UNIVERSITY OF ZAGREB
**FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING**

MASTER THESIS num. 802

# ANALYSIS OF ALGORITHMS FOR DETERMINING TRUST AMONG FRIENDS ON SOCIAL NETWORKS

Mirjam Šitum

Vienna, June 2014

Thesis was created at University of Vienna during the study period of the student in the international exchange program Erasmus.

Rad je izrađen na Sveučilištu u Beču tijekom studijskog boravka studentice u okviru programa međunarodne razmjene Erasmus.

**SVEUČILIŠTE U ZAGREBU**
**FAKULTET ELEKTROTEHNIKE I RAČUNARSTVA**
ODBOR ZA DIPLOMSKI RAD PROFILA

Zagreb, 10. ožujka 2014.

# DIPLOMSKI ZADATAK br. 802

Pristupnik: **Mirjam Šitum (0036454376)**
Studij: Informacijska i komunikacijska tehnologija
Profil: Telekomunikacije i informatika

Zadatak: **Analiza algoritama za određivanje povjerenja među prijateljima na društvenim mrežama**

Opis zadatka:

Usluga društvenog umrežavanja na Internetu, podržana web-zasnovanim platformama kao što su Facebook, Twitter, LinkedIn i druge, temelji se na društvenom grafu. Društveni graf se sastoji od čvorova koji predstavljaju korisnike usluge društvenog umrežavanja te grana koje predstavljaju vezu, odnosno prijateljstvo, između para korisnika čije čvorove promatrana grana spaja. Grane u društvenom grafu najčešće poprimaju binarne vrijednosti težina, tako da težina 0 označava situaciju kada grana ne postoji, odnosno težina 1 kada grana postoji. Na taj način se veze međusobno ne razlikuju.
Vaša je zadaća analizirati algoritme za određivanje povjerenja među prijateljima na društvenim mrežama. Takvi algoritmi omogućavaju izgradnju društvenog grafa s granama koje mogu poprimiti kontinuirane vrijednosti težina, pri čemu težina grane između nekog para čvorova opisuje razinu povjerenja među korisnicima reprezentiranim tim čvorovima. Nadalje, tako izgrađeni društveni graf koji je karakteriziran granama koje mogu poprimiti kontinuirane vrijednosti težina omogućuje identifikaciju skupa prijatelja kojima ego-korisnik društvene mreže najviše vjeruje. Prilagodite izabrane primjere algoritama za izračun povjerenja kontekstu internetskog okružja te ih programski izvedite koristeći Facebook API za interakciju s društvenom web-stranicom Facebook. Evaluirajte rezultate izvođenja implementiranih algoritama za izračun povjerenja.
Svu potrebnu literaturu i uvjete za rad osigurat će Vam Zavod za telekomunikacije.

Zadatak uručen pristupniku: 14. ožujka 2014.
Rok za predaju rada: 30. lipnja 2014.

Mentor:

Doc. dr.sc. Vedran Podobnik

Prof. dr.sc. Wolfdieter Merkl

Djelovođa:

Doc. dr.sc. Lea Skorin-Kapov

Predsjednik odbora za diplomski rad profila:

Prof. dr.sc. Igor Sunday Pandžić

**UNIVERSITY OF ZAGREB**
**FACULTY OF ELECTRICAL ENGINEERING AND COMPUTING**
MASTER THESIS COMMITTEE

Zagreb, March 10th, 2014

# MASTER THESIS ASSIGNMENT No. 802

Student: **Mirjam Šitum (0036454376)**
Study: Information and Communication Technology
Profile: Telecommunication and Informatics

Title: **Analysis of Algorithms for Determining Trust among Friends on Social Networks**

Description:

Internet social networking services, supported by web platforms such as Facebook, Twitter, LinkedIn and others, are based on a social graph. Social graph consists of nodes that represent users of social networking services and edges that represent a connection, i.e. friendship, between pairs of users whose nodes the observed edge connects. Edges in the social graph are usually characterized with binary value weights, i.e. weight 0 if an edge does not exist, or weight 1 if there is an edge. In that way, there is no difference among links that connect users of social networking services.
Your assignment is to analyse algorithms for determining trust among friends on social networks. Such algorithms allow construction of social graphs with edges that can be characterized with continuous value weights, where the weight of an edge between a pair of nodes describes a level of trust among users represented by these nodes. Furthermore, the social graph constructed in such a way allows identification of a set of friends in which the ego-user of a social network trusts most. Adapt the selected examples of algorithms for trust calculation to the Internet context and implement them using the Facebook API for interaction with the social networking web site Facebook. Evaluate the results of execution of implemented trust calculation algorithms.
All of the resources will be provided to you by the Department of Telecommunications.

Thesis submitting date: June 30th, 2014

Mentor:

Prof. Vedran Podobnik, PhD

Committee Chair:

Prof. Igor Sunday Pandžić, PhD

Prof. Wolfdieter Merkl, PhD

Committee Secretary:

Prof. Lea Skorin-Kapov, PhD

Content

# Introduction

Social network interactions have become regular in the lives of people and the biggest credit for this goes to the rapid development of social networks such as Facebook. These networks create a social graph of users which is usually consisted of users and their friendships. But, information that can be extracted from this network is usually much richer than just binary friendship relations. One of the most important human interactions is trust which has its place in computer science and social networks. Trust among users in social networks has been of high interest, not just in computer science, but in psychology and sociology as well. Knowledge about trust in social networks can also be used in recommendation systems, which makes it even more interesting for researchers.

This work gives a general evaluation of different algorithms for computing trust among users in social network Facebook. It puts big emphasize on peer-to-peer trust computation algorithms of which some have been adapted for social network environment. An application which collects user data from Facebook and calculates different trust algorithms has been developed for purpose of different trust algorithms evaluation.

In the chapter 1, a general theoretical review of trust has been given. Different properties and aspects of trust have been presented and described. Chapter 2 provides a review of different existing approaches of trust computation in peer-to-peer systems and social networks. Chapter 3 presents different algorithms that have been adapted and implemented in this work, that compute trust among users on Facebook. These algorithms include direct algorithms which compute trust based on direct interactions between friends on Facebook, and peer-to-peer algorithms. Chapter 4 describes the components of developed application and in the end last chapter gives a complete analysis and evaluation of algorithms.

# 1. Trust

One of the major components of human interaction is trust. There are many definitions of trust. One of the definitions is that trust is a measure of confidence that an entity will behave in expected manner, despite the lack of ability to monitor or control the environment in which it operates (Singh et al., 2007). Psychological definition of trust says that the trust is psychological state of the individual where he risks being vulnerable to the trustee based on the positive expectations of the trustees intentions or behavior (Rousseau et al., 1998). In sociology, trust is defined as a "bet" about the future contingent actions of the trustee (Dumouchel, 2005). Finally, there is trust in computer science, where it can be classified in two categories: user and system. User trust is subjective expectation an entity has about another's future behavior, (Mui, 2003), while system trust is the expectation that a device or system will faithfully behave in a particular manner to fulfil its intended purpose. This work refers to the trust as a user trust in computer science. There are two main types of user trust in computer science: direct trust and recommendation trust, also referred to as peer-to-peer trust (Nepal et al., 2013). Direct trust is based on the direct experience of the member with another party, while recommendation trust is based on the propagative properties of the trust.

## 1.1. Types of Trust

Trust has many different aspects. When we talk about trust aspects, we consider a perspective from which we look at a trust. This perspective often gives different semantics to the trust (Sherchan et al., 2013).

- Calculative aspect of trust
  - Defines trust as the result of a calculation on behalf of the member to maximize his stakes in the interaction (Tyler et al., 1996). This aspect of trust can mainly be found in economics where the trust is described as a chance of gain to the chance of loss in ratio with the amount of the potential loss to the amount of the potential gain.

- Relational aspect of trust
  - This aspect of trust is a result of repeated interactions between the member and trustee. In computer science, this aspect of trust is called direct trust, trust based on direct interactions between two parties. (Rousseau et al., 1998)
- Emotional aspect of trust
  - Defines trust as the security and comfort in relying on a trustee (Kuan et al., 2005). In psychology, it is an outcome of direct interpersonal relationships between member and a trustee.
- Institutional aspect of trust
  - Defines trust as a result of an institution providing environment that encourages cooperation between members and penalizes misbehaviors (Lewis et al., 1985). Such supports can exist at organizational level and societal level as legal systems that protect individual rights and property.
- Dispositional aspect of trust
  - Dispositional aspect of trust recognizes that, over the course of their lives, people develop generalized expectations about the trustworthiness of other people (Rotter, 1967).

This work will be exploring relational aspect of trust and how the interactions between member and trustee on social network Facebook reflect on their trust towards each other.

## 1.2. Properties of Trust

Trust can have different properties. It can vary with time or be subjective. Properties of the trust have great influence in determining what kind of trust is being researched and modeled. Most important trust properties are (Sherchan et al., 2013):

- Context specific
  - Trust can be context specific in its scope. If person A trusts person B as his doctor, he will not trust him as his mechanic to fix his car. So person A is trustworthy towards person B in the context of seeing a doctor, but not in the context of fixing its car. When modeling trust among Facebook user, we consider it to be context specific. In this work, there will be conclusions about contexts in which trust is found on Facebook. We can also say, since this trust is calculated by interactions on Facebook, that it has Facebook context.

- Dynamic
  - Trust can increase or decrease with time and new experiences, interactions or observations (Staab et al. 2004). New experiences are more important than the old ones, since old experiences may become irrelevant with time. In computer science, various techniques are used for modeling dynamicity of trust. In this work, this property is not in the center of research. This work will calculate trust from interactions on Facebook in a period of time without tracking its changeability with time.
- Propagative
  - This is a very important property of trust and it is explored in many works. It can be described with next example. If person A trusts person B and person B trusts person C, then A, who does not know C, can derive some amount of trust on C based on how much she trusts B and how much B trusts C. Because trust is usually propagative, its information can be passed from one member to another in social network, creating trust chains. It is similar as the "word of mouth" propagation of information for humans (Abdul-Rahman et al., 2000). This is one of the most important properties of trust for this work because we explore peer-to-peer algorithms for determining trust among Facebook users. Propagative property is essential for this type of algorithms. In this work, we will explore how the trust is propagated through the network of Facebook users and how accurately trust can be computed with this property.
- Aggregative
  - While propagation of trust along social chain says that a member can form some trust on a member that is not directly connected to it, it does not say how to behave when we have several chains of trust recommending different amount of trusts towards trustee. This information, from multiple chains, has to be composed to form final amount of trust. Golbeck proposes a trust composition function based on the structure of trust relationships (Golbeck, 2005). In this work, Golbecks method is one of several which will be adapted and explored to research if the trust calculated from user interactions on Facebook shows propagative and aggregative properties.

- Subjective
  - In general, trust can be subjective. If person A gives its opinion of someone, than trust computed from that opinion is subjective. The subjective nature of trust leads to personalization of trust computation, where preferences of the member have a direct impact on the computed trust review. This work will not be computing trust from subjective reviews of the members because the trust is calculated with interactions between users. However, there will be conclusions about the correspondence of the trust calculated from user interactions of Facebook and user subjective preferences toward other users.
- Asymmetric
  - Trust is typically asymmetric. One member may trust another more than s/he is trusted back. Asymmetry occurs because of differences in people perceptions, opinions and beliefs.
- Self-reinforcing
  - Trust is also usually self-reinforcing. Members usually act positively with other members whom they trust. Also, if the trust is really low between members it is highly unlikely that they will interact with each other, leading to even less trust on each other (Yu et al., 2000).
- Event sensitive
  - Trust usually takes a long time to build, but a single high-impact event may destroy it completely (Nepal et al. 2010). That means that trust has event sensitive nature. Despite this being an interesting property of the trust, it is not well researched and it will also not be researched in this work.

## 1.3. Trust Representation

When we are talking about trust representation, we are thinking about different approaches in grading members trust towards another member. These approaches can be divided into two main categories: probabilistic and gradual. Probabilistic approaches deal with a single trust value in black and white fashion and compute probability that the trustee can be trusted. (Rici et al., 2011) On the other hand, gradual approaches estimate trust values to some extent, as opposed to being either right or wrong. In gradual approaches trust values are not interpreted as probabilities, but rather values where higher value corresponds to a higher trust in an agent. Differences in trust representation have higher effect in

computation of trust by peer-to-peer algorithms where even the whole algorithms depend on trust representation.

## 1.4. Information Sources

There are three main sources of trust information is social networks: (Scerchan et al., 2013)

- Attitudes
  - Attitudes represent an individual's degree of like or dislike for something. They are positive or negative view of some entity. Attitudes are derived from user's interactions.
- Experiences
  - Experiences describe perception of the member in their interactions with each other. Experiences are often used as an information source in peer-to-peer networks while computing trust between nodes in the network. These experiences are usually peers ratings of quality of interactions with other peers.
- Behaviors
  - Behaviors are identified by patterns of interactions and its main information source is interaction. It can depend on type of interaction, frequency of interaction, or change of interaction. All these behaviors can be used to determine and compute trust between members in network.

## 1.5. Trust Evaluation Models

There are different ways of computing trust among members of a network. They can be categorized in three groups: network-based trust models, interaction-based trust models and hybrid trust models.

### 1.5.1. Network Structure Trust Models

In this trust evaluation model, a trust network is created for each member. With this model, members are represented as nodes in the graph, and trusts among members are represented as directed edges. Various approaches are then used to determine trust between any two nodes in the network. These approaches exploit propagative nature of trust. One example of network-based trust model is Tidal Trust where trust is derived between two people in social network who are not directly connected. This approach is based on the premise that

the neighbors with higher trust ratings are likely to agree with each other about the trustworthiness of a third party. As mentioned before, this approach will be adopted in this work to test and evaluate properties of trust in Facebook, such as propagation and aggregation of trust.

### 1.5.2. Interaction-based Trust Models

Some models only use interaction within the network to compute social trust. In these models, we can distinguish two types of trust: popularity trust and engagement trust. (Nepal el al., 2011) Popularity trust refers to the acceptance and approval of a member in the community, representing the trustworthiness of the member from the perspective of other members in the community. Engagement trust refers to the involvement of the member in the community, representing the trust the member has towards the community. Combination of popularity trust and engagement trust forms the basis for determining the social trust in the community.

### 1.5.3. Hybrid Trust Models

Interaction based social trust models only consider interactions in the community to compute trust, but ignore the social network structure. Social network structure can provide important information about how members relate to each other and it can be a significant source of information for social trust computation. This is why some models consider both graph structures and interactions within the social networks to compute social trust and are called hybrid trust models.

# 2. Existing Algorithms Review

In this chapter we present a review of some of the currently available literature about existing approaches in trust computation on social networks or peer-to-peer networks.

Eigen Trust is network-based algorithm for determining trust among peers in peer-to-peer network. In this algorithm, global reputation of each peer is given by the local trust values assigned to peer *i* by other peers weighted by the global reputations of the assigning peers. This algorithm is used in P2P networks and shows subjective property because it uses peers subjective ratings of interactions with other peers to compute trust. Also, in (Kamvar el al., 2003) Eigen Trust has been used to make a research considering its aggregative and propagative properties.

Peer Trust is another example of network-based algorithm for computing trust. In this algorithm, every peer uses several information factors to determine trust towards other peers. Peer obtain feedback from other peers, total number of transactions peer has with other peers an etc. This algorithm shows subjective properties since it is also computed from peer's experiences of communication with other peers. In (Xiong et al. 2004) this algorithm is presented with dynamic properties since it takes into account the changes of trust through time. In this algorithm, past experiences became less important with time. Specifics of peer interactions determine steps of this algorithm, what makes it hard to adapt for analysis on users in social network Facebook.

Another approach is presented in (Yu et al., 2004). In this work, another method for rating peers in peer-to-peer networks is presented. It explores ideas of every peer rating interaction with another peer with a rate from interval [0,1] and effectively aggregate noisy ratings in a presence of dishonest voters. This paper proposes an idea of combining direct trust between two peers and propagated trust obtained from a network. This combination could provide more accurate results. This is why this idea will be used in this work to see if the combination of direct and propagated trust can produce better results.

Tidal Trust is another example of network-based algorithms for trust computing, but it differs from previously mentioned algorithms. While previous algorithms work with peer-to-peer networks, this algorithm was researched on social networks. (Golbeck el al., 2005).

This algorithm uses propagative properties of trust to compute trust from trust. It differs from Eigen Trust and Peer Trust with trust values being gradual, rather than probabilistic.

(Josang et al., 2006) is another approach to trust computation problem. This is an example of trust computing algorithm with both, trust and distrust. However, algorithm is too complex to be adapted to Facebook trust computation.

(Chaverlee et al., 2008) explores another method for trust calculation. This paper distinguishes relationships quality from trust. It proposes the tracking of user behavior over time to encourage long-term good behavior and penalize misbehavior in social networks.

(Liu et al., 2008) is an example of work which only uses interactions in determining trust among entities. It is known as epinions case study and brings an analysis of user interactions on specific social network and trust computed from these interactions. Since the algorithm used in this case study is specific for the interactions used on this social network it will not be further studied in this work.

Last example of algorithm for computing trust is given in (Nepal et al., 2011). This work emphasizes different types of interactions and their research. In that work, difference between popularity trust and engagement trust is introduced.

