**CHAPTER 1**

**INTRODUCTION**

In this chapter, the concept of Congruent Networks is discussed. After the introduction to the domain, the challenges and opportunities that existed during the implementation of this project are stated. We then present the problems that have been identified for this domain. For the problems that have been identified, we have selected a few problems for our implementation of this project. We then describe all the problems for which we have attempted to provide a solution.

* 1. **Introduction to Congruent Networks**

According to the fundamental theory of Social behavior of humans, we create our own social network. We form a network of our ‘friends’. A ‘friend’ does not mean a classmate or a colleague. A ‘friend’ in this context, means a person who is involved with us in some social behavior. A social link can be established based on various parameters such as hobbies, folksonomy, profile, views etc. These networks have a lifetime, which is usually long, and exists till we have at least one friend. As our social behavior changes, the friends in our network will change, and this will result in social networks being dynamic. A network will be strong when the right ‘friends’ are connected. This is possible only in a context where the friends are tied together, stronger in a circumstance to perform a task.

The force that ties us stronger in a network can be called as “Congruence” and such networks can be called as “Congruent Networks”. The general definition of congruence is: identical in form. in agreement or harmony. A feeling of inherent trust comes from someone who is congruent. Whether intentional or not, congruence often attracts one to another person. People seek friends who conduct themselves congruently in their presence. Students get inspiration from teachers who are congruent with them.

* 1. **Egocentric Congruent Networks**

Egocentrism is found in all people, but the extent of ego can vary from person to person, and it also depends on the context. Egocentric people see themselves and their own interests or opinions as most important. Decisions that are taken based on one’s own information, is regarded to be more important than the decisions that are taken with other people’s information as well.

Although much social network analysis attempts to examine complete, large scale networks (ones in which every connection between network members is visible), we consider a more

restrictive view of using egocentric network analysis technique, which examines only people’s immediate neighbors and associated interconnections. This helps us to learn about how individuals correspond with their own social networks.

An ego network is a mini-network or immediate neighborhood surrounding ego that can, perhaps, reveal something important about the social world from the ego’s perspective. We might guess, for example, that if the social units are persons, those in the ego network are the ones that have the greatest impact on ego’s attitudes, norms, values, goals and perceptions of the world. Moreover, those are the ones to whom ego must turn to seek information, help and support. In some sense, an ego network is a map of ego’s personal social world. It shows something about how that ego is tied in to the larger human society.

* 1. **Challenges and Opportunities**

The profile of a user can keep changing as the user’s social actions changes. Some social networking sites are providing services which keeps the users to stick onto the sites. The actions made by users on the social network sites indicate their presence in the online world. Nowadays, users are trying to increase their Social Index**,** which means that their availability on social networking sites is more frequent and they like to be connected socially**.**

Current electronic social networks provide very few services that utilize the Social Index of the user effectively. They are mainly targeting the users as commercial entities, with a lot of targeted advertisements. These services, to the extent are providing one way benefits to the commercial users (with advertisements). Many social networking sites are available today in the market, each providing a particular set of services. Thus there is a need to create some kind of a network which is capable of providing the aggregated and combined services, with mutual benefits to the users (regular) as well as the commercial entities. The current networking sites are ineffective in capturing **“contexts”**. A context is some circumstance which provides mutual benefits to the stakeholders.

Different people perceive ego in different ways. Some people are more egocentric, and are more worried about what benefits they reap, whereas some people are more sociocentric, and are more concerned about the society as a whole, and not just themselves. The main challenge is the way

in which we are able to capture this ego, and how we match the ego of one person with that of another person. The other main challenge is to generate the egocentric congruent network for the context that is given by the user.

The user may not be in a position to express the context as a single word or a phrase. It is easier for the user to provide a simple description of the context, and let us decide on what context the user is in. Thus the challenge here is to provide a natural language interface to the user, through which the user can describe his context. If the context of the user is already known to us, we can also recommend some possible contexts, out of which the user can select his context. Through the same interface, the user can also add in some properties to the context of the user.

The next challenge here is to measure the degree of an egocentric congruent network is the extent to which the nodes in the network, are congruent with each other. The information of each user is expressed as his profile, and based on the profiles of the users in the congruent network, the degree of the egocentric congruent is calculated. This degree is a measure of how closely related are the people in the network.

The other challenge here is to measure the stability of the egocentric congruent network is a measure of the number of nodes in the network that may not be congruent throughout the entire lifetime of the context. Each node in the network also has a lifetime, and the node should exist in that context, throughout the lifetime of the context. The number of nodes that leave or join the congruent network, or are becoming incongruent with the context, is a measure of the stability of the congruent network.

* 1. **Problems Identified**

After analyzing the introduction to congruent networks, and the challenges and opportunities that existed during the implementation of this project, we list a few problems which we have indentified:

* + - Creation of Egocentric Networks
    - Handling the perception of ego by different people
    - Creation of context
    - Capturing the ego of a person
    - Matching the ego between people
    - Matching the ego of a person with the context in which he is in
    - Creation of Egocentric Congruent Networks for the given context
    - Measuring the degree and stability of the Egocentric Congruent Network
  1. **Applications**

Some areas where egocentric congruent networks are applicable are mentioned below:

* + - Matrimonial services
    - Human relationship management
    - Team formation
    - Team productivity
    - Friend suggestion

**CHAPTER 2**

**PROBLEM DEFINITION**

This project, as depicted by its title, mainly focuses on the creation of egocentric congruent networks and measurement of the degree and stability of the same. Since congruent networks are formed by using some existing social networks, they are widely applicable in all the areas that can make use of the social networks for any purpose. But, this purpose has to be beneficial mutually to all the stakeholders involved. The implementation can be used for finding the congruent people to perform a particular task. Consider the situation of suggesting a friend to a person. In this case, we use electronic social networks to understand the behavior of that person, and recommend people to him or her, people who match their behavior, as a friend. Then the challenge of this project would be to find the people who may really be his or her friend.

**2.1** **Creation of Egocentric networks**

Once the social network data is collected and stored, egocentric networks are generated for each of the individual people in the network. This is done based on the profile information of each of the users. Based on the information of all the users, we generate pools of data, which contains all the information of the users in the network. The profile of each user is in the form of attribute-value pairs. In the profile of each user, some attributes are identified as egocentric attributes. For a particular user, we identify which attributes of the user, are egocentric, and match them with the attributes in the pools, to identify similar ego attributes. Based on this matching, we can create two levels of egocentric networks. The first level egocentric network consists of users who have the same attribute value, whereas the second level egocentric network consists of users who have similar attribute values, and this can be based on a threshold value. The first level egocentric network is given a higher preference than the second level egocentric network. The same is done for all the users, and we generate egocentric networks for all the users.

**2.2 Handling the perception of ego by different people**

Different people perceive ego in different ways. Some people who are more egocentric by nature, are more worried about what benefits they get, whereas some people are more sociocentric, and are more concerned about the entire society as a whole, and not just about their personal benefits. The main challenge is the way in which we are able to capture this ego, and how we match the ego of one person with that of another person. To do this, we identify some attributes of the user profile as egocentric, which contribute to the ego of a person, and some attributes as non egocentric, which do not contribute to the ego.

**2.3 Creation of context**

The user can use an existing context or he can create a new context. The user can enter a brief description of the context we is willing to choose and he can create a new context. Each context is also associated with certain properties. The user can also add the properties to the context. Based on the context and its properties, the further matching is done.

**2.4 Recommendation of context**

It is also possible that the user is not able to express the context in which he is in, as a single term or phrase. Hence the user can write a small description of his context, and we recommend him a term or phrase for his context. The user can choose this context, or can give a term or phrase of his own. The same is also done for the properties of that context as well.

**2.5 Capturing the ego of a person**

It is difficult to capture the ego of a person. The level of ego varies from person to person, depending on their behavior. Each user profile consists of attributes, and some of these are identified as egocentric attributes. Some attributes that mainly contribute to the ego of a person include age and gender. It is commonly found that a person is congruent with another person of the same age, or a slight deviation from his or her age. It is also commonly found that people are more congruent with other people of the same gender. The reason behind this can also be the difference in ego between the two people.

**2.6 Matching the ego between people**

Once the attributes that contribute to the ego of a person are identified, ego matching is done, to check if the two people are congruent with each other or not. People with similar ego often are more congruent than people with dissimilar ego. Thus we can eliminate some people whose ego does not match, thereby creating a stronger egocentric congruent network.

**2.7 Creation of Egocentric Congruent Networks for the given context**

Once the egocentric networks are created, the user enters the context in which he is in. For this context, we will generate an egocentric congruent network, by considering the ego of the person as well as the given context. This egocentric congruent network will consist of the people who belong to that context and are congruent with each other.

**2.8 Measuring the degree and stability of the Egocentric Congruent Network**

Once the egocentric congruent network is generated, it is then checked for its degree and stability. The degree of the egocentric congruent network will tell us as to how congruent the people in the network are, with each other. The stability of the egocentric congruent network tells us to what extent the congruent network changes throughout the lifetime of the context. This tells us how stable and strong the network is.

**CHAPTER 3**

**LITERATURE SURVEY**

**3.1 Survey experience**

3.1.1 Social networks

A social networking service is a platform to build [social networks](http://en.wikipedia.org/wiki/Social_network) or [social relations](http://en.wikipedia.org/wiki/Social_relation) among people who may share interests, activities, backgrounds, or real-life connections. A social network service consists of a representation of each user (often a profile), his/her social links, and a variety of additional services. Most social network services are [web-based](http://en.wikipedia.org/wiki/Web-based) and provide means for users to interact over the [Internet](http://en.wikipedia.org/wiki/Internet), such as [e-mail](http://en.wikipedia.org/wiki/E-mail) and [instant messaging](http://en.wikipedia.org/wiki/Instant_messaging). [Online community](http://en.wikipedia.org/wiki/Online_community) services are sometimes considered as a social network service, though in a broader sense, social network service usually means an individual-centered service whereas [online community](http://en.wikipedia.org/wiki/Online_community) services are group-centered. Social networking sites allow users to share ideas, pictures, posts, activities, events, and interests with people in their network.

Web-based social networking services make it possible to connect people who share interests and activities across political, economic, and geographic borders. Several websites are beginning to tap into the power of the social networking model for [philanthropy](http://en.wikipedia.org/wiki/Philanthropy). They provide a means for connecting otherwise fragmented industries and small organizations without the resources to reach a broader audience with interested users. Social networks are providing a different way for individuals to communicate digitally. These communities of hypertexts allow for the sharing of information and ideas, an old concept placed in a digital environment.

