Xaao: A Novel Approach To Reify "Community Linked" and Enable Data Self-Governance

Tawfiq Khali, Ching-Seh (Mike) Wu
Computer Science and Engineering
Oakland University
Rochester, Michigan. USA
Tawfiq.Khalil@hp.com, cwu@oakland.edu

ABSTRACT: *In this paper we present a novel and noble approach to revitalize the sense of community that began to diminish due to our modern and fast pace life style. In order for any community to flourish, members have to collaborate and work together. Innovative and cutting edge technologies should be used to benefit and not harm our personal lives, families and the society we live in. Community Linked is a framework that relies on cutting edge technologies and our innovative solution to embrace the community spirit and help achieve its goals. In this paper, we review the different link patterns in the World Wide Web. Then, we provide a partial list of user stories to prescribe the desired functionalities of the system. Next, we present a thorough literature review on the related key areas of the framework and finally present the framework including our innovative XaaO (everything as an object) annotation that we used in Community Linked framework.*

Keywords: Community, Cloud Computing, Graph Database, Linked Data, Everything as an Object, XaaO, DDDIA

Received: 30 May 2014, Revised 5 July 2014, Accepted 15 July 2014

© 2014 DLINE. All Rights Reserved

1. Introduction

The term community in this paper refers to entities sharing the same locality; although it can also encompass entities having common interests but the emphasis is on the former. Entities include people, merchants, organizations, local authorities and others. In a PEW survey on the role of the internet in achieving group goals in the preceding 12 months, 84% of respondents believe the internet played a role in getting a candidate elected to office, 74% believe the internet played a role in solving or changing an issue in society at large and 64% believe the internet played a role in solving or changing a local problem [1]. Another PEW survey shows that 72% of users of different social networking sites use it multiple times a day [2]. Common features of social networks include the ability of users to create a list of "*friends*," *update* "*status*," add comment, "*like*" another user's content, and post messages [2]. Current social networks have evidently been successful in linking people on a global and dispersed level. However, a survey with experts on the future of social relations shows positive and negative feelings about the internet and social networks [3]. For example, "*What I already miss the most is personal contacts in the real world. Too much communications through the internet can damage real quality of life. I fear this will be worse in* 2020." (Bernhard Adriaensens). "*The internet's effect on relationships is paradoxical. It strengthens our relationships with distant friends and relations through social networks and email, but may damage the relationships of those nearer to us as always-on technologies and applications eat into family and social time*." (Mary Joyce). The result of the survey indicates that the benefits of the internet and social networks will outweigh the negatives over the next decade.

Community Linked fills the gaps found in the current social networks and capitalize on their success. It is not controlled by a single entity serving its objectives. It is an open platform accessible and maintained by all community members based on their subscription and access level. The core objective of Community Linked is to unite the community (residents, merchants, health providers, local authorities, etc.). It exploits cutting edge technologies such as cloud computing, NoSQL and our state-of-the-art solution XaaO (everything as an object) to represent and universalize data that represents entities.

Figure 1. Community Linked Objective

This paper is organized into six parts. Section II reviews the link patterns in the World Wide Web as a background study to show how these patterns motivated our framweork. Section III presents some of the user stories of the framework and the limitless opportunities that it can provide for the well-being of a community. Section IV provides a literature review on the key areas related to our framework. Section V presents the framework which includes the suggested deployment model, security considerations, data management, XaaO and DDDIA (i.e. Data Description, Discovery, Integration and Aggregation). Finally, we conclude the paper with experimental results (i.e. section VI) and suggested future work (i.e. section VII).

2. Background Study

The World Wide Web (WWW) has enjoyed phenomenal growth and has received wide global adoption without regard to factors such as age, ethnicity and location of its users. In our previous study, we identified the following link patterns observed in the World Wide Web [4]:

Pattern 1: linking documents and anchors within the document explicitly via hyperlinks

Pattern 2: linking people to documents and services implicitly via search engines

Pattern 3: linking applications to web services, and web services to other web services via mashup, choreography and orchestration

Pattern 4: linking people via proprietary online social network services

Pattern 5:linking objects: linking data, that is represented as objects, among applications via URL and object's unique id

Pattern 6: linking data via semantic web, microformats and microdata techniques

The understanding of these link patterns provides a foundation for our framework, in terms of how data is published and linked, and motivates our effort to create a new, scalable and reusable solution for communities to collaborate and become closer than ever before. Current social networks like Myspace, LinkedIn and Facebook have been successful in connecting people with

common interests (music, profession, friendship, etc.) regardless of proximity. Connected people in these social network services may never talk directly to each other and get to know each other except for being connected and reading others statuses and comments.

Many other solutions have been implemented to embrace the sense of community and encourage collaboration among community members such as neighborhood watch and nixle [5], [6]. These solutions are focused on building a safer community through open and direct communication between local authority and citizens which is a great step in building a stronger and safer community. However, these solutions are limited in scope again and do not unlock the full potential of having full collaboration between community citizens, businesses, health providers, schools and local authority.

Community Linked framework is presented in this paper to strengthen and integrate communities, as well as, fulfill the dream of many parents as Bill Clinton articulated it "When I think about the world I would like to leave to my daughter and the grandchildren I hope to have, it is a world that moves away from unequal, unstable, unsustainable interdependence to integrated communities - locally, nationally and globally - that share the characteristics of all successful communities.

3. User Stories

Web 2.0 and smart phones made it possible for people to subscribe to online services to stay informed about things they are interested in. As an example, local news channel can send messages about latest news. Weather service can send warning messages about severe weather conditions. nixle sends messages about public safety issues from local police. Many other disconnected services are available to perform a specific task. Community Linked is a framework to provide a collaboration platform to unify all these efforts to build a strong community. Some of the user stories of this framework include:

• As a community member, I want to have one place I can subscribe to so I can learn about my community

• As residents, we want to have a common place where we can share ideas so we can help making our community better

• As a resident, I want to know about any outbreak diseases in my community and preventative measures so I can stay healthy

• As a resident, I want to know about any school closure so I can avoid going to school while it is closed

• As a resident, I want to know when criminals flee from the police so I can stay vigilant and safe

• As a resident, I want to be informed about occurring events in my community so I can participate when interested

• As a health department, we want to identify restaurants having bad reviews from customers who experienced health issues after eating at their locations so we can investigate and take appropriate actions

• As a health department, we want to have data about the community to help with investigating disease outbreaks „h As a police department, we want to communicate to residents when searching for criminals to seek help and ask them to stay vigilant

• As a weather department, we want to communicate any weather warnings or watches to residents to keep them prepared

• As a business owner, I want to reach out to the community so I can promote my business and better understand my clients

• As a government, we want to have open and direct communication with the community to educate them on issues and proposals related to their community and get their feedback without any mediators

• As a religious entity, we want to stay in touch with community issues and concerns to offer spiritual guidance and support

• As a research institution, we want to have access to data related to all the different aspects of the community tohelp us identify patterns that could provide guidance to current and future research

4. Literature Review

Semantic web and Graph mining has become an important topic of research because of the significant growth of data and the need to enable machines understand data, relationships, thus be able to extract knowledge and infer new relationships. There is a numerous amount of literature based on several hundred years of mathematical and scientific study introducing new and enhanced techniques for graph classification, clustering, matching and frequent pattern mining. On the other hand, application domains are diverse and it is a critical step during the application design to choose and customize the appropriate algorithm for the application.

Community detection in social networks is an application of graph clustering. It attempts to identify groups where nodes (entities) within the same group are similar whether based on the edge behavior (node clustering) or the overall network structure behavior (graph clustering). Fortunato, Santo presented a thorough analysis on the different algorithms that can be applied for community detection, and concluded that this problem is considered very hard and has not yet been satisfactorily solved, despite all the available literature and algorithms on this topic [15]. New frameworks and applications must consider this problem in an early stage and craft a design that embraces the concept of community due to its importance and complexity specifically for applications that are based on users' interactions.

Graph classification aims to predict class label (category) for the whole graph and the nodes within the graph (label propagation). It is different from clustering since it does not aim to partition the graph into sub-graphs. Classification on the graph level can be unsupervised (based on similarity) or supervised (based on training set) [16-18]. Label Propagation exploit sparsity of the adjacency matrix [19], [20]. Label propagation is more efficient than its predecessor methods such as diffusion kernels. Diffusion Kernels determines similarity between two nodes by calculating the commute time of random walks between them [21]. It is not efficient in large network since it takes O (n3) time to compute. Zhou et al. [20] introduced label propagation algorithm with a key idea that every node iteratively propagate its label (class) to its neighbors until a global stable state is reached. It is computed by simultaneously solving linear equations with a sparse coefficient matrix. The time needed for the computation is nearly linear to the number of non-zero entries of the coefficient matrix [22]. It is important to denote that this solution is a semi-supervised or transductive since it relies on the labeled nodes and the structure of the unlabeled nodes to determine the classification of the unlabeled nodes.

Wang et al. [23] presented a review of algorithms for keyword search on graph data. There are three challenges presented when performing keyword search on graph. First, identify the sub-graphs that match the keywords. Second, rank results that best answer the query. Finally, ensure query efficiency. The effectiveness of graph search algorithms including the bidirectional algorithm [24], BANKS [25], BLINKS [26] and ObjectRank [27], [28] is highly dependent on the structure of the graph and the distribution of keywords in the graph. BLINK avoids blind exploration on the graph by creating an index structure that keeps track of the keyword reachable information. ObjectRank, extension of PageRank, uses labels to classify nodes and edges. Keywords are included in the node which give the node authority based on the keywords and that authority flows to other nodes according to their semantic connections. It is important to note that the node is not normalized and it contains all the attributes along with the keywords altogether in one place.

Link recommendation is a prominent research area in social networks. Link recommendation enriches the user experience by providing the user with a list of possible interesting connections instead of manually searching for them and possibly missing out desired connections. There are common approaches to implement this feature. One approach relies on classification methods that performs feature extraction on the nodes and edges to create training sets, positive and negative, to be fed to the model to be trained, in order to predict where the nodes will make a link [30].

Another dominant approach is based on ranking algorithms that assign higher ranking score to nodes that have a link to the node of interest (n) and a lower ranking score for the nodes that are not linked to (n). PageRank[32] and random walk with restart [33], [34] are popular algorithms for ranking nodes on the graph. PageRank is based on a key idea that a link from one node to another is considered as a vote or endorsement. A highly linked nodes has higher importance than nodes with fewer links. In-links from high ranked nodes count more than links from lower ranked nodes. Many proposed frameworks for link recommendation were also based on using random walk algorithm [35], [36]. Yin et al. [35] introduced a framework that augments the original graph by adding a node (Va) for each attribute in the original node (Vo) and a new edge from (Vo) to (Va). Next, the framework runs random walk with restart algorithm on the augmented graph using the attributes (Va) and graph structure. Backstrom et al. [36] introduced a method to make the random walk biased by using node and edge information to compute edge strength; thus, the random walk is more likely to visit "*positive*" than "*negative*" nodes. Preferential attachment is also another popular method for predicting links in the graph [37]. It has three common measures: degree centrality, betweeness centrality and closeness centrality. Degree centrality is the number of incoming links to the node which measures local centrality and reveals how influential and informal leader the node is in the graph. Betweeness Centrality measures how frequent a node occurs in between other nodes which could signify that the node is a gatekeeper or broker. Closeness Centrality is a global measure which consider a node that is close (i.e. shortest path) to many other nodes in the graph is globally central.

Our framework builds on the observations made in this section. It has a hierarchical architecture that organizes community members under communities to simplify the process of further community detection (Figure 5). It also introduces XaaO concept

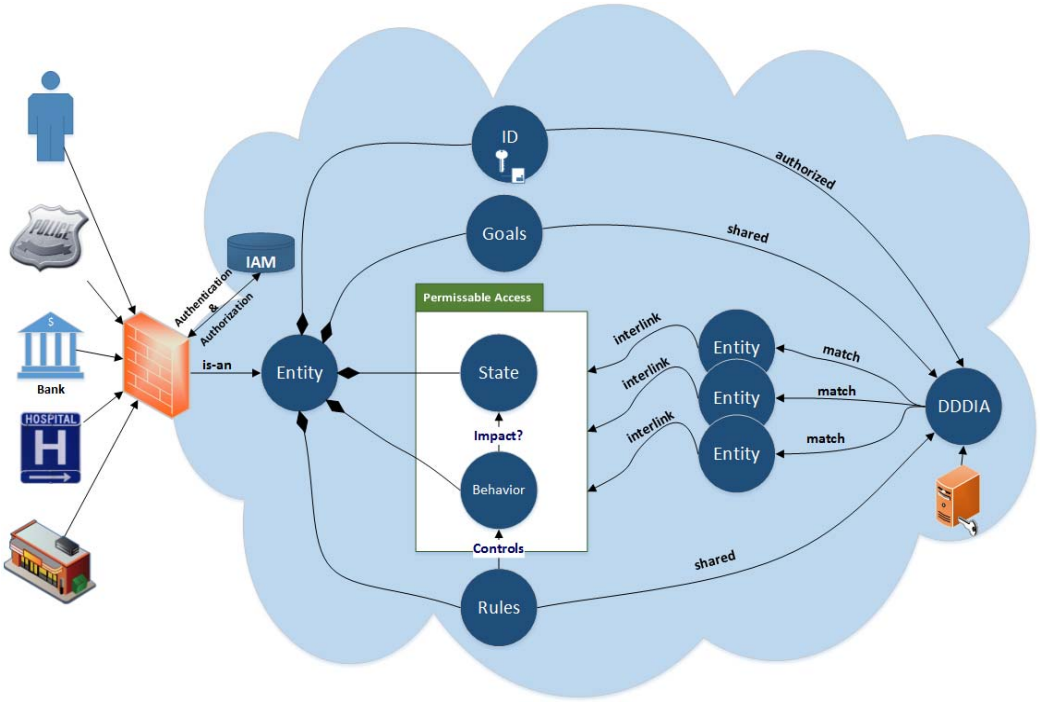


Figure 2. Community Linked High-level Architecture

that enables data to be self-governed, has one or more types (i.e. labels) to alleviate the need for classification, and has goals and tags to improve link recommendation.