Table 2.1 Existing algorithms categorization

| | Application Domain | Trust Evaluation | Information Collection | Trust / Distrust | Trust Representation |
|---|---|---|---|---|---|
| **Eigen-Trust** | P2P | Graph Based | Experience | Trust | Probabilistic |
| **Peer-Trust** | P2P | Graph Based | Experience | Trust | Gradual |
| **Yu** | P2P | Graph Based | Experience | Trust | Probabilistic |
| **Golbeck** | Social Networks | Graph Based | Experience | Trust | Gradual |
| **Josang** | P2P | Graph Based | Experience | Trust and Distrust | Probabilistic |
| **Power-Trust** | P2P | Graph Based | Experience and Behavioral | Trust | Gradual |
| **Chaverlee** | Social Networks | Interactions | Behavioral | Trust | Probabilistic |
| **Liu** | Social Networks | Interactions | Behavioral | Trust | Gradual |
| **Nepal** | Social Networks | Interactions | Behavioral | Trust | Gradual |

As we can see in table **Error! Reference source not found.**, there are many axes by which e can categorize algorithms for computing trust. Algorithms can be categorized to algorithms which are implemented in P2P domain problem or social network domain problem. First five algorithms are graph based and use propagative and aggregative properties of the trust to compute trust values. Last three algorithms use only direct interactions among members to compute trust. Interaction-based algorithms also use behaviors to compute trust, while mostly all peer-to-peer algorithms use peer's experiences of communication with other peers. Mostly all algorithms work only with trust, only Josang's algorithm computes both trust and distrust. In the end, they are equally divided by trust representation, where probabilistic and gradual approaches are equally represented.

Table 2.2 Properties od trust in existing algorithms

| | Context-specific | Dynamic | Propagative | Aggregative | Subjective |
|---|---|---|---|---|---|
| **Eigen-Trust** | - | - | + | - | + |
| **Peer-Trust** | - | + | + | - | + |
| **Yu** | - | - | + | + | + |
| **Golbeck** | - | - | + | + | + |
| **Josang** | - | + | + | + | + |
| **Power-Trust** | - | - | + | + | + |
| **Chaverlee** | - | + | - | - | - |
| **Liu** | - | + | - | - | - |
| **Nepal** | - | + | - | + | + |

Table 2.1 shows a review of properties that were researched in papers that covered these algorithms.

# 3. Used Algorithms for Trust Calculation

The goal of this work is to develop several models of algorithms for computing trust among Facebook users. In the center of this research are peer-to-peer algorithms, but since they depend on direct trust among Facebook users, direct algorithms will also be important part of the research. In this chapter, first we define direct algorithm for computing trust between Facebook user and his friends. After that we will adapt several peer-to-peer algorithms that will use this interaction-based direct algorithm and propagative and aggregative properties of the trust to compute trust values among Facebook users that are not directly connected, that are not friends. In the end, we will bring up few ideas of how to combine direct interaction-based algorithm with graph-based peer-to-peer algorithms to produce better results of computing trust among users on Facebook. These algorithms work with user interactions on Facebook, which means that they collect information from behaviors and not experiences, and are not subjective.

## 3.1. Interaction Based Algorithms

These algorithms calculate trust between user and his friends on Facebook based on their interactions in social network. These interactions can be divided into two groups: interactions made by user towards friends and interactions made by friends towards user. Since the data collected from Facebook is consisted of user posts and photos we can define an interaction of user A to user B as user A’s comments, likes or tags on user B’s post or photo. User, whose trust towards another user, is computed is called ego user.

Table 3.1 List of interactions between ego-user and his firends

| **Interactions of friend towards ego user** | **Interactions of ego user toward friends** |
|---|---|
| Friend posts on user timeline | User posts on friends timeline |
| Friend likes post made by user | User likes post made by friend |
| Friend comments on post made by user | User comments on post made by friend |
| Friend is tagged in post made by user | User is tagged in post made by friend |
| Friend commented on a photo of user | User commented on a photo of friend |
| Friend liked photo of user | User liked photo of friend |
| Friend is tagged in photo of user | User is tagged in photo of friend |

Table 3.1 shows list of interactions between ego user and his friends. They are divided into two groups: interactions of ego user towards his friends and interactions of ego user's friends towards him.

Direct algorithm is computed from these interactions. Since these two types of interactions have differences in their collecting, which will be described later, they will define two different direct algorithms; one computed from interactions of ego user which is called engagement algorithm, and another computed from interactions of friends towards ego user which is called popularity algorithm. Both of these algorithms are computed same, they only differ in types of interactions. Trust value between ego user and his friend A is computed by this expression:

$$trust(ego\ user, friend_A) = \frac{\sum_{i \in I} i(A) * w_i}{\sum_{i \in I} w_i} \quad (1)$$

where I is the set of interactions of friend A towards ego user for engagement algorithm, and set of interactions of ego user towards friend A for popularity algorithm. i(A) is the number of interactions *i* towards ego user from friend A for engagement algorithm and number of interaction towards friend A form ego user for popularity algorithm. $w_i$ is the weight of the interaction *i*. Weights are different for every action and are calibrated in the later chapters of this work.

When computing both of these algorithms there are some limitations that have to be put forward. Due to the Facebook permission system, after user gives his permissions to the application, only his posts and photos can be collected, which means that only friends interactions towards him (comments, likes, tags) can be collected too. This means that ego user's interactions toward a specific friend will be collected only when that friend enters a system. Even if user actions towards friends could be fetched, it would be impractical to fetch all this data in one session of one user. This is why this work will only use friend's interactions towards ego user in determining his trust towards his friends. Table 3.2 shows final list of interactions used to compute trust value among Facebook users based on their interactions.

Table 3.2 List of interactions used for direct algorithm

| **Description of action** |
|---|
| Friend likes post made by user |
| Friend comments on post made by user |
| Friend is tagged in post made by user |
| Friend commented on a photo of user |
| Friend liked photo of user |
| Friend is tagged in photo of user |

## 3.2. Peer-to-peer Algorithms

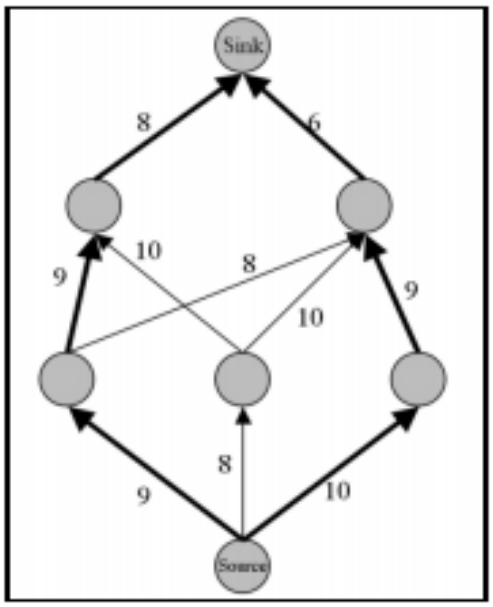
Peer-to-peer algorithms for computing trust are used for calculating trust between members (peers) that are not directly connected. Users from social network can be represented as nodes in a graph, and friendships as edges that connect two nodes in the graph. It is possible to treat this network as a network of peers and use algorithms for determining trust in peer-to-peer network to determine trust in social networks.

### 3.2.1. Tidal Trust Algorithm

The adaptation of Tidal Trust algorithm found in (Golbeck, 2005) will be presented here. Tidal Trust is an algorithm for computing inferred trust amongst users in social network. For this peer-to-peer algorithm to work, first a trust amongst users that are directly

connected has to be defined. In Facebook, two directly connected users are two friends on Facebook. Trust between users that are connected is direct interaction-based trust presented in the previous chapter, and it will be denoted as $t_{i,j}$, where $i$ is the user who has some trust $t_{i,j}$ towards his friend $j$. With this definitions, we can construct a directed graph where the nodes represent users, and edges connect two users that are friends and at least one of them has trust towards the other one greater than zero. The graph is directed since the trust has its direction. One user can have different value of direct trust towards his friend than that friend has towards him.

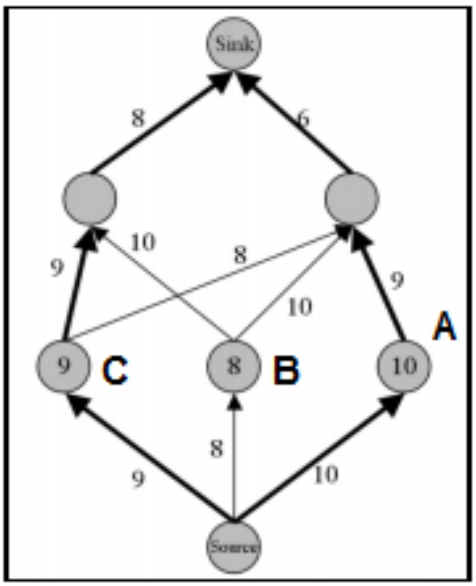
This algorithm, as an example of peer-to-peer algorithm, can compute trust between two users that are not directly connected, that are not friends. When computing trust amongst two not directly connected users, or when computing trust among two users without considering their direct trust, we call the first user 'source' and trustee user 'sink'. This algorithm's goal is to determine source's trust towards sink, using the graph structure.




Picture 3.1 Graph structure created while computing Tidal Trust

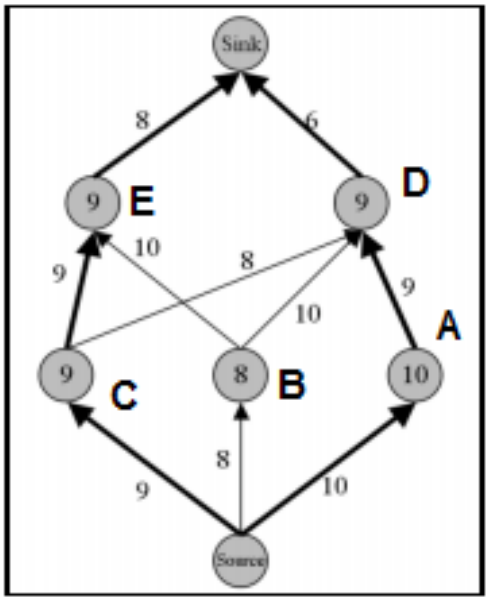
Picture 3.1 shows simple graph constructed for Tidal Trust calculation. Nodes represent users and edges connect two users that are friends and where one user has trust value towards the other. Tidal Trust algorithm consists of two parts: finding a path from the source to the sink while rating nodes while on the way to the sink, and, after the sink is found, aggregating trust backwards towards the source.

The first part of Tidal Trust algorithm starts from the source. Source looks into all of his neighbors searching for a trust value towards the sink. Sources neighbors are his friends towards who he has some trust value. If some neighbor has a trust value towards the sink then the sink is found. If the sink is not found, every neighbor of the source asks his neighbors if they have their trust value towards the sink. We call source's neighbors as nodes at level 1, their neighbors as nodes on level 2 etc. This procedure continues in recursively manner for every level, searching for a sink. While doing this procedure, every node in current level are given ratings by nodes in previous level. Picture 3.2 shows the state of the graph after the first step of Tidal Trust algorithm.




Picture 3.2 Graph after first step of Tidal Trust algorithm

The neighbors of the source, on level 1, are rated by source on level 0. The source rates his neighbors with his trust value towards them. In this example, node C is rated 9 because source has his trust value towards him valued 9. After every node at level 1 has been rated and none of them has found the sink, the algorithm continues to the next step. Every node from level 1 asks all of its neighbors if they are connected to the sink. These neighbors are nodes on level 2 and they are also rated, this time by nodes on level 1. Their ratings are calculated different than the ones from source. Every node from level 1 gives its rating to his neighbor in level 2 and the value of that rating is the minimum between his rating and his trust towards that neighbor. If some node from level 2 has more than one predecessor in level 1, then his rating is the maximum of the ratings that his predecessors gave him.




Picture 3.3 Graph after a first part of Tidal Trust algorithm has finished

Picture 3.3 shows the graph state after the first part of the algorithm is finished and the sink is reached. In this example, node E has been rated by its predecessors C and B. C rated E with the minimum of his rating (9) and his trust towards E (9) which is 9. B rated E with the minimum of his rating (8) and his trust towards E (10) which is 8. The final rating of E is the maximum between these two ratings and its value is 9.

This first part of algorithm continues inn recursive manner as BFS. The goal of this part of algorithm is to dynamically determine the threshold value of the trust. The threshold value is determined once the sink is reached. This value is defined as the maximum rating of the nodes in the last level of the graph that are connected to the sink, that have some trust towards the sink. In this example, both nodes E and D, have trust towards the sink valued 9. The threshold is the maximum between these ratings and in this example, its value is 9.

Once the trust threshold is established the algorithm continues with its second part where the trust threshold is used. In the second part of the algorithm, trust towards the sink is propagated through all levels back to the source. Every node in the graph, except the nodes in the final level of the graph that have direct trust values towards the sink, calculate their trust values towards the sink by this expression:

$$t_{i,k} = \frac{\sum_{j \in adj(i) | t_{i,j} \geq max} t_{ij} * t_{jk}}{\sum_{j \in adj(i) | t_{i,j} \geq max} t_{ij}} \quad (2)$$

Where max is the trust threshold, $t_{i,k}$ is the trust between the nodes i and k, and j are all neighbors of node i.

Nodes that are connected to the sink have their values towards the sink. Nodes that have them as neighbors will calculate their trust values towards the sink using the formula above, as weighted average of trust between them and their neighbors that have their trust values towards the sink, and trust that their neighbors have towards the sink. The trust threshold is used to filter out the nodes with low rating. In these calculations only the nodes with rating value above trust threshold are used. This is recursively repeated for every level of nodes, until the source is reached and his trust value towards the sink is computed.

We can observe that this algorithm stops searching for the sink as soon as it founds it. Also, this algorithm favors nodes with higher trust values on their path to the sink. (Golbeck, 2005) proposes that shorter propagation paths yield more accurate trust estimates, and that paths containing higher trust values yield better results too. This is why only the shortest paths to the sink are used in computation and why threshold is dynamically determined to favor nodes with higher trust values.

```
function calc_sink_trust(source, sink, nodes, visited, direct_trusts) {
	sink_found = false;
	treshold = 0;
	new_nodes = array();
	foreach (nodes as node) {
		foreach (node->neighbours as neighbor) {
			if(neighbor->Id == sink->Id) {
			    sink_found = true;
			    node->sinkTrust = neighbor->trust_previous_to_this;
			    node->reachedSink = true;
			    if(node->Rating > treshold) {
			      treshold = node->Rating;
			    }
			}
		}
	}
	if(sink_found) return $treshold;
	else {
		new_nodes = fetch_unvisited_neighbours(sink, source, nodes, v
	isited, direct_trusts);
		if(count(new_nodes) == 0) return 0;
		treshold = calc_sink_trust(source, sink, new_nodes, visited,
	direct_trusts);
		foreach (nodes as node) {
			if(node->Rating < treshold) continue;
			sum1 = 0; sum2 = 0;
			foreach (node->neighbors as neighbor) {
				if(neighbor->reachedSink == true) {
				    node->reachedSink = true;
				    sum1 = sum1 + neighbor-
				>trust_previous_to_this * neighbor->sinkTrust;
```

```
                        sum2 = sum2 + neighbor-
                    >trust_previous_to_this
                    }
                }
                if(sum2 != 0) node->sinkTrust = $sum1 / $sum2;
            }
            return treshold;
        }
}
function tidal_trust(source, sink, direct_trusts) {
        nodes = fetch_unvisited_neighbours(sink, source, nodes, visited
direct_trusts);
        visited.push(source);
        treshold = calc_sink_trust(source, sink, nodes, visited, direct_tru
sts);
        sum1 = 0; sum2 = 0;
        foreach (nodes as node) {
            if(node->Rating < treshold) continue;
            if(node->reachedSink == true) {
                sum1 = sum1 + node->trust_previous_to_this * node-
            >sinkTrust;
                sum2 = sum2 + node->trust_previous_to_this;
            }
        }
        if(sum2 != 0) return ($sum1 / $sum2);
        else return 0;
}
```

Code 3.1 Pseudocode of Tidal Trust algorithm

Code 3.1 shows pseudocode of Tidal Trust algorithm. This implementation of this algorithm uses propagative and aggregative properties of trust. Also, this algorithm does not take into account dynamic properties of trust. Since it uses direct trust based on interactions and user behaviours this trust is also not subjective, but it is asymmetric. Trust values in this algorithm have no upper bound and are gradual.

### 3.2.2. Mole Trust Algorithm

Mole trust is second example of peer-to-peer trust algorithm, adaptation of algorithm found in (Rici et al., 2011). This type of algorithm is similar to Tidal Trust. The difference between this algorithm and Tidal Trust is in determination of threshold and determination of search for sink. Tidal Trust determines threshold dynamically and ends its search for sink once it founds it. Mole Trust does not stop there, but it rather continues to look for sink further more. In this algorithm not all paths to the sink have the same length because this algorithm does not stop when the sink is reached. Mole Trust algorithm stops its search for the sink when it reaches depth d. Once the depth d is reached, algorithm continues with its second part and computes trust values towards the sink in the similar manner as Tidal Trust. The difference in aggregation of trust is in trust threshold value which is not

determined dynamically, but is set to a static value which is usually 60% of the highest rating of the nodes that are directly connected to the sink.

Mole trust has the same properties as Tidal Trust algorithm, it just explores them in different manner.

### 3.2.3. Eigen Trust Algorithm

Eigen Trust algorithm is probabilistic graph-based trust algorithm for computing trust amongst nodes in peer-to-peer network. (Kamvar et al., 2003) uses this algorithm to determine global trust values for nodes in peer-to-peer network. Peers mutual transactions are used to compute local trust values between two peers. These transactions can be either satisfactory or unsatisfactory. $s_{ij} = sat(i,j) - unsat(i,j)$ is defined as local trust value between two peers i and j, where sat(i,j) is the number of satisfactory transactions and unsat(i,j) is the number of unsatisfactory transactions between peers i and j. These local trust values have unlimted domain, and they can also be negative. Eigen Trust demands them to be normalized to values between 0 and 1. Normalized values of trust between nodes i and j are given by this equation:

$$c_{i,k} = \frac{\max(s_{i,k}, 0)}{\sum_{j \in adj(i)} \max(s_{i,j}, 0)} \tag{3}$$

Where $s_{ij}$ is the local trust value and $c_{ij}$ is the normalized local trust value between nodes $i$ and $j$. We can see from the expression that all trust values are from interval [0,1]. An adaptation, made in this work, uses interaction-based direct trust values as local trust values between nodes. These values are nomralized so that their domain is [0,1].