The advent of [social networking](http://en.wikipedia.org/wiki/Social_networking) platforms may also be impacting the way(s) in which learners engage with technology in general. The use of online social networks by school libraries is also increasingly prevalent and they are being used to communicate with potential library users, as well as extending the services provided by individual school libraries. Professional use of social networking services refers to the employment of a network site to connect with other professionals within a given field of interest. Social networking services like [LinkedIn](http://en.wikipedia.org/wiki/LinkedIn) geared towards companies and industry professionals looking to make new business contacts or keep in touch with previous co-workers, affiliates, and clients. Other network sites are now being used in this manner, [Twitter](http://en.wikipedia.org/wiki/Twitter) has become a mainstay for professional development as well as promotion and online Social networking services support both the maintenance of existing social ties and the formation of new connections.

Much of the early research on online communities assume that individuals using these systems would be connecting with others outside their preexisting social group or location, liberating them to form communities around shared interests, as opposed to shared geography. Other researchers have suggested that the professional use of network sites produce “[social capital](http://en.wikipedia.org/wiki/Social_capital).” For individuals, social capital allows a person to draw on resources from other members of the networks to which he or she belongs. These resources can take the form of useful information, personal relationships, or the capacity to organize groups. As well, networks within these services also can be established or built by joining special interest groups that others have made, or creating one and asking others to join.

Social networks exhibit strikingly systematic patterns across a wide range of human contexts. Although genetic variation accounts for a significant portion of the variation in many complex social behaviors, the heritability of egocentric social network attributes is unknown.

3.1.2 Social network analysis

The roots of social network analysis are found in the mathematical study of graph theory (such as the work of Erdos, Harary and Rappaport) and empirical studies of social psychology (Bott, Heider and Moreno). While the former group were charting various axioms between abstract nodes and lines, the latter found nodes and lines to be a sensible way to map concrete relationships between individuals. As the field matured in the latter half of the twentieth century these two groups converged on a series of metrics and methods to tease out underlying structures from complex empirical phenomena.

Social network analysis is a rapidly expanding interdisciplinary paradigm, much of which is taking place with online data. Social network analysis has emerged in the past half-century as a compelling complement to the standard toolkit of social science researchers. At its foundation is a belief that explanations for social organization are not to be found in innate drives or abstract forces. Instead we can look to the structure of relationships that constrain and enable interaction alongside the behaviors of agents that reproduce and alter these structures. Social network analysis is particularly well suited to understanding online interaction. There are two key facts about online interaction that make it particularly amenable to social network analysis - the nature of online interaction and the nature of digital information.

We can learn a lot about people from how they talk, to whether they talk, to sociable, well connected people or to unconnected individuals. Online communication is an increasingly important way to get information from, and keep in contact with, each other. Communication through a computer can be recorded and then analyzed.

Network analysis began to mature in the 1970s. In 1969, Stanley Milgram published his Small World experiment, demonstrating the “six degrees of separation”. In 1973, Mark Granovetter’s published the landmark “The Strength of Weak Ties” which showed empirically and theoretically how the logic of relationship formation led to clusters of individuals with common knowledge and important ’weak tie’ links between these clusters. The following two decades saw explosive growth in the number of studies that either alluded to or directly employed network analysis.

One of the most basic notions governing the structure of social networks is homophily, the principle that we tend to be similar to our friends. (or the tendency for people to choose relationships with people who have similar attributes). Homophily provides a fundamental illustration of how a network’s surrounding contexts can drive the formation of its links. Nodes are connected in the network depending on the several characteristics, these can be listed as follows

* Nodes in the network are similar to you along racial and ethnic dimensions
* similar in age
* similar in characteristics that are more or less mutable
* places they live
* occupations
* their levels of affluence
* their interests
* beliefs,
* opinions
* relationship

3.1.3 Communities of Interest

Communities of interest are groups of people who share a common identity, or a group of people who share a common experience. Hence people who share some common interests can form a community of interest.  Participation in a community of interest can be compelling, entertaining and create a ‘sticky’ community where people return frequently and remain for extended periods.

3.1.4 Communities of Purpose

A community of purpose is a [community](http://en.wikipedia.org/wiki/Community) of people who are going through the same process or are trying to reach a similar goal. Such communities serve a functional purpose, smoothing the path of the member for a limited period surrounding a given activity. Members of the community assist each other by sharing experiences, suggesting strategies and exchanging information on the process in hand.

3.1.5 Connection maps by LinkedIn - InMaps

InMaps are used to visualize how your network looks like. It would be interesting to find out whether one’s connections form clusters or groups. It would be nice if one could see the way all connections are related to each other and also be able to identify the elusive hubs between one’s professional worlds. InMaps are capable of doing this.

3.1.6 Ego networks

Ego networks have a simple but constrained structure, which makes it simple for us to collect data, but we will have fewer powerful tools with which to analyze the data. Techniques that have been used to analyze ego networks are usually based on density, connectivity or the attributes of the neighboring nodes, or combinations of all the three.

3.1.7 Ego network analysis

Ego network analysis focuses on network characteristics such as size, relationship strength, density, [centrality](http://en.wikipedia.org/wiki/Centrality), [prestige](http://en.wiktionary.org/wiki/prestige) and roles such as [isolates, liaisons](http://en.wikipedia.org/wiki/Isolates), and [bridges](http://en.wikipedia.org/wiki/Bridge_(interpersonal)). Such analyses, are most commonly used in the fields of [psychology](http://en.wikipedia.org/wiki/Psychology) or [social psychology](http://en.wikipedia.org/wiki/Social_psychology_(sociology)), [ethnographic](http://en.wikipedia.org/wiki/Ethnographic) [kinship](http://en.wikipedia.org/wiki/Kinship) analysis or other [genealogical](http://en.wikipedia.org/wiki/Genealogy) studies of relationships between individuals.

3.1.8 Interest graphs

An interest graph is an online representation of the specific and varied things in which an individual is interested. Interest graphs have perceived utility and value because of the premise that people’s interests are a major aspect of who they are, forming part of their personal identity, and can be used as indicators of such things as what they might want to do or buy, where they might want to go, or who they might want to meet, follow or vote for.

3.1.9 Network visualization techniques

Visual representation of social networks is important to understand the network data and convey the result of the analysis. Many of the analytic software have modules for network visualization. Exploration of the data is done through displaying nodes and ties in various layouts, and attributing colors, size and other advanced properties to nodes. Visual representations of networks may be a powerful method for conveying complex information, but care should be taken in interpreting node and graph properties from visual displays alone, as they may misrepresent structural properties better captured through quantitative analyses.

3.1.10 Collection of user profile information from existing social networks

The main focus of this project is to identify the users of a social network who are closely related to each other in some context. This would mean that the information of every user in the social network has to be collected. This is the first step in order to generate a congruent network of people. For this project, the information of users has been collected from Facebook using the Graph API. The Graph API is the primary way to collect data from the [social graph](https://developers.facebook.com/docs/opengraph/) of Facebook. It is a low-level HTTP-based API that can be used to query data and post new stories.

In order to collect a user’s information from Facebook using the Graph API, the access token of the user has to be obtained. The access token is a random string that identifies a user and provides information about granted permissions. Once the user access token is obtained, we can collect the information of the user, as well as the information of the user’s friends on Facebook, using a HTTP request.

**3.2 Focused literature survey**

A social network is the connection of people by a computer network, and internet will be the larger social network. The main point is that a social network is composed of actors and ties between them. Ties in a social network are numerous and of different natures: familial ties, lifelong friend ties, marital ties, business partner ties, that are important for people to obtain the fundamentals of identity, affection, emotional and material support, i.e. the recognition of their existence by others. However, the commitment of individuals is superficial, limited to the reasons of the local interaction. As a consequence, ties in a social network are "socially-oriented" like in the real life, trust does not play an important role, and individuals generally belong to several social networks

[Social networking](http://personalweb.about.com/od/easyblogsandwebpages/ss/2007topsnsites.htm) is a way for one person to meet up with other people on the internet. That's not all though. Some people use social networking sites for meeting new friends on the internet. Other's use it to find old friends. Then there are those who use it to find people who have the same problems or interests they have.

We can learn a lot about people from who they talk to — whether they talk to sociable, well connected people or to unconnected individuals, for example. Is the person part of a large group whose members all talk to each other, or does he or she bridge social worlds? Examining such connections between people can teach us how they operate socially.

Online communication, in particular, is an increasingly important way to get information from, and keep in contact with, each other. Communication through a computer can be recorded and then analyzed; we can later use that analysis to develop systems that are more aware of how people interact online. Although much social network analysis attempts to examine complete, large scale networks, ones in which every connection between network members is visible, we take a more restrictive view. Using egocentric network analysis techniques, which examine only people’s immediate neighbors and associated interconnections, helps us learn about how individuals correspond with their social networks.

3.2.1 Functions Served by Ego Networks

Ego networks mainly serve the following purposes:

* Social support
* Emotional and material aid
* Companionship
* Information.
* Sense-making
* How to interpret the world
* Social control
* Ensuring that egos behave according to norms
* Access to resources
* Entrepreneurs draw on their contacts to get the clients and employees and consultants they need
* Behavioral models
* You tend to talk like the people you talk to
* You become aware of choices from the people in your ego network

By an egocentric network, we mean one that represents data about the relationships of a single person in respect to others.

* + 1. Social network analysis

Social network analysis offers a powerful framework for detecting and interpreting social relationships online. They are accompanied by a host of analytic techniques ranging from simple centrality scores to sophisticated multilevel modeling. Yet gathering these networks is a time-intensive and challenging task. Online networks make this task somewhat easier through the use of passive networks (such as email stores and web pages), but the increase in efficiency leads to additional challenges about when to stop collecting, and what sorts of relations are substantively meaningful. Overcoming these challenges takes patience, a good dose of technical skills with scripting languages or custom software and some trial and error. In return the results, as seen by many of the aforementioned studies, can inform our understanding of the interpersonal structures that affect online participation and online life in general. Yet, the techniques are relevant beyond the digital domain. The discovered structures mirror and are a part of everyday life. It is not merely a gaze to distant shores, but a more crystallized view to the here and now.

* + 1. Context in a social network

There is a large number of social networks. Depending on whom you are communicating with and how you want your message to be received by the recipient(s), you would choose your tool from a lot of choices. Implementing and analyzing a context is also another challenge. Not many social networks have been successfully able to bring the context into consideration.