5. Community Linked Framework

Community Linked framework relies on our innovative XaaO annotation to represent data as entities that have characteristics such as state, behavior, goals and rules. XaaO is inspired by object oriented programming principles and can be used as a standard for data communication using semantic web ontology and storage in graph databases.

5.1 Deployment Model

Community Linked framework requires an environment that allows access for multiple groups to collaborate and work toward a common goal. Initially, computing resources do not exist and it is hard to predict the need (without over/under provisioning) to determine upfront cost and distribute the cost among community sponsors. In addition, the environment must be flexible and able to automatically respond to increasing demands by providing additional computing resources.

Cloud Computing model has many features available that are in line with the framework needs and automatically meet its goals without the overhead of implementing these requirements [7], [8]:

• **Ubiquitous access:** Access through a widely available network supporting heterogeneous client devices which requires transformation of transport and messaging protocols via cloud broker service

• **On demand:** Depending on the specified requirements and thresholds, computing resources are automatically provisioned without human intervention

• **Multitenancy:** Ability to assign and reassign (share) computing resources (virtual, physical) based on the consumer requirements. This means that a computing resource can be shared simultaneously among multiple consumers

• **Elasticity:** Ability of scaling in/out computing resources depending on changing demands

• **Measured usage:** Ability to track, record and report the usage of computing resources for SLA reporting and billing purposes

• **Resiliency:** Providing a failover system that distributes redundant implementations of computing resources on different physical locations within the same cloud or across multiple clouds to improve availability and reliability

There are three primary cloud delivery models to choose from for delivering Community Linked in the cloud:

• **Software as a Services (SaaS):** Application is available in the Cloud environment and cloud consumer has no administrative access to the underlying infrastructure with the possible exception of configuring application settings

• **Platform as a Service (PaaS):** Providing a ready-made-environment that contains pre-packaged products and tools to build and support the application. Consumer has limited administrative access to deploy and configure the application

• **Infrastructure as a Service (IaaS):** Providing high level of control over self-contained infrastructure centric computing resources (virtual and not physical resources).

These delivery models can have variations such as Data as a Service (DaaS) and Hardware as a Service (HaaS). It is best for Community Linked to identify a PaaS cloud provider that supports its needs in regards to development tools (.Net, Java, SQL Server, neo4j, Hadoop, etc.) and alleviate the burden of managing infrastructure computing resources.

Finally, cloud providers offer 4 different deployment models:

• **Private:** owned by a single organization to centralize access to its computing resources by different departments in the organization. Administration of the environment is managed by internal or third party staff and it can exist on premise or off premise.

• **Public:** publically accessible and owned by a third party cloud provider

• **Community:** limited access to community members who share responsibility for defining and evolving the environment. Community members may jointly own the cloud or it could be owned by third party cloud provider. In addition, it may exist on premise or off premise.

• **Hybrid:** a combination of two or all of the above deployment models

An essential requirement of Community Linked is to allow access to multiple groups to collaborate and work toward a common goal. It is apparent that Community deployment cloud model is a natural fit for the framework. Community businesses, authority, health providers and others can share the cost of the environment by utilizing the "*measured usage*" service available in the cloud or by a flat rate. Community members will also need to define the roadmap for the environment and its capabilities depending on the common short and long term goals.

5.2 Authentication & Authorization

Information security in Community Linked framework is an essential requirement that must be met to protect confidentiality, integrity and availability of members' information. The process of verifying the identity of the user and determining what access (s) he has must accommodate the three different types of Community Linked users who requires different level of authentication:

1. **Residents:** Basic TLS and user ID & password

2. **Government agencies:** Dual TLS and user ID & password

3. **Businesses, organizations and institutions:** Either basic or dual TLS, and user ID & password.

Transport Layer Security (TLS) is a connection oriented protocol that provides a secure channel between the client and the server. TLS and SSL are most widely recognized as the protocols that provide secure HTTP (HTTPS) for Internet transactions and it can be used in the cloud environment. The authentication process is described as follows [9]:

1. The client and server exchange initial Hello messages

2. The protocol requests the certificate from the server and verifies it once received

3. In case of dual TLS authentication, The server requests the certificate from the client and verfies it once received

4. The client sends the predefined secret encrypted using the server's public key which is the client key exchange.

5. The client and the server complete mutual handshake and the initial encryption parameters.

In dual TLS authentication, the client is also being asked to prove his identity by supplying a certificate. This is an additional security that will be required for critical community members like government agencies. Client certificate can be assigned on the

agency top level or on the department level. It is cumbersome to standardize dual TLS authentication for all community members since it requires purchasing a certificate from Certificate Authority (CA) and additional setup when making the connection. Community members can also request a higher standard of authentication based on their security standards.

In addition to TLS authentication, a community member is required to supply a valid user ID and password to complete the authentication process and initiate the authorization process based on the user's specified access roles in identity and access management (IAM) system also known as identity store. IAM establishes a central cloud security mechanism that performs authentication and authorization based on its identity store along with roles and access control rules for its stored identities.

5.3 Everything as an Object (XaaO)

XaaO (pronounced as zhao) is a novel concept that we introduce in this paper and is adopted by Community Linked framework. It is designed to represent everything (including community members and data) as an object. Each object has the following characteristics:

5.3.1. Identification: This is a structure that consists of a unique ID to uniquely identify the entity (automatically generated) and contains a reference to its key (i.e. public key) in the key store. It can also contain other optional information depending on the community needs. Identification is only accessible to DDDIA and has the following representations based on the community member type:

• **(Sub)Community:** ID that uniquely identifies the (sub) community among other communities within Community Linked. Each (sub) community has also a unique key in the key store.

• **Member:** ID that uniquely identifies the member in the community and across communities within Community Linked framework after being authenticated and authorized through IAM. Member has a key in the key store only if the member is using dual TLS authentication otherwise it will be null.

• **Data:** ID that uniquely identify the messages between community members and it automatically inherits the key from its originator if available.

5.3.2 Goals: Used to describe the purpose of the object and help to quickly interlink it to other entities. It is only accessible to the entity owner and DDDIA. There are two formats for a goal:

• **Tags:** keywords separated by commas.

• **User story format:** As an (entity) I want to (goal) so I can (reason).

Augmented Backus–Naur Form (ABNF) is widely used as a definition language for the Internet Engineering Task Force (IETF) communication protocols therefore we use it in this paper to formalize the syntax for goals format [10], [11].

5.3.3. State: contains object's properties (name, size, location, tags, etc.). It is important to denote the difference between tags in XaaO's state and the tags in XaaO's goals. Tags in state are originally defined by the object's owner and if object's rules and access level (described in V.C-7) permit, other entities can view the tags and add to them. It has an attribute that contains the ID of the tag creator. The purpose of tags in state is to make the entity discoverable and enable other entities to find it and link to it.

5.3.4. Behavior: contains the interactions and relationships with other objects. Interactions can be classified depending on the community domain. For example, post a message, reply to message, comment on a message, comment on a comment, "*like*" a message or a comment, and follow an entity whether it is a community, member or a message. Access to entity's behavior is limited based on object's rules and access level.

5.3.5. Rules: this is a vital characteristic of the object to enact self-governance along with access level control. It has conditions on who the object can interlink with (i.e. entity type) and the type of relationships. Rules are only accessible to the entity's owner and DDDIA. In addition, rules are encrypted for additional security with the entity's key if available. Rules can follow predefined syntax that can be continually evolving and socialized among Community Linked members once DDDIA can interpret it. Figure 4 presents an initial ABNF grammar that represents rules. Rules can be defined to indicate the type of entities that the object can auto interlink with (i.e. managed by DDDIA) and what actions other entities can perform on it based on their trust

level.

• E.g. allow auto_interlinked with school low - "*Low*" trust level will allow entities with higher trust levels

```
goals-rule = tags-rule / userStory-rule
;— — —Tags Rule— — — — — — —— — — — — — —
tags-rule = tag * (delimiter SP tag)
tag = *VCHAR
delimiter = ","
;— — —User story Rule— — — — — — — — — — —
userStory-rule = introTxt entity wantTxt goal reasonTXT reason
introTxt = ("As a"/"As an") SP;
entity = 1*VCHAR SP ; Entity can be restricted to allow only
                    ; the available entities such as
                    ; community, member, message
wantTxt = ("I" / "we") SP "want to" SP;
goal = 1*VCHAR SP ; goal can also be restricted to the
                    ; available interactions in the system
reasonTXT = "so that" SP;
reason = 1 * VCHAR ;
— — — — — — — — — — — — — — — — — — — — — — — — — — — — —
VCHAR = %x21-7E
SP = %x20
```

Figure 3. ABNF grammar for goal rules

```
rule = rule-with-others/rule-on-self
rule-with-others = action relationshipA prepositionA entity trustlevel
rule-on-self = action relationshipB prepositionB self trustlevel
action = ("allow"/ "prevent") SP;
relationshipA = "auto_interlinked" SP;
relationshipB = ("comment"/ "reply"/ "like") SP;
prepositionA = "with" SP;
prepositionB = "on" SP;
entity = ((1 * VCHAR * (delimiter entity))/ "all") SP;
; entity: either specify the entity or apply the rule to all entities
self = "self"SP;
trustlevel = ("low"/ "medium"/ "high")
delimiter = ","
;— — — — — — — — — — — — — — — — — — — — — — — — — — — —
VCHAR = %x21-7E
SP = %x20
```

Figure 4. ABNF grammar for entity rules

• E.g. prevent auto_interlinked with person medium - "*Medium*" trust level will prevent entities with lower trust level

• E.g. allow auto_interlinked with person high

• E.g. allow comment on this high

- This rule allows only highly trusted entities to comment on this. "*this*" refers to the entity itself which could be a community, a member or a message

5.3.6 Type: each object has a type (i.e. instance of a class). The type acts as a template for the object and defines its structure. Web Ontology Language (OWL) and class definition in any object oriented programming language can be used to implement that when the object is published or internally processed (by DDDIA) respectively.

5.3.7 Access Level: Entity and its state and behavior can have the following access levels:

• **Public:** available to everyone under Community Linked. This includes other (sub) communities.

• **Internal:** available to everyone within the same (sub) community.

• **Protected:** available only to the entities that it is linked to explicitly (does not include the auto interlinked entities by DDDIA)

• **Private:** available only to the entity and DDDIA. In case of behavior, it also includes the other entity that it is interacting with.

• **DDDIA:** available only to DDDIA.

Public, internal and protected access levels can be prefixed by a keyword "*trusted*" to restrict the access further to only trusted entities that use dual authentication or became trusted based on centrality ranking. It can also be suffixed with a keyword "*+W*" to allow write access as in the case of tags inside the entity's state. The default prohibits write access unless specified explicitly. Rules will also add additional restrictions as long as they do not conflict with access level.

XaaO	Access Level	Defaults	Misc.
XaaO Entity	All ("*trusted*" keyword is optional)	Public with no write access	Has a type
Identification	DDDIA	• ID is auto assigned • key for data entities is inherited from its originator	Key, if available, is used to decrypt rules
Goals	Private	Private	Used to interlink entity with others
State	All ("*trusted*" keyword is optional)	Public with no write access	Tags allow entity to be discoverable
Behavior	All ("*trusted*" keyword is optional)	Public with no write access	Has a type to classify interactions
Rules	Private	Private	Encrypted when the originator has a key
Access Level	A mechanism to specify the level of access that other entities have on the data; whether on the entity or the attribute level	• Default is "–W" • "+W" allows write access and this would be useful for tags	• DDDIA • Private • Public [trusted | +W] • Internal [trusted |+W] • Protected [trusted |+W]

Table 1. Xaao High-level Summary

5.4 Data Structure

Community Linked can consist of one or more communities. Each community consists of one or more members and zero or more sub-communites. Community and members are described using XaaO model. The structure does not allow for orphan community (i.e. not part of Community Linked or a parent community) or member (i.e. not part of a (sub)community). Members in the

community can belong to one or more communities and sub-communities; however, they have different XaaO representation but they should be marked as the same entity either by using "*SameAs*" relationship or having a reference ID as an attribute in XaaO's state to the other entity. Figure 5 presents the hierarchical structure of Community Linked. It includes an example of Member2 that belongs to Community2 and Communityn. In addition, Communityn includes on the same level members and a sub-community (i.e. Communityn.1). Members cannot include other members.