Table 3.3 Types of interactions used for local trust values in Eigen Trust algorithm

| **Description of action** |
|---|
| Friend x likes post made by user |
| Friend x comments on post made by user |
| Friend x is tagged in post made by user |
| Friend x commented on a photo of user |
| Friend x liked photo of user |
| Friend x is tagged in photo of user |

Table 3.3 shows list of actions used for computation of normalized direct trust values between two friends on Facebook. If the set of all types of actions is I, then we say that set of all types of normalized actions is IW. If we denote $i(x), i \in I$ as the number of one type of action I then we can define:

$$iw(x) = \frac{i(x)}{\sum_{y \in Y} i(y)} \tag{4}$$

Where iw(x) is the normalized number of correspondent action I of user x, and Y is the set of all friends of ego user that have some action towards ego user. Finally, we can define trust of ego user towards his friend x as:

$$trust(ego\ user, friend_x) = \frac{\sum_{iw \in IW} IW(x) * w_{iw}}{\sum_{iw \in IW} w_{iw}} \tag{4}$$

It can be seen from the expression that this trust values are from interval [0,1]. This algorithm will be denoted as normalized direct algorithm.

After an adaptation of calculation of local trust values of Eigen Trust algorithm we continue by adapting its peer-to-peer calculations. Again, user whose trust value algorithm is searching is called source, and the target user is called sink. In Eigen Trust, source first looks for sink's trust value in his neighbours. Neighbours of the node are his friends in social network Facebook. Trust of the source towards the sink is calculated by this expression:

$$t_{ik} = \sum_{j} c_{ij} * c_{jk} \tag{4}$$

Where i is the index of the source, k is the index of the sink, and j are nodes that are sources neighbours that have trust values towards the sink. This trust value towards the sink is calculated by checking only one level of sources neighbour nodes. This procedure continues by every neighbour of the source checking for the sink trust values in all of their neighbours. Calculation that computes trust by checking two levels of neighbours is given by expression:

$$t_{il} = \sum_{j} c_{ij} * c_{jl} + \sum_{j} c_{ij} \sum_{k} c_{jk} * c_{kl} \tag{4}$$

Where i is the source, l is the sink, j are the sources neighbours and k are the neighbours of sources neighbours. This procedure can continue recursively until some set depth d. Eigen trust values require that trust values are from interval [0,1] because this computation does multiplication of trust where longer paths to the sink have more multiplications. Because of this it is needed that the value of the trust gets smaller by every next level, and that is provided by normalizing trust values of direct algorithm to interval [0,1].

```
function find_sink_trust(source, sink, direct_trusts, nodes, visited, tru
st, iteration) {
    next_level_nodes = [];
    new_trust = 0;
    foreach (nodes as node) {
        visited.push(key);
        node_neighbors = get_node_neighbors(node, direct_trusts);
        foreach (node_neighbors as node_neighbor) {
            if(node_neighbor in visited) continue;
            if(node_neighbor == sink) {
            new_trust = new_trust + node.trust_previous_to_this * node_ne
      ighbor.trust_previous_to_this;
            }
            node_neighbor.trust_previous_to_this = node.trust_previous_to
      _this * node_neighbor.trust_previous_to_this;
            next_level_nodes.push(node_neighbor);
        }
    }
    if(iteration == depth) return trust * new_trust;
      return eigen_calc_sink_trust(source, sink, direct_trusts, new_nodes
      , visited, trust + new_trust, iteration+1)
}
function eigen_trust(source, sink, direct_trusts) {
    source_neighbors = get_node_neighbors(source, direct_trusts);
    visited = [];
    visited.push(soruce);
      trust_value = find_sink_trust(source, sink, direct_trusts, source_n
      eighbors, visited, 0, 0);
    return trust_value;
}
```

Code 3.2 Pseudocode of Eigen Trust Algorithm

Eigen Trust algorithm is different from the previous two because it is a probabilistic algorithm. It also uses propagation and aggregation as properties of the trust, but explores them in different way than previous two algorithms. Difference between this implementation and the one in the literature is that this algorithm uses local trust values computed from interactions rather than experiences. Also, Eigen Trust algorithm used in peer-to-peer network calculates global trust value for every user, while this adaptation calculates trust values between two users in social network.

## 3.3. Combination of algorithms

In this chapter we analyze what combinations or variations of the algorithms could provide more precise results of computing user's trust towards his friends. Peer-to-peer algorithms are usually used to compute trust towards the users that are not directly connected to the ego user, and while this is their main purpose, they can also be used to compute the trust of ego user's friends. When computing ego user's trust towards a friend by peer-to-peer algorithm, we discard his trust towards the friend and try to compute it from the neighbor nodes, that is, we try to compute it from the network. While doing this, we consider the network information about the ego user's trust towards the friend and discard the direct trust information. Also, when computing direct trust of ego user towards his friend, we discard the network information.

User interactions are not always capable to show precise results of user's trust toward a friend. Sometime it is because we are limited to some subset of interactions and sometimes it is because some users are just not active enough on Facebook and their trust results can be misleading. Network structure can provide additional information about ego user's trust toward his friend by taking into account other, propagated trusts in the network. This way, lack of information about the user interactions can be covered by the trust information collected from the network structure and peer-to-peer algorithms. Also, peer-to-peer algorithm performance highly depends of the structure of the network. Some parts of the networks can be poorly covered and trusts that should be computed from that part of the network could be missing. This is way combination of direct trust information from user interaction with his friends and peer-to-peer trust information computed from network structure could be the best solution in finding a general trust values between ego user and his friends. This is why direct trust algorithm results will be combined with peer-to-peer

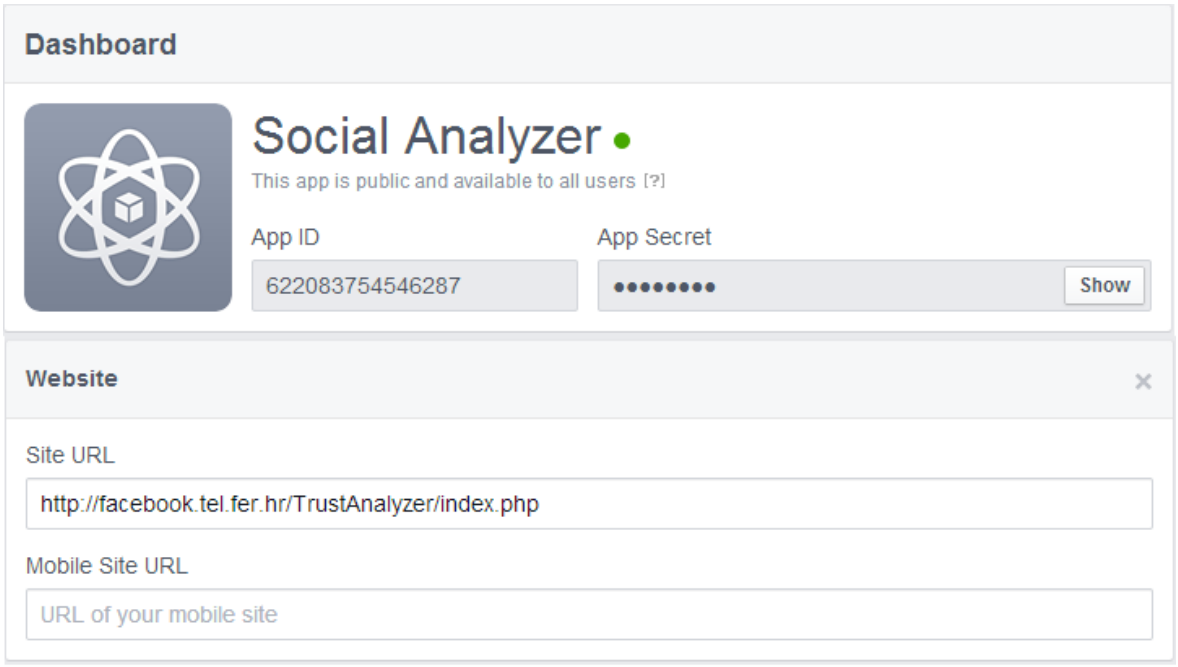
trust algorithm results and analyzed to see if their results are more precise than those of individual algorithms.

# 4. Application System for Trust Computation

For analysis of trust computing, an application has been developed, that enables collecting data from Facebook, storing data, calculating trust and collecting survey data filled in by users. This application has been developed in PHP language and uses relational database MySql for data storage. In this chapter, first something about Facebook specific parts of the application is explained after which the general architecture of the application is presented. Furthermore, a data model with description of the data will be given. Finally this chapter will end with description of the application and instructions of how to use the application.

## 4.1. Facebook Component

This application is web application, but it is also a Facebook application, which means that users can login to the system with their Facebook credentials and that they can give certain Facebook permissions to the application. With these permissions, application can make requests to Facebook and collect Facebook data about the user.




Picture 4.1 Part of the Facebook dashboard for Facebook applications

First part in creating this application was creating a Facebook application. By doing this, developer is provided with an application id and application secret for his application. This data is unique identifier of this Facebook application. It is used in every query and request made towards Facebook. Also, developer has to choose on which platform he will build his Facebook application and provide a valid URI to his application.

In order to use this application user has to give permission that application requires him to give. For purposes of this work, next permissions were requested from user:

- user_about_me
  - application can access user's basic data such as his birthday, gender, name, id etc.
- user_friends
  - application can access list off user's friends on Facebook.
- user_photos
  - application can access list of user photos as well as the data associated with these photos such as comments, likes and tags on photos.
- read_stream
  - application can access user posts on his timeline and on other user's timeline, as well as the data associated with these posts, such as comments, tags and likes on posts.

After user has granted requested permissions, application can collect his Facebook data. This is done with Facebook Graph API which provides queries for most of user data on Facebook. Queries used in this application are:

- /me
  - Return user's basic profile:
- /me/friends
  - Returns list of user's Facebook friends
- /me/photos
  - Returns all photos with user tagged in them
- /me/posts
  - Return all posts made by user

All responses are returned in JSON format. First query returns information about user; his id, name, gender and his birthday. Second query returns list of user's friends; their ids and

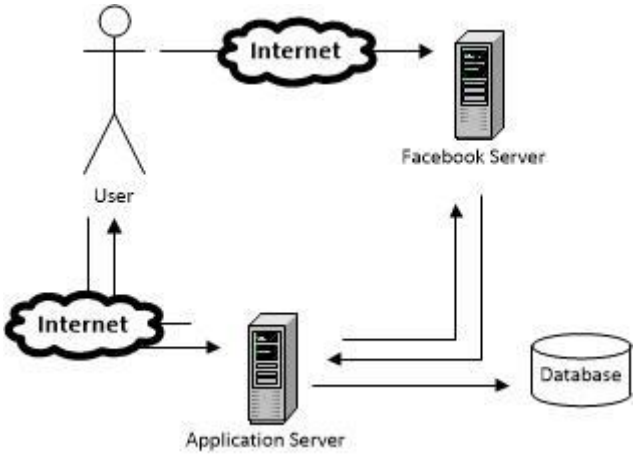
names. Third and fourth query return list of posts and photos of user. They provide ids of photos and posts and their content, but more importantly they provide list of users who commented, liked or are tagged in these posts and photos. Example of data for one post returned by Facebook API is given in Picture 4.2


```
{
  "id": "100001175623464_526632204052629",
  "from": {
    "id": "100001175623464",
    "name": "Mirjam Šitum"
  },
  "message": "Hvala svima na lipin željama za rođendaaaan :) <3",
  "type": "status",
  "created_time": "2013-06-26T10:27:48+0000",
  "likes": {
    "data": [
      {
        "id": "100001796284014",
        "name": "Josip Marić"
      },
      {
        "id": "10202455431161177",
        "name": "Marin Andrić"
      },
      {
        "id": "795207853832447",
        "name": "Josipa Maleš"
      },
      {
        "id": "793414427349390",
        "name": "Mia Korda"
      },
      {
        "id": "454503301361575",
        "name": "Marko Situm"
      }
    ]
  }
},
```


Picture 4.2 One Facebook post returned in JSON format by Facebook API

## 4.2. Application Architecture

Main components of the application system are web server, database and Facebook server.




Picture 4.3 System architecture

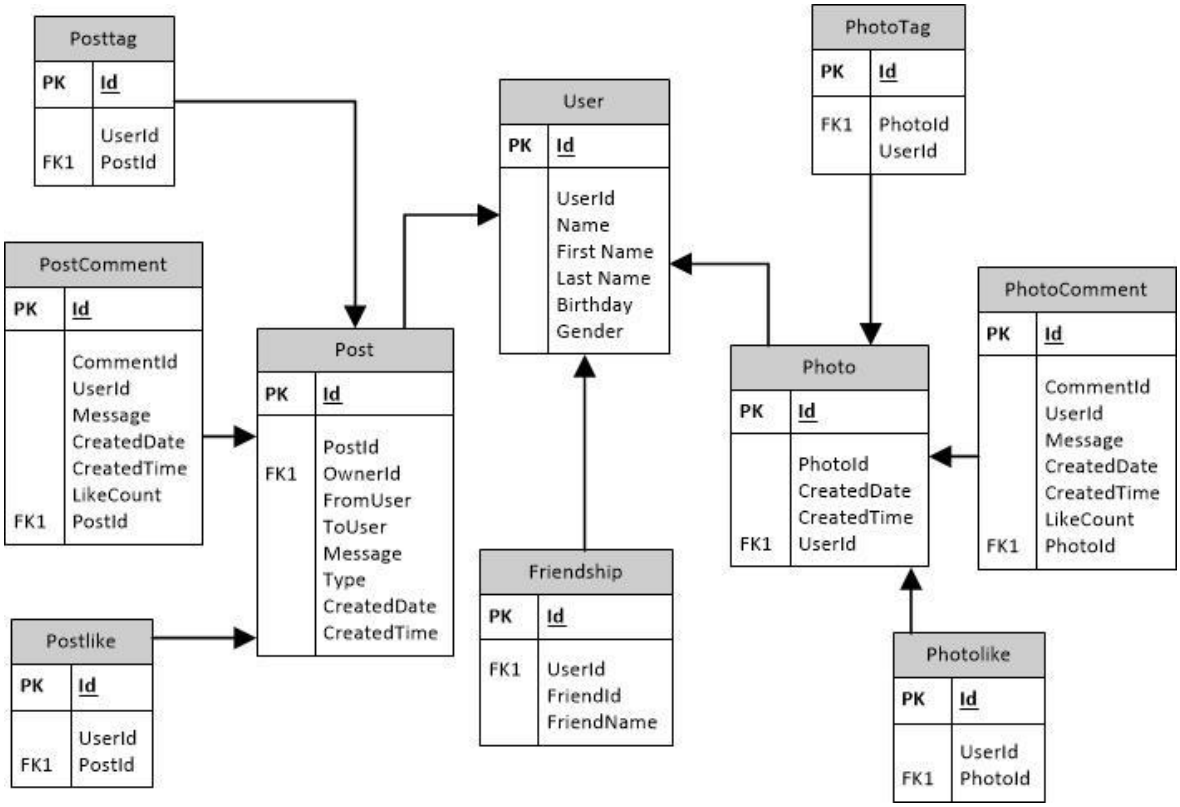
In Picture 4.3 a general flow of communication between main entities of the application is given. User communicates with application server, but he also communicates with Facebook server when he is giving certain permissions to the application. Application server mainly communicates with Facebook server while fetching data from Facebook about user. He also has to communicate with the database to store and fetch stored data. By its functionalities, this application can be divided into three main parts: collecting and storing data, computing trust and survey.

### 4.2.1. Collecting and Storing Data

This is the most demanding part of the application considering the duration of its runtime and complexity of data parsing. After user grants application with requested permissions, application makes requests to Facebook via Facebook API. Most complicated part of data collecting is parsing user posts. Since query which returns user posts has many different types of data in its response, it is important to filter only useful data. Useful data in this context are user's posts on his wall or on wall of others. Due to Facebook API's inconsistent structure of the data in response this data had to be filtered by application. Other data such as user's new friendships or activity had to be discarded.

#### 4.2.1.1 Data Model

This application uses MySql database to store data. In Picture 4.4 an ER-model of the database is given. Data collected from Facebook is transformed to relational data and stored to database. Central table of the database is *user* with basic data about the user. In *friendship* table, a list of user's friends is stored, with their ids and their names. In *photo* and *post* tables, there are lists of user's photos and posts. Every post and photo has an attribute *UserId* , which connects every photo or post to certain user in table *user*. There are also tables *PostTag*, *PostComment* and *PostLike*, which store tags, comments, and likes for every post. *Photo* table is the same as *post* but only for user photos.




