



Text Analysis for the visualisation of large twitter data

MIDDLESEX UNIVERSITY

MODULE CODE:
CMT4141.

STUDENT:
HARISH MUPPALLA.

STUDENT NUMBER:
M00332278.

PROJECT SUPERVISER:
DR. KAI XU.

Department of Computing and Multimedia
Technology, Middlesex University.

Acknowledgement

This dissertation get completed with guidance and the help of several individuals, who in one way or another contributed and extended their valuable support in the preparation and completion of this study.

*My First and foremost gratitude to my Project Supervisor **DR. KAI XU** Senior Lecturer, Department of Computing and Multimedia Technology, Middlesex University. Whose guidance and encouragement is unforgettable. I sincerely pay my gratitude for teaching new concepts of visual analytics area. The suggestions for improvement in work and review feedback have given me knowledge, to step ahead to next level of study.*

*I owe my deepest Gratitude to **DR. CARL EVANS** Director of Postgraduate Studies, Department of Computing and Multimedia Technology, Middlesex University. Who taught me design patterns and Object oriented programming concepts through java, His perfection in teaching enlighten me in implementation work with utmost interest also helped to improve my skill set. He has been the inspiration as I hurdle all the obstacles in the completion of project. I learnt a lot from course works and assignments framed by Dr. Carl Evans, which laid fundamental building block for code implementation.*

*It is grateful to thank **DR. RALPH MOSELEY** Senior Lecturer, Department of Computing and Multimedia Technology, Middlesex University .who introduced internet programming concepts, web technologies, and databases like **MYSQL** to me. His style of teaching though experimental lab works, was given me a scope of learning from mistakes.*

*It is an honour to me to pay respect and thanks to **MR. ED CURRIE**. Head of Department, Department of Computing and Multimedia Technology, Middlesex University. Who taught me functional programming (Haskel), his experienced teaching given the opportunity for leaning functional logic building ability, helps in next level study in programming.*

*I would like to thank **DR. FRANCO RAIMONDI** Senior Lecturer, Department of Business Information Systems. His patience and sincerity towards teaching and also the support in lab works is remarkable. His cardinal way of understanding problem to provide solution is very much appreciative.*

*I would like to show my gratitude to **Mrs. BRONWEN CASSIDY** (lab Instructor module CMT 4161 & CMT 4451) for her patience and steadfast encouragement to complete course works in lab. She is responsible for seeding interest in me for learning through practicing on machine. She is a very good human being and a good tutor in clarifying doubts in lab exercises with utmost attention, I learned a lot from her.*

*I would like to thank **DR. ELKE DUNKER-GASSEN** (Principle Lecturer) & **Miss. NALLINI SELVARAJ** (Tutor) Department of Computing and Multimedia Technology, Middlesex University. For teaching me postgraduate and professional skills, by the knowledge I earned in the study of module CMT 4021, built confidence to review various references to get inference by concluding. This area of study enlightens my skills, the course work assignments helped in learning as well as in documentation and report writing.*

*Finally and most importantly My **PARENTS**, I want to pay them my deepest gratitude, love and respect. Who have always supported and encouraged me in every walk. They believed in me, in all my endeavours and who so lovingly and unselfishly cared for me and my sister.*

Table of Contents:

Table of Contents:	3
1.0 ABSTRACT:	5
2.0 Introduction:	6
3.0 Literature Review:	8
3.1 Accessing tweets from the Twitter:	8
3.2 Annotations Extraction:	10
3.3 Geo-Coding of a location:	11
3.4 Sentiment analysis:	12
3.5 Significant or key phrases extraction:	13
4.0 Project Requirement Specifications:	15
4.1 Requirements	15
4.1.1 Project scope:	15
4.1.2 Software Requirements:	15
4.1.3 Functional Requirements:	16
4.1.4 Non-Functional Requirements:	17
4.2 Use cases:	18
5.0 Analysis and Design:	20
5.1 System Design	20
5.2 Overview of the proposed system design:	21
5.2.1 NERextraction:	22
5.2.2 Geocode:	22
5.2.3 Senticalculate:	22
5.2.4 Significant phrases:	22
5.3 Security concerns:	23
5.4 Databases:	23
5.5 CentralTweetCollector Class Diagram:	24
6. Implementation and Testing:	26
6.1 Implementation:	26
6.1.1 Planning or approach for implementation	26
6.1.2 Design patterns:	26
6.1.3 TweetCollector:	26
6.1.3.1 Storage of persistent data using DAO design pattern:	28
6.1.3.2 Storage of persistent data without using DAO design pattern:	28
6.1.3.3 Storing the persistent data:	29
6.1.4 TextAnalysisComponents:	29
6.1.5 GeoCode:	30
6.1.6 NERextraction:	33