* + 1. Ego networks

In analysis the entire network, the goal is often to describe the characteristics of the network, and ask why certain individuals occupy a particular location in the network. (E.g., why do people always reply to him? Are there multiple subgroups in this network?) By contrast, ego network analysis is comparative in nature. One examines the differences in the size, shape and quality of a number of personal networks or ego networks. These networks are commonly captured by sampling from a population. In this regard they are akin to traditional surveys as one would similarly want a representative (even stratified) random sample from a population. Each sampled case in this context is referred to as “ego”, and the nodes connected to ego are referred to as “alters”. One can either capture a star network (which is merely the ties to ego) or a full personal network (which includes the ties between alters). One can unobtrusively collect personal networks in social software sites, communication and web pages. In each case one captures a list (such as a friend list) and then checks to see who on this list is also tied to each other. Active collection of personal networks can make use of a number of pre-existing interview and survey techniques.

**CHAPTER 4**

**PROJECT REQUIREMENTS DEFINITION**

**4.1 Project Perspective**

Users get themselves registered to our system. During this process of registration, users create profiles of their own which also includes their id from the social networks. With the help of this id that is provided by the users, their data from social networks is fetched, analyzed and stored in a format called as User Description Format (UDF).

Once the user inputs the context, our system identifies the stored UDFs which satisfy the constraints and conditions of the given context. The given context has to be relevant to the user and also has to provide some kind of benefit to the user.

Once the users are registered with our system and the UDFs are generated, and the user enters a brief description about the context, the system identifies the contexts which are congruent with the UDFs and recommends the same to users. The user can either accept it or reject it. These contexts which are accepted or rejected are stored along with UDF and they can be reused.

When the users input contexts to our system, the UDF may not have the necessary attributes. In such situations, the user is also responsible to enter the attributes that are required for their contexts. We recommend these contexts to those users who are congruent with some of the contexts defined by this user. If the users have recently registered with our system, then we recommend these contexts to the people who are congruent with the contexts which have some of the attributes of newly defining context.

Once the UDF is defined and updated with the contexts, generating a congruent network for some given context can be simple as selecting those UDFs that have the given context with them. This is only possible for the predefined contexts. For the newly defined contexts, all UDFs will have to be analyzed and UDFs having the required attributes are selected.

Once the UDFs are refined, the congruence weight is calculated for each UDF. The UDFs which have a congruence weight greater than a threshold (defined by user) are identified to be as congruent.

**4.2 Project Functions**

The major facilities that are provided by our system are the following:

* Creation of Egocentric Networks
* Handling the perception of ego by different people
* Creation of context
* Capturing the ego of a person
* Matching the ego between people
* Matching the ego of a person with the context in which he is in
* Creation of Egocentric Congruent Networks for the given context
* Measuring the degree and stability of the Egocentric Congruent Network

**4.3 User Classes and Characteristics**

The users of this system will be able to register themselves, and thereby create a profile of their own, according to the UDF. The user can also create a context, by entering a brief description about the context, and then can choose the contexts from those that are recommended to the user, or the user can give a term for the context. Similarly, the user can also add properties to the contexts.

* 1. **Operating Environment**

4.4.1 Hardware requirements

* Processor : Intel Core i3 and above
* RAM : 2GB and above
* Hard disk : 20GB and above

4.4.2 Software requirements

* Operating System : Windows or linux
* Database : MongoDB
* Language : Python 2.7
* Access to different social networking sites
* Python packages : Pymongo, NLTK and BeautifulSoup
* Browser : Firefox 20.0.1 and above

**4.5 Design and Implementation Constraints**

All users must be connected with the system in all the appropriate social networks. In this project, Facebook, Twitter and LinkedIn are used as the primary social networks. Social connections are treated differently by different social networks. Facebook interprets people as friends whereas Twitter interprets as followers and LinkedIn as connections. Hence, the user must be a friend of our system in Facebook, a follower on Twitter and a connection on LinkedIn.

**4.6 User Documentation**

For this project to be effective, users should be able to express their requirements as contexts. The system will provide users with guidelines in order to convert the requirements into corresponding contexts. As contexts are expressed in terms of attributes, sample contexts defined with attributes will be provided.

**4.7 Assumptions and Dependencies**

Users will portray themselves differently in different social networks. This portrayal of the user will depend on the purpose that is intended and also on the benefits that can be reaped from the various social networks. Here representation is the quantity of information which the user is willing to show to others who are in the same social network.

Since the users can have different levels of benefits from this system, the strength of their connection with the system will also vary. Based on the strength of the connection the users will have, recommendations can be made. Recommendations, once done, will have to be beneficial to the users. As recommendations can be beneficial, we assume that the users will accept the recommendations made. The decision to accept or reject the recommendations, solely depends on the choice of the users.

**CHAPTER 5**

**SOFTWARE REQUIREMENTS SPECIFICATION**

**5.1 External Interface Requirements**

5.1.1 User Interfaces

* Interface for creating new contexts or using existing contexts
* Interface for registering new users into the system
* Interface for changing the profile of a user
* Interface for viewing generated statistics

5.1.2 Software Interfaces

* Operating system : 64–bit linux 2.4 or above
* Database : MongoDB
* Language : JyThon (java Python)
* PyMongo package for python – this also requires nose, setuptools and sphinx packages for python
* NLTK package for python
* BeautifulSoup package for python
* Web 2.0 standards

**5.2 Functional Requirements**

5.2.1 Generation of egocentric networks, for each node

* For individual nodes, egocentric networks are generated, taking into consideration only the attributes of that particular node.
* Each ego network can be visualized as a star network.
* The attributes of the central node is called as distinguishing attributes of the egocentric network in that social network.

5.2.2 Elimination of egocentric networks based on the required context

* Some of the egocentric networks that do not relate with the required context are eliminated.
* This is done by matching the attributes of the context, that are derived from the context tree, and the distinguishing attributes of the individual egocentric networks.

5.2.3 Reduction in the number of nodes of each egocentric network, based on the context

* The number of nodes in the remaining egocentric networks is reduced.
* This is done by matching the attributes of the individual nodes with that of the context, and this gives us a weight.
* Those nodes whose weights are below a threshold are eliminated.
* This results in context specific egocentric networks.

5.2.4 Generation of egocentric congruent network

* From the remaining ego networks that belong to that context, all the nodes in all the networks are merged to generate the egocentric congruent network.
* Else, the most suitable context specific egocentric network can be suggested as egocentric congruent network.

5.2.5 Measurement of the degree and stability of the egocentric congruent network

* Once the egocentric congruent networks are generated, the degree and stability of the network is measured in order to understand the characteristics of the network.
* The degree can be measured by determining the level of congruence between the nodes in the network.
* The stability of the network is measured by determining the changes that happen in the network in the lifetime of the context.
* Higher the variations that happen in the network, lower is the stability of the network.

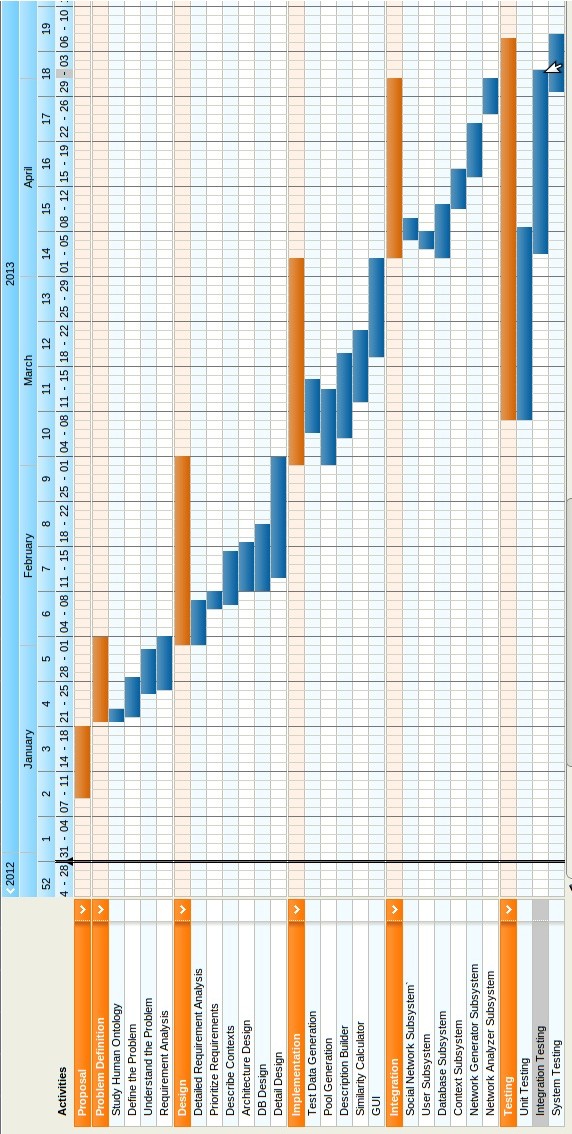
**5.3 Non functional requirements**

Since this project is web based and it involves various social networking sites, the system has to meet many non functional requirements. But, due to the limitation of time and infrastructure, our implementation has been restricted to meet the following non functional requirements:

* Implementation is capable of handling up to 10,000 user profiles.
* Should support at least 100 simultaneous users.
* Should be capable of handling at least 100 to 1000 contexts.
* The response time for generation of the egocentric congruent network should be less than 3 minutes.
* Profiles that have been changed by the users, should be updated within a time of 24 hours.

**CHAPTER 6**

**GANTT CHART**

****

***Fig. 6.1*** *Gantt chart*

**CHAPTER 7**

**SYSTEM DESIGN**

**7.1 Block Diagram**

The system is divided into the following subsystems:

* Social Network Subsystem
* User Subsystem
* Database Subsystem
* Context Subsystem
* Network Generator Subsystem
* Network Analyzer Subsystem

Social Network

User

UDF

Context

Network Generator

Network Analyzer

***Fig 7.1*** *Block diagram of the system*

7.1.1 Social Network Subsystem

The social network subsystem represents the social networking sites from which the user profile information is collected. A person will interact with other people by performing certain social actions on such social networking sites

7.1.2 User Subsystem

The user subsystem will represent the users of this system. The user will interact with our system by creating their own profiles which will be monitored by our system.

7.1.3 Database Subsystem

The database subsystem will store all the profile information of all the users in the User Description Format (UDF) and also all the information regarding the contexts, and its properties will be stored in the database.

7.1.4 Context Subsystem

The context subsystem represents the context that the user is in. The context will also consist of a list of properties that will describe that context in a meaningful manner.