Picture 4.4 ER model of the data

### 4.2.2. Computing Trust

This part of application is consisted of all algorithms used in this research. It uses the data prepared by the *Collecting and Storing Data* part of the application. Besides main algorithms implementations that are center of this research, such as direct trust algorithm based on friends actions towards ego user, Tidal Trust algorithm, Mole Trust algorithm and

Eigen Trust algorithm, this part of application also provides implementations of the methods for calibrating weights in direct algorithm. There are several scripts that are part of this part of application:

- *count.php*
  - this script provides functionality of summing up user data; it can count number of likes, comments, tags etc. User's data is counted only once, when his data is collected. This counted data is stored to database and used in computation of direct algorithm. Concrete implementation of direct algorithm is given in *algorithms.php.* User's data is counted and summed up by database queries. One example of those queries is given by Code 4.1.
- *algorithms.php*
  - an interface that provides functions for all four algorithms: direct, Mole, Tidal and Eigen. This script uses other scripts with real implementations of algorithms: mole.php, eigen.php and tidal.php
- *mole.php*
  - implementation of Mole Trust algorithm
- *tidal.php*
  - implementation of Tidal Trust algorithm
- *eigen.php*
  - implementation of Eigen Trust algorithm
- *monte-carlo.php*
  - implementation of weights calibration for direct algorithm

```
SELECT postlike.UserId, friendship.FriendName, count(*) as number FROM
postlike JOIN friendship ON postlike.UserId = friendship.FriendId  JOIN
post ON post.PostId = postlike.PostId WHERE friendship.UserId = ? AND
post.OwnerId = ? AND FromUser = ? AND postlike.UserId <> ? GROUP BY
postlike.UserId, friendship.FriendId ORDER BY number DESC;
```

Code 4.1 SQL query that fetches number of likes of user's friends on his posts

### 4.2.3. Survey Implementation

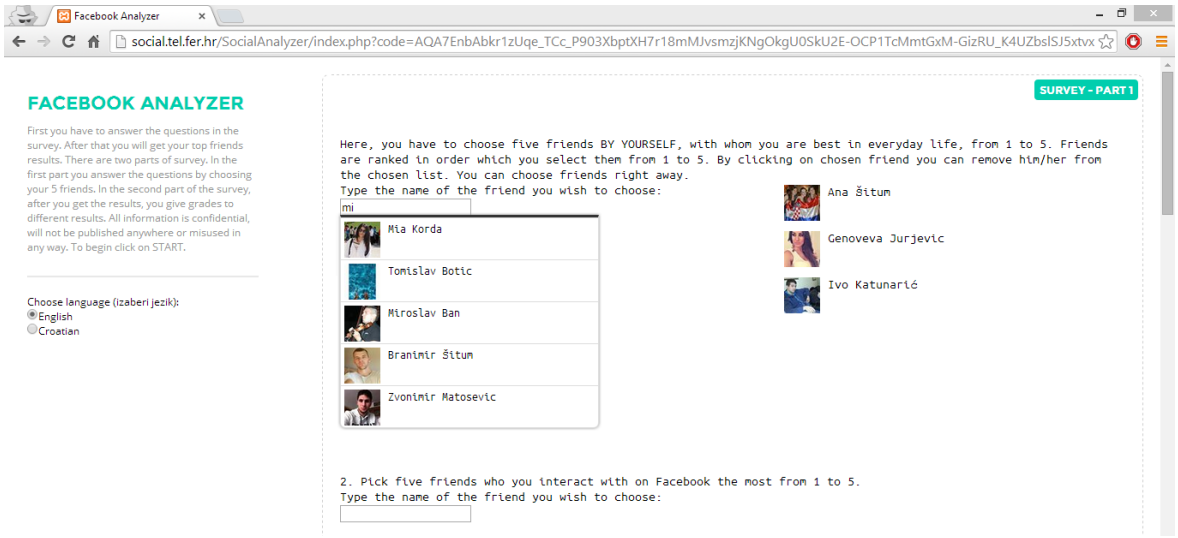
Last part of application is the survey part. There are two main types of surveys, whose data is used in application. First one is a survey whose data was used for weight calibration of

direct algorithm and second one was used for evaluation and comparison of chosen algorithms. Survey content is available in two languages, English and Croatian.

## 4.3. Instructions of How to use the Application

This application has been used for two main parts of research:

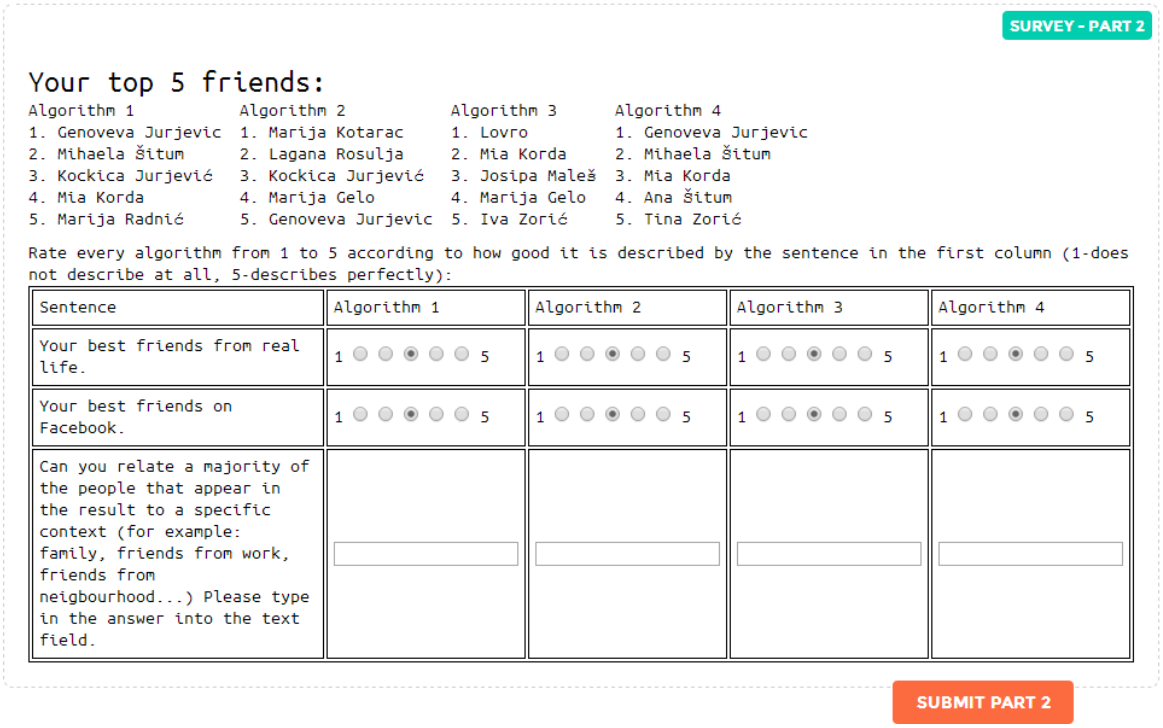
- Direct algorithm weights calibration
  - Application was collecting data of users and their answers to one question in the survey, no algorithms were calculating their results
- algorithms evaluation and comparison
  - Application was still collecting and storing data about the users, but the survey was extended with more questions for the main algorithm analysis. Also all the algorithms were implemented to show the results to the user.




Picture 4.5 Application interface during the weight calibration data collection

Picture 4.5 show the application interface in the first part of the research. General structure of the web page consists of: description, survey and results. Description part is located on the left part of the web page. It tells user about the application and gives him directions of how to use it. Two other parts are located on the right. In the upper part of the web page is the survey. In the first part of the research, the survey only had one question, but the second part of the research there were five questions located in this part of the web page. These questions were all of the same type, asking the user to choose five of his friends. User can choose his best five friends with the search bar which enables him to find his Facebook friends. The result part is located in the lover part of the web page. This part

shows the results of the algorithms to the user. While the data is collecting and the algorithms are calculating, user is instructed to wait for the results. Picture 4.6 shows the results part of the web page in the second part of the research. User can see the results of the algorithm after they finish and he can answer the second part of the survey after he gets his results.




Picture 4.6 Results part of the web page

# 5. Algorithm Analysis

In this chapter, complete review of analysis of algorithms will be presented. First there will be a review of used measures for determining algorithm performances and precision. After that all steps in research will be described and finally the results of algorithm evaluation will be shown.

## 5.1. Metrics for Algorithm Evaluation

This subchapter presents four different measures for trust evaluation: Kendal Tau distance, rank difference match count and exact count.

### 5.1.1. Kendal Tau Distance

The Kendall Tau distance is a metric that counts the number of pairwise disagreements between two ranking lists (Kendal, 1938). The larger the value of the distance, the more different the lists are. The Kendall tau ranking distance between two lists L1 and L2 is:

$$K(t_1, t_2) = \left|\left\{(i,j) : i < j, \left(t_1(i) < t_2(j) \wedge t_2(i) > t_2(j)\right) \vee \left(t_1(i) > t_1(j) \wedge t_2(i) < t_2(j)\right)\right\}\right| \quad (4)$$

Where $t_1(i)$ and $t_2(i)$ are the rankings of the element i in L1 and L2. The value of $K(t_1, t_2)$ is 0 if two lists are identical and $\frac{n*(n-1)}{2}$ if one list is the reverse of the other.

Table 5.1 Example with two lists and indexes of elements in the lists

| user | 1. list indexes | 2. list indexes |
|---|---|---|
| A | 1 | 3 |
| B | 2 | 4 |
| C | 3 | 1 |
| D | 4 | 2 |
| E | 5 | 5 |

The Table 5.1 shows two lists and indexes of their elements. The Table 5.2 shows an example of calculation of Kendal Tau distance.

Table 5.2 Example of Kindal Tau distance calculation

| **Pair** | (A,B) | (A,C) | (A,D) | (A,E) | (B,C) | (B.D) | (B,E) | (C,D) | (C,E) | (D,E) |
|---|---|---|---|---|---|---|---|---|---|---|
| **First list indexes** | 1 < 2 | 1 < 3 | 1 < 4 | 1 < 5 | 2 < 3 | 2 < 4 | 2 < 5 | 3 < 4 | 3 < 5 | 4 < 5 |
| **Second list indexes** | 3 < 4 | 3 > 1 | 3 > 2 | 3 < 5 | 4 > 1 | 4 > 2 | 4 < 5 | 1 < 2 | 1 < 5 | 2 < 5 |
| **Count** | - | X | X | - | X | X | - | - | - | - |

From the definition of the Kendal Tau distance and the Table 5.2 it can be seen that he Kendal Tau distance for this example is four.

### 5.1.2. Rank Difference

This measurement is calculated from user's answers in a survey on questions that ask to choose top 5 friends ranked from first to fifth which make one list for this measurement (Flitz, 2014). Other list for the measurement is given by the algorithms result which ranks user's friends from first to last. Table 5.3 shows an example of rank difference calculation. First, the differences between user's rankings and algorithm rankings are computed. After that the differences are summed up and give the total difference that is the result of rank difference measurement.

Table 5.3 Example of Rank difference calculation

| **Friends** | **User Perception Ranking** | **Algorithm Ranking** | **Difference** |
|---|---|---|---|
| A | 1 | 53 | 52 |
| B | 2 | 8 | 6 |
| C | 3 | 1 | 2 |
| D | 4 | 15 | 11 |
| E | 5 | 3 | 2 |
| **Total Difference = 73** | | | |

### 5.1.3. Match Count

Match Count counts the matches between user perception of the top 5 friends and the algorithm results for the top 5 friends. We make a difference between a top 5 match count and an exact match count (Flitz, 2014). Table 5.4 shows an example of Match Count calculation. Top 5 match count counts only how many elements are the same in the lists. Exact match count counts only how many elements in the lists are at the same index.

Table 5.4 Example of Match Count calculation

| **Friend** | **User Perception Ranking** | **Algorithm Ranking** |
|---|---|---|
| A | 1 | 20 |
| B | 2 | 2 |
| C | 3 | 5 |
| D | 4 | 15 |
| E | 5 | 3 |
| **Match Count = 3** | | |
| **Exact Match Count = 1** | | |

## 5.2. Direct Algorithm Weight Evaluation

### 5.2.1. Weight Calibration

The research of direct interaction-based trust computation algorithm that calculates trust between users on Facebook is given in (Skorin et al., 2013). Different types of interactions have been used to calculate trust between friends. Types of interactions used in (Skorin et al., 2013) differ from the types of interactions used in this work, but the approach of determining the weights of the algorithm will be similar. The method used in (Skorin et al., 2013) is an example of Monte Carlo method in which all possible solutions from determined solution space are checked and evaluated to see what inputs of weights give best solution.

Direct algorithm used in this work has six different types of interaction and six weights for every type of interaction: weight for likes on ego user posts, comments on ego user posts,

tags in ego user posts, likes on ego user photos, comments on ego user photos and tags in ego user photos. The weights have been set to the value 10 and all solutions have been calculated for weights +-10 from the initial values of weights. This approach is similar to the one in (Skorn et al., 2013). For every weight entry a goodness of a solution has been defined by a Rank difference measurement.

For this part of research the application was tuned to fetch and store user's data and to calculate the ranking of user friends. Also, the application had one question which asked users to choose their top 5 friends from 1 to 5 who are their best friends on Facebook. 100 users fill in the survey question and had their data collected. For every user, number of likes, comments and tags on his posts, as well as the number of likes, comments and tags on his photos, made by his friends, was counted and stored.

The calibration of weights was done according to this one specific question that was asked from users to fill. This is because the goal of the direct algorithm was to correspond with the best friends of user on Facebook. The goal of computed weights is to show how much influence every type of interaction on Facebook has on user's trust towards his Facebook friends. This is why the weights are calibrated considering this question and this algorithm. Later, direct algorithm is just a tool to see how different peer-to-peer algorithms explore propagation of this trust through the network.

### 5.2.2. Implementation and Results

Weights calibration has been done by evaluating goodness of the direct algorithm based on rank difference measurement for every weights input from the interval [1,20] for every weight, with the step 0.5. For every different input, algorithm would give different order of user's friends. The weights input that gave the best result was chosen to be the final combination of weights and it is used to compute direct trust values from that point on.

The goodness of the solution is computed from the goodness of rankings of friends of all users in the system. The goodness of the rankings of friends of one user is given by the rank difference measurement. Top 5 friends chosen by the user present the first list for the rank difference calculation, and the algorithm results provide the second list for the rank difference calculation. The higher the rank difference the lower the goodness of the ranking. The goodness of the solution is then reciprocal to the sum of the rank differences of all users in the system.

Since this computation is complex because ranks of friends have to be computed for every user and for every weights input, the calibration of weights took 60 hours to complete. After it finished, the best solution was found and the weights which gave that solution were chosen. Table 5.5 shows the results of the weight calibrations.

Table 5.5 Weight values for direct algorithm

| Weight | Description | Value |
|---|---|---|
| W1 | Likes on user posts | 2.5 |
| W2 | Comments on user posts | 5 |
| W3 | Tags on user posts | 10.5 |
| W4 | Likes on user photos | 1 |
| W5 | Comments on user photos | 12 |
| W6 | Tags on user photos | 2 |

## 5.3. Survey

In this chapter, a survey used for evaluation of algorithms will be presented. The survey was divided into two parts. First part of the survey is consisted of questions where user has to choose five friends, ordered from 1 to 5, that satisfy the answer to the specific question. This part of the survey is filled in before the data is fetched and before the algorithms compute their results. The second part of the survey is filled in after the results of the algorithms, because it has questions that are concerning the algorithm results:

The first part of the survey has five questions that are available in English and Croatian language. Their content is shown in Table 5.6.

Table 5.6 Questions from the first part of survey

| | Questions in English | Questions in Croatian |
|---|---|---|
| 1 | Here, you have to choose five friends BY YOURSELF, with whom you are best in everyday life, from 1 to 5.Friends are ranked in order which you select them from 1 to 5. By clicking on chosen friend you can remove him/her from the chosen list. You can choose friends right away. | Ovdje morate sami odabrati 5 prijatelja s kojima ste najbolji u stvarnom životu. Prijatelji će biti rangirani po mjestu na kojem ih stavite od 1-5. Pritiskom na ime prijatelja možete ga maknuti iz liste za odabir. |

| | | |
|---|---|---|
| 2 | Pick five friends who you interract with on Facebook the most from 1 to 5. | Odaberite 5 prijatelja s kojima najčešće interaktirate na Facebooku i rangirajte ih od 1. do 5. mjesta. |
| 3 | You are planning a trip around the world. Choose 5 friends to take with you by preference from 1 to 5 ? | Planirate putovanje oko svijeta. Odaberite 5 prijatelja koje biste najrađe poveli sa sobom i rangirajte ih 1. do 5. mjesta |
| 4 | Who would you trust the most to take care of your apartment and/or pet while you are away? Choose 5 friends from 1 to 5. | Kome biste najviše vjerovali da za vrijeme vašeg putovanja da Vam se brine za stan i/ili ljubimca? Odaberite 5 prijatelja i rangirajte ih od 1. do 5. Mjesta. |
| 5 | You are in a good mood to see a good movie. Which five friends would you call to recommend you one. Choose from 1 to 5. | U raspoloženju ste za gledanje dobrog filma i tražite preporuku od prijatelja. Odaberite 5 prijatelja koje ćete najprije nazvati za to i rangirajte ih od 1. do 5 mjesta. |

The second part of the survey has three questions that are asking about the results that are shown after the algorithms give the results. First two questions ask the user to rate every algorithm from 1 to 5 of how good its results are described by the sentence from the question. First question's sentence is *Your best friends from real life* and the second question's sentence is *Your best friends on Facebook.* The third questions asks the user to place the majority of the people from result in some context (example: family, friends from work, friends from childhood…) for every algorithm. It is obvious that these questions require the results from the algorithms and that is why they are asked after the results.

## 5.4. Results

In this chapter the results of algorithms evaluation are shown. First there is summary of tested algorithms after which the collected data is described. Finally, all results of algorithms performance and precision are shown.

### 5.4.1. Tested Algorithms

In previous chapter, there were four algorithms presented: Direct trust algorithm with two variants: normalized algorithm and normal direct algorithm, Tidal trust algorithm, Mole trust algorithm and Eigen trust algorithm. In this chapter we present the results of the

analysis of these algorithms. The direct trust algorithm with normalizeds interaction values is denoted as Direct2.

Beside these presented algorithms, a research about the combination of direct and peer-to-peer algorithms was also made. For this reason we constructed three combinations of direct algorithms and peer-to-peer algorithms. Direct trust algorithm results were combined with Tidal trust and Mole trust algorithms. Values for trust towards one friend, given by a Direct algorithm, were summed up with the values of trust towards the same friend given by peer-to-peer algorithm. This experiment was done to check if two algorithms combined can produce better results than every algorithm by itself. Direct trust algorithms may provide information that peer-to-peer algorithms cannot and vice versa. Combination of Mole trust algorithm and Direct algorithm is denoted as Combo1, where combination of Tidal trust algorithm and Direct algorithm is denoted as Combo3. Direct2 algorithm is combined with Eigen trust because they both have normalized values and it is denoted as Combo2.