6.1.7 SentiCalculate:	34
6.1.8 DisplayTokennization (significant phrases/words)	35
6.2 Testing:.....	37
At level one:	37
At level two:	38
At Level Three:	40
7.0 Project Evaluation:.....	42
7.1 Requirement specification evaluation:.....	43
7.2 Performance testing results:	43
7.3 Performance evaluation:.....	44
7.4 Project Demonstration:	45
8.0 Critical Evaluation of project and self reflections:	49
9.0 Conclusion:	50
Future work:	50
10.0 References:	51
Appendices.....	53

1.0 ABSTRACT:

Communication is a key factor in today's human life, due to time constraints physical interaction between people is not possible. This gap is filled by the technology through 'social networking' sites it's very easy to get access to interact other based on their interests. Many applications are getting releasing with new features day-by-day from vendors, to provide efficient usability and user friendliness. Visualisation is a new trend setter of information representation, the back bone of visualisation is data.

This project proposed a new system that delivers large database of Social Networking Site (SNS) called 'Twitter'. Many Third party application are building based on SNS like Twitter, they need to have processed data from their operational purpose. The main stream of the applications is visualisation applications. This project gives more beneficial solution by providing in-depth detailed information of data. In this context this implementation serves processed information of tweets accessed from Twitter Server.

Here processing the tweet involves extraction of metadata of tweet, geocoding the physical address in a tweet, analysing the sentiment of content in the tweet text and extracting the significant and key phrases from a text. This application is an integrated system used to get connect and access tweets from Twitter to get processed text analysis components. After all the Information Extracted and NER (Named Entity Recognition) text analysis from tweet, are stored into a persistence database. This document discussed in review the contemporary and early works and studies related to Text analysis for and efficient procedures in extracting vital aspects of information. Here Object Oriented programming and Design patterns are used in implementation of this system, with proper testing and validation are performed at three levels, both normal and performance test results are evaluated to achieve a sophisticated system.

2.0 Introduction:

The Growth and advancement of information technology gear-up in providing tremendous amount of data to next level from diverse streams in the form of creation, storing and validating. One of the good consequences is availability of incredible amounts of data, which is not possible earlier. There is evidence of negligence or not taken proper care in conveying knowledge from the data. The designing approaches, pattern and representations are not efficient to communicate data so far. Suitable Remedy for this problem is Visualisation it is an art fame work of scientific design approach of creative innovation of emotional involvement in communication.

Visualisation is aimed at human understanding in processing the information efficiently and effectively. The accelerated expansion of ‘social networks’ (example twitter) makes possible, to transfer and share information to multiple user’s very fast with less cost. The potential outcome of social networking facilitates a user to reach and interact millions of other users. Companies are building Third party applications, which are experimental in delivering tools to benefit user. It helps to study the opinions, user views, new ideas, public interests, and their focused activities of millions of user round the globe. Marketing firms also get involved in analysing user inputs and exercising over public sentiment, and the brake out of latest trends in the masses in upgrading the products and services. The raw material in building the third party application is bulk volumes of data that has to process to get information. The extraction of information from raw data put extra burden on applications that impairs effective utilisation of available data. Text analysis may also refer as text mining for text analytics, to improve quality persistence and adds sense to the meaning of data. Text analytic is a superset of Information retrieval and lexical analysis of data.

This work proposed text analysis implementation for information extraction (IE) from data by proper evaluation techniques to reduce the unwanted noisy data. Segregated the extracted information based on classification of usability. Discussed and reviewed contemporary tools and relative text analysis factors, like sentiment analysis, extraction of annotations and identification of significant phrases over the data. Examined various procedures and developed the suitable procedure for geo-coding, in demand of contextual preference to twitter.

Evaluate various classifiers in view of developing sentiment analyser. This document evaluates the available API’s to get access data from the twitter, and implementation of suitable procedure to build database of social network data (twitter). To make it useful for visualisation of twitter data, which is efficient and effective in utilisation and maintenance? Also examined and compared existing gazetteers and Entity extraction libraries. For a task of implementing NER (Names Entity Recognition) to extract annotation specific to the defined patterns and formats after proper analysis of input. Sentiment analysis have insight to identify the positive and negative sense in the text, the evaluation focuses mainly on the behaviour aspects and words or phrases that means the human emotions. This work

simplified to the process to sentiment analysis after proper review on contemporary analysis to classify sentiment over text.