5.4.1 Data Management

Traditionally, many applications use relational databases to manage information. Relational databases rely on tables to manage similar data into rows and columns. Rows are uniquely identified inside the table via a primary key. It is a good practice to normalize tables to reduce data redundancy and increase data integrity. Foreign keys are used to link tables (entities) to each other. Normally, tables are joined in the query when selecting data that resides in multiple tables by utilizing the foreign key relationship. For performance reasons, database architects typically denormalize tables to reduce joins and improve performance, which subsequently causes data redundancy and may negatively affect data integrity. Relational databases have been for decades successful in meeting the complex demands of large scale applications and will continue to be used to manage data for small, mid and large size applications. However, applications that are based on interactions and relationships (i.e. network model) are better implemented in a database management system that embraces relationships and has similar topology, and that is graph databases. Graph databases naturally support network topologies such as social networks. In a graph database model, real instances are depicted and it does not contain a general entity relationship model (schema) as in relational database models. This makes it flexible by allowing new nodes and relationships to be added without the need for migration or downtime to update the schema as it is the case in relational databases. This flexibility adds an overhead on the application side to ensure data integrity.

The Community Linked graph model consists of three building blocks:

• **Nodes:** represent entities that have a type such as community, member and message. Member can also have subtype such as person, school, hospital, police, library, restaurant and grocery.

• **Relationships**: unidirectional edges connecting entities (index free adjacency) that has a type to classify relationships under categories. For example, "*follows*" type includes "*follows*" and "*blocks*". "*likes*" type includes "*likes*" and "*dislikes*". "*message*" type includes subtypes such as "*comment*", "*alert*", "*issue*", "*vote*", "*note*". The "*note*" Subtype can have "*post*", "*forward*", "*delete*" and "*read*".

• **Properties:** represented in the form of a key-value pair and is available for both nodes and relationships. The value in the property can be an array of primitives. Adding properties to the relationship provides a great value by representing metadata (such as the strength of the connection) between entities. For performance considerations, it is important to note that XaaO state tags, rules and identification were normalized and presented in separate nodes having a relationship to link them to the entity. In addition, XaaO's state attributes that have different access level can be presented in two ways. First, the attribute is stored in a separate node and an edge with access level is pointing to it from the entity. Or, the attribute has an array such that the first element is the attribute value and the second is the access level. However, the latter approach is not recommended due to performance reasons.

Figure 6 clarifies the relationship between Community Linked, graph model and XaaO. The process for building a graph model that represents Community Linked starts with identifying its user stories (e.g. III). For illustration purposes, we will present the first user story "*As a community member, I want to have one place I can subscribe to so I can learn about my community*". We can identify in this user story two entity types and one relationship. Member and Community are two entity types. Follow can be a type of a relationship. Appendix A and B presents the graph model and the script used to generate it following the guidelines discussed in this paper [12].

5.4.2 Data Publishing

Data publishing in this context refers to the technology of making data available on the web to be understandable and consumable by machines and humans in a standard format. This is a bidirectional process that includes exporting and importing data from and to Community Linked to and from other entities.

This section presents an ontology that represents Community Linked hierarchical structure including XaaO annotation. It is important to note the difference between semantic web technologies and how OWL and RIF rules are different from XaaO rules.

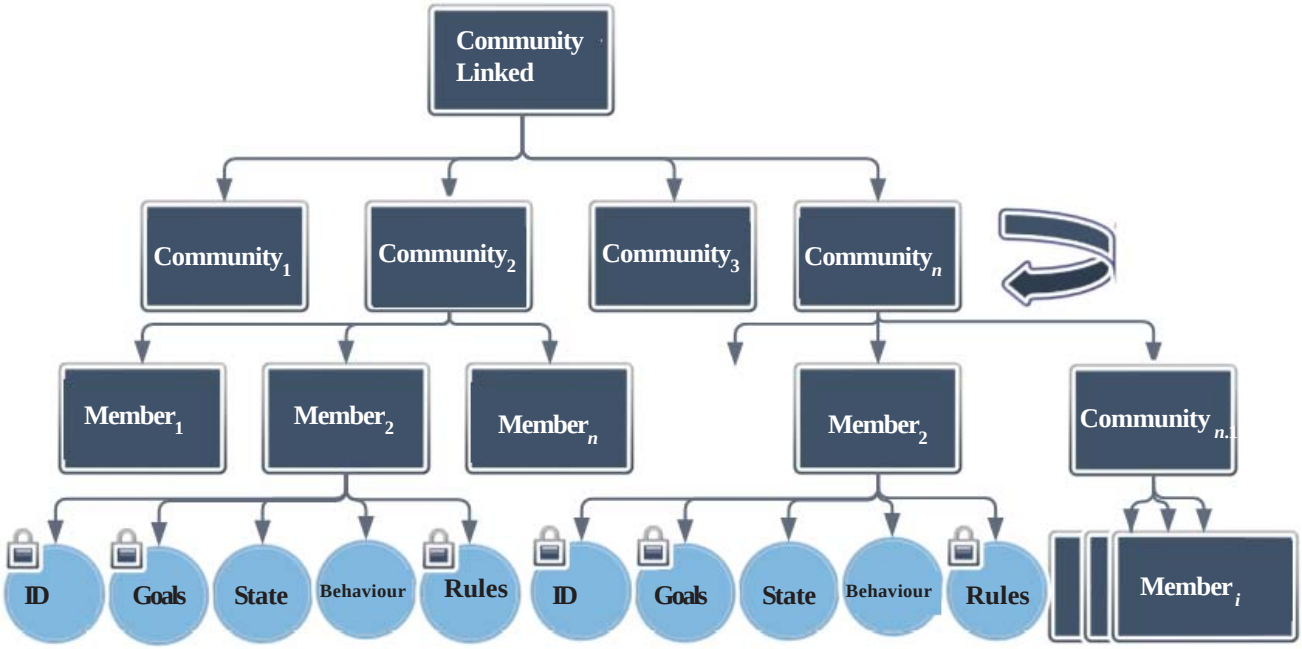


Figure 5. Community Linked hierarichal view structure

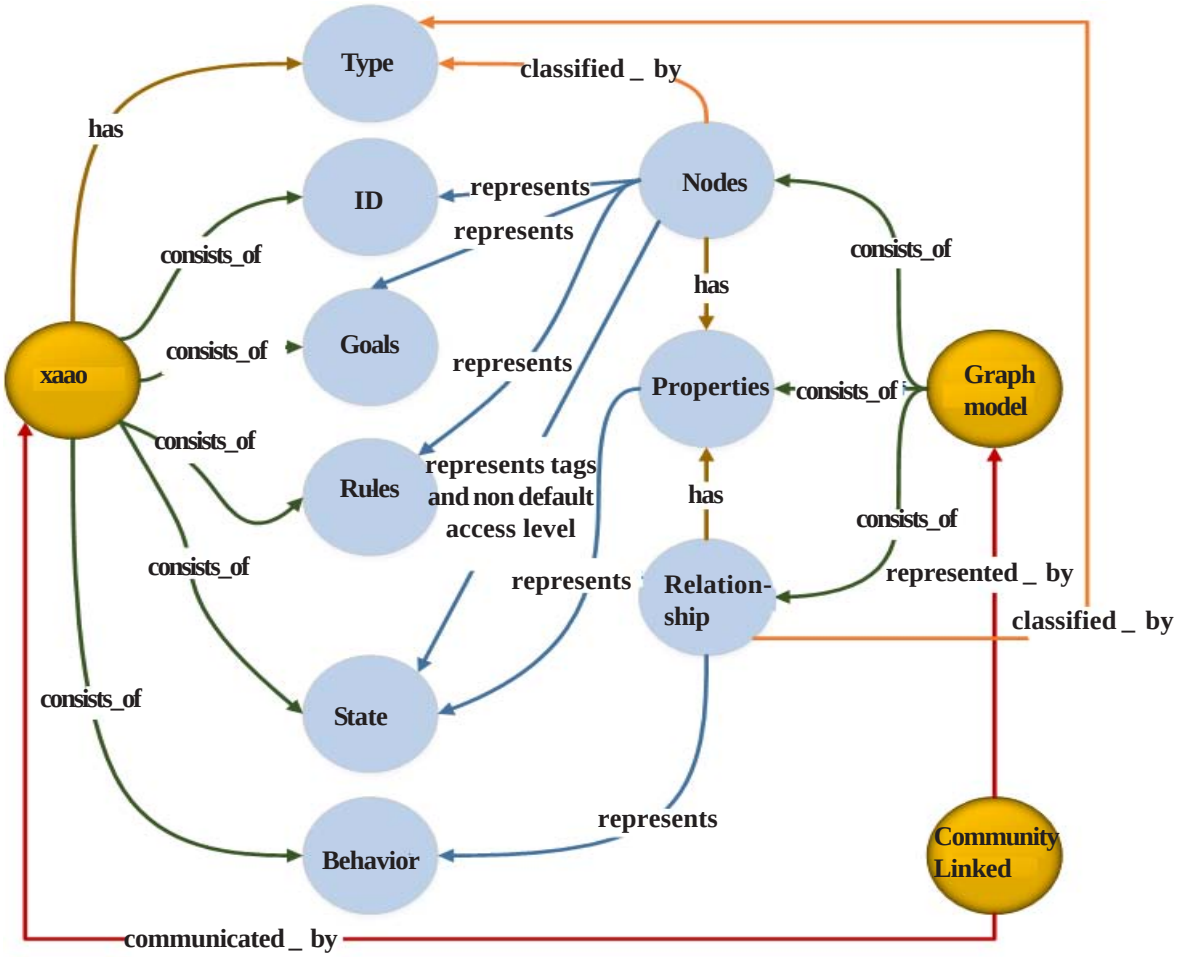


Figure 6. Relationship between Community Linked, XaaO and Graph Model

The World Wide Web Consortium (W3C) objective is to help standardize a semantic web technology stack to support Web 3.0 vision and Linked Data approach. The current standard uses Web Ontology Language (OWL) as vocabularies to define concepts (classes, subclasses and individuals) and their relationships. Resource Description Framework (RDF) as a graph-based data format is used for describing entities in the form of triples: subjects (nodes), predicates (edges), and objects (nodes can be a literal value or a referring to another subject). SPARQL is used as a query language for RDF. Rule Interchange Format (RIF) supports reasoning over data by defining and interchanging rules between systems to enrich the language. OWL and RIF rules provides a general mechanism to discover (infer) new relationships based on existing ones [13]. On the other hand, XaaO's rules are not intended to make the language richer but is focused on providing self-governance to the data. Appendix C provides a conceptual model for the Community Linked OWL that was created for this framework. In addition, appendix D presents a physical graph representation for the OWL using OntoGraf plugin. OntoGraf was added to Protégé Ontology Editor which was used to develop the ontology [14].

5.5 Data, Description, Discovery, Integration and Aggregation - DDDIA

DDDIA (pronounced triple DIA) is responsible for integrating data from different sources and aggregate it. It has two main services:

5.5.1 Integration Service

Community Linked User Interface (UI) and other external client systems can invoke the Integration Service via different endpoints to publish data that conforms to the protocol described in the data publishing section. Integration Service is an orchestrator service that sequentially invokes the following services:

• **Extractor Service:** Responsible for parsing and validating the data according to the XaaO annotation guidelines and corresponding OWL classes.

• **Transformer Service:** Responsible for transforming and preparing the data to be loaded into the search index and database. It also identifies the existing tags and goals in the system that belong to the same type and have similar meaning to the newly added ones to add a link (i.e. same as) between them.

• **Loader Service:** Responsible for connecting to the search index and database to load the data from the transformer service. Loader service asynchronously invokes the Centrality Ranking Service and send a list of the recently updated or added entities and connections.