7.1.5 Network Generator Subsystem

The network generator subsystem will generate the egocentric congruent network, by considering the egocentric attributes of the user who is considered. Once the egocentric congruent network is generated, the network analyzer will take the network as input.

7.1.6 Network Analyzer Subsystem

The network analyzer subsystem will take the generated egocentric congruent network as input, and it will analyze the network for its degree and stability. Once they are measured, the results will be displayed back to the user.

**7.2 Architecture**

As mentioned earlier, it is mandatory for social networks to exist before congruent networks are generated. With these social networks, context matching is done. Only the nodes that match the given context are retained and remaining nodes are discarded. In order to do this, clustering is done with the attributes of the context. Nodes that are closer in the “context space” are retained. Further, with the remaining nodes, congruence value for each of the nodes with the given context is calculated. Nodes having a congruence factor greater than a threshold are considered congruent with the context and the others are not.

User Profile Data

Congruence value calculator

Data Extractor

Context matcher

Egocentric congruent network generator

Degree and stability checker

***Fig 7.2*** *Architecture for the proposed system*

The major phases to generate egocentric congruent networks are as follows:

* Analyzing and optimizing social networks
* Defining the context
* Context matching
* Calculating the degree of congruence
* Generating congruent networks
* Measuring the degree and stability of the network

7.2.1 Analyzing and optimizing social networks

There are many social networking sites where users share some information which they like with their individual networks or friends. Users are generally identified by their profiles. User actions are analyzed and recorded in a format that is called as User description format or UDF. UDF, along with the result of analysis, also contains profile data. It is important to consider that the user knows what information of theirs can be shared and it is completely under their discretion. Forming a congruent network does not necessarily mean that information will be shared. Sharing is considered to be an action where information of the user is made available to others based on the individual's choice. Only those attributes that are necessary for the creation of congruent networks by maintaining privacy are considered. The focus of this model is to generate a network of congruent people, without actually sharing their data. Only the relevant information that will assist the creation of congruent network in a particular context, is analyzed and/or created and then utilized.

The social network data that is collected may have redundant instances of users, their followers or similar users. The UDF takes care of the redundant data, and represents it as a numerical value after collecting and analyzing the social network data. Consider Twitter as an example. Instead of storing all the tweets of a user, it is better to keep only the most recent tweets and keep the classification/domain count for all the remaining tweets. Classification/domain count of a context is a number which indicates the count of actions related to the context/domain. This demands that, it is necessary to identify the related contexts/domain for each of the users and the necessary services have to be provided for enabling the change of contexts/domains. The content of UDF is dynamic in nature, thereby it can keep getting updated.

7.2.2 Defining the Context**s**

Contexts are defined by properties. A property is an attribute that is associated with both the user as well as the context. Attributes can be either quantitative or qualitative. Quantitative attributes can be measured whereas qualitative attributes are not directly measurable, but they can be derived through observation and analysis. A context is defined as a combination of these attributes with a minimum threshold value for each of the attributes. Properties of the context and the attributes of the user need not be the same. It is possible to match the property of a context with the attributes of a user. Property of a context can be defined as a combination of attributes. Consider the following example. It is required to form a team in order to organize a social awareness program. The context can now be defined with the property “Social participation” and with the attributes as ‘Interested in social work’, ‘participated in group activity’ and ‘organized a community development program’. Expressing the context property in terms of user attributes is done by identifying the actions on social networking sites which are recognizable, recordable and related to that context. For quantitative properties, they are directly expressed as the ‘necessary value’. Context can be generic or specific.

7.2.2.1 Generic Context

Generic context is a context having various sub contexts under it. These contexts are usually applicable to all the users. “Social Context” is one of the examples of generic contexts. Generic contexts are defined with multiple properties each of which covers multiple attributes. Generic context takes each attribute as equally important. One optimization that can be implemented is to consider only attributes that cover sub contexts.

7.2.2.2 Specific Context

Specific context has well defined properties and cannot be decomposed further. Specific context usually captures single purpose.

7.2.3 Context Matching

Once the UDF is defined for all the users and contexts are expressed in terms of attributes, context matching is more or less similar to filtering the UDFs for the contexts. For contexts with only quantitative attributes, matching can be as simple as an SQL query. For contexts that have mixed attributes, matching considers the result of analysis of social actions and other related attributes. Only the nodes/users satisfying these constraints are retained for the further process. Context Matching is a kind of clustering where clustering is made on attributes covering the properties of contexts. The challenges here are i) multiple attributes ii) the order of attributes. One possible way to handle the above challenges is to perform clustering on distinguishing important attributes and let the next phase to filter out further.

7.2.4 Calculating the degree of congruence

This is the important stage where congruent networks differ from other electronic social networks. In the previous stage we filter the nodes which do not satisfy the constraints of context. Here, we consider the user as “the ultimate decision maker” and follow reverse direction to what was explained in the previous stage. No matter how important the context is, it is the user who finally determines the usefulness of the context. This stage will assist the user to determine that importance. Unless the nodes are present at centre of the “context space”, where they are already congruent, congruence with the context depends on users. The degree of congruence can now be calculated in two ways:

* Collect inputs from the user and do binary classification for classifying him/her as congruent or non congruent.
* Let the user himself/herself agree or disagree with the context.

When all of the incongruent nodes are filtered out, it is important to give weights to the remaining nodes with respect to the context. The congruent weight can be calculated as, per node basis or as per attribute basis. This decision, again, depends on the context and availability of attributes. For a node, the degree of congruence in the form of weight is calculated as the cumulative sum of all the attributes. For attribute congruence, the calculation is done for each of the individual attributes.

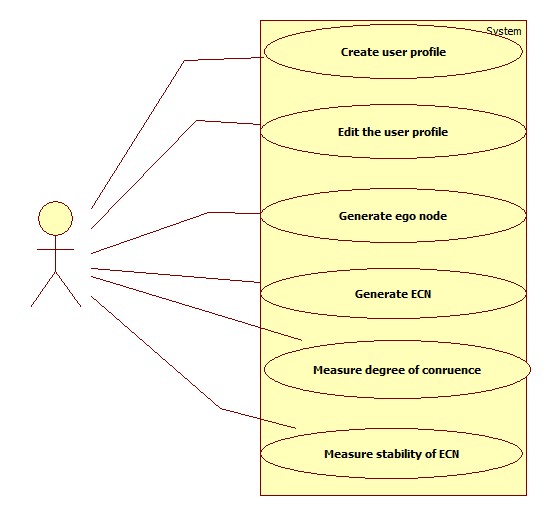
7.2.5 Generation of congruent networks

All the nodes remaining after the previous stage will be congruent with the context. The context is added to all the nodes in the ‘Congruence’ field in their UDF. Adding the context in the nodes provides the advantage of reuse. When the same context is used multiple times, we can directly fetch the congruent nodes without repeating the process, unless new nodes are added due to the dynamic nature of social networks. The issue with this approach is to identify when the node shall become incongruent with that context. This can be solved by monitoring the health of each node, at regular intervals, and update the ‘congruent contexts’ accordingly.

7.2.6 Measuring the degree and stability of the network

Once the congruent network has been generated, the next stage will determine the degree and stability of the network. The degree is a measure of how congruent the nodes in the network are, with each other. By matching the attributes of the nodes in the congruent network, the system can determine the degree of the congruent network. The stability of the network is a measure of the number of changes that happen in that network within the lifetime of the context. If the number of nodes, that have become incongruent in the same context, is large or the variations that happen within the network is more, the network is said to be unstable and hence its stability would be less. Higher the stability of the network, more congruent are the nodes of the network, with each other.

**7.3 Use case diagram**

****

***Fig 7.3*** *Use case Diagram for the proposed system*

The user of this system will be able to perform the above mentioned functions. The user is able to register with the system, by creating a profile. This profile of the user can be edited by the user at some later point of time. The user will also be able to create an ego node, which will represent him in the network. The user can also create contexts and determine the egocentric congruent network for that context. The system also lets the user to measure the degree and stability of the generated congruent network.

**CHAPTER 8**

**DETAILED DESIGN**

**8.1 Application architecture**

The Input 🡪 Process 🡪 Output model is followed for each of the following stages:

* Generation of ego networks
* Creation of new contexts
* Use of existing contexts
* Elimination of ego networks
* Reduction of nodes
* Generation of Egocentric Congruent Network(ECN)
* Measurement of degree and stability of ECN
  + 1. Generation of ego networks

Input Process Output

**User profile**

**Attribute matching for node**

**Ego network for each node**

***Fig 8.1.1*** *Generation of Egocentric networks*

The user profile is taken as input and attribute matching is done. From this, ego networks are generated for each of the users. The ego networks are given as the output from this.

* + 1. Creation of new contexts

Input Process Output

**Write Context Description**

**Write Property Description**

**Fetch Suitable Name**

**Update Context Base**

**Store context in database**

**Display network**

***Fig 8.1.2*** *Creation of new contexts*

The user can enter the context description and the property description of each property of the context. Once this is done, the system is able to provide a name that suits the context, and also a name for the given property description. The context name, context description, and property names and property descriptions are stored in the database and the corresponding network will be displayed to the user.

* + 1. Use of existing contexts

Input Process Output

**Prepare Description**

**Print Results**

**Find Matching Contexts**

***Fig 8.1.3*** *Use of existing contexts*

Once the user enters the context description, the system will suggest some pre existing contexts to the user. This is done by the system, by matching the context description entered by the user with the context description stored in the database. The best matching contexts are recommended to the user, and the user can select the context, if the user wishes to.

* + 1. Elimination of ego networks

Input Process Output

**Ego networks**

**Context**

**Context present or not**

**Context matching**

**Ego networks**

**( Some eliminated )**

**Score calculation**

***Fig 8.1.4*** *Elimination of ego networks*

Once the user enters the context, context matching is done to identify which contexts are relevant, and which are not. By doing this, some ego networks are eliminated and the rest are retained.

* + 1. Reduction of nodes

Input Process Output

**Ego networks**

**Reduced nodes in ego network**

**Node matching context**

**Context**

**Score calculation**

***Fig. 8.1.5*** *Reduction of nodes in egocentric networks*

From the remaining networks, the nodes that do not match the context are eliminated. Thereby, node reduction is done in the remaining ego networks. This is done based on the score that is calculated for each of the nodes in the remaining ego networks.