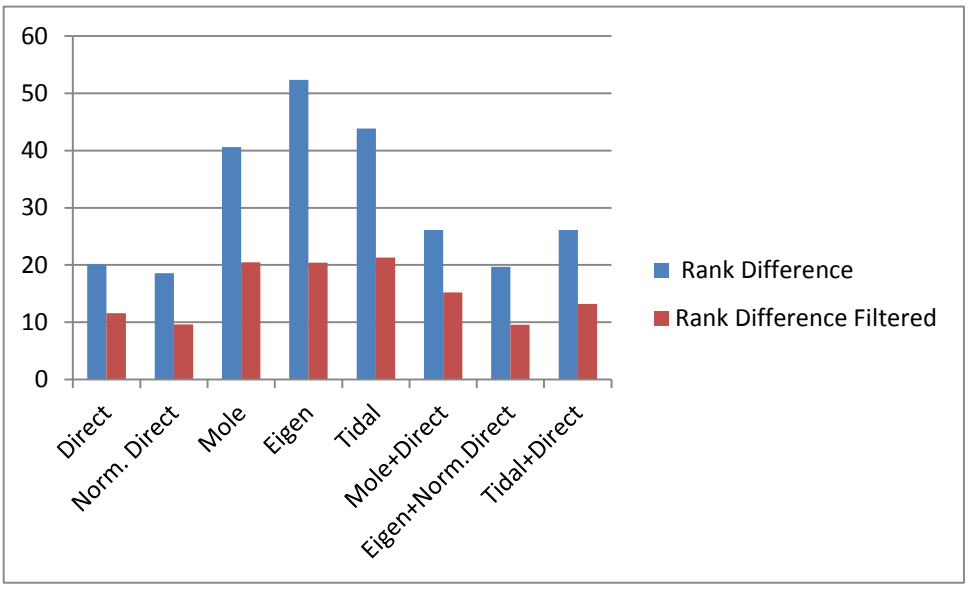
### 5.4.2. Collected Data

For the evaluation of algorithms, users were asked to fill the survey presented in the chapter above. 104 users filled the survey and their data was combined with the data of the users from the previous survey used to calibrate weights of direct algorithm. There were total of 215 users in the system when the algorithms were evaluated. 108 of them were male and 107 were female. These users formed total of 83864 friendships of which 42403 had some direct trust value assigned to the friendship. There were 34656 photos collected with 71806 comments, 186827 likes and 111777 tags. Considering the posts, there were 62747 posts collected with 46873 comments, 158183 likes and 3618 tags.

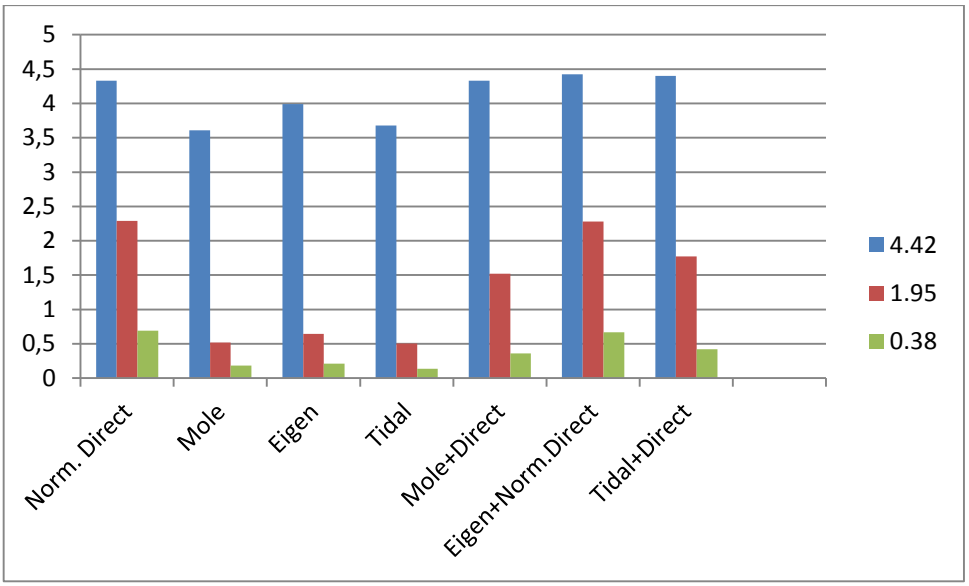
### 5.4.3. Results of the First Part of the Survey

All algorithms results were conducted for five questions from the first part of the survey. For every algorithm, results of trust computations were used to rank the friends for every user that filled the first part of the survey. The results of peer-to-peer algorithms were conducted by dismissing the direct trust value of user towards his friend and computing it from the network by peer-to-peer algorithm. The ranking of user friends were used to compare with the results that users provided in the five questions asked in the first part of the survey.

Users had to choose five friends as an answer to a question. For the Kendal Tau distance, those same five friends were taken these five friends were taken from the algorithm's ranking and the distance was calculated using these two lists: user rankings and algorithm rankings for these five friends. Distances were calculated for every user and average distance was calculated by summing every distance of every user and dividing the sum by number of users. Rank difference measure was also calculated form user's list of five friends and algorithm results, and the average ranking difference was calculated too, by dividing the total sum of all ranking distances with number of users times five to get the average ranking distance for one user's friend. Also, we calculated average ranking distance with dismissing ranking distances for friends that have distance values over 100. This was done to see the precision of algorithms when not taking into account the friends who have really high distances. These friends are considered to be mistakes made by the algorithms because of the sparse peer-to-peer network in peer-to-peer algorithms or because of the inactivity of the users in direct algorithms. Count match and exact match were calculated as described and their average result by user is given. The next set of pictures show the results of the algorithm analysis for the first part of the survey.

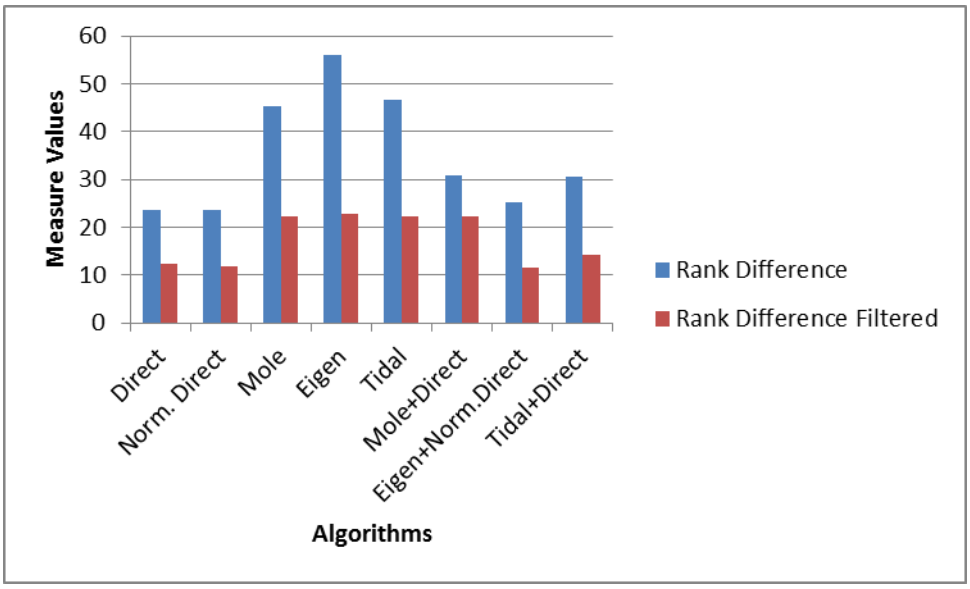



Picture 5.1Rank difference measure for question „best friend in real life"

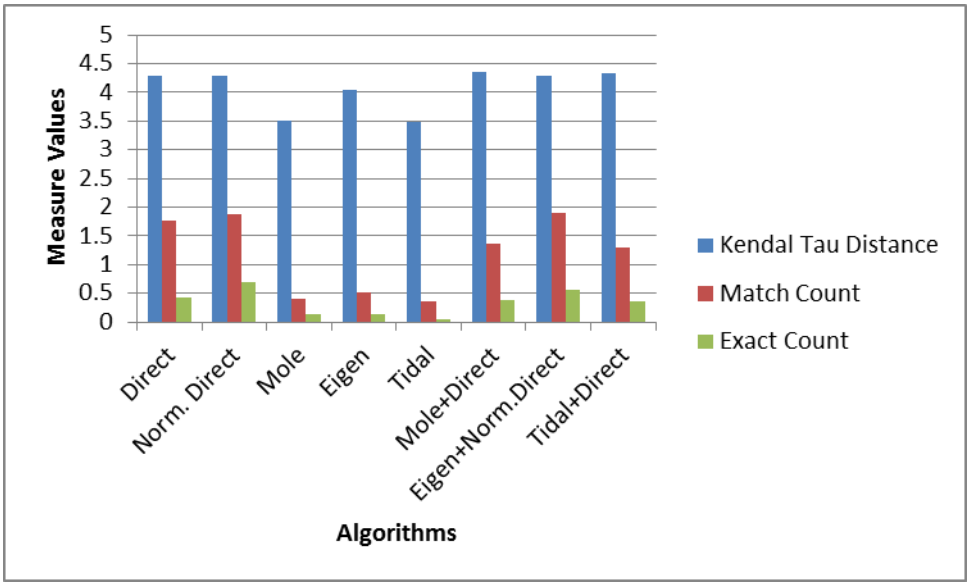



Picture 5.2 Rest of the measures for question „best friends in real life“

Picture 5.1 and Picture 5.2 show the results of the measures for the two direct algorithms, three peer-to-peer algorithms and their combinations for the first question from the survey, which asked users to choose top 5 friends whom they are best with in real life. Rank difference measure shows that direct algorithms give better results than peer-to-peer algorithms where Direct2 algorithm is slightly better than Direct algorithm. Tidal trust and Mole trust algorithms show better precision than Eigen trust although Eigen trust combined with Direct2 is the most precise algorithm. Peer-to-peer algorithms are less precise and the reason for that is that for some part of the users the network structure was too sparse for peer-to-peer algorithms to conduct good results. Nevertheless, Kendal Tau distance showed that peer-to-peer algorithms show signs of better ordering of friends than direct algorithms. The results for the first question, where users had to choose five friends whom they are best with in real life, showed good results for direct algorithms and combinations of algorithms, especially for the combination of Direct and Eigen algorithm. This algorithm showed 46% of guesses of top 5 user’s friends.



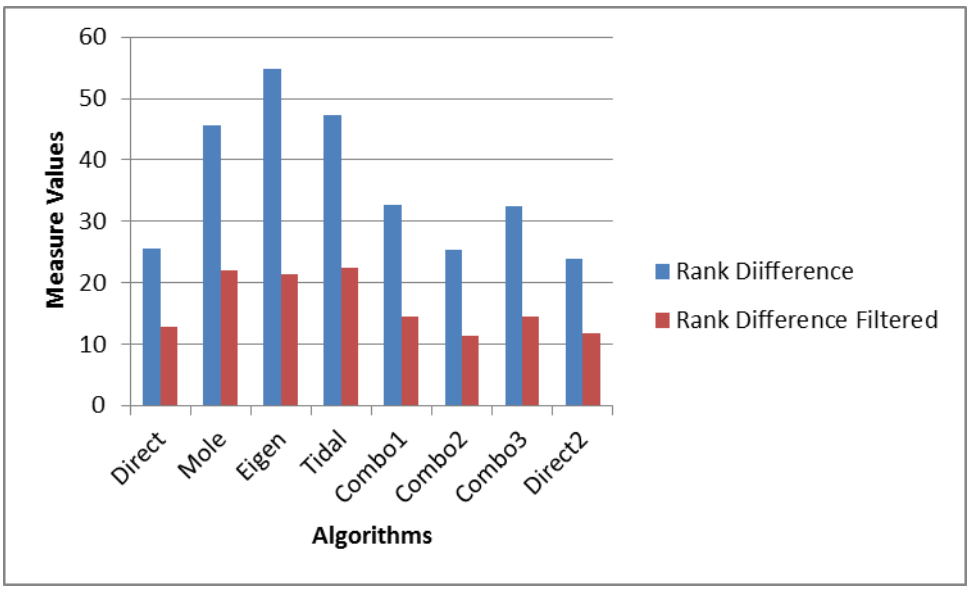

Picture 5.3 Rank difference measure for question „best friends on Facebook“



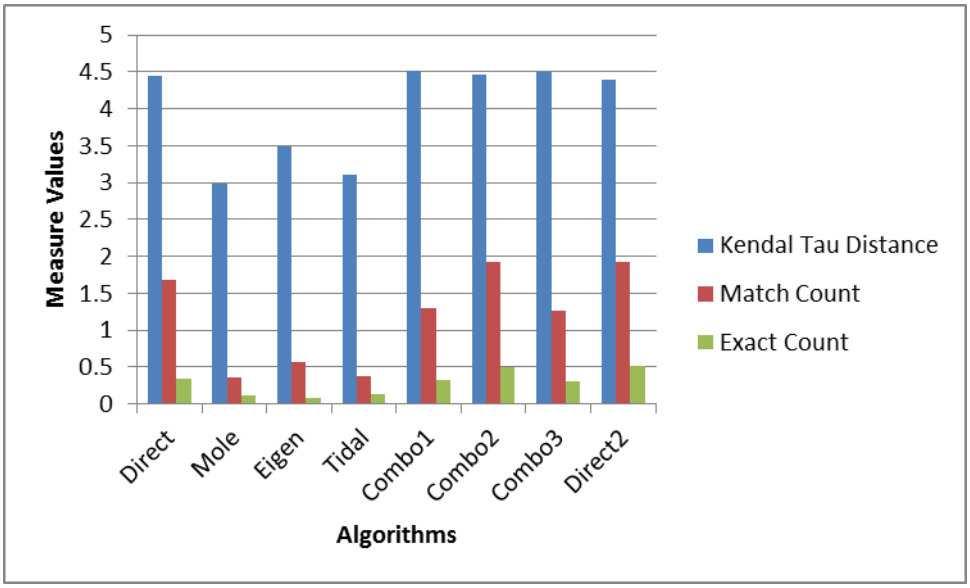

Picture 5.4 Rest of the measures for question „best friends on Facebook“

The second question asked users to choose top 5 friends whom they interact the most with on Facebook. Algorithm performances were rather similar to ones for the first question and are given in Picture 5.3 and Picture 5.4, but surprisingly they were less precise for this question than for the first one. All of the algorithms showed better results for the first question. Direct algorithms and the combination of Direct2 algorithm and Eigen algorithm again gave best results with match count measure little less than 40% this time. We can also see that the Eigen trust algorithm gives much better precision when we discard friends who have rank difference bigger than 100, which tells us that Eigen trust algorithm makes

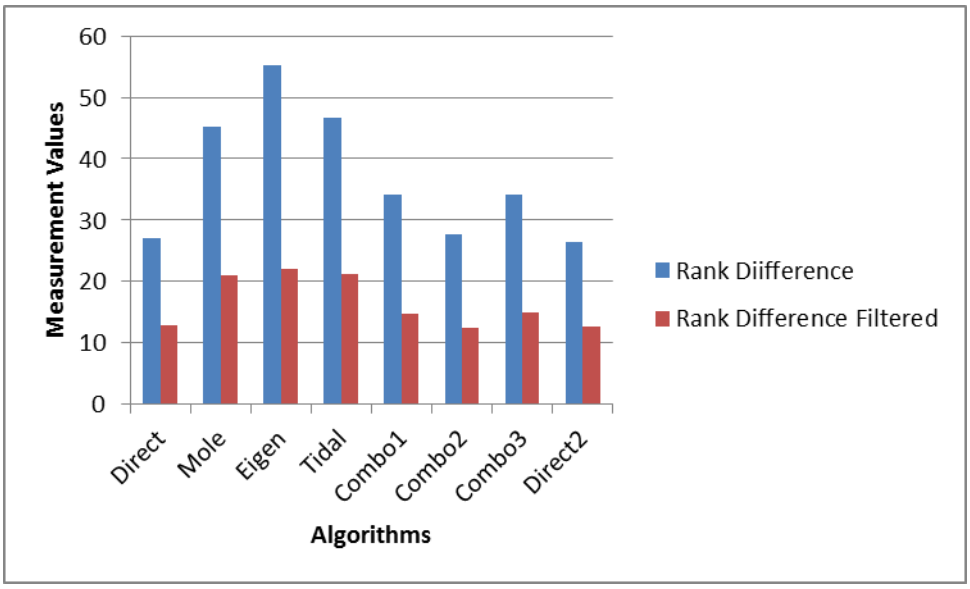
more crucial mistakes for top 5 users. Next six pictures show the results for the last three questions.