• **Centrality Ranking Service:** Responsible for determining degree centrality for the entity node and rank it according to the community members within the same community. The service uses equation (1) to determine entity centrality. The values (i.e. α and β) of E_+ and $E\_$ depends on the agreeable configuration by the community initiators. Afterwards, the service updates the trust level according to the ranking level. The update of the trust level is an example of how the entity's behavior impacts its state as depicted in Figure 2.

$$\sigma_d(x) = \sum_{i=1}^{n} e_{ix} \begin{cases} e_{ix} = 1 \text{ if } e \in E_n \\ e_{ix} = \alpha \text{ if } e \in E_+ \\ e_{ix} = \beta \text{ if } e \in E\_ \end{cases} \quad E_+, E_n, E\_ \text{ defined in V.E-2.viii} \qquad (1)$$

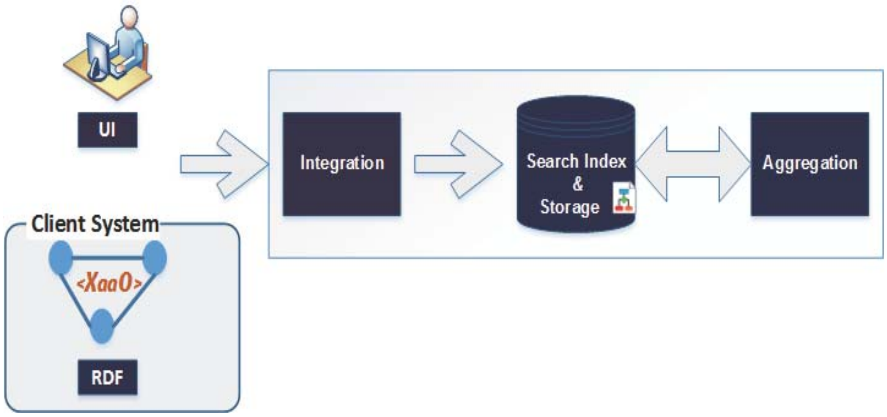


Figure 7. DDDIA Services Interactions

The user interface is a separate component of the Community Linked system and is a consumer of the integration service. It enables members to enroll, login, connect and interact as follows:

• **Enrollment:** community members can subscribe to a community within Community Linked and indicate whether they previously have subscribed to another community. Then, they enter their user information, authentication requirements including certificate options, and finally their goals, tags and rules.

• **Login:** enrolled community members can login to Community Linked using their valid credentials and have access to view their authorized content 3

• **Connect & Interact:** Logged in members can post new content and include goals, tags, rules for it, search for other members and information, and make decisions on whether to accept or reject recommendations for new connections to other members and content based on their goals and preferences. They have the ability to collaborate with their connections on issues and events concerning the community, hence become active and informed members in the community.

2. Aggregation Service

Community Linked relies on aggregation service to unite community members and achieve their goals. XaaO annotation and the graph database model were designed to support the aggregation service in its mission and simplify the discovery process.

• Problem Formulation:

i. Given a directed graph $G = (V, E)$ such that V is the set of all vertices (i.e. nodes) and E is the set of all edges (i.e. links) in the digraph.

ii. $V \triangleq \{ CL \cup C \cup M \cup D \mid \exists\, !\, cl : CL \in V \}$

• $M \subset C$, $\forall c_0 : C \subset (\text{cl}:C \vee CL) \mid c_0 \neq c_1$

• CL is the root of the digraph and represents Community Linked.

• C is a community

• M is a community member

• D is data node $\triangleq \{ D_g \cup D_t \cup D_r \cup D_m \cup D_{id} \cup D_a \}$ that consists of goals, tags, rules messages, identification and other normalized attributes.

iii. $\forall cl : CL \mid \exists\, eg : E = [cl , D_g] \wedge \exists\, et : E = [cl , D_t] \wedge \exists\, er : E = [cl , D_r] \wedge \exists\, eid : E = [cl , D_{id}] \wedge \exists\, ed : E = [cl , D_d]$

iv. $\forall c : C \mid \exists\, eg : E = [c , D_g] \wedge \exists et : E = [c , D_t] \wedge \exists\, er : E = [c , D_r] \wedge \exists\ eid : E = [c , D_{id}] \wedge \exists\ ed : E = [c , D_d]$

v. $\forall m : M \mid \exists\, eg : E = [m , D_g] \wedge \exists\, et : E = [m , D_t] \wedge \exists\ er : E = [m , D_r] \wedge \exists\ eid : E = [m , D_{id}] \wedge \exists\ ed : E = [m , D_d]$

vi. AccessLevel = { DDDIA, Private, Public, Public+W,

vi. AccessLevel = { DDDIA, Private, Public, Public + W, trustedPublic, trustedPublic + W, Internal, Internal + W, trustedInternal, trustedInternal + W, Protected, Protected + W, trustedProtected, trustedProtected + W }

• $\forall\ v : V, e:E \mid al\ (v,e) \in$ AccessLevel

• $\forall\ a$: *Attibute* $\mid a \in d : D \wedge al\ (a) \in$ AccessLevel

vii. TrustLevel = { Low, Medium, High }

• $\exists\ !\ a$:*Attibute* $\in D_{id} \mid tl\ (a) \in$ TrustLevel

viii. $E \triangleq \{ E_+ \cup E_n \cup E_- \mid E_+ \cap E_n \cap E_- = \varnothing \wedge E_n = E_- (E_+ \cup E_-) \}$

• E_+ : a predefined set of positive edges (e.g. "*follow*", "*like*")

• E_-: a predefined set of negative edges (e.g. "*block*")

• E_n : all the other edges that are not in $E+$ and $E-$

• Objective:

Find $R\ (G) = G' = (V', E')$ such that G' is an augmeted digraph of G that includes auto interlinked edges between entities based

• **Proposed Solution:**

Recommendation Function R:

```
Input: G = (V, E)
Output: G' = (V', E')
1: ..
2: Initialize G'= G;
3: foreach (Community C in CL)
4:             Process(C, G');
5: ..
```

Process (sub)community function:

```
Process(C,G'):
1: // C is the (sub)community id in G'
2: ..
3: if (C.hasSubComm)then
4:      foreach (Community subC in C)
5:          Process(subC, G');
6: else
7:     Call Map/Reduce Job to process C
```

Map Function:

```
Map (string inputLine, MapperContext context):
1: // inputLine: serialized graph into JSON string
and read from a file
2: Graph G' ←
JsonConvert.DeserializeObject<Graph>(inputLine);
3: foreach (Node N in G')
4: if (N.type ∈ M) then
5: foreach (Node N' ∉ N.OutLinkList)
6: if (N'.type ∈ M and
         N'.isAccessibleBy(N)) then
7: // 1- include only community member nodes and
candidates
8: // 2- accessible based on:
9: // - access level and
10: // - rule-with-others
11:             EmitKeyValue (N, N');
```

Reduce Function:

```
Reduce (Node N, List<Node> candidates):
```

```
1: // key: node, values: a list of candidate
     nodes to be interlinked
2: ..
3: double neighborSimScore : = 0.0;
4: int cntCnMembers : = 0;
5: foreach (Node C in candidates)
6: {
7: //First, get similarity score between N
goals and C tags
8: directSimScore ← GetSimScore(Ng, Ct);
9: //Next, get average similarity score between
N goals and Cn goals
10: foreach (Node Cn ∈ C.InLinkList)
11: {
12:  if (Cn.type ∈ M) then
13:     {
14:  neighborSimScore += GetSimScore(Ng, Cng);
15:           cntCnMembers++;
16:     }
17: }
18: avgNeighborSimScore ← neighborSimScore /
cntCnMembers;
19: //Finally, Calculate the final score for
the candidate
20:  finalScore ← α * directSimScore +
β...avgNeighborSimScore;
21: // α + β equals to 1
22: EmitLine ("N" + "C" + "finalScore");
23: }
```

on their access level and matching goals to tags and rules.

Recommendation function *R* uses MapReduce framework to process each community. MapReduce provides a simple and automatic environment for parallel processing when there is no dependency between data. It is based on one master that is responsible for scheduling and coordination, and multiple workers that are assiged by the master to perform either map or reduce tasks based on their availability. The framework is perfectly suited for the cloud environment and has also other advantages such as load balancing and fault tolerance.

The Map function produces intermediate results that contains the community member node and its potential candidate community member connections. Candidacy is based on two factors. First, there is no existing explicit link from the node "*N*" to the candidate community member node. Second, the candidate node can be accessed by "*N*" which must meet the access level criteria (i.e. Public [trusted | + *W*], Internal [trusted | + *W*], and rule which can also restict further the type of "*N*" (e.g. organization, person, etc).

The MapReduce framework provides two builtin methods before the Reduce Funtion is invoked. The reduce worker, once assigned by the master, sorts the intermediate results by key (i.e. specific node) and group the candidate nodes associated to it in a list.

The Reduce function is responsible for assigning a ranking score for each candidate community member in the list. The ranking score is based on two factors. First, the similarity between the goals of node "*N*" and the tags of the candidate. Second, the similiarity between the goals of node "*N*" and the goals of the candidate's existing explicit inbound connections of community members. The rationale behind this is that if node "*N*" goals are similar to the goals of the candidate's inbound connection of community members then there is a high propabaility that "*N*" would be interested in making a connection with that candidate since they have similar goals. There are several techiques used to determine similarity such as Jackard, Sørensen and Jaro-Winkler Similarity. However, Community Link aggregation service algorithm uses Cosine function due to its relevance to the application and its efficiency. For example, Jaro-Winkler includes the sequence of the words being matched in the similarity calculation and that is not needed for our application. In addition, the algorithm is not interested in finding what needs to be changed in the compared lists in order to make them similar as in the Hamming distance and Levenshtein distance algorithms.

$$COS(\theta) = \frac{A.B}{\|A\| . \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} (A_i)^2} \sqrt{\sum_{i=1}^{n} (B_i)^2}} \quad (2)$$

To illustrate, let's have "*A*" as a list of goals in the node and "*B*" as the list of tags of the candidate node.

- A = {news, education, health}
- B ={health, entertainment}
- $A \cup B$ ={ news, education, health, entertainment}
- Converting the lists *A* and *B* into frequency of occurrence vectors

$$A = \begin{bmatrix} 1 \\ 1 \\ 1 \\ 0 \end{bmatrix}, B = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 1 \end{bmatrix}$$

- $A.B = 1 * 0 + 1 * 0 + 1 * 1 + 0 * 1 = 1$
- $\|A\| = \sqrt{1^2 + 1^2 + 1^2 + 0^2}$
- $\|B\| = \sqrt{0^2 + 0^2 + 1^2 + 1^2}$
- $\|A\| . \|B\| \approx 2.45$
- $cos(\theta) \approx 0.41$

The final recommendation score between a node "*N*" and its candidate connection "N'"can be determined by equation (3). The sum of the coefficients "α" and" "β" is equal to 1 and they are used to specify the percentage of impact that their associated expressions have on the overall score. The first expression that is associated with "α" produces the similarity score between "*N*" goals to "N'" tags. The second expression that is associated with β produces the average similarity score between "*N*" goals to "N''" (i.e. inbound community members connected to "N'") goals. "*n*" is the number of inbound community members connected to "N'".

$$R\,\text{Score}(N, N') = \alpha * \frac{N_g \cdot N_t'}{\|N_g\| . \|N_t'\|} + \beta * \frac{\sum_{i=1}^{n} \frac{N_g \cdot N_{ig}''}{\|N_g\| . \|N_{ig}''\|}}{n} \quad (3)$$

6. Experimental Results

We present in this paper many novel ideas that require proof of concepts to prove their feasibility and efficincy. In data management section, we implemented a physical graph model that encompasses the guidelines presented in that section including XaaO annotation using Neo4j graph database. Data publishing is also another critical section that requires proof of concept. We depicted a conceptual model for the ontology and implemented it using Protégé Ontology Editor.

Aggregation service is another critical area in the Community Linked framework that also requires a proof of concept. We

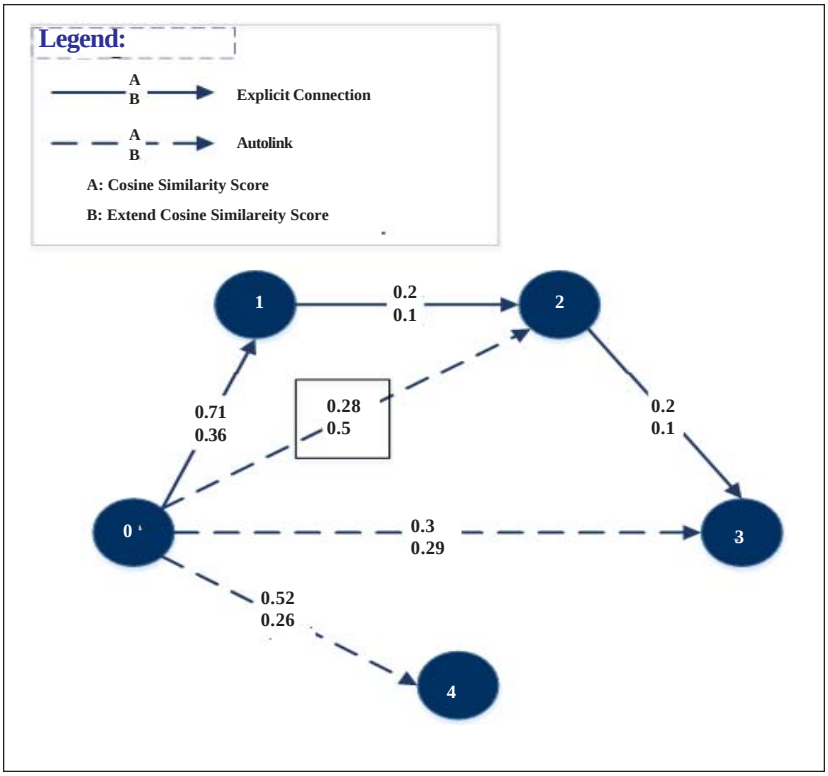


Figure 8. Algorithm Results on a Small Dataset

implemented the service using MapReduce framework in Visual Studio 2013. Appndix E presents the reveresed engineered assembly dependency diagram based on the developed code. In addition, we used Windows Azure to setup HDInsight cluster that consits of 4 data nodes. The cluster's heap size is 270.69 MB / 3.56 GB and its capacity is 3.91 TB. Each worker node in the cluster has a capacity of 1000 GB.