3.0 Literature Review:

Present information world delivers reporting through automation, minimises human effort to analyse the text. Ongoing research facilitates user friendly procedures in implementing systems, to extract information from the textual content. In the present context of analysing the text and extract information with respect to application requirements is essential. Visualisation needs a processed logical or statistical data to represent in visual format helps users to understand massive volume of data effortlessly. Current work focused to analyse the texts of large volume sources like twitter (social networking database), it throws lot more questions in implementing the system, prerequisite of development is to analyse relative works and existing research or earlier proposed systems. Factors which are significant in review of already committed works like reliability, usability, flexibility and complexity. Specification of current work proposal requires a proper study on various aspects that influence intended implementation. Some are categorized into the following

- Connecting twitter to get access tweets, in this regard review of available web-services, API (application programming interface) and libraries is to be conducted. Request and response types, authenticated services, user accessibility constraints and limitations have to be studied.
- Conversion of a physical address in to Geo-coordinates.
- Scrutinizing the parsers and existing procedures for Extraction of user defined Annotations in a text.
- Text analysis of large volumes collection of tweets, sentiment analysis over input text and prediction of sentiment in text.
- Contemporary parsers, existing analyser and Extracting significant or key phrases of a given input text.

Validating the methods, processes and algorithms developed periodically over a span has to review with comparative study helps in concluding, and formulating assumptions for intended application.

3.1 Accessing tweets from the Twitter:

Accessing Tweets from a Twitter is primary for building a database to get processed and extract information. Twitter has 3 types of API's REST API, Search API and Streaming API. Each has different usability REST API allows user to access twitter core data Search API grants methods to communicate the Twitter search. Streaming API assures long-span connection to get access huge volume of tweets. API's in Twitter is http based requests that too GET method is required in data retrieval.

Twitter API (dev twitter 2011) provides Search API and Streaming API for accessing Tweets, Search API provides recent Tweets with relevance to the search key and Tweet index of recent 6 to 9 days. Were as Streaming API gives the real time continuous stream of all Tweets, but it doesn't filter Tweets that are relevance. Limitation are laid on the users request frequency rate for both Search and Streaming API's, which not disclosed due to

abuse and needless usage. The request limit can check in the response header, so that it varies over time and overall requests to get access.

Twitter API (dev twitter 2011) facilitates two ways to get access Tweets, through Authenticated and unauthenticated requests. Search API supports unauthenticated and Streaming API need to have authentication. As far as authentication is concerned about types of Tweets, here we have public status and protected status Tweets. Search API present public status tweets on the other hand Streaming API present s both public and protected status Tweets. Request rate limit authenticated user-requests laid on user and for unauthenticated user-requests limit is laid on IP (ip address of the system). Client can request statuses at maximum of 3200 by REST API and 1500 statuses (response tweets) through Search API. Haewoon, lee & Housing (2010) have clearly explained about the functionality, operation and usability of twitter and also briefed about background processing to user. There is evidence (Haewoon, lee & Housung, 2010) that (1) Maximum number of requests from the user to twitter is 10,000 per hour from each IP address. (2) It is advised that tweet collector from the twitter to limit their request rate to the prescribed 10,000 requests/hour and to maintain time delay in between request for better results without any duplication.

Twitter API (dev twitter 2011) gives scope of implementation of custom applications though broad spectrum on programming language Libraries and packages, java in particular the best in implementing object-oriented programming. Twitter4j API(twitter4j, 2011) is one of the java library for implementing custom application on Twitter, Twitter4j is feasible and flexible library for getting connected to Twitter, and communicate from custom application via Twitter4j to Twitter.

Twitter facilitates bifurcation of tweets into public and protected, public statuses tweets are from user accounts which are not protected, and protected is from protected user accounts. Protected statuses need user authentication credentials to get access Search API supports for public statuses.

Twitter API (dev twitter 2011) gives response to requests in “JSON”, “XML” and “ATOM” formats, parsing the output are in need of specific to the method you are using to extract. In twitter response, out put some field are not guaranteed to return the value may it contain null, if value of the corresponding field value is not available to return? The http response codes may be witnessed in the output, by specifying the status of the user request. Twitter4j (twitter4j, 2011) provides an implementation of java libraries to parse the GET responses like JSON, XML etc. Metadata of the tweet also implanted in response of a search query, it's vital in understanding the information stated in the tweet.

(dev twitter 2011) have evidenced and analysed that every tweet is not geo-tagged (geographic coordinate's latitude and longitude), but some tweets are exclusively geo-tagged in responses through Search API. It's purely optional to the user in stating the geo-location, because of user perspective and privacy to unable the disable this geo-tagging feature while tweeting through twitter.

3.2 Annotations Extraction:

My objective is to extract annotations from the Tweet text and the contemporary implement them for finding the annotations. Alias-i (2008) and Cunningham et al (2011) have proposed the corpus (document) and datasets and stated a mechanism for chunking text into predefined chunks based on specified regular expression or tokenising. Cunningham et al (2011) have given a solution for NER (Name Entity Recognition) with the help of Annie gazette but input text should be a textual document. Alias- i (2008) and Cunningham et al (2011) to extract annotations we need to train the system by specifying entity trained files or files of gazetteer lists.

