner, *The Structure of the Nervous System of the Nematode Caenorhabditis elegans*. Philosophical Transactions of the Royal Society of London B **314** (1165), pp 1–340, 1986.

[15] D. J. Watts, S. Strogatz *Collective dynamics of 'small-world' networks*. Nature **393** (6684), pp 440–42, 1998.

[16] P. Erdős, A. Rényi, *On the evolution of random graphs*. Publications of the Mathematical Institute of the Hungarian Academy of Sciences **5**, pp 17–61, 1960.

[17] M. Barthelemy, L. A. N. Amaral, *Small-world networks: Evidence for a crossover picture*. Physical Review Letters **82** (15), 3180, 1999.

[18] A. L. Barabási, R. Albert, *Statistical mechanics of complex networks*. Reviews of Modern Physycs **74** (1), pp 47-97, 2002.

[19] P. W. Holland, S. Leinhardt *Transitivity in structural models of small groups*. Comparative Group Studies **2** (2), pp 107-24, 1971.

[20] R. D. Luce, A. D. Perry *A method of matrix analysis of group structure*. Psychometrika **14** (2), pp 95-116, 1949.

[21] L. C. Freeman *A set of measures of centrality based on betweenness*. Sociometry **40** (1), pp 35-41, 1977.

[22] J. P. Onnela, J. Saramaki, J. Hyvonen, G. Szabo, D. Lazer, K. Kaski, J. Kertesz, A. L. Barabási, *Structure and tie strengths in mobile communication networks,* Proceedings of the National Academy of Sciences **104** (18), pp 7332-36, 2007.

[23] K. Choromański, M. Matuszak, J. Miękisz, *Scale-Free Graph with Preferential Attachment and Evolving Internal Vertex Structure*. Journal of Statistical Physics **151** (6), pp 1175-83, 2013.

[24] A. Clauset, C. R. Shalizi, M. E. J Newman, *Power-law distributions in empirical data*. SIAM Review **51**, pp 661-703, 2009.

[25] A. Turing, *Computing Machinery and Intelligence*. Mind **LIX**, 1950.

[26] M. Davis, *Unicode Text Segmentation*. Unicode Standard Annex **#29**, http://www.unicode.org/reports/tr29/, 2012.

[27] J. B. Lovins, *Development of a Stemming Algorithm*. Mechanical Translation and Computational Linguistics **11**, pp 22–31, 1968.

[28] J. Levin, P. Milgrom, *Online Advertising: Heterogeneity and Conflation in Market Design*, American Economic Review **100** (2), pp 603-07, 2010.

[29] M. F. Porter, *An Algorithm for Suffix Stripping*. Program **14** (3), pp 130–37, 1980.

[30] Oxford English Corpus, *The OEC: Facts about the language*. accessed and archived 24 July 2013, (http://web.archive.org/web/20130724181442/http://oxforddictionaries.com/words/ oec-facts-about-the-language).

[31] G. K. Zipf, *The Psychobiology of Language*. Oxford, England: Houghton Mifflin, 1935.

[32] G. K. Zipf, *Human Behavior and the Principle of Least Effort*. Cambridge, Massachusetts: Addison-Wesley, 1949.

[33] C. D. Manning, H. Schütze, *Foundations of Statistical Natural Language Processing*. MIT Press, 1999.

[34] L. A. Adamic, B. A. Huberman, *Zipf's law and the Internet*. Glottometrics **3**, pp 143-50, 2002.

[35] L. Egghe, *Untangling Herdan's law and Heaps' law: Mathematical and informetric arguments*. Journal of the American Society for Information Science and Technology **58** (5), pp 702-09, 2007.

[36] A. Kornai, *Zipf's law outside the middle range*, in J. Rogers, *Proceedings of the Sixth Meeting on Mathematics of Language*, pp 347-56. University of Central Florida, 1999.

[37] J. Milička, *Type-token & Hapax-token Relation: A Combinatorial Model*. Glottotheory. International Journal of Theoretical Linguistics **1** (2), pp 99-110, 2009.

[38] E. Fischer, S. J. Arnold, *More than a Labor of Love: Gender Roles and Christmas Gift Shopping* Journal of Consumer Research, **17** (3), pp 333-45, 1990.

[39] M. J. Arnold, K. E. Reynolds, *Hedonic shopping motivations*. Journal of Retailing **79** (2), pp 77-95, 2003.

[40] S. Gebert, R. Pries, D. Schlosser, K. Heck, *Internet Access Traffic Measurement and Analysis*. TMA'12 Proceedings of the 4th international conference on Traffic Monitoring and Analysis, Springer-Verlag Berlin, Heidelberg, 2012.

[41] M. A. Carskadon *Adolescent Sleep Patterns: Biological, Social, and Psychological Influences.*. Cambridge University Press, 2002.

[42] F. Zhou, S. Mahler, H. Toivonen, *Simplification of Networks by Edge Pruning*. Bisociative Knowledge Discovery, Lecture Notes in Computer Science **7250**, pp 179-98, 2012.

[43] H. Toivonen, S. Mahler, F. Zhou, *A Framework for Path-Oriented Network Simplification*. in P.R. Cohen, N.M. Adams, M.R. Berthold, IDA 2010. LNCS, **6065**, pp 220–31, Springer, Heidelberg, 2010.

[44] P. G. Lind, M. C. González, H. J. Herrmann, *Cycles and clustering in bipartite networks*. Physical Review E **72**, 056127, 2005.

[45] P. Jaccard, *Étude comparative de la distribution florale dans une portion des Alpes et des Jura*. Bulletin de la Société Vaudoise des Sciences Naturelles **37**, pp 547-79, 1901.

[46] A. Masoudi-Nejad, F. Schreiber, Z. R. Kashani *Building Blocks of Biological Networks: A Review on Major Network Motif Discovery Algorithms*. IET Systems Biology **6** (5), pp 164-74, 2012.

[47] R. Milo, U. Alon, *et al.*, *Network Motifs: Simple Building Blocks of Complex Networks*. Science **298** (5594), pp 824-27, 2002.

[48] S. Mangan, A. Zaslaver, U. Alon, *The coherent feedforward loop serves as a sign-sensitive delay element in transcription networks*. Journal of Molecular Biology **334** (2), pp 197-204, 2003.

[49] P. Eichenberger, *et al.*, *The program of gene transcription for a single differentiating cell type during sporulation in Bacillus subtilis*. PLoS Biol. **2** (10), pp 1664-83, 2004.

[50] T. I. Lee, *et al.*, *Transcriptional regulatory networks in Saccharomyces cerevisiae*. Science **298** (5594), pp 799-804, 2002.

[51] L. A. Saddic, *et al.*, *The LEAFY target LMI1 is a meristem identity regulator and acts together with LEAFY to regulate expression of CAULIFLOWER*. Development **133** (9), pp 1673-82, 2006.

[52] L. A. Boyer, *et al.*, *Core transcriptional regulatory circuitry in human embryonic stem cells*. Cell **122** (6), pp 947-56, 2005.

[53] U. Alon, *Network motifs: theory and experimental approaches*. Nature Reviews Genetics **8** (6), pp 450-61, 2007.