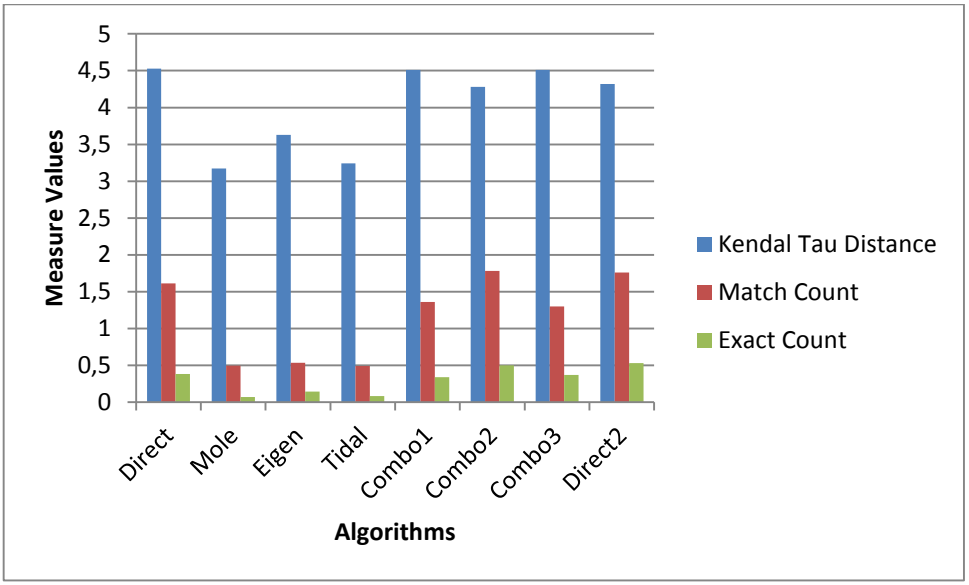

Picture 5.5 Rank difference measure for question „travel companions“



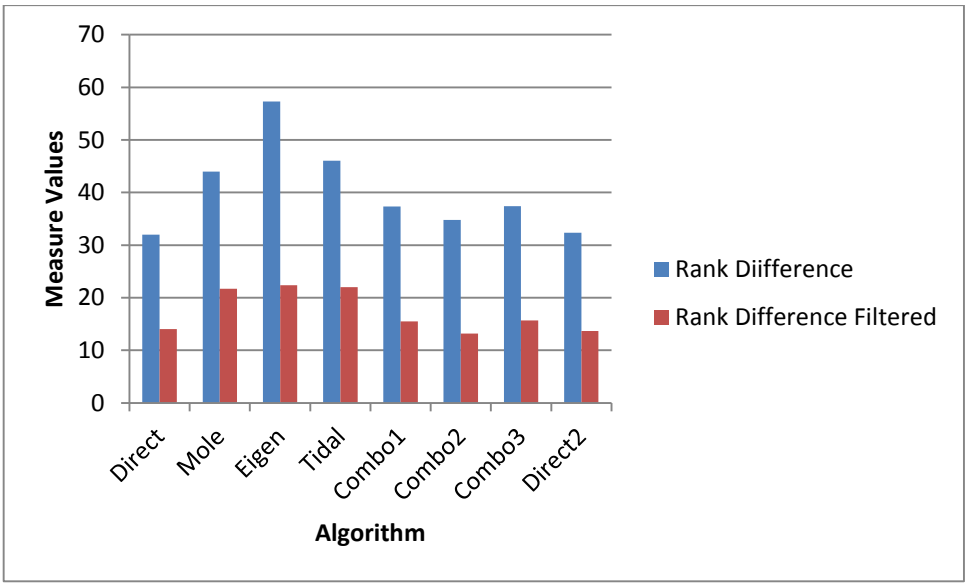

Picture 5.6 Rest of the measures for question „travel companions“



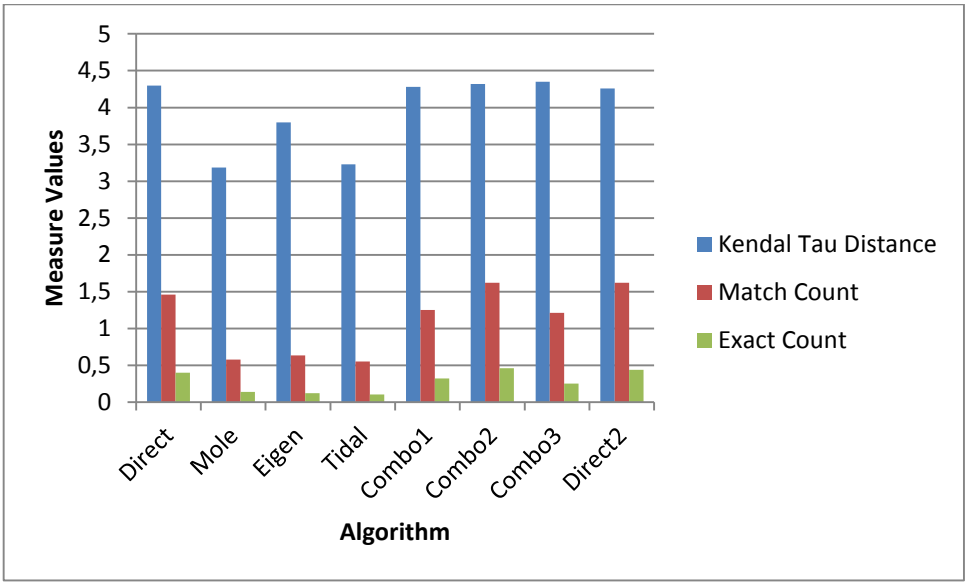

Picture 5.7 Rank difference measure for question „watching over property“




Picture 5.8 Rest of the measures for question „watching over property“



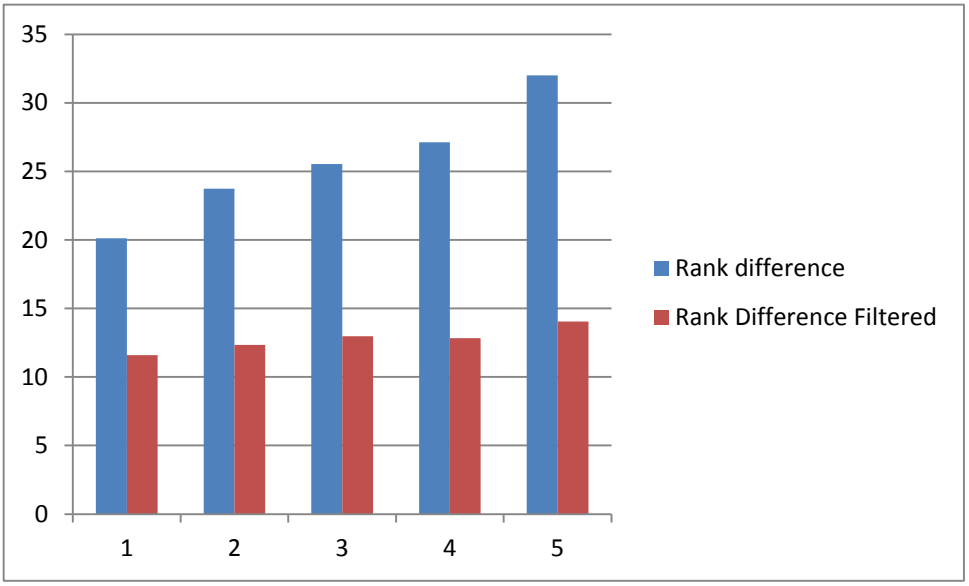

Picture 5.9 Rank difference measure for question „music recommendation“



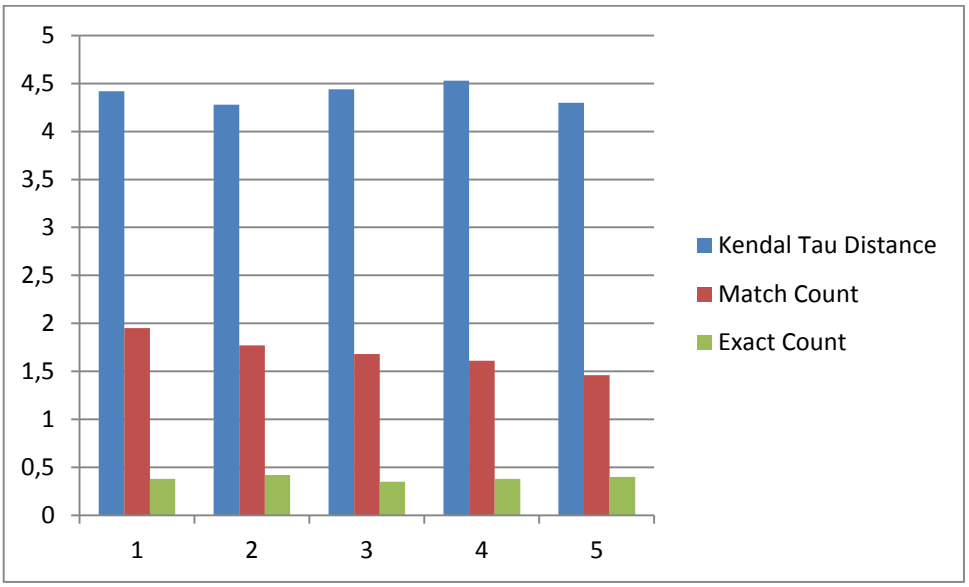

Picture 5.10 Rest of the measures for question „music recommendation“

These pictures show that direct algorithms and their combination with peer-to-peer algorithm have slightly less precise results on last three questions than on the first two questions. The worst performance was found for the question that asks about friends who would make a music recommendation. The question about friends who would take care about user's apartment and/or pet had slightly better results and the question that asked about friends whom with user would go to a trip had best results amongst these three questions. Peer-to-peer algorithms showed less fluctuation on these three questions, having more or less similar results on all of them. Their results varied; Mole and Tidal trust algorithm did best on the last question and Eigen trust did best on fourth question. Eigen

trust algorithm still had worst precision, but its combination with Direct2 algorithm did best when it comes to match count measurement and also had the same performance to the Direct2 algorithm when it comes to the rank difference measurement.



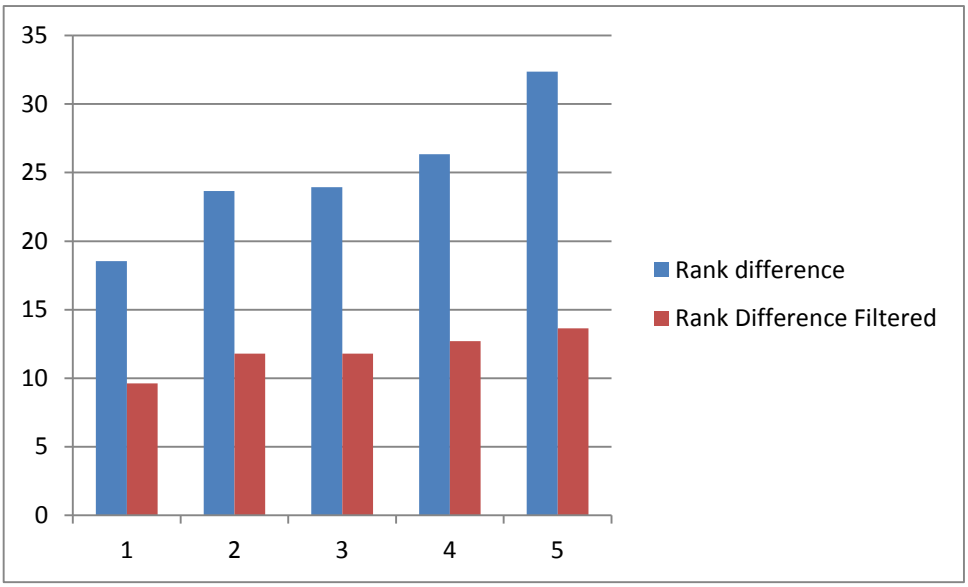

Picture 5.11 Rank difference measure for Direct algorithm for all questions



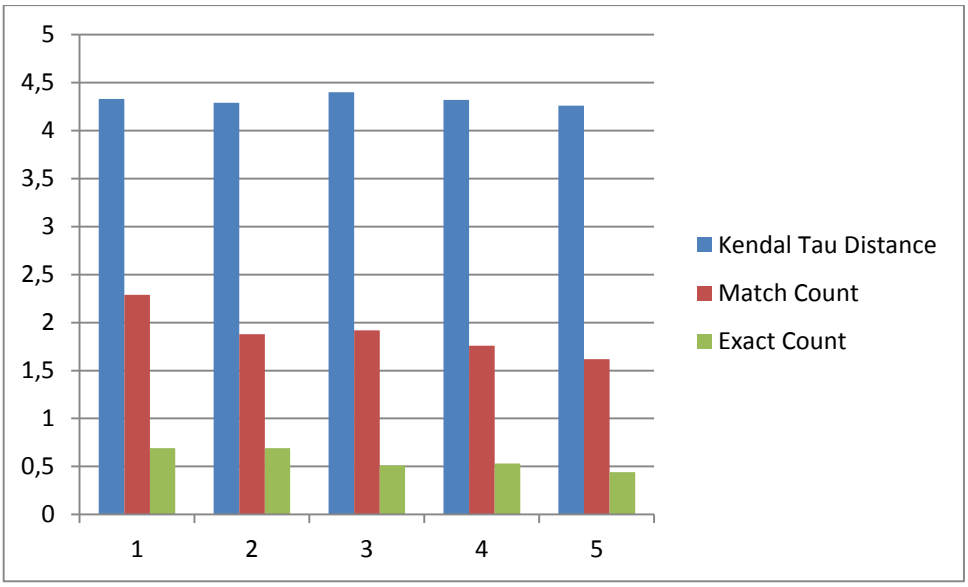

Picture 5.12 Rest of the measures for Direct algorithm for all questions

Picture 5.11 and Picture 5.12 show the results of measures for Direct algorithm for all questions. We can see that direct algorithm gave best results on first question where average rank difference had the lowest value and average count match had the highest value. For context questions from third to fifth, Direct algorithm did best for question three, which asked user to choose friends whom they would like to go on a trip the most with lowest rank difference and highest count match values. Algorithm had worst

performance on fifth question considering friends who would user like to make them a music recommendation. Direct2 algorithm had similar results as Direct algorithm with slightly better precision. It also had best performance on first question and worst on fifth what we can see in



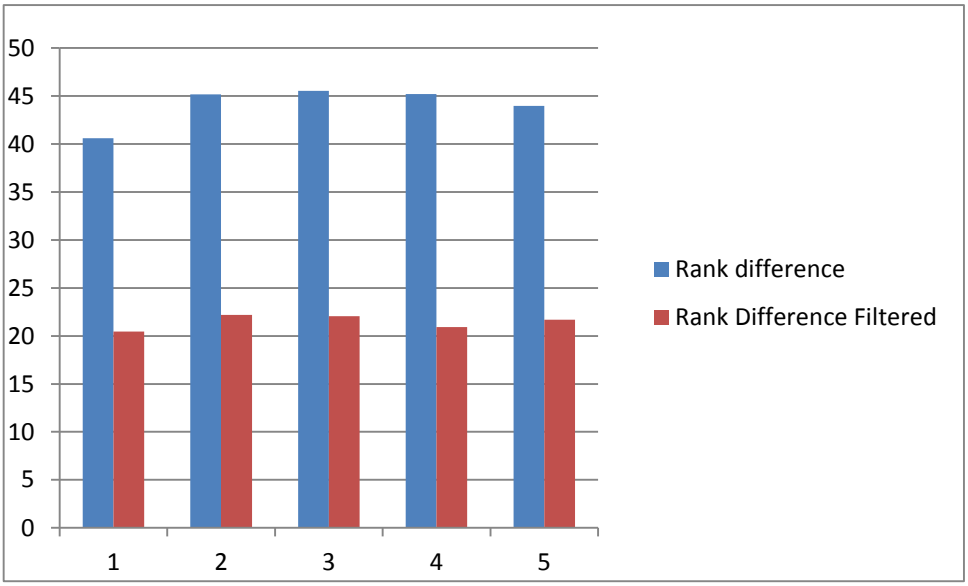

Picture 5.13 Rank difference for Direct2 algorithm for all questions



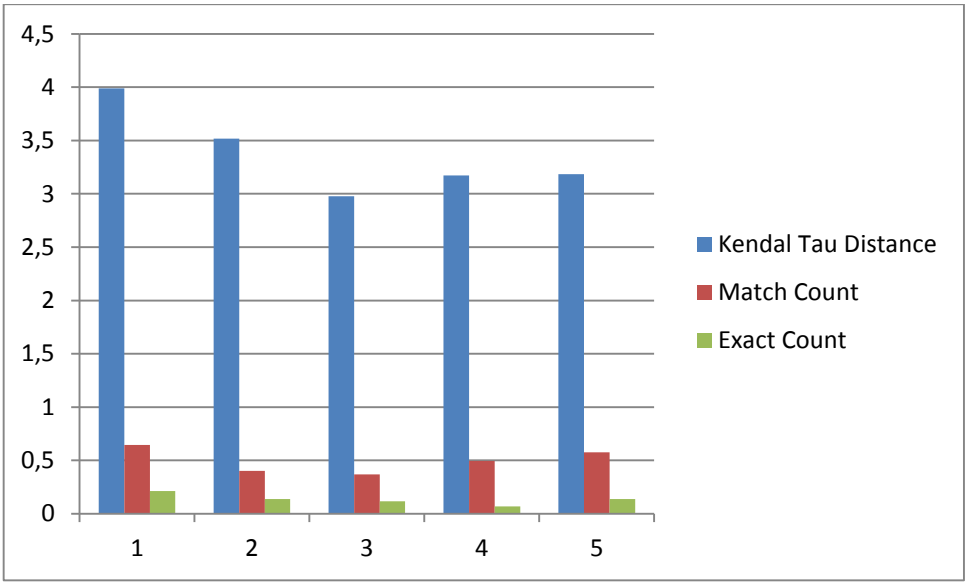

Picture 5.14 Rest of the measures for Direct2 algorithm for all questions

Peer-to-peer algorithm which didn't show as good results as direct algorithms also had interesting results when considering context questions. Mole trust and Tidal trust algorithms were most precise with fifth question with lowest average rank difference and highest count match, where Mole trust showed slightly better results than Tidal trust. The reason for this could be found in the network incompleteness which affects Tidal trust

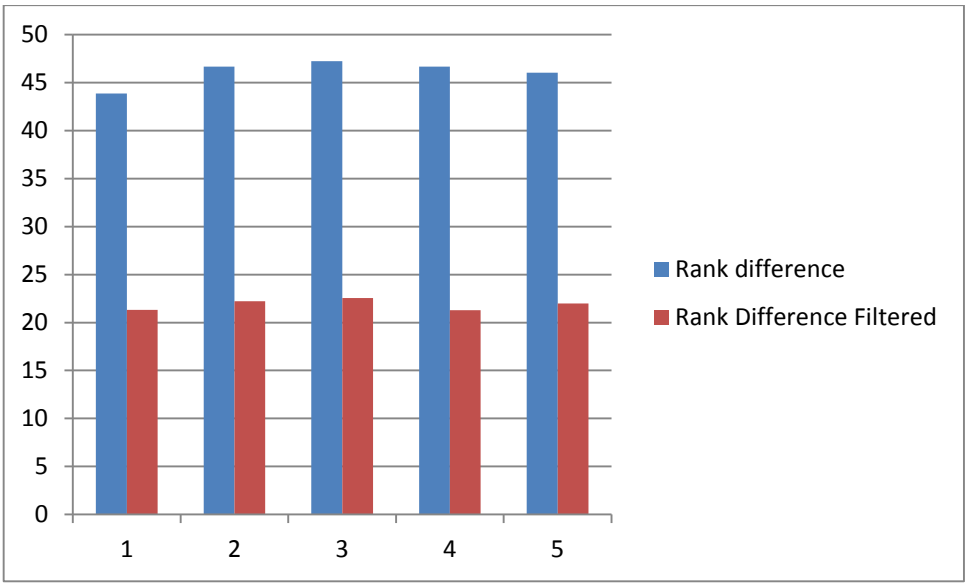
more because it stops at the first sight of sink node. The results for these two algorithms for all questions were given in Picture 5.15, Picture 5.16, Picture 5.17and Picture 5.18. Eigen trust was the least precise algorithm for all questions, but considering the context questions it gave best results for question number three which had lowest average rank difference and lowest Kendal tau distance.