* + 1. Generation of Egocentric Congruent Network(ECN)

Input Process Output

**Reduced Ego networks**

**Context**

**Node merging**

**Egocentric congruent networks**

**Network merging**

***Fig.8.1.6*** *Generation of egocentric congruent networks*

After the node reduction stage is done, the remaining networks (and nodes) are merged to generate the egocentric congruent network.

* + 1. Measurement of degree and stability of ECN

Input Process Output

**Egocentric congruent network**

**timeline**

**Degree measurement**

**Statistics**

**Stability measurement**

***Fig 8.1.7*** *Measurement of degree and stability of ECN*

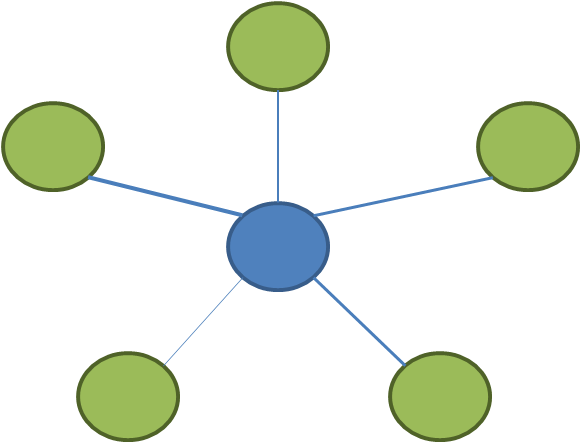
Once the egocentric congruent networks are generated, the degree to which the nodes are congruent is calculated and the stability of the ECN is also calculated. The result of these measurements will provide us the statistics that can be analyzed by us.

**8.2 Modules**

8.2.1 Generation of ego networks

Once we have the user profiles of all the users in the required UDF, the ego networks are generated for all the users by following the below mentioned steps:

* All the attributes that are static (or will not change) and all the ones that are dynamic (change often), are identified.
* Attribute matching is done for all the users in the network, with the attribute of the user, and two levels of matching can be identified. The first level indicates that the matching is exact and the second level indicates that the matching is similar but not exact.
* The ego attribute values that come under first level are given a higher preference to the ones that come under second level.
* Based on this matching, the users who match exactly are identified and the users who match similarly are identified. From this, a star network can be created for the user under consideration. This will give us the ego network for the user who is under consideration.

****

***Fig.8.2.1*** *Ego Network*

8.2.2 Context creation

Context creation is done with the help of recommendation context creation is expected to be done by context users. This is done in 2 steps:

* Naming
* Properties identification

8.2.2.1 Naming

* User enters the description of the contexts. Using this description, a term is found for that description by making a call to the reverse dictionary.
* The context description is saved in the context tree/database.
* The reverse dictionary returns n terms (usually 10) that will match the description given by the user.
* If the user agrees with the results retrieved, he uses one of the terms as the context name.
* If the user doesn’t agree with the results retrieved, he may enter his own name or the user will repeat the procedure with some different description. In such a case, the context database will be updated.

8.2.2.2 Properties identification

Context is some situation or circumstance which is useful for the user. To make it useful, right users have to be found. In order to do that, the context is expressed in terms of properties. There are two types of properties.

* Quantitative Properties
* Qualitative Properties

Quantitative Properties are directly measurable.

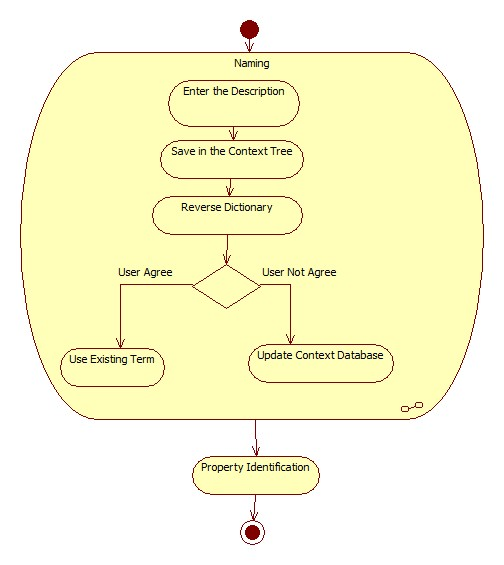
* The system will recommend the available attributes.
* User chooses one of the attribute as the property name.
* Use enters values. These values can be:
  + Tuple, List
  + Range (Numerical)
  + Single value
* Attribute, along with values, is stored in the context database.
* The same process is repeated for the required number of times.

Qualitative Properties are not directly measurable.

* User enters the description of the property.
* Using the reverse dictionary look up, the property names are suggested to the user.
* The property description is saved in the context database.
* If the user accepts any of the suggested names, it is stored in the database.
* If the user doesn’t agree with the results, the user will provide a name, or repeat the same procedure with different description, and the context database gets updated.

The following assumptions are made:

* Reverse dictionary look up can be made.
* Context database is kept updated.
* Description of the context name, along with description of attributes, is important in order to generate congruent networks.



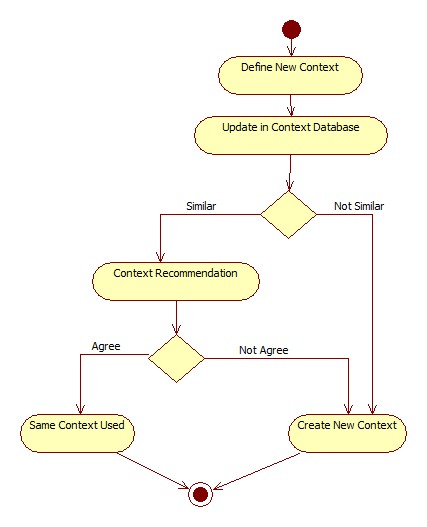
***Fig. 8.2.2*** *Activity diagram for creating of new contexts*

8.2.3 Use of existing contexts

* When a user wants to define a new context, he enters description of the context.
* This description is compared with the description of the existing contexts in the database.
* If the descriptions are found to be similar, these contexts will be recommended to the user, so that the user can select the existing context.
* Recommendation also includes properties of the context, along with their description.
* If the agrees, the same context is used. A congruent network would have already been generated, and the same congruent network will be used with some updates.

The following assumptions are made:

* Facility exists to compare a context and find similar contexts.
* Facility exists to update congruent networks.



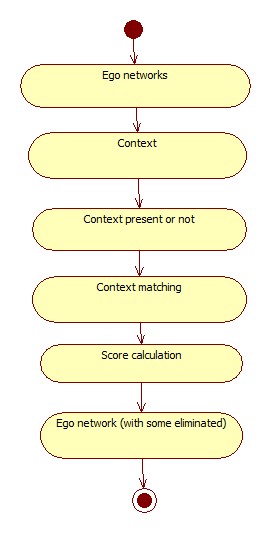
***Fig. 8.2.3*** *Activity diagram for using an existing context*

8.2.4 Elimination of ego networks

* For the given context, identify the attributes that describes that context best
* Context matching is done between the attributes of the context and the attributes of the ego node in the egocentric network
* If CA ∩ NA < Threshold, then eliminate the ego network

CA

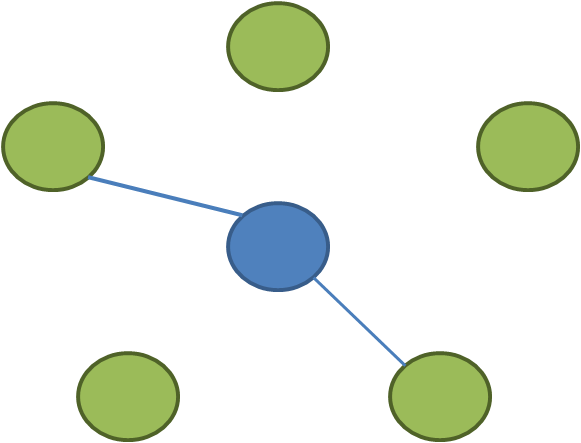
* Only the ego networks whose score is above this threshold value are considered for the next steps.



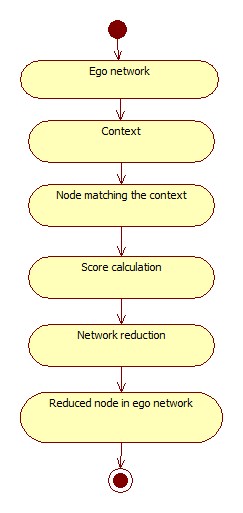
***Fig. 8.2.4*** *Activity diagram for elimination of Ego networks*

8.2.5 Reducing the number of nodes in each of the remaining egocentric network

* For each of the remaining egocentric network
* Nodes whose score is less than the threshold are eliminated and we are left with the ego centric network for the particular context.
* Hence, we derive the context specific egocentric network
* At the end of this step we would be left with certain number of context specific egocentric network



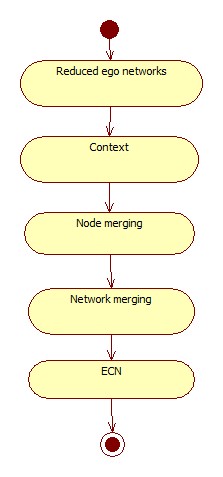
***Fig.8.2.5.1*** *Ego network with nodes reduced*



***Fig.8.2.5.2*** *Activity diagram for reduction of nodes in ego network*

8.2.6 Generation of ECN

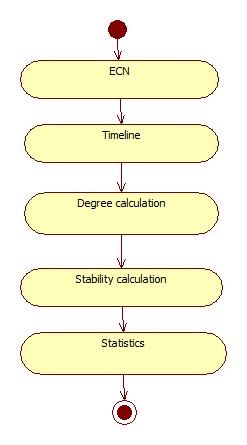
* We consider all the contest specific egocentric networks for this step.
* A network of all the nodes, in the remaining networks, is formed to generate the egocentric congruent network.



***Fig.8.2.6*** *Activity diagram for generation of ECN*

8.2.7 Determine the degree and stability of ECN

* Degree tells us to what extent the nodes of the ECN are congruent with each other
* Stability of the ECN tells us the number of nodes that “join” or “leave” the congruent network, is the specified timeline
* To simulate this, the user profile information is updated, and based on this modification, the ECN regenerated for the same context. The number of nodes that “come into “ECN or “go out” of the ECN, tell us about the stability
* Higher this number, lower is the stability of the ECN

****

***Fig. 8.2.7*** *Activity diagram for measurement of degree and stability of ECN*

**8.3 Database Design**

The users profile is stored in a format called as User Description Format or UDF.