The goal of the Aggregator service is to measure similarity between two entities based on their goals, tags and explicit connections using our algorithm. In this section, we present three experiments. The experiments were conducted on a stochastic community that consists of one million members and each member was assigned three to ten random goals and tags. Community members also had six hundred thousand explicit connections among them. "α" and" "β" in formula (3) were equally set to "0.5". This should vary based on the configuration setup of the real community.

In part one of the expriment, we run the algorithm on a small number of members to show the details of the results. The MapReduce Job ran on a sub-graph that conisits of 5 members who have 3 explicit connections. Figure 8 depicts the results of the algorithm and appendix F presents the details. Solid line edges represent explicit connections and dotted line edges represent a recommended "*Autolink*" conncetion. Each edge has two scores "*A*" and "*B*".

- "*A*" measures the direct similarity between the source node goals and the target node tags.
- "*B*" is the result of formula (3) which includes half of "*A*" and half of the average simiarity between source node goals and target node's explicit neighbors' goals.

We observe in this experiment the negative impact of not having explicit inbound connections and positive impact of having inbound connections that have similar goals to the goals of the source node .

The second part of the experiment is executed on a larger dataset that consists of 100 community memebers who have 86 explicit connections. Appendix G displays the counts of cosine and extended cosine similarity scores. It is evident that adding the nodes' inbound connections to the formula changed the recommendations and introduced new connections that have better chance to be a good match not only based on the direct target node's tags but also its inbound explicit connections' goals.

	Link Patterns in the World Wide Web						
	Linking Documents	**Linking People to**	**Linking services**	**Linking People**	**Linking Objects**	**Linking Data**	**Linking Communities & Beyond Documents (our solution)**
Web Version	Web 1.0	Web 1.0	Web 2.0	Web 2.0	Web 2.0	Web 3.0	Web 3.0 empowerd by XaaO
Link Type	Explicit	Implicit	Implicit	Implicit	Explicit	Explicit	Implicit & Explicit
User Access	Read Only	Read Only	Read/Write & Application Level	Read/Write	N/A Only Application Level	Read & Application Level	Read/Write & application level based on XaaO rules and access control
Link Mechanism	Hyperlinks	Search engines	UDDI Search and auto discovery	Request/accept a connection	Object GUID	URI	Data is self-governed via XaaO and managed DDDIA to create manage links
Impact	Global Document Space	Ease of access to information and service providers	Service Mashups	Social Network	Global Object Space	Global Data Space	Extensible and polymorphous Community Linked
Shortcomings/ Challenges	• No typed links • No collaboration capabilities. • Not understandable by machines.	• Crawlers are used to centralize and index documents instead of realtime or near-realtime lookup. • Search algorithms mainly rely on links which may be used for other purposes. • Web usage and content mining are used to optimize results. • Web content mining must accommodate different languages and locales. • Spurious efforts to increase visibilty (spam) must be blocked. • Users are not able to provide feedback on results.	• Optimize dynamic service composition using semantic and social web techniques. • Quality of Service verification	• Published data is not represented semantically • Sentiment analysis, opinion mining and community detection. • Optimize data management for intelligent recommendations based on the network topology	• No typed links between objects. • Object interactions and state dependencies are constrained by domain and application logic. • No metadata is used to describe the object	• Limited expressivity in OWL and RDF • Link maintenance • Data integration and aggregation • Lack of support for integrity constraints and closed-world querying • Automatically interlink similar data	• XaaO annotation to be adopted by W3C OWL standards for defining “Thing” base class. • Generalize the solution to be extensible and customizable based on the specific solution needs • Avoid single ownership to ensure that the objectives of all community members are addressed.

Table2. Link Patterns Evaluation Results

Finally, we compare the performance of using a single Map/Reduce task in the Map/Reduce job versus 16 Map and 8 Reduce tasks. MapReduce framework provides a great benefit to safely parallelize the execution of the algorithm and provides fault tolerance. Appendix H presents the results of this performance test based on a dataset of 100 members with 60 explicit connections. It shows that it took the jobs that end with 63(templeton controller job)/64(stream job) 52 minutes and 10 seconds to process the data, whereas it took 65(templeton controller)/66(stream job) only 21 minutes and 1 second to process the same data with higher Map/Reduce tasks. This is a 60 % performance improvement which will be significant on a large scale of data.

7. Conclusion and Future Work

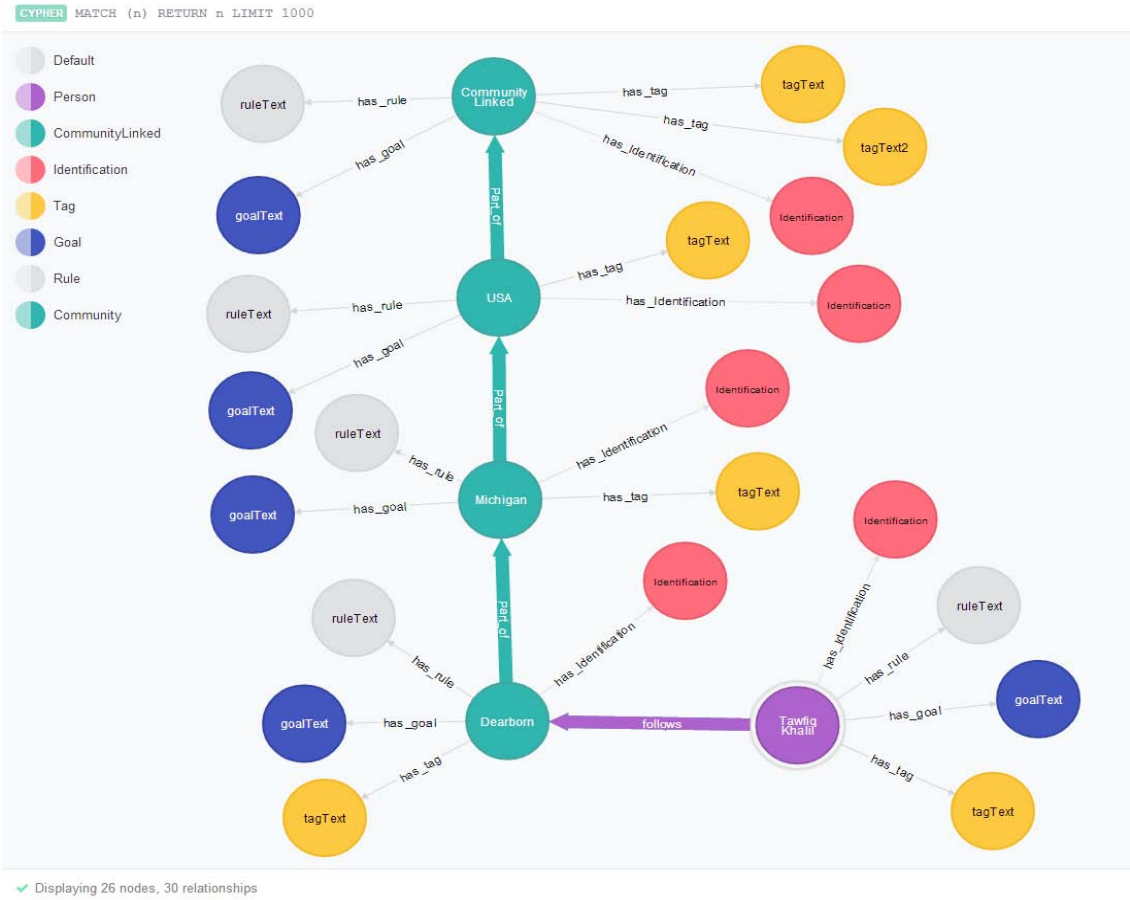
There are many lessons can be learned from the success and the evolution of the World Wide Web. Web 1.0 has been successful

in linking documents, and people to documents and services. On the other hand, it did not empower the user to collaborate and contribute to the application. This shortcoming was rectified in Web 2.0; where the user became an integral part of the application and the web became a collaboration platform. Yet, there is another issue that began since the inception of the World Wide Web and was not addressed in Web 2.0. The web has become a massive decentralized environment that contains semantic-free documents. The dependence on hyperlinks between documents can be misleading and does not reveal the desired intelligence from the documents. Other complex web mining techniques are limited due to scarcity of meta-data on the documents. The objective of Web 3.0 is to fulfil this need by applying semantics to data in order to improve knowledge discovery. Linked data was the de facto adopted approach by W3C to semantically represent and connect data by utilizing Semantic Web technology stack. Yet, the decentralized nature of the web and lack of self-governance created many chaotic repositories that contain billions of triplets and thousands of ontologies. As a results, many dangling and unreachable links exist which require extensive amount of processing to navigate through a machine friendlier linked data. In addition, data mapping and translation is required to understand data based on the ongoing created and updated ontologies.

CKAN is a prominent open source data management solution that attempts to bring order to the Semantic Web and centralize data. It provides an efficient way for data publishers to publish their datasets and provide metadata to help making their data discoverable. Tags, groups and organizations are three main facilities to organize and control access to data. Tags along with other metadata can be used to search for data. In addition, publishers can assign their dataset to a group to classify it and enable searching by category. An organization can also register its members and set access for them on its published datasets. A dataset can have private or public access. Organizations can select private option to restrict access to its members only.

The primary objective of this paper is to shed light on the importance of local community on our life, and the countless opportunities a community has when its members are united and informed about their local issues and challenges. Current research literature that focuses on social web has not targeted local communities; instead, focused on connecting distant people together and was limited in its scope (e.g. networks of professionals, music and friends). It is our ambition to instigate researchers to focus on bringing local communities together. Community Linked framework with its novel XaaO annotation reifies these noble concepts. It is based on the lessons learned from Web1.0 through Web 3.0. It aims to create a scalable and reusable collaboration environment where users are empowered as in Web 2.0. It also intends to embody the data with meta-data as in Web 3.0. It shares CKAN's objective to centralize and integrate data to make it discoverable. However, it is distinct in its vision and techniques. The choice of Community Cloud as a deployment model makes community members not only contributors to the content of the application but also partners in setting the community goals and its capabilities. Utilizing OWL and extending "*Thing*" class to include XaaO annotation does not only add semantics to the data but also make it self-governed. XaaO is enabling data to have its own goals, tags, state, behavior, rules, type(s) and access level not only on the entity level but also on a more granular level that includes data attributes. In reality, rules controls behavior and behavior impacts state. DDDIA embraces this reality in its integration and aggregation services when determining trust level and recommending connections. Graph databases are schema-free to provide flexibility but that can limit its potential. Adopting XaaO in the graph model will normalize and standardize the stored data which simplifies and improves data access and knowledge discovery. Finally, the hierarchical representation of Community Linked has many advantages. It naturally supports community detection and classification. Members are always connected to a community which simplifies the process of determining degree centrality that reveals influential and informal leaders in the community and affect their trust level.

Community Linked introduces a new initiative and pattern in the World Wide Web. Many research initiatives can build on its philosophy and vision. XaaO annotation can be enhanced further to better support social network analysis. OWL "*Thing*" class can be standardized to include XaaO structure. Aggregation Service algorithm can be easily expanded to not only recommend members to members but also members to accessible messages and messages to messages as well. Authentication and authorization can be improved and simplified based on the continuous advancements in information security. Community Linked can be polymorphous and applied to various scenarios. For example, a large enterprise can adopt this model to have communities as its organizations and sub communities as its departments to better connect its employees and achieve their department's goals. In fact, XaaO annotation and our initiative to make data self-governed can play a big role in the next version of the World Wide Web.