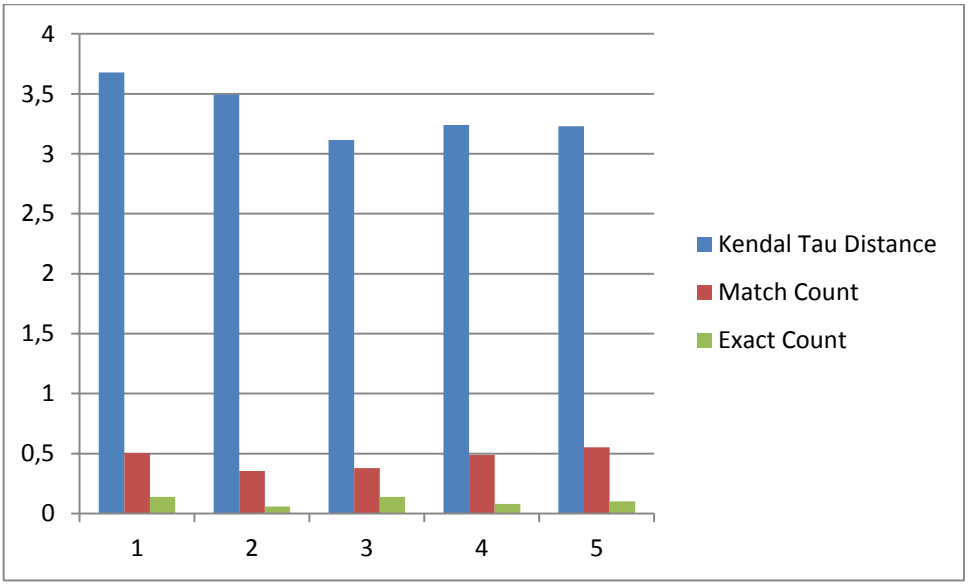

Picture 5.15 Rank difference measure for Mole trust for all questions



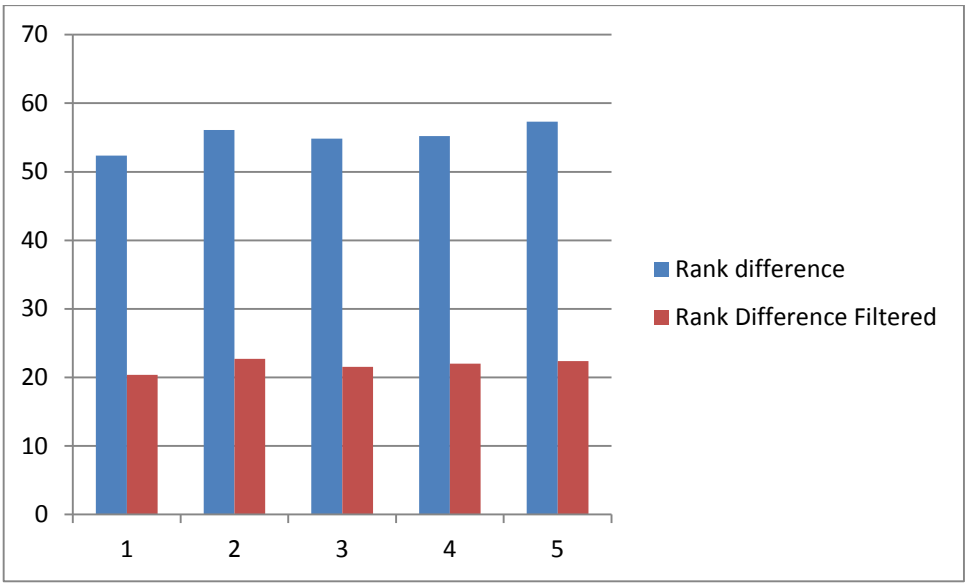

Picture 5.16 Rest of the measures for Mole trust for all questions



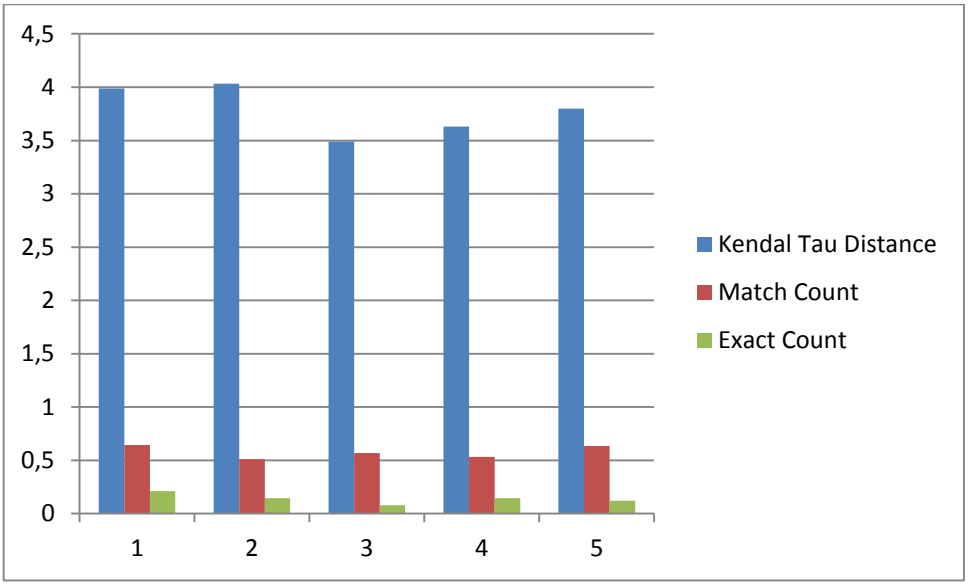

Picture 5.17 Rank difference measure for Tidal trust for all questions




Picture 5.18 Rest of the measure for Tidal trust for all questions




Picture 5.19 Rank difference measure for Eigen trust for all questions




Picture 5.20 Rest of the measures for Eigen trust for all questions

### 5.4.4. Results of the Second Part of the Survey

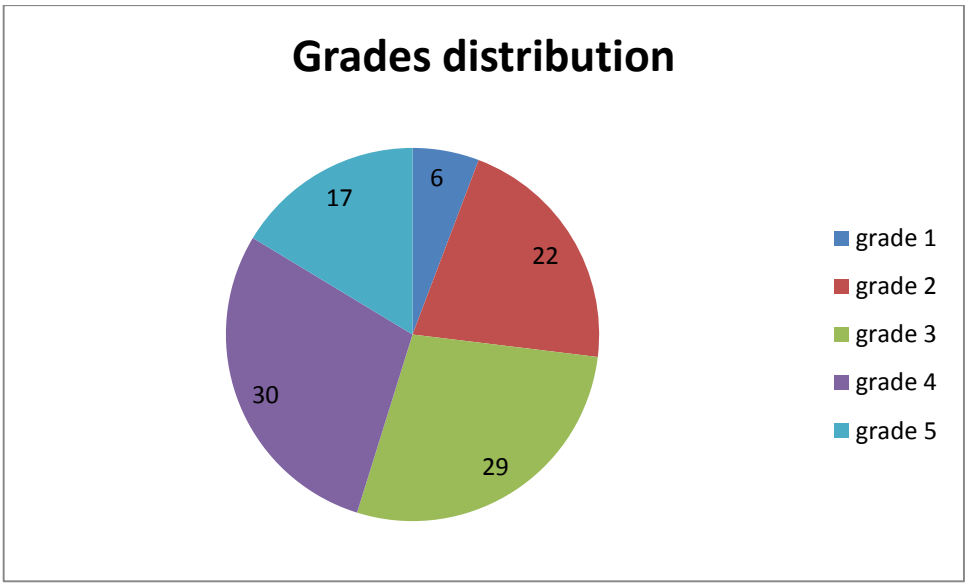
Second part of the survey was consisted of user's ratings of the algorithms. In this part of the survey, we asked users to grade four algorithms from the best grade – 5 to the worst grade 1. 104 users did this and we calculated the average grade for every algorithm. Because there were not enough users in the system to enable peer-to-peer algorithms to compete with direct algorithm, we asked user to grade direct algorithm and three combinations of direct algorithm with peer-to-peer algorithms. There were combinations of

Mole, Eigen and Tidal trust peer-to-peer algorithms with direct trust algorithm. The results of the grading are given in the Table 5.7.

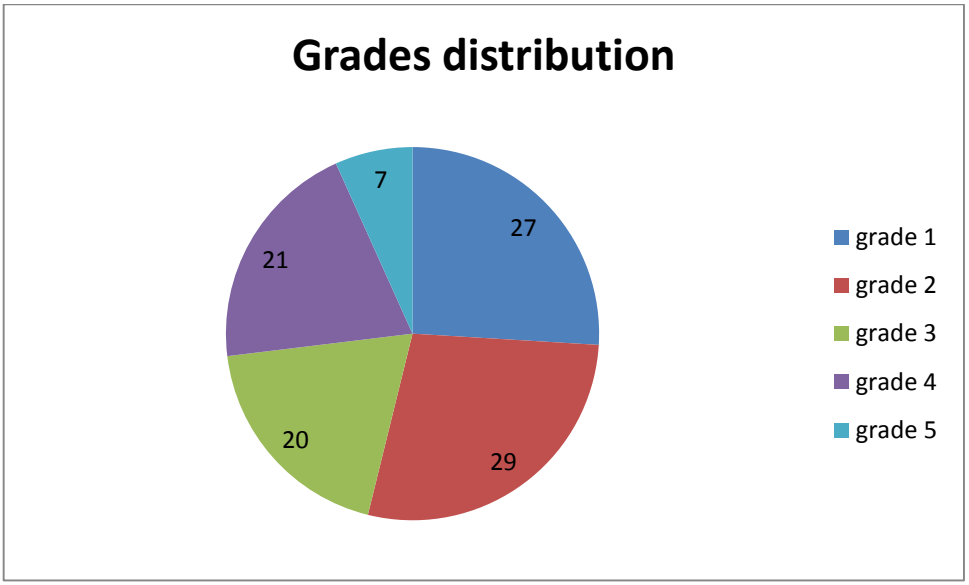
Table 5.7 Average grades of algorithms

| **Algorithm** | **Average grade for sentence *Your best friends from real life*** | **Average grade for sentence *Your best friends from Facebook*** |
|---|---|---|
| Direct trust algorithm | 3.2885 | 3.3269 |
| Combination of Mole trust and direct trust algorithm | 2.5384 | 2.6731 |
| Combination of Tidal trust and direct trust algorithm | 2.6154 | 2.5962 |
| Combination of Eigen trust and direct2 trust algorithm | 3.5096 | 3.3846 |

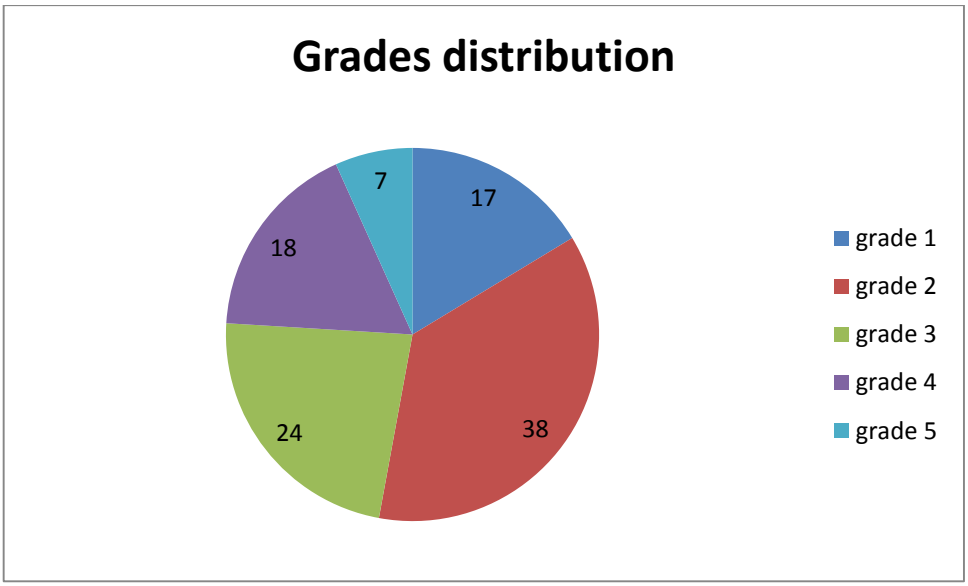
This table shows that the direct algorithm and his combination with the Eigen trust algorithm was best rated by the users, despite the fact that Eigen trust algorithm itself had worst rank difference results. The combination of direct trust and Eigen trust algorithms gave better results than the direct trust algorithm itself, which shows that peer-to-peer algorithms have potential to improve results of direct trust algorithms. Also, Mole trust algorithm and Tidal trust algorithm combination with direct trust, which were less graded than the other two algorithms, showed good results. It is interesting to see than Mole trust combination did better for ranking friends that are user's best friends from real life, and Tidal ranking combination did better in ranking user's friends that are his best friends on Facebook. Next set of pictures shows the distribution of users grades for algorithms and for two sentences.



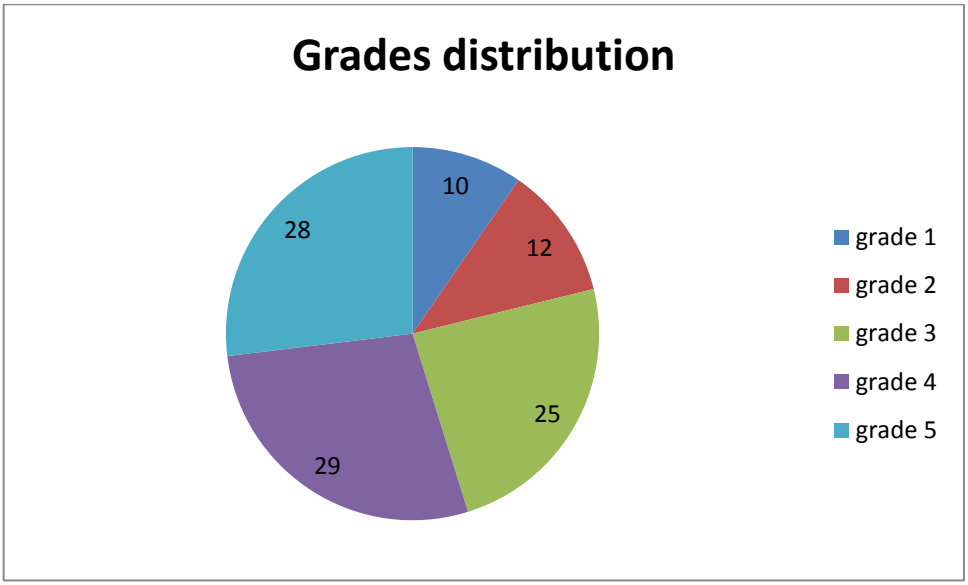

Picture 5.21 Grades for direct algorithm for sentence „best friends from real life“



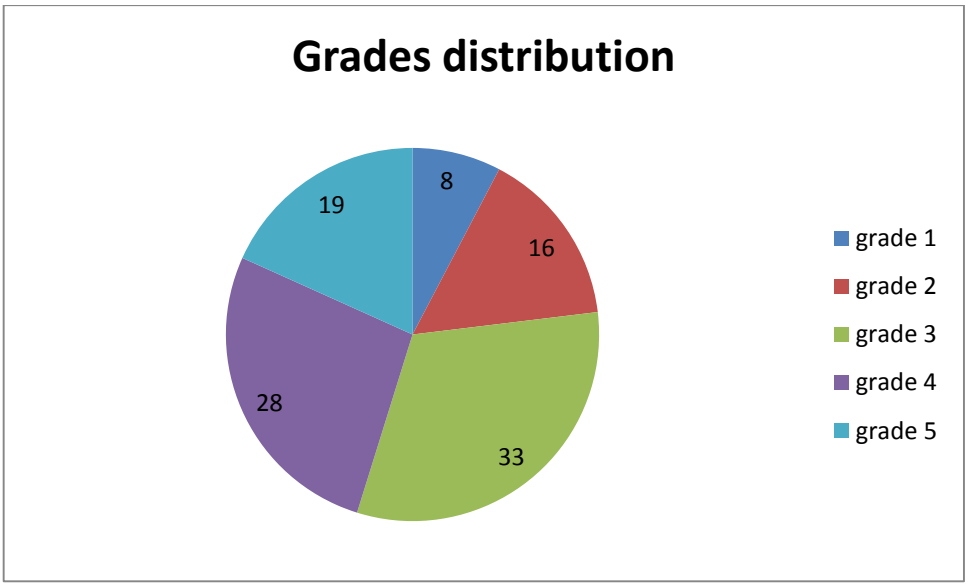

Picture 5.22 Grades for Mole algorithm combination for sentence „best friends from real life“