The contexts are also stored in the MongoDB database, along with the properties of the context. The format for this is as shown below:

|  |  |
| --- | --- |
| Name | <Context name> |
| Description | <Context desctiption> |
| PropertyName | [List of property names] |
| PropertyDescription | [List of corresponding property descriptions] |

The 13 pools that have been identified are also stored in MongoDB database.

* Activities

|  |  |
| --- | --- |
| Word | <activity name> |
| Wiki | <activity content from wikipedia> |
| Category | <category of the activity> |
| wolframalpha-description | <description of the activity from wolframalpha> |
| wolframalpha-category-list | <categories to which the activity belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the activity from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Books

|  |  |
| --- | --- |
| Word | <book name> |
| Wiki | <book content from wikipedia> |
| Category | <category of the book> |
| wolframalpha-description | <description of the book from wolframalpha> |
| wolframalpha-category-list | <categories to which the book belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the book from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* College

|  |  |
| --- | --- |
| Word | <college name> |
| Wiki | <college information from wikipedia> |
| wolframalpha-description | <description of the college from wolframalpha> |
| wolframalpha-category-list | <categories to which the college belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the college from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Graduate school

|  |  |
| --- | --- |
| Word | <graduate school name> |
| Wiki | <graduate school information from wikipedia> |
| wolframalpha-description | <description of the graduate school from wolframalpha> |
| wolframalpha-category-list | <categories to which the graduate school belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the graduate school from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* High school

|  |  |
| --- | --- |
| Word | <high school name> |
| Wiki | <high school information from wikipedia> |
| wolframalpha-description | <description of the high school from wolframalpha> |
| wolframalpha-category-list | <categories to which the high school belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the high school from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Events

|  |  |
| --- | --- |
| Word | <event name> |
| Wiki | <event information from wikipedia> |
| Location | <location of the event> |
| wolframalpha-description | <description of the event from wolframalpha> |
| wolframalpha-category-list | <categories to which the event belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the event from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Games

|  |  |
| --- | --- |
| Word | <game name> |
| Wiki | <game information from wikipedia> |
| Category | <category of the game> |
| wolframalpha-description | <description of the game from wolframalpha> |
| wolframalpha-category-list | <categories to which the game belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the game from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Hometowns

|  |  |
| --- | --- |
| Word | <hometown name> |
| Wiki | <hometown information from wikipedia> |
| Latitude | <latitude of the place> |
| Longitude | <longitude of the place> |
| wolframalpha-description | <description of the place from wolframalpha> |
| wolframalpha-category-list | <categories to which the place belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the place from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Interests

|  |  |
| --- | --- |
| Word | <interest name> |
| Wiki | <interest information from wikipedia> |
| Category | <category of the interest> |
| wolframalpha-description | <description of the interest from wolframalpha> |
| wolframalpha-category-list | <categories to which the interest belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the interest from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Locations

|  |  |
| --- | --- |
| Word | <location name> |
| Wiki | <location information from wikipedia> |
| Latitude | <latitude of the place> |
| Longitude | <longitude of the place> |
| wolframalpha-description | <description of the place from wolframalpha> |
| wolframalpha-category-list | <categories to which the place belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the place from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Movies

|  |  |
| --- | --- |
| Word | <movie name> |
| Wiki | <movie information from wikipedia> |
| Category | <category of the movie> |
| wolframalpha-description | <description of the movie from wolframalpha> |
| wolframalpha-category-list | <categories to which the movie belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the movie from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Music

|  |  |
| --- | --- |
| Word | <music name> |
| Wiki | <music information from wikipedia> |
| Category | <category of the music> |
| wolframalpha-description | <description of the music from wolframalpha> |
| wolframalpha-category-list | <categories to which the music/song belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the music from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

* Work

|  |  |
| --- | --- |
| Word | <work name> |
| Wiki | <work information from wikipedia> |
| wolframalpha-description | <description of the interest from wolframalpha> |
| wolframalpha-category-list | <categories to which the interest belongs, from wolframalpha> |
| wordnet-synsets-list | <synonym set list of the interest from wordnet> |
| wordnet-definitions-list | <definitions from wordnet> |

Apart from this, the similarity scores that are calculated for each pool, are also stored in the MongoDB database. This is stored in the form of a hash table of hash tables.

Pool name

{‘word1’:{‘word2’:sim\_score1,’word3’:sim\_score2,…’word(n)’:sim\_score(n-1)}}, {‘word2’:{‘word3’:sim\_score,’word4’:sim\_score,….’word(n)’:sim\_score}},….,

{‘word(n-1)’:{‘word(n)’:sim\_score}}

|  |  |  |  |
| --- | --- | --- | --- |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

***Fig. 8.3.1*** *Similarity matrix for each of the pools*

**CHAPTER 9**

**IMPLEMENTATION**

**9.1 Implementation choices**

The system has a three tier architecture.

The options for the programming language for implementation mainly included C++, Java, C# and Python.

The system needed a natural language processing subsystem at the back end. The options available were Apache Open NLP, Sharp NLP and NLTK.

The considered choices for implementing the front end were Camelot, PySide, PyObjC, QT, PyQT and wxPython.

**9.2 Reasons for choosing a particular implementation**

Python was chosen as the language for implementing the back end due to its support of various NLP libraries and UI frameworks. Also, application programming using python was considered more suitable. The Python regex library is also used in the implementation, and was found very useful.

NLTK, or Natural Language Toolkit was chosen for this purpose due to its easy-to-use interfaces with many corpora and lexical resources such as wordnet, CMU dictionary, Brown Corpus, etc. NLTK also comes with a suite of text processing libraries for classification, tokenization, stemming, tagging, parsing, and semantic reasoning.  NLTK is a free, open source, community-driven project available for Windows, Mac OS X, and Linux. NLTK is suitable for linguists, engineers, students, educators, researchers, and industry users alike.

The front end, i.e. the UI was created using PyQT version 4.10.3PyQT allows creation of the GUI with the help of QT designer that is easy to use. The .ui file can then directly be converted to .py files by using pyuic.

**9.3 Data structures used**

The data structures used to store the following details are:

* Misused words: A list with each element being a list itself. Each element has three strings – the misused word, it’s tag, correct word
* Misused phrases in Indian English: A list with each element being a list of two strings: the first one is the incorrect phrase, second is the suggested correction.
* Comments generated: The comments are stored in a list with each element containing the three things:
  + The sentence for which the comment was generated as a string
  + The severity of the error found
  + The Comment as a string
* Floweriness:
* Tokenised sentences: The nltk.sent\_tokenize() function returns a list of tokenised sentences.
* Tokenised words: The nltk.word\_tokenize() function returns a list of tokenised words.
* Tagged words: The nltk.pos\_tag () function returns a list of tagged tuples.
* Readability:
* Statistics about the text: All the statistics about the text such as number of sentences, number of words, number of syllables, number of polysyllables and number of characters are stored in a list.
* Syllable counting: A list is used for finding the number of syllables in the text, further used to calculate the readability.

**9.4 Communication mechanisms used and their structure**

* QT Action

The QAction class provides an abstract user interface action that can be inserted into widgets. In applications many common commands can be invoked via menus, toolbar buttons, and keyboard shortcuts. Since the user expects each command to be performed in the same way, regardless of the user interface used, it is useful to represent each command as an *action*.

* QT Signal/Slot mechanism

The signal/slot mechanism has the following features:

* + A signal may be connected to many slots.
  + A signal may also be connected to another signal.
  + Signal arguments may be any Python type.
  + A slot may be connected to many signals.
  + Connections may be direct (ie. synchronous) or queued (ie. asynchronous).
  + Connections may be made across threads.
  + Signals may be disconnected.

**9.5 Psuedocode**

**CHAPTER 10**

**INTEGRATION**

**10.1 Integration strategies**

The development part of this project follows Bottom up approach. The bottom up approach is used for building small components, and finally to integrate them into one complete system. Initially the backend implementation was done. Each small component was implemented in python. API's were defined for better interaction and better integration with the frontend as well as for interaction between the backend modules.

To integrate all the components, top down integration approach is used. Top down approach is better for integration, and for every scenario, the corresponding backend API to be used, is also known. The User interface functionalities are linked with backend one after the other one, in a sequential manner. For the frontend, HTML has been used and the backend coding has been done using Python. As python was used, integration was a challenging task. No python frameworks such as Django were used. Python is implemented as a web service by using CGI scripting. After that, the User Interface and the backend CGI scripting was integrated in a phase wise or stage wise manner.

**10.2 User Data Generation**

Using Graph API, the Facebook Access token is obtained. Using this access token, the user profile information and his connected users’ information is fetched in a JSON format. From this JSON data, the required field values is queried and stored in database as User Profile(UP) format.

Users data in UP format

Fetch the User’s Information from facebook

Get Facebook Access token

***Fig. 10.1*** *User data generation*

**10.3 Pools Creation**

From the user profile stored in the database, each user’s information is fetched. This information is parsed and user attributes are retrieved for the different pools. The pools are created by the attribute names of the user fetched from the database. These pools are filled with attribute values. This is done for each of the users’ information and the various pools are generated.

User profile

Get Users Attributes

Parse the Attributes

pool

***Fig 10.2*** *Pool creation*

**10.4 Description Builder**

The pools information present in database is fetched. Each pool is parsed and the attribute values are taken, the description for the attribute value is fetched from the Wikipedia, WolframAlpha and Wordnet.

Get Attributes from Pools

Description for the Attributes

pool

Description builder

***Fig.10.3*** *Description builder*

**10.5 Similarity calculation**

Similarities are calculated between the context entered by the user and the contexts in the database. The user enters the description of the context and the attributes for that context. This context will be matched with the existing contexts in the context database, and the best matching contexts are recommended to the user.

**10.6 Add Context**

In this User Interface component, the Context description is taken as input from the user. First, contexts are recommended from the existing contexts. If the user does not select the context from it, using the Reverse Dictionary the possible context names are suggested. User can select the suggested context name or user can enter the new context name for the description, and the context name and context description are stored in the database. After this, the user is asked to enter the Property description, similar as Context description the property name is suggested to user. Here too, the user can select the suggested property name or he can enter new property name and update the same in the database for that context.

Suggest Names

Get Context Description

Ask for Property Name and Description

Select the Context Name for the Description

Context database

***Fig. 10.4*** *Add Context*

**10.7 Browse Context**

In this User Interface component, the context names present in database are fetched from the database and displayed. User can click any of the context name displayed, to see the context description. The required context name from the context list is to be entered to choose the context.