APPENDIX A: Graph DB Query Results

Appendix B: Graph DB Creation Script

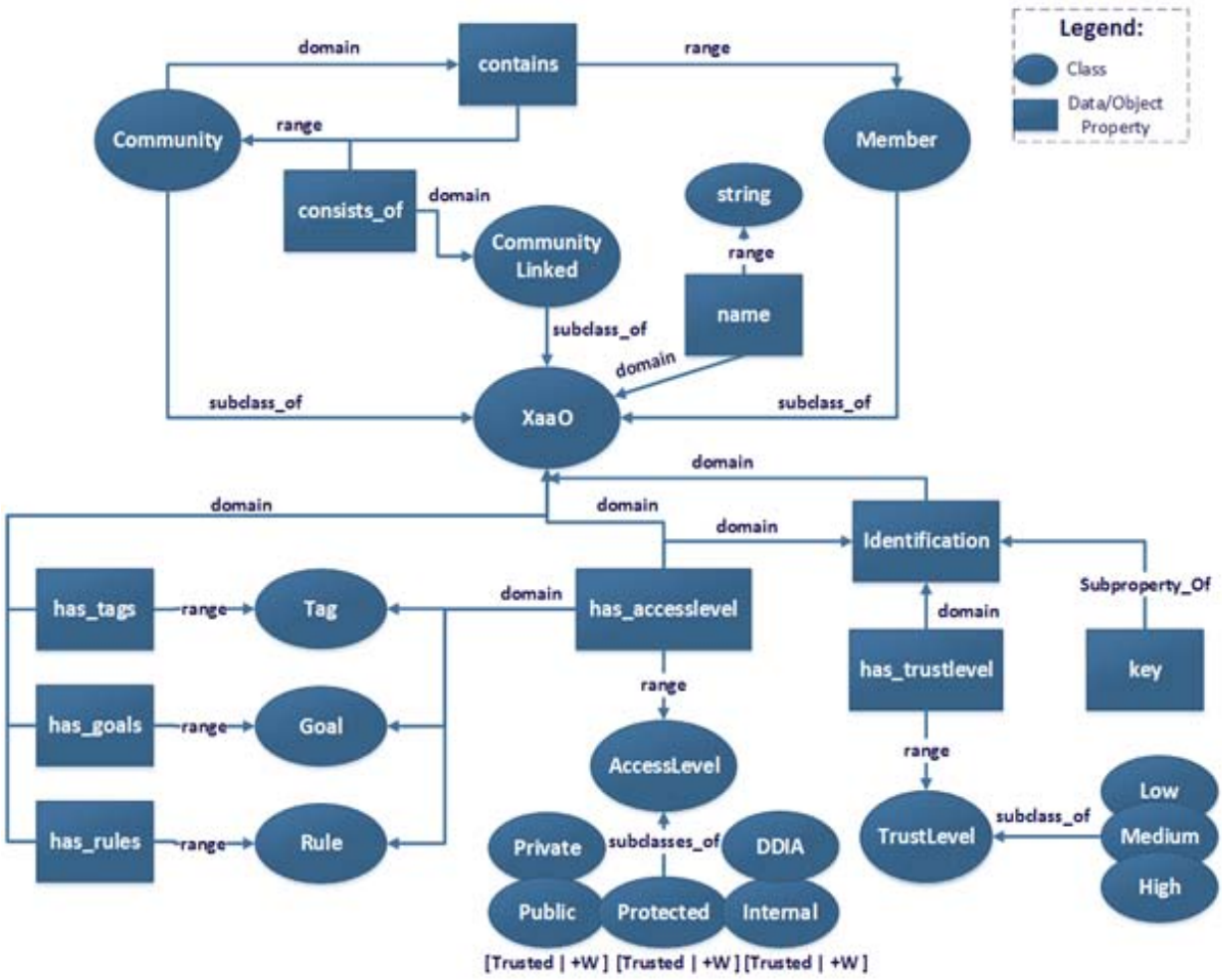
```
//************************************************************************//
// creating the Community Linked node which will be the root of the graph. //
// root should not represent any entity because it is not recursive and that //
// allows future scalability                                              //
//************************************************************************//
CREATE (Root:CommunityLinked{accesslevel:'public',
      name:'Community Linked'
      //Default access level for attribute is public.
      //it can be written as name:['CommunityLinked', 'accesslevel:protected']
      //for other access type for the attribute. It is an array where the
      //second element is the access level prefixed with string "accesslevel:".
      //or it can be normalized to a relation and node following the core relations
      below
})
//*******************************************//
// creating the core relations for the root                    //
//*******************************************//
```

```
 //Identification: Unique ID is automatically generated. This node will contain the rest of the information
CREATE (RootID:Identification{name:'Identification',key:'encrypted',trustlevel:'high'})
CREATE UNIQUE (Root)-[:has_Identification{accesslevel:['DDDIA']}]->(RootID)

//Tags: this is the XaaO state tags. The node has tag info, the relation has the access level
CREATE (rootTag1:Tag{name:'tagText'})
CREATE UNIQUE (Root)-[:has_tag{accesslevel:['public']}]->(rootTag1)
CREATE(rootTag2:Tag{name:'tagText2'})CREATE UNIQUE (Root)-[:has_tag{accesslevel:['public']}]->(rootTag2)

//Goals: this is the XaaO Goals. The node has Goal info, the relation has the access level
CREATE (rootGoal1:Goal{name:'goalText'})
CREATE UNIQUE (Root)-[:has_goal{accesslevel:['private']}]->(rootGoal1)

//Rules: this is the XaaO Rules. The node has rule info, the relation has the access level
CREATE (rootRule1:Rule{name:'ruleText'})
CREATE UNIQUE (Root)-[:has_rule{accesslevel:['private']}]->(rootRule1)


//*************************************************************************//
//creating Community node for USA                                          //
//*************************************************************************//
CREATE (USA:Community{accesslevel:'public',
      name:'USA',
      population:'313,914,040',
      language:'English'
            //tags as a state are represented as another node to optimize performance
 })
//*****************************************************//
// creating the core relations for the USA Community   //
//*****************************************************//
//Identification: Unique ID is automatically generated. This node will contain the rest of the information
CREATE (USAID:Identification{name:'Identification', key:'encrypted',trustlevel:'high'})
CREATE UNIQUE (USA)-[:has_Identification{accesslevel:['DDDIA']}]->(USAID)

//Tags: this is the XaaO state tags. The node has tag info, the relation has the access level
CREATE (USATag1:Tag{name:'tagText'})
CREATE UNIQUE (USA)-[:has_tag{accesslevel:['public+W']}]->(USATag1)

//Goals: this is the XaaO Goals. The node has Goal info, the relation has the access level
```

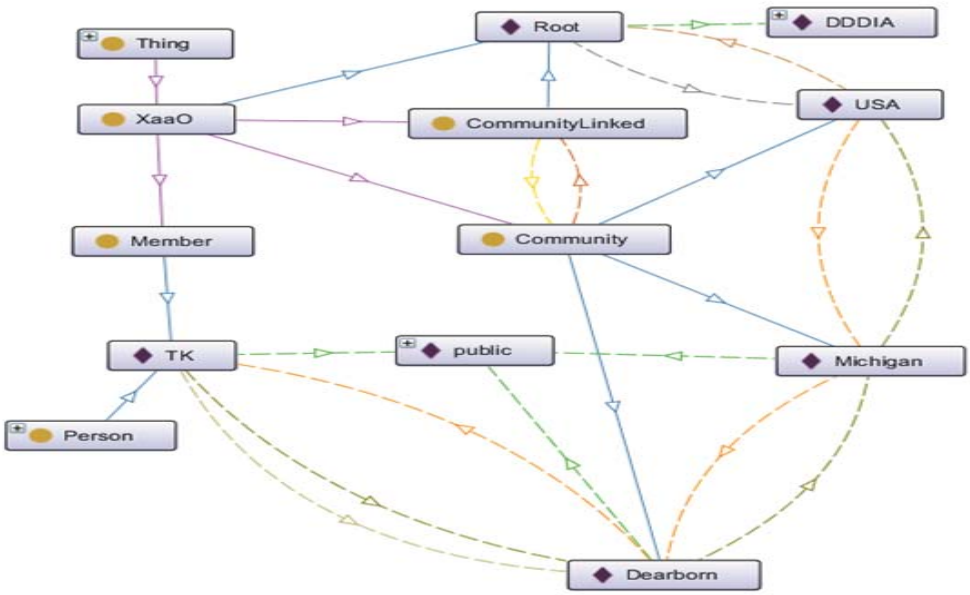
```
CREATE (USAGoal1:Goal{name:'goalText'})
CREATE UNIQUE (USA)-[:has_goal{accesslevel:['private']}]->(USAGoal1)

//Rules: this is the XaaO Rules. The node has rule info, the relation has the access level
 CREATE (USARule1:Rule{name:'ruleText'})
CREATE UNIQUE (USA)-[:has_rule{accesslevel:['private']}]->(USARule1)


//************************************************************************//
 // Connecting USA Community to Root CommunityLinked //
//************************************************************************//
CREATE UNIQUE (Root)-[:consists_of{accesslevel:['public']}]->(USA)
CREATE UNIQUE (USA)-[:Part_of{accesslevel:['public']}]->(Root)


//************************************************************************// /
/ creating Community node for Michigan State as a subcommunity //
//************************************************************************//
CREATE (Michigan:Community{accesslevel:'public',
      name:'Michigan',
      population:'9,883,640'
            //tags as a state are represented as another node to optimize performance
})
//*********************************************************//
// creating the core relations for the Michigan Community //
//*********************************************************//
//Identification: Unique ID is automatically generated. This node will contain the rest of the information
CREATE
(MichiganID:Identification{name:'Identification',key:'encrypted',trustlevel:'high'})
CREATE UNIQUE (Michigan)-[:has_Identification{accesslevel:['DDDIA']}]->(MichiganID)

//Tags: this is the XaaO state tags. The node has tag info, the relation has the access level
CREATE (MichiganTag1:Tag{name:'tagText'})
 CREATE UNIQUE (Michigan)-[:has_tag{accesslevel:['public+W']}]->(MichiganTag1)

//Goals: this is the XaaO Goals. The node has Goal info, the relation has the access level
CREATE (MichiganGoal1:Goal{name:'goalText'})
CREATE UNIQUE (Michigan)-[:has_goal{accesslevel:['private']}]->(MichiganGoal1)

//Rules: this is the XaaO Rules. The node has rule info, the relation has the access level
CREATE (MichiganRule1:Rule{name:'ruleText'})
CREATE UNIQUE (Michigan)-[:has_rule{accesslevel:['private']}]->(MichiganRule1)
```

```
//************************************************************************//
// Connecting Michigan Community to USA Parent Community                  //
//************************************************************************//
CREATE UNIQUE (USA)-[:consists_of{accesslevel:[‘public’]}]->(Michigan)
CREATE UNIQUE (Michigan)-[:Part_of{accesslevel:[‘public’]}]->(USA)

//************************************************************************//
// creating Community node for Dearborn city as a subcommunity //
//************************************************************************//
CREATE (Dearborn:Community{accesslevel:’public’,
      name:’Dearborn’,
      population:’98,153'
            //tags as a state are represented as another node to optimize performance
})
//*******************************************************//
// creating the core relations for the Dearborn Community //
//*******************************************************//
//Identification: Unique ID is automatically generated. This node will contain the
rest of the information
CREATE
(DearbornID:Identification{name:’Identification’,key:’encrypted’,trustlevel:’high’})
CREATE UNIQUE (Dearborn)-[:has_Identification{accesslevel:[‘DDDIA’]}]->(DearbornID)
//Tags: this is the XaaO state tags. The node has tag info, the relation has the access
level
CREATE (DearbornTag1:Tag{name:’tagText’})
CREATE UNIQUE (Dearborn)-[:has_tag{accesslevel:[‘public+W’]}]->(DearbornTag1)

//Goals: this is the XaaO Goals. The node has Goal info, the relation has the access
level
CREATE (DearbornGoal1:Goal{name:’goalText’})
CREATE UNIQUE (Dearborn)-[:has_goal{accesslevel:[‘private’]}]->(DearbornGoal1)

//Rules: this is the XaaO Rules. The node has rule info, the relation has the access
level
CREATE (DearbornRule1:Rule{name:’ruleText’})
CREATE UNIQUE (Dearborn)-[:has_rule{accesslevel:[‘private’]}]->(DearbornRule1)

//************************************************************************//
// Connecting Dearborn Community to Michigan Parent Community //
//************************************************************************//
CREATE UNIQUE (Michigan)-[:consists_of{accesslevel:[‘public’]}]->(Dearborn)
CREATE UNIQUE (Dearborn)-[:Part_of{accesslevel:[‘public’]}]->(Michigan)

//************************************************************************// /
```