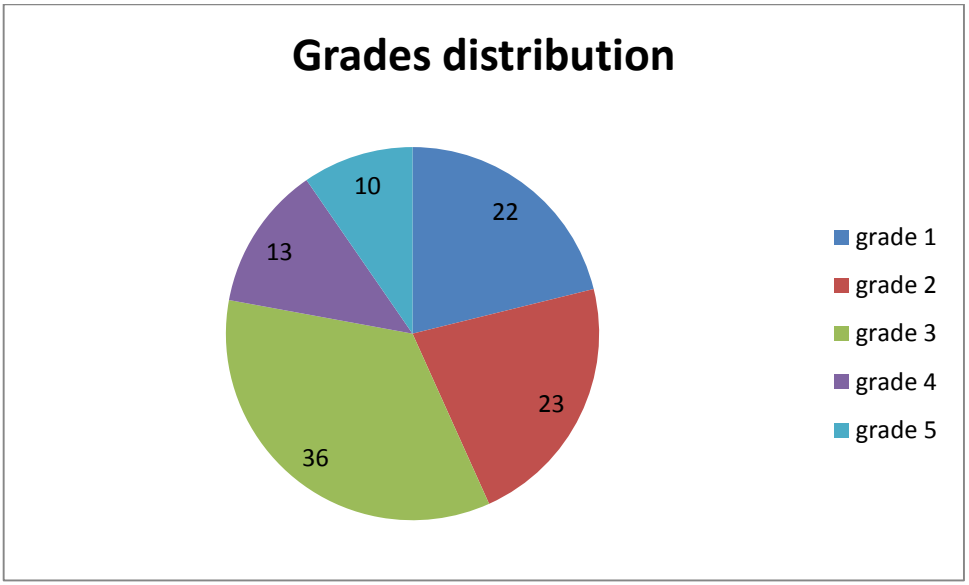

Picture 5.23 Grades for Tidal algorithm combination for sentence „best friends from real life“



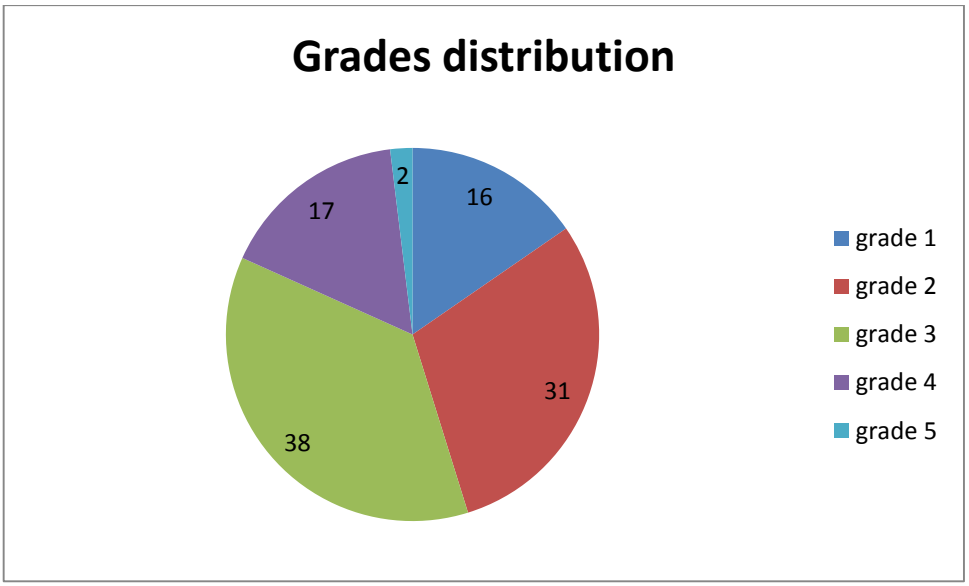

Picture 5.24 Grades for Eigen algorithm combination for sentence „best friends from real life“



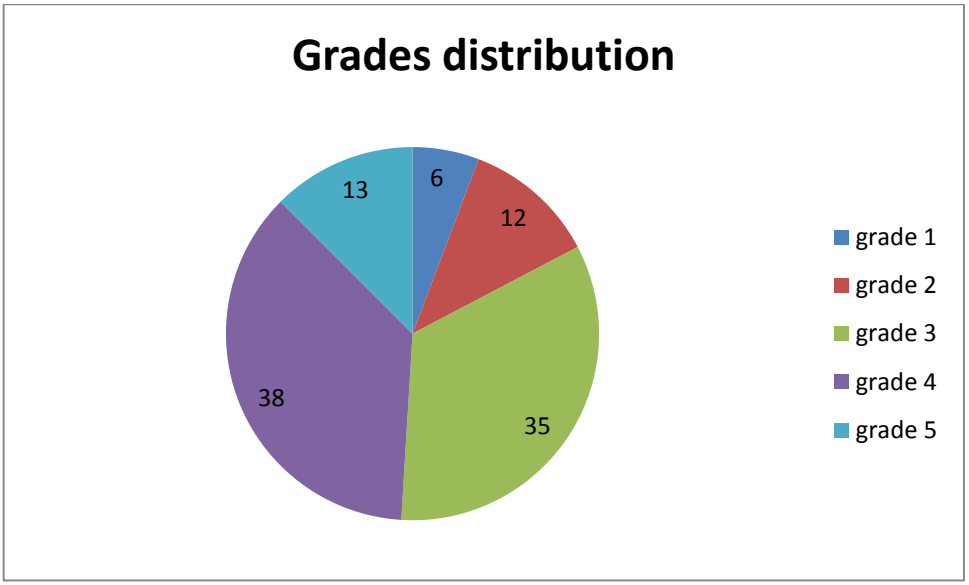

Picture 5.25 Grades for direct algorithm for sentence „best friends from Facebook“




Picture 5.26 Grades for Mole algorithm combination for sentence „best friends from Facebook“




Picture 5.27 Grades for Tidal algorithm combination for sentence „best friends from Facebook“




Picture 5.28 Grades for Eigen algorithm combination for sentence „best friends from Facebook“

This graphs with grades distribution shows that direct algorithm and his combination with Eigen trust is very accurate with most of the users rating the algorithms with grades bigger than 2. Mole trust algorithm combination had the most grades of value 1 on both sentences which shows that it has the highest number of big mistakes with rankings. Tidal trust combination had les bit mistakes, but also was not precise enough and had less scores of 5 than any other algorithm.

Last part of the research and the second part of the second part of the survey was asking users to fill their own interpretation of algorithm results. They were asked to put algorithm results in some context. For direct algorithm 82 users put some description where most of

them put general description of results as friends. 35 users said that the results show their friends from school or college and few said friends from work. 10 users said that the list shows their family and 7 said that it shows friends from neighborhood. 81 users gave their description of second algorithm. 30 of them said that the rankings show their friends from school, college or work and 6 users said that rankings show their friends from neighborhood and 6 said that is showed their family. 79 users gave their description of third algorithm, where 30 of them said that the rankings show their friends from school, college or work, 9 of them said it shows their family and 6 of them said it shows their friends form neighborhood. 78 users fill in the description for fourth algorithm. 27 of them said it shows their friends from school, work or college, 9 said it showed their family and 6 of them said it showed their friends from neighborhood. For fourth algorithm there were the most inputs that said the results show their best friends form everyday life or friends they trust the most where 7 users said that.

# 6. Conclusion

Human relations are one of the most important parts of human life. Sciences as psychology and sociology are studying, among other things, human interaction and how it effect human life. Social networks, such as Facebook provide variety of possibilities of human interactions and relationships analysis. One of the most important parts of human interaction is trust. In this work, we made an analysis of trust on social network Facebook using interactions of Facebook users on Facebook. These interactions include: likes, comments and tags of friends on user photos and posts. Actions such as tags in posts and comments on photos showed to be more important than likes on posts and photos in determining trust among user and his friend. Also, we can conclude that normalization of interactions, when computing direct trust among user and his friends, can lead to better results. This work also gave implementations of three peer-to-peer algorithms: Mole trust algorithm, Tidal trust algorithm and Eigen trust algorithm. Analysis of these algorithms, as well as direct algorithms and their combination was given. This analysis was done with a survey on 100 users. What we could conclude is that algorithms in give meaningful results and can calculate trust among user and his friends on Facebook and in real life. Peer-to-peer algorithms showed less accurate results than direct algorithm, which is a result of network incompleteness. These algorithms would probably be much more accurate with the growth in network of users. Another thing to conclude is that peer-to-peer algorithm show good ordering of top 5 friends of user and that combination of direct trust algorithms and peer-to-peer algorithm can give better results. Such is the example of Eigen trust algorithm and direct trust algorithm. When talking about different contexts of trust amongst Facebook users, it is difficult to give precise conclusions. Algorithms in this work give good trust results about general user trust towards his best friends in real life, but slightly less accurate when we talk about trust in context such us trust towards someone who can recommend some music or someone who you could trust to keep your house safe. This research is good base for future work on this subject where bigger datasets could provide better results and where different contexts of trust, or combinations of trust algorithms could be evaluated.

# Literature


[1] KAMVAR, S.D., SCHOLSSER, M. T. AND GARCIA-MOLINA, H. The eigentrust algorithm for reputation management in p2p networks. *In Proceedings of the 12th International Conference on World Wide Web*. New York, (2003), 640-651.

[2] XIONG, L. AND LIU, L.Supporting reputation-based trust for peer-to-peer electronic communities.*IEEE Trans. Knowl. Data Engin.* (2004), 843-857.

[3] YU, B. SINGH, M, AND SYCARA, K. Developing trust in large-scale peer-to-peer systems. *In Proceedings of the 1stIEEE Symposium on Multi-Agent Security and Survivability*. IEEE Computer Society, 1-10., (2004).

[4] GOLBECK, J. A. Computing and applying trust in web based social networks. Ph.D. thesis, University of Maryland at College Park, MD., 2005.

[5] JOSANG, A., HAYWARD, R. AND POPE, S. Trust network analysis with subjective logic. *In Proceedings of the 29th Australasian Computer Science Conference*. Australian Computer Society, Hobart, Australia, (2006), 85-94.

[6] CAVERLEE, J., LIU, L., AND WEBB, S. Socialtrust: Tamper-resilient trust establishment in online communities. *In Proceedings of the 8thACM/IEEE-CS Joint Conference on Digital Libraries*. ACM Press, New York, (2008), 104-114.

[7] LIU, H., LIM, E.-P., LAUW, H. W., LE, M.-T., SUN, A-, SRIVASTAVA, J. AND KIM, Y. A. Predicting trusts among users of online communities: An epinions case study. *In Proceedings of the 9th ACM Conference on Electronic Commerce*. ACM Press, New York, (2008), 310-319.

[8] NEPAL, S., SHERCHAN, W., AND PARIS, C. STrust: Atrust model for social networks. *In Proceedings of the 10th IEEE International Conference on Trust, Security and Privacy in Computing and Communications*., (2011), 841-846.

[9] SHERCHAN, W., NEPAL, S. AND PARIS, C. 2013. A Survey of trust in social networks. ACM Comput. Surv. 45, 4, Article 47 (August 2013)

[10] RICI, F., ROKACH, L., SHAPIRA, B., KANTOR, P.B. Trust and Recommendations. *In Recommender Systems Handbook (pp. 645-675)*, London , (2011).

[11] SKORIN, J., MARJIĆ, M., HUMSKI, L., SKOČIR, Z., Korištenje interakcije na društvenoj mreži za predviđanje odnosa u stvarnom životu. Zagreb, (2013).

[12] FLITZ, A., Online Social Network Capabilities for a Platform-Independent Cloud Storage System for Multi-Usage. *Master Thesis*, University of Zurich, Switzerland, (2014).

[13] SINGH, S. AND BAWA, S. 2007. Privacy, trust and policy based authorization framework for services in distributed environments. *Int. J. Comput. Sci*. 2,2, 85-92.

[14] KENDALL, M., G. A new measure of rank correlation. *Biometrika*, 30(1/2):81-93, June, 1938.

[15] ROUSSEAU, D. M., SITKIN, S. B., BURT, R. S, AND CAMERRER, C. Not so different after all: A cross-discipline view of trust. *Academy Manag. Rev.* 23, 3 (1998), 393-404.

[16] DEMOUCHEL, P., Trust as an action. *Euro. J. Sociol.* 46, (2005), 417-428.

[17] MUI, L., Computational models of trust and reputation: Agents, evolutionary games, and social networks. Ph.D. thesis. 2003.

[18] TYLER, T. R. AND DEGOEY, P., Trust in organizational authorities: The influence of motive attributions on willingness to accept decisions. In *Trust in Organizations: Frontiers of Theory and Research,* R. M. Kramer and T.R. Tyler, Eds., Sage Publications, Thousand Oaks, CA, 1996.

[19] KUAN, H. AND BOCK, G., The collective reality of trust: An investigation of social relations and networks on trust in multi-channel retailers. In *Proceedings of the 13th European Conference on Information Systems.*, 2005.

[20] LEWIS, J. D. AND WEIGERT, A. Trust as a social reality. *Social Forces 63, 4*, (1985), 967-985.

[21] ROTTER, J. B. A new scale for the measurement of interpersonal trust. *J. Personality 35, 4*, (1967), 651-665.

[22] STAAB, S., HWANG, K., ZHOU, R. AND KWOK, Y.-K. Trusted p2p transactions with fuzzy reputation aggregation. *IEEE Internet Comput. 4*, 6, (2005), 24-34.

[23] ABDUL-RAHMAN, A. AND HAILES, S. Supporting trust in virtual communities. *In Proceedings of the 33rd Hawaii International Conference on System Sciences.* IEEE Computer Society, (2000),1-9.

[24] YU, B., SINGH, M. P. An evidential model of distributed reputation management. *In Proceedings of the 1st International Joint Conference on Autohomous Agents and Mulit-Agent Systems.* ACM Press, New York, (2002), 967-985.

[25] NEPAL, S., ZIC, J., LISU, D., AND JANG, J. Trusted computing platform in your pocket. *In Proceedings of the IEEE/IFIP International Conference on Embedded and Ubiquitous Computing.* IEEE Computer Society, Los Alamitos, (2010), 967-985.

# Summary


### Analysis of algorithms for determining trust among friends on social networks

Trust is the one of the major components of human interaction. This work gives an analysis of trust among friends on social network Facebook as well as the theoretical review of most important components of trust in general. Different algorithms are implemented and evaluated, both direct trust algorithms and peer-to-peer trust algorithms. Direct trust algorithm are the one that compute trust between friends on Facebook based on their interaction, and peer-to-peer trust algorithms compute trust between any two users in the network of Facebook users. Three adaptations of peer-to-peer algorithm are made in this work: Tidal trust algorithm, Mole trust algorithm and Eigen trust algorithm. Results of these algorithms are also combined with the results of direct trust algorithms for purposes of exploring new, more precise algorithm results. All of the algorithms were evaluated using the survey which was filled in by 100 users. Users choose their top 5 friends in 5 different contexts and their answers were used to evaluate algorithms with different measures: Kendal Tau distance, Rank difference, Count match and Exact match. In the end, some conclusions and evaluation of results was given with open possibilities for further work.



**Keywords:** trust, peer-to-peer, Facebook, social network, interaction, propagation, weight calibration, survey, ranking, distance measures

# Sažetak


## Analiza algoritama za određivanje povjerenja među prijateljima na društvenim mrežama

Povjerenje je jedno od glavnih komponenti ljudske interakcije. U ovom predu prezentirana je analiza povjerenja medu prijateljima na društvenoj mreži Facebook, kao i teretski pregled najvažnijih teoretskih komponenti povjerenja. Razliciti algoritmi su implementirani i evaluriani, direktni algoritmi i algoritmi ravnopravnih sudionika. Direktni algoritmi racunaju povjerenje izmedu prijatelja na Facebooku na temelju njihovih interakcija, dok algoritmi ravnopravnih sudionika racunaju povjerenje izmedu bilo kojih korisnika u mreži. Tri adaptacije algoritama ravnopravnih sudionika su ucinjene u ovom radu: Tidal algoritam, Mole algoritham i Eigen algoritham. Rezultati ovih algoritama su kombinirani sa rezultatima direktnog algoritma u svrhu istraživanja mogucnosti poboljšanja rezultata. Svi algoritmi evaluirani su uz pomoc upitnika kojeg je ispunilo 100 korisnika. Korisnici su birali 5 najboljih prijatelja u 5 razlicitih konteksta i njihovi odgovori su korišteni pri evaluaciji algoritama sa cetiri razlicite mjere: Kendal Tau udaljenost, rang udaljenost, broj poklapanja i broj tocnih poklapanja. Na kraju su dani zakljucci i evaluacija rezultat, kao i mogucnosti daljnjeg rada.



**Ključne riječi:** povjerenje, ravnopravni sudionici, Facebook, društvena mreža, interakcija, propagacija, kalibriranje težina, anketa, rangiranje, mjera udaljenosti

# Attachment

### Software Installation

Developed application is a web and Facebook application developed in PHP. Whole project can be copied in the apache server application directory, for example, directory htdocs of XAMPP. Also, maximum execution time of server has to be set to at least five minutes because some users have big amount of data that needs to be collected. Another configuration requires to set the redirect_uri variable in config.php script to the start page of the application on the new host. This uri also has to be set on the Facebook page of the application

### Instructions of How to Use

Application has only one page and that page first provides user with an option to login to the application. User can login with his Facebook account. Before he is logged in, user has to give some Facebook permissions, which the application requires him to do. Once he is logged in, user can start to fill in the survey while his friends rankings are being calculated. When he is finished with the filling of survey questions and when the algorithms have finished with their calculation, user can grade the results of algorithms. In the end, user can submit his answers by clicking on submit button.