Suggest Context Names

Select the Context

Context

Database

***Fig.10.5*** *Browse context*

**10.8 Generate Network**

Context description, properties and property description for the corresponding context are fetched. The fetched property description is compared with the pools description. The pool description which matches the properties is fetched from database. These selected pools property values are compared with the user profile. The user profiles which match the property values are selected. The selected user IDs and their names are used to generate the graph. Nodes in the network graph generated represent users congruent to the context. Graph generated contains users who match to the context, containing connections between them. User can click any of the nodes present in graph. The user information of the clicked node is fetched from the database and displayed.

Users’ id to Graph UI

Match (Property Value, User Property Value)

Fetch the Properties for the given Context

Match (Property Value with User Property Value)

Context

database

User Profile

***Fig. 10.6*** *Generate Network*

User Data Collection

User Profile

Pools Creation

Pools

Description Builder

Add Context

Context database

Browse Context

Generate Graph

***Fig. 10.7*** *Integrated System*

The user data collected from Facebook is inserted into MongoDB in required format. From this data, pools are generated. Description of every attribute value in all the pools is generated from Wikipedia, Wordnet, and WolframAlpha and stored. For every pool, similarity between attribute values is calculated and stored.

When a new context is added by a user, the context database is updated accordingly. When a user wants to use an existing context, he can browse and select the context. Once the user has entered a context, his Egocentric congruent network is generated by considering the user’s profile information and the context in which he is in. This network is generated by matching the user’s profile with other users who are also in that context. The Integrated system which generates the network is as shown in the diagram above.

**CHAPTER 11**

**TESTING**

**11.1 Unit Testing**

11.1.1 File options

* *Test Definition:* Checking file options of new, open, save, close
* *Entry criteria:* The application must be open
* *Exit Criteria:* The application is closed.
* *Test input:* Name of the document
* *Test Output:* New document is created/existing document is opened, modified, saved and closed

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected Result** | **Actual Result** | **Conclusion** |
| If the currently open file is modified “save?” option must appear | User must be given the option of saving current file | User is given the option of saving current file | Success |
| Create a blank new document | Blank new document is created and opened | Blank new document is created and opened | Success |
| Open an existing file | Existing file opens. | Existing file is opened | Success |
| Open a non-existent file | Failure in opening file | File opening failed | Success |
| Upon opening an existing file, display of file name in title bar | Display of file name is title bar | File name is displayed in title bar | Success |
| Saving a file after it is modified | Give save option | Save option given | Success |
| Closing an unsaved file | Give save option | Save option given | Success |

11.1.2 File’s edit options

* *Test Definition:* Editing a file*.*
* *Entry Criteria:* The application is open*.*
* *Exit Criteria:* All edit options are tested
* *Test Input:* Text
* *Test Output:* Text

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected Result** | **Actual Result** | **Conclusion** |
| Copying/cutting and pasting after selecting text | Write same text again | Text is copied/cut and pasted | Success |
| Copying text when nothing is selected | Disable copy option | Copy option disabled | Success |
| Undo when no changes are made in the editor | Disable undo option | Undo option disabled | Success |
| Redo when there is no prior undo | Disable redo option | Redo option disabled | Success |
| Redo changes undone | Perform redo | Redo is performed | Success |

11.1.3 Formats

* *Test Definition:* Typing text in a format*.*
* *Entry Criteria:* A format UI is open*.*
* *Exit Criteria:* Format data appears in editor window
* *Requirement:* “OK” is pressed in the format window
* *Test Input:* Text
* *Test Output:* Text in editor window

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected Result** | **Actual Result** | **Conclusion** |
| Filling all text boxes in the format | Write text in the order entered, into the editor window | The text appears in the order entered, into the editor window | Success |
| Leaving a few text boxes blank in the format | Text written, along with help text for those fields left blank | Text written, along with help text for those fields left blank appears in editor window | Success |
| All fields left blank | Write help text for all fields in to editor window | Help text for all fields appears in editor window | Success |

11.1.4 Stylistic comments

* *Test Definition:* Stylistic comment list
* *Entry Criteria:* A list of stylistic comments
* *Exit Criteria:* Highlighting text and displaying comments only of the colour specified
* *Requirement:* Comments of all colours and multiple comments for the same sentence
* *Test Input:* Text and colour option
* *Test Output:* Text highlighting

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Expected Result** | **Actual Result** | **Conclusion** |
| Clicking a comment | Highlighting the sentence associated with the comment | The sentence associated with the comment is highlighted | Success |
| Changing the text and clicking a related comment | No highlighting | No text is highlighted | Success |
| Choosing a comment colour option | Displaying comments only of that colour | Comments only of that colour are displayed | Success |

11.1.5 Stylistic analysis

* *Test Definition:* Detection of stylistic errors
* *Entry Criteria:* A piece of text
* *Exit Criteria:* Completion of stylistic analysis
* *Requirement:* Grammatically correct text
* *Test Input:* Text data
* *Test Output:* Presence or absence of stylistic error

11.1.5.1 Test cases for rules of style

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Actual Result** | **Conclusion** |
| Detection of passive voice | A sentence with simple passive voice | Passive voice detected | Success |
|  | A sentence in active voice | Passive voice not detected | Success |
| Detection of a preposition ending | A sentence with a preposition at the end | Preposition ending detected | Success |
|  | A sentence without a preposition at the end | Preposition ending not detected | Success |
| Detection of multiple tenses in the same sentence | A sentence with multiple tenses | Tense inconsistency detected | Success |
|  | A sentence in a single tense | Tense inconsistency not detected | Success |
| Detection of misuse of words | A sentence with “then” instead of “than” | Misused word detected | Success |
| Detection of Indian English | A sentence with “yesterday night” instead of “last night” | Indian English detected | Success |
| Detection of long sentences | A sentence with length > 25 | Long sentence detected | Success |

11.1.5.2 Test cases for stylistic statistics

|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Actual Result** | **Conclusion** |
| Floweriness | A sentence with 2 adjectives and 3 adverbs | Floweriness is 5 | Success |
| Obscurity | A sentence with 2 infrequent or rare words of English | Obscurity is 2 | Success |

**11.2 Integration testing**

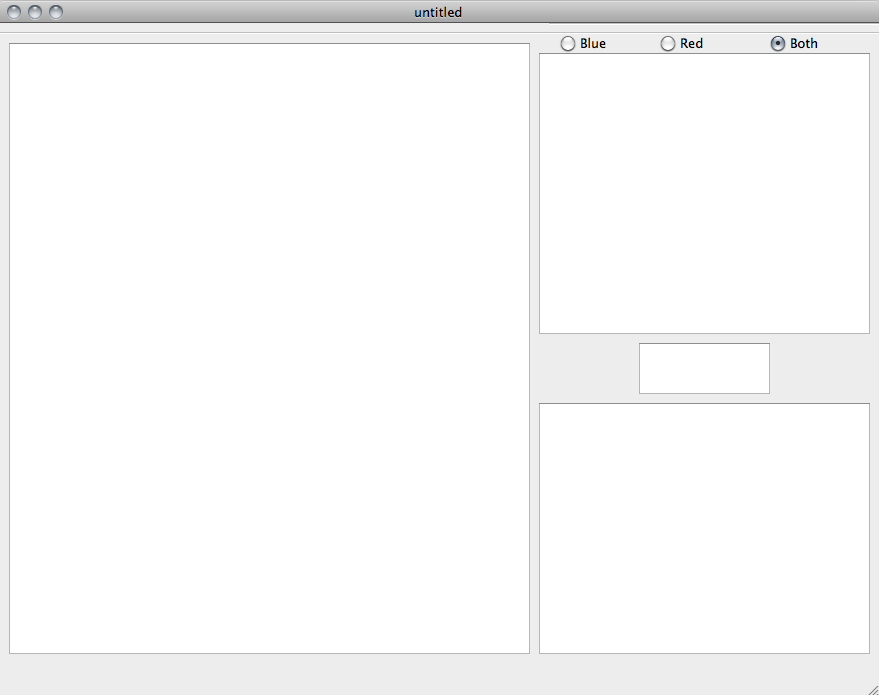
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Component** | **Number of Test Cases** | **Test case Description** | **Result** | **Conclusion** | **Issues faced and fixed/ Comments** |
| Opening text file | 4 | Create a new file | Text file created | Success |  |
|  |  | Open an existing file | Text from existing file loaded into the editor window | Success | No comments |
|  |  | Try to open an existing file and then cancel | Existing file not opened | Success |  |
|  |  | Type into an untitled file and then save it | File saved in the target directory | Success |  |
| Evaluating Readability | 4 | Evaluate readability of a blank document by clicking on Evaluate readability | Readability is 0 | Success | No Comments |
|  |  | Evaluate readability of the text while typing | Appropriate readability displayed | Success | No comments |
|  |  | Close the document and reload it. Check readability now | Readability remains the same as the previous time | Success | No Comments |
|  |  | Check readability while typing by enabling auto-evaluate | Readability evaluated every time there is the space bar is pressed | Success | Event is triggered when the space is encountered |
| Setting Target Readability | 3 | Set readability as Children  Test a document with a higher readability | Readability comments change colour | Success | No comments |
|  |  | Test an existing document for different readabilities. Do not click Evaluate | No change | Success | No comments |
|  |  | Test an existing document with different readabilities by pressing evaluate every time. | Colour changes for everything except target readability | Success | No Comments |
| Generating comments | 11 | Give a poem as a text | Proper comments not generated | Failed | Limitation of the parsing system being used, in this case NLTK |
|  |  | Use multiple tenses in the same sentence | Appropriate comments generated | Success | The writer should stick to one tense in a sentence |
|  |  | Multiple tenses across sentences in a paragraph | No Comments generated | Success | Checking of tenses is a sentence level functionality |
|  |  | Sentence with 30 words | “Sentence may be too long” comment generated | Success | The sentence should not be more than 25 words. |
|  |  | Sentence with 20 words | No comment generated | Success | No comments |
|  |  | Sentence with 3 adjectives in a sentence | Floweriness: 3.0 per sentence | Success | Floweriness = Number of Adjectives per sentence |
|  |  | Sentence with no adjectives | Floweriness  : 0.0 per sentence | Success | Floweriness = Number of Adjectives per sentence |
|  |  | Sentence with passive voice | Passive Voice comment generated | Success |  |
|  |  | Sentence in active voice | Passive voice comment not generated | Success |  |
|  |  | Sentence with 3 rare words in a sentence | Comment: Obscurity: 3.0 per sentence | Success | Obscurity = Number of words with frequency <5 per sentence |
|  |  | Sentence with 0 rare words in a sentence | Comment: Obscurity: 0.0 per sentence | Success | Obscurity = Number of words with frequency <5 per sentence |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