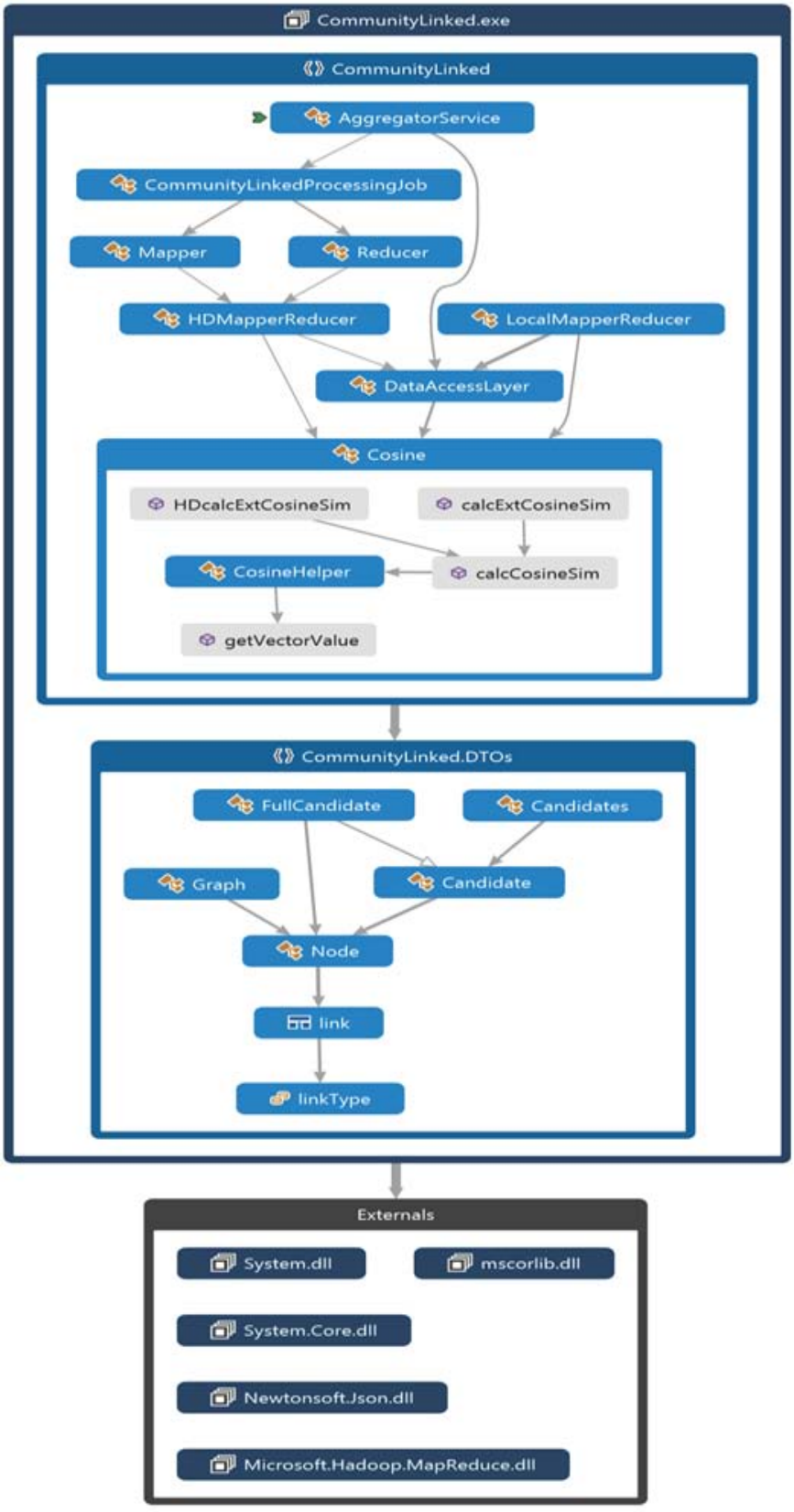
```
/ creating Member node for Tawfiq as a member and Person Type //
 //*************************************************************************//
CREATE (TK:Member:Person{accesslevel:'public',
      name:'Tawfiq Khalil',
      Gender:'Male'
            //tags as a state are represented as another node to optimize performance
})
//*********************************************************//
// creating the core relations for Tawfiq as a Member & Person //
//*********************************************************  //
//Identification: Unique ID is automatically generated. This node will contain the rest of the information
CREATE (TKID:Identification{name:'Identification',key:'',trustlevel:'low'})
CREATE UNIQUE (TK)-[:has_Identification{accesslevel:['DDDIA']}]->(TKID)

//Tags: this is the XaaO state tags. The node has tag info, the relation has the access level
CREATE (TKTag1:Tag{name:'tagText'})
CREATE UNIQUE (TK)-[:has_tag{accesslevel:['public+W']}]->(TKTag1)

//Goals: this is the XaaO Goals.The node has Goal info, the relation has the access level
CREATE (TKGoal1:Goal{name:'goalText'})
CREATE UNIQUE (TK)-[:has_goal{accesslevel:['private']}]->(TKGoal1)
//Rules: this is the XaaO Rules. The node has rule info, the relation has the access level
CREATE (TKRule1:Rule{name:'ruleText'})
CREATE UNIQUE (TK)-[:has_rule{accesslevel:['private']}]->(TKRule1)


//*************************************************************************//
// Connecting TK to Dearborn Community and establish follow relationship per user story //
//*************************************************************************//
CREATE UNIQUE (Dearborn)-[:consists_of{accesslevel:['public']}]->(TK)
CREATE UNIQUE (TK)-[:Part_of{accesslevel:['public']}]->(Dearborn)
CREATE UNIQUE (TK)-[:follows{accesslevel:['public']}]->(Dearborn)
```



Appendix C: Web Ontology Language (OWL) Conceptual Model



Appendix D: OWL Physical Representation Via OntoGraf



Appendix E: Aggregator Service: High-level Code & Assembly Dependency Diagram

Appendix F: Algorithm Results on a Small Dataset

Node 0		Node 1		Node 2		Node 3		Node 4	
Goals	Tags	Goals	Tags	Goals	Tags	Goals	Tags	Goals	Tags
BP	Impeachment	BP	BP	Recall	Recall	Elections	Recall	Elections	Elections
Glenn Beck	John McCain	Glenn Beck	Glenn Beck	LGBT	LGBT	Mitch McConnell	LGBT	Mitch McConnell	Mitch McConnell
							Bush		
Global Warming	Haifa	Global Warming	Global Warming	Bush	Bush	DNC	Photos	DNC	DNC
Elections	Religious Right	Teaching	Teaching	Photos	Photos	Glenn Beck		Glenn Beck	
Mitch McConnell	Humor	Veterans	Veterans	International	International		International		
	Debate	Harry Reid	Harry Reid	Top Comments	Top Comments			New Jersey	
		Impeachment	Impeachment	Taxes	Taxes		Top Comments	Minnesota	
		Protest	Protest	South Carolina	South Carolina		Taxes	Supreme Court	
		Elections	Elections	Elections	Elections		Elections	Japan	
		Mitch McConnell	Mitch McConnell	Mitch McConnell	Mitch McConnell		Mitch McConnell		

Direct Cosine Similarity (0->1)		
Union	Node 0 goals	Node 1 tags
BP	1	1
Glenn Beck	1	1
Global Warming	1	1
Elections	1	1
Mitch McConnell	1	1
Teaching	0	1
Veterans	0	1
Harry Reid	0	1
Impeachment	0	1
Protest	0	1
Cosine Score	0.71	

Direct Cosine Similarity (1->2)		
Union	Node 1 goals	Node 2 tags
BP	1	0
Glenn Beck	1	0
Global Warming	1	0
Teaching	1	0
Veterans	1	0
Harry Reid	1	0
Impeachment	1	0
Protest	1	0
Elections	1	1
Mitch McConnell	1	1
Recall	0	0
LGBT	0	0
Bush	0	0
Photos	0	0
International	0	0
Top Comments	0	0
Taxes	0	0
South Carolina	0	0
Cosine Score	0.2	

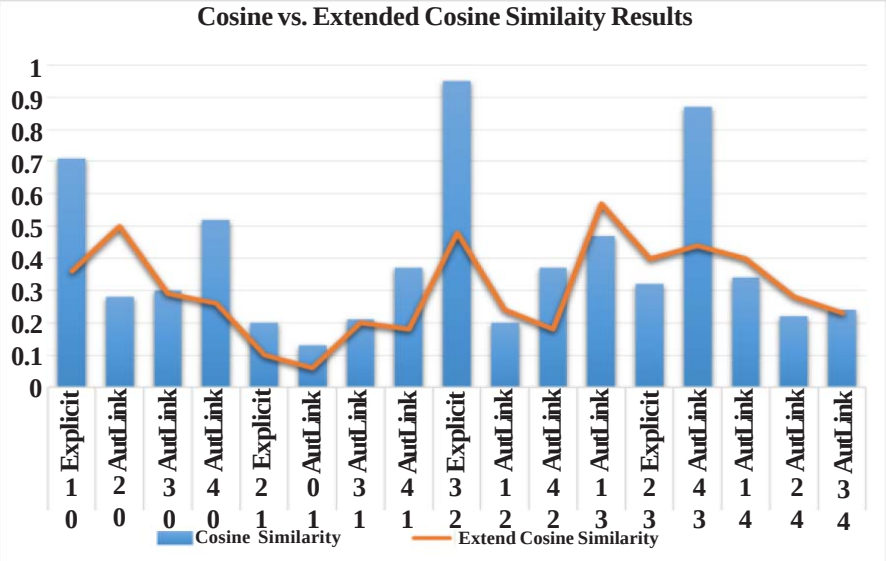
Direct Cosine Similarity (2->3)		
Union	Node 2 goals	Node 3 tags
Recall	1	1
LGBT	1	1
Bush	1	1
Photos	1	1
International	1	1
Top Comments	1	1
Taxes	1	1
South Carolina	1	0
Elections	1	1
Mitch McConnell	1	1
Cosine Score	0.95	

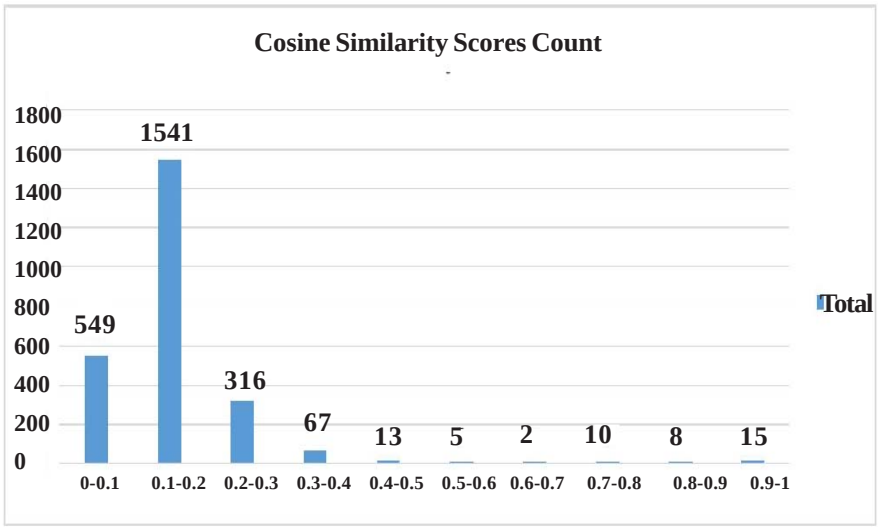
Direct Cosine Similarity (3->4)		
Union	Node 3 goals	Node 4 tags
Elections	1	1
Mitch McConnell	1	1
DNC	1	1
Glenn Beck	1	0
Cosine Score	0.87	

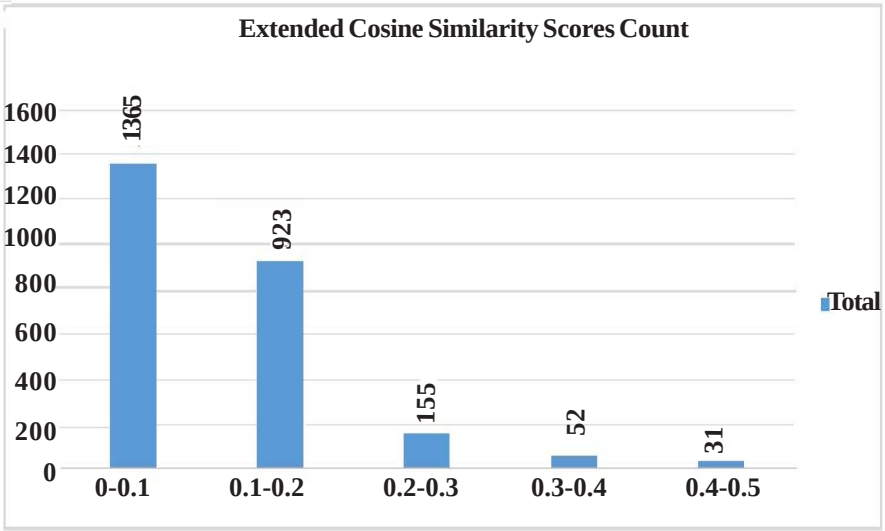
Direct Cosine Similarity (0->2)		
Union	Node 0 goals	Node 2 tags
BP	1	0
Glenn Beck	1	0
Global Warming	1	0
Elections	1	1
Mitch McConnell	1	1
Recall	0	1
LGBT	0	1
Bush	0	1
Photos	0	1
International	0	1
Top Comments	0	1
Taxes	0	1
South Carolina	0	1
Cosine Score	0.28	

Direct Cosine Similarity with Node 2 neighbor (0->1)		
Union	Node 0 goals	Node 0 goals
BP	1	1
Glenn Beck	1	1
Global Warming	1	1
Elections	1	1
Mitch McConnell	1	1
Teaching	0	1
Veterans	0	1
Harry Reid	0	1
Impeachment	0	1
Protest	0	1
Cosine Score	0.71	

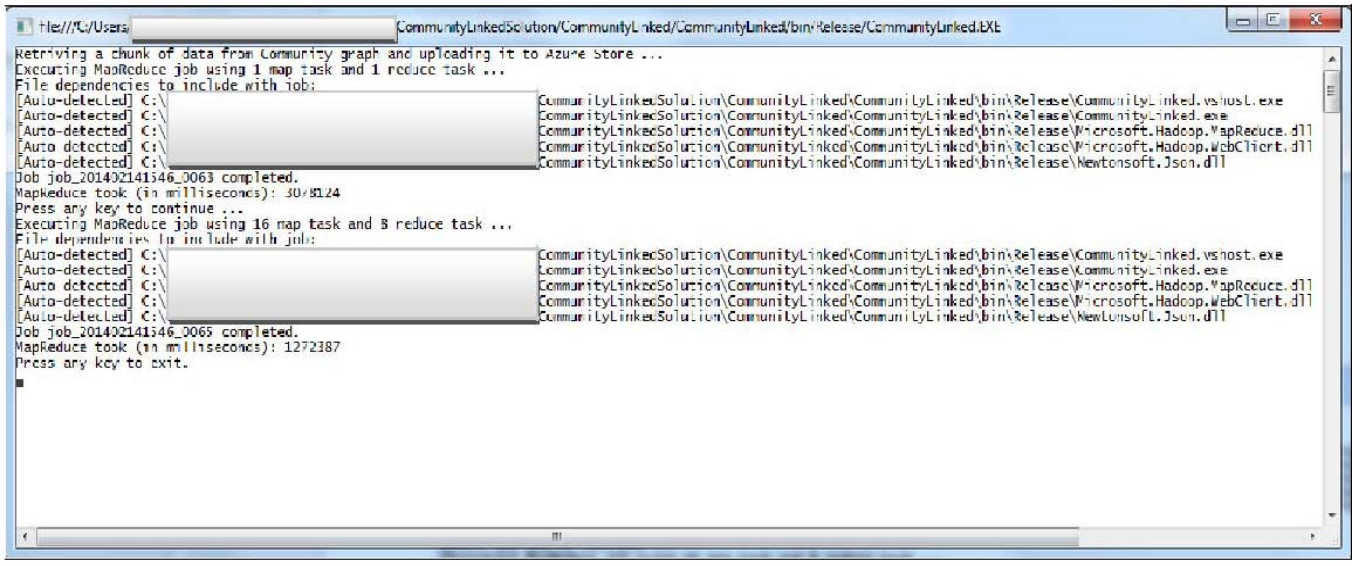
Extended Cosine Score = 0.5 * 0.28 + 0.5 * 0.71 = 0.5







Appendix G. Algorithm Results on a larger Dataset



Hadoop job_201402141546_0064 on jobtrackerhost

User: tawfiq_khalil
Job Name: streamjob1453681849373132493.jar
Job File: hdfs://namenodehost:9000/mapred/staging/tawfiq_khalil/.staging/job_201402141546_0064/job.xml
Submit Host: RD00155DC0DB5E
Submit Host Address: 100.66.136.68
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Sun Feb 16 04:13:04 GMT 2014
Finished at: Sun Feb 16 05:03:44 GMT 2014
Finished in: 50mins, 40sec
Job Cleanup: Successful
Job Scheduling information: 0 running map tasks using 0 map slots. 0 additional slots reserved. 0 running reduce tasks using 0 reduce slots. 0 additional slots reserved.

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	2	0	0	2	0	0 / 0
reduce	100.00%	1	0	0	1	0	0 / 0

Hadoop job_201402141546_0066 on jobtrackerhost

User: tawfiq_khalil
Job Name: streamjob5868925047110200885.jar
Job File: hdfs://namenodehost:9000/mapred/staging/tawfiq_khalil/.staging/job_201402141546_0066/job.xml
Submit Host: RD00155DC0F252
Submit Host Address: 100.66.112.71
Job-ACLs: All users are allowed
Job Setup: Successful
Status: Succeeded
Started at: Sun Feb 16 05:07:39 GMT 2014
Finished at: Sun Feb 16 05:28:08 GMT 2014
Finished in: 20mins, 29sec
Job Cleanup: Successful
Job Scheduling information: 0 running map tasks using 0 map slots. 0 additional slots reserved. 0 running reduce tasks using 0 reduce slots. 0 additional slots reserved.

Kind	% Complete	Num Tasks	Pending	Running	Complete	Killed	Failed/Killed Task Attempts
map	100.00%	17	0	0	17	0	0 / 0
reduce	100.00%	8	0	0	8	0	0 / 0

Appendix H: Performance Analysis Results

References

[1] Rainie., Lee., Kristen Purcell., Aaron Smith. (2011). The social side of the internet. Pew Internet and American Life Project.

[2] Hampton., Keith.,Goulet, Lauren Sessions., Rainie, Lee ., Purcell, Kristen. (2011). Social networking sites and our lives. Pew Internet and American Life Project.

[3] Anderson., Quitney, Janna., Rainie, Lee. (2010). The future of social relations. Pew Research Center's Internet & American Life Project 2.

[4] Khalil, Tawfiq., Wu, Ching-Seh (Mike). (2013). Link Patterns in The World Wide Web. *International Journal of Information Technology and Management Information Systems* (IJITMIS). 4 (3) 96-113.

[5] Crime Prevention. Law Enforcement. National Sheriffs' Association, n.d. Web. 22 Dec. 2013. <http://www.sheriffs.org/content/crime-prevention>.

[6] Nixle., Nixle- Building Safer Communities Together. Nixle, n.d. Web. 22 Dec. 2013. <http://www.nixle.com/>.

[7] Erl, Thomas., Mahmood, Zaigham., Puttini, Ricardo. (2013). Cloud Computing: Concepts, Technology & Architecture. Prentice Hall. Edition 1. May 20.

[8] Liu., Fang., Tong, Jin., Mao, Jian., Bohn, Robert., Messina, John., Badger, Lee., Leaf, Dawn . (2011). NIST cloud computing reference architecture. NIST special publication 500, 292.

[9] Thomas., Stephen. SSL and TLS Essentials. Wiley Computer Publishing. (2000).

[10] RFC Editor. Official Internet Protocol Standards. Official Internet Protocol Standards. RFC Publication, n.d. Web. 29 Dec. 2013. <http://www.rfc-editor.org/rfcxx00.html>.

[11] Crocker, D., Ed., Overell. P. (2013). ABNF- RFC 5234. ABNF- RFC 5234. Network Working Group, Jan. 2008. Web. 29 Dec. <http://www.rfc-editor.org/rfc/rfc5234.txt>.

[12] Download and Install Neo4j. Neo4j. The Neo4j Community., n.d. Web. 15 Nov. 2013. <http://www.neo4j.org/download>.

[13] Semantic Web, W3C, http://www.w3.org/standards/semanticweb/. Web. Nov. 2013

[14] The Protégé Ontology Editor and Knowledge Acquisition System. The Protégé Ontology Editor and Knowledge Acquisition System. Stanford Center for Biomedical Informatics Research (BMIR) at the Stanford University School of Medicine, n.d. Web. 07 Dec. 2013. <http://protege.stanford.edu/>.

[15] Fortunato, Santo. (2010). Community detection in graphs. Physics Reports 486 (3) 75-174.

[16] Tsuda., Koji., Taku Kudo. Clustering graphs by weighted substructure mining. *In*: Proceedings of the 23rd International conference on Machine learning. ACM, 2006.

[17] Borgwardt., Karsten M., Nicol N., Schraudolph.,Svn Viswanathan. (2006). Fast computation of graph kernels. *In*: Advances in neural information processing systems, p. 1449-1456.

[18] Saigo., Hiroto., Sebastian Nowozin., Tadashi Kadowaki., Taku Kudo., Koji Tsuda. (2009). gBoost: a mathematical programming approach to graph classification and regression. *Machine Learning* 75, 1, 69-89.

[19] Zhu., Xiaojin., Zoubin Ghahramani., John Lafferty. (2003). Semi-supervised learning using gaussian fields and harmonic functions. *In:* ICML, 3, p. 912-919.

[20] Zhou., Dengyong., Olivier Bousquet., Thomas Navin Lal., Jason Weston., Bernhard Schölkopf. (2004). Learning with local and global consistency. Advances in neural information processing systems 16, 16, 321-328.

[21] Kondor., Risi Imre., John Lafferty., Diffusion kernels on graphs and other discrete input spaces. In ICML, 2, p. 315-322. 2002.

[22] Tsuda., Koji., Hyunjung Shin., Bernhard Schölkopf. (2005). Fast protein classification with multiple networks. Bioinformatics 21, no. suppl 2. ii59-ii65.

[23] Wang., Haixun., Aggarwal, Charu C., (2010). A survey of algorithms for keyword search on graph data." In Managing and Mining Graph Data, p. 249-273. Springer US.

[24] Kacholia., Varun., Pandit, Shashank., Chakrabarti, Soumen., Sudarshan, S., Rushi Desai., Karambelkar, Hrishikesh . (2005). Bidirectional expansion for keyword search on graph databases. *In*: Proceedings of the 31st International conference on Very

large data bases, p. 505-516. VLDB Endowment.

[25] Bhalotia., Gaurav., Arvind Hulgeri., Charuta Nakhe., Soumen Chakrabarti., Shashank Sudarshan. (2002). Keyword searching and browsing in databases using BANKS. *In*: Data Engineering. *In*: Proceedings. 18th International Conference on, p. 431-440. IEEE.

[26] He., Hao., Haixun Wang., Jun Yang., Philip, S,. Yu. (2007). BLINKS: ranked keyword searches on graphs. *In*: Proceedings of the 2007 ACM SIGMOD International Conference on Management of data, p. 305-316. ACM.

[27] Balmin., Andrey., Vagelis Hristidis., Yannis Papakonstantinou. (2004). Objectrank: Authority-based keyword search in databases. *In*: Proceedings of the Thirtieth international conference on Very large data bases-30, p. 564-575. VLDB Endowment.

[28] Hristidis., Vagelis., Yao Wu., Louiqa Raschid. (2013). Efficient Ranking on Entity Graphs with Personalized Relationships. 1-1.

[29] Aggarwal., Charu, C., Haixun Wang., eds. (2010). Managing and mining graph data. 40. Springer.

[30] Mozer., Michael, C. (2003). Optimizing classifier performance via an approximation to the Wilcoxon-Mann-Whitney statistic.

[31] Tong, Hanghang., Faloutsos, Christos., Pan, Jia-Yu .(2006). Fast random walk with restart and its applications.

[32] Page, Lawrence., Brin, Sergey., Motwani, Rajeev., Winograd, Terry. (1999). The PageRank citation ranking: bringing order to the web.

[33] Minkov., Einat., William, W,. Cohen. (2007). Learning to rank typed graph walks: Local and global approaches. *In*: Proceedings of the 9th WebKDD and 1st SNA-KDD 2007 workshop on Web mining and social network analysis, p. 1-8. ACM.

[34] Li., Cuiping., Jiawei Han., Guoming He., Xin Jin., Yizhou Sun., Yintao Yu., Tianyi Wu. (2010). Fast computation of simrank for static and dynamic information networks. *In*: Proceedings of the 13th International Conference on Extending Database Technology, p. 465-476. ACM.

[35] Yin., Zhijun., Manish Gupta., Tim Weninger., Jiawei Han. (2010). A unified framework for link recommendation using random walks. In Advances in Social Networks Analysis and Mining (ASONAM), 2010 International Conference on, p. 152-159. IEEE.

[36] Backstrom., Lars., Jure Leskovec. (2011). Supervised random walks: predicting and recommending links in social networks. *In*: Proceedings of the fourth ACM international conference on Web search and data mining, p. 635-644. ACM.

[37] Abbasi, Alireza., Hossain, Liaquat., Leydesdorff, Loet. (2012). Betweenness centrality as a driver of preferential attachment in the evolution of research collaboration networks. *Journal of Informetrics* 6 (3), 403-412.