**CHAPTER 12**

**STORY BOARD**

**12.1 Blank document page**

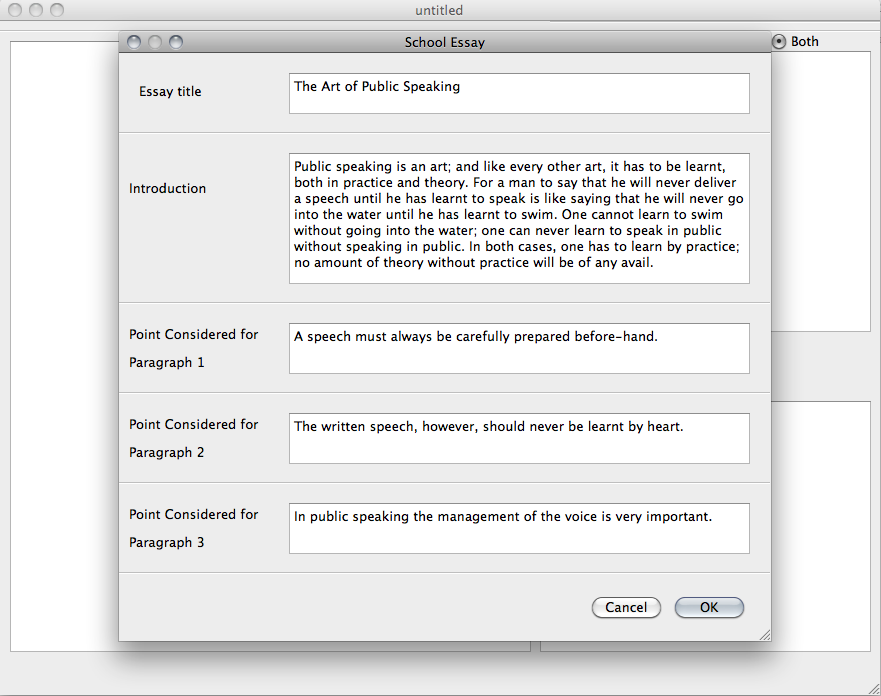
When the user first opens the application, a blank editor appears. From here, the user can chose to type in the blank document and hence create a blank file or to open an existing file. If an existing file is opened the contents of the file appear in the editor section.



***Fig. 12.1*** *Blank doc page*

**12.2 Writing in a template**

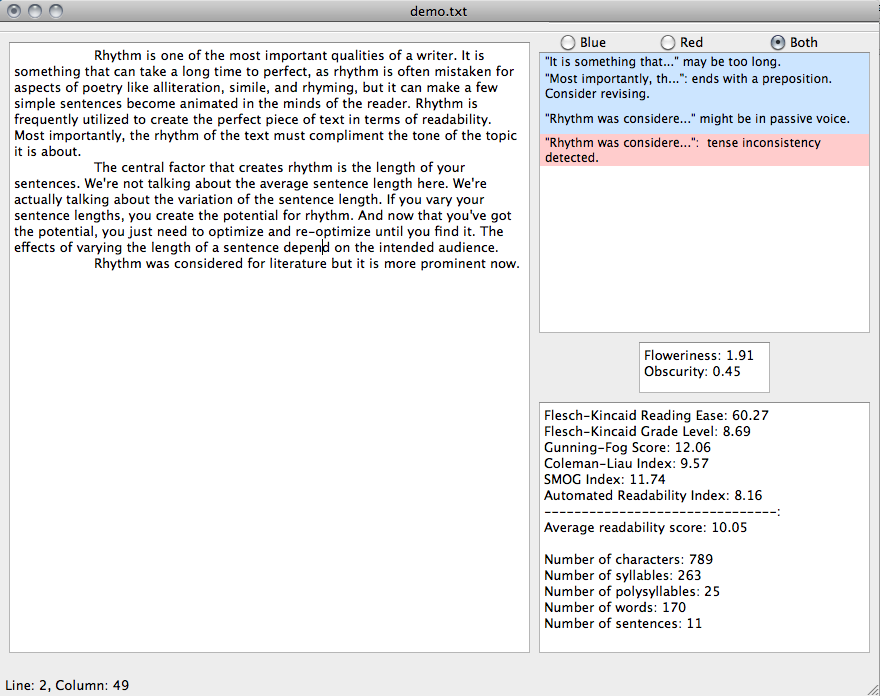
The user can chose from several templates. Here, a sample of school essay is considered. After the fields are filled in, the contents appear on the editor section. Along with the fields, additional help text also appear.



***Fig.******12.2*** *School essay template*

**12.3 Evaluating for stylistics and readability**

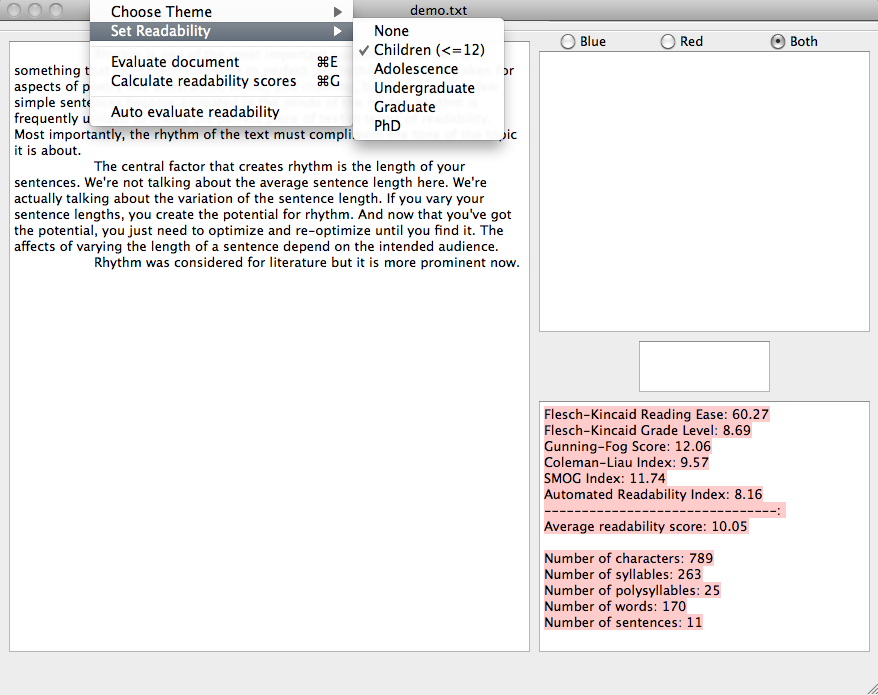
The user has completed typing a document. The application has evaluated the text and has printed comments for those sentences violating rules of style. Each comment is given a colour according to the severity of error. Also, the application gives the readability scores of the same piece of text along with statistics like number of words and sentences.



***Fig. 12.3*** *Stylistic comments and readability scores*

**12.4 Setting target readability**

If the user wishes to monitor the readability score as he types the document, he may set the target readability score. The colour of the readability box changes if the actual readability score is not in the target range.



***Fig. 12.4*** *Choosing target readability score*

**CHAPTER 13**

**CONCLUSION**

Congruent networking is a novel concept, which can be used to improve the quality of service in a particular context. Social networks have many disadvantages that can be solved by the use of congruent networks. The use of congruent networks helps us in providing the best possible people who are related to us, in that particular context. The advent of congruent networking will provide some new areas for research to be done. Some applications of this field include the following Maximizing team productivity, Assignment problem, Formation of context based Knowledge groups, Human relationship management, Comparative study of behavioral v/s Biological parameters to identify congruence, Product - Entity based congruence, Event Managed based Congruence, Pre-Market Analysis, Trend based Market analysis, Matrimonial services, Recruitment Services, E-teaching and E-learning, Graph theory based approach for relationship analytics etc.

The implementation takes into account the ego-factor of the users of the social network, and will generate an appropriate egocentric congruent network for the given context, for that user. The level of ego varies from person to person, and also varies depending on the context. Some people tend to be more egocentric during certain situations, whereas some people are more sociocentric. The congruent network will change depending on the context as well. This approach is more efficient when we consider egocentric people, that is, people who are more interested in themselves than others.

Similar contexts are implemented using the Jaccard similarity. It takes the description from the user and matches it with the description that is available in the database. In order to generate the network, the system compares the similarity of the context’s property’s description with the pools’ attributes stored in the database. The best 5 matching attributes are chosen for each of the properties for the given context. These attributes are looked for in the users’ database. The selected candidates are assumed to be congruent with the user considered. The mapping also shows an expected result, with an efficiency of 50%-80% for the users. Generation of the Egocentric congruent network takes approximately 5 to 10 seconds as it involves mapping of properties with the existing contexts, and mapping of the context with the users.

**CHAPTER 14**

**FUTURE SCOPE**

As the project comes under a new and upcoming research domain, there is a lot of scope for more enhancements as well as more research opportunities exist. Congruent networking, as a research topic, will be very helpful in the near future. Our implementation of this project has been restricted by the information collected from the various social networking sites. Due to the unavailability of time and infrastructure, we have not been able to implement this project by considering the scalability aspect. In this section some of the enhancements possible to our implementation in different sections, starting with functional improvements, followed by new approaches for the generation of congruent networks, have been listed.

**14.1 Functional Improvements**

This section lists some of the improvements that can be incorporated to the existing implementation of this project:

* Collect information from various social networking sites, including Twitter, LinkedIn, Quora, Google+ etc.
* Collect more information from Facebook which may include posts, likes, comments, etc
* Include category information provided by WolframAlpha and other services while performing similarity calculation.
* Generate a larger description for the values of the attributes in the pool database
* Use of different similarity algorithms to calculate the similarity scores and combine similarity scores of various algorithms to provide better results.
* Classification algorithms can be implemented for the matching of the pools with the context.
* Implement the context using more efficient ways.
* Implement more natural language algorithms to find the intentions behind the context.
* Generation and use of the network, using existing network generation tools, for a better visualization and better user experience.
* Use different technologies to implement the backend in a more efficient manner.
* Make the project compatible with every browser.

**14.2 New approaches**

This section lists a few new approaches that can be incorporated into our implementation.

* Host the application on the cloud and include a large number of users.
* Collect user feedback and incorporate them.
* Define the context in different ways.