The mechanism of indentifying the annotation is based on the matching of the trained file content with textural words of respective annotation type of corresponding files. Alias-i (2008) has used external training files with data on annotation, where as Cunningham et al, (2011) have used internal mechanism to mention the gazetteer index with the lists. Both Alias (2008) and Cunningham et al, (2011) stated that there is no provision of finding annotation of provided input simple text but it limits usability. Cunningham et al (2011) have said in their context that in defining the training data the usability has to analysed first and Alias-i (2008) has mentioned that segregation entities have to be taken into different lists or files while preparing the training.

The release (Cunningham et al, 2011) specified only trained mechanism in extracting annotations from the text document, were as it not stated for untrained mechanism. In concern simple Text annotation with various discipline data, (Alias- i, 2008) (Cunningham et al, 2011) complicates the procedure of defining the training data. Nadeau & Terney (2006) have defined the “Entity noun ambiguity” and resolved it by implementing algorithm called “Aliasing resolution algorithm”, it explains entity boundary detection in the course of unsupervised system to extract annotations and stated that it is not comparative to complex system.

Stanford’s (Jenny Finkel, 2006) implemented natural language processing resources for text engineering and have mainly focused on processing of natural language in to a spectrum contents like parts of speech, translators, word segmentation, classifiers etc. In comparison to (Alias- i, 2008) and (Cunningham et al, 2011) the scope is limited in (Jenny Finkel, 2006). Features of (Nadeau & Terney, 2006) and (Jenny Finkel, 2006) are relative in the context of information extraction from the corpus. Jenny Finkel (2006) has customised the implementation of the code and made reusable or user friendly in different contexts. Extraction of annotations in a simple text is defined clearly in (Jenny Finkel, 2006) and some models have been discussed, which in general we make use of them for every textural input data.

As discussed earlier there is every need to walk through the code for customisation, apart from the models in the discussion. If custom implementation demands more annotation apart from models, there are alternative options to go for custom models which are mentioned in Jenny Finkel (2006). One factor that effects the performance is the training source, be cautious about the size of the training files. Main inference is the developer has to be

Cautious over the no entity type lists in a training file, because delay time in extracting annotation is proportional to the training data size. Query execution time crucial in designing the databases, efficient use of memory builds application efficiency so, to be selective in framing the annotation types on priority basis.

3.3 Geo-Coding of a location:

Geo-coding plays an important role in representation of physical address on visual animated maps. Earth surface is divided in horizontal and vertical angles, the horizontal lines represent latitude and vertical lines represent longitude. For latitude the equator is taken a reference point as 0 Degree and towards poles end 90 Degrees, the Greenwich (prime meridian) and total 360 Degrees span of vertically into equal halves of 180 Degrees of east and 180 Degrees of west. Geo-coding coordinates are decimal values of latitude and longitude. As the objective of this work it demands for geo-coding (converting location or address in to latitude and longitude coordinates) the contemporary mechanism is to make use of the API's having functionality and huge data corresponding to the geographic coordinates.

In this context it's necessary to analyse the available resources, evaluate the relative functionality, usability and flexibility in customization of the resource. In which way the available research satisfies the user assumption in building a new system, by updating requirements of specific scenario in the available system. As per Dr.Ela Dramowicz (ela Dramowicz,2004) the address need to analysed taken care of providing information like street name, postal code or the area name, example county, district . Which need to be conscious over providing approximate address string at least in, finding the geo-coordinates of an address? In (ela Dramowicz, 2004) there is a discussion of three methods in finding geo-coordinates they are through street address, postal codes and boundaries, which is interesting, but not briefed about the implementation.

The popular geo-coding API available in use is "Google geo-coding" (Mono marks, 2010) and "yahoo place finder" (yahoo 1.0 2010) both are providing web-services to find the geo-coordinates of the user query. Mono marks (2010) and yahoo 1.0 (2010) have provides services which require authorisation and both have similarity in http request to the respective URI and response formats of JSON and XML. As the service is on commercial basis and to control load of unlimited request from users, they place restriction over the accessibility by limiting the user requests. Mono marks (2010) is meant to have client-side purpose by limiting the 2500 requests / day for each IP address, whereas yahoo 1.0 (2010) was concerned for server-side limited 50,000 requests/day for the user application. Mono marks (2010) policy guidelines states that using geo-coding results without plotting on Google map is prohibitive. In comparison Mono marks (2010) and yahoo 1.0 (2010) both are efficient and accurate but Mono marks gives best results.

Goldberg & Wilson (2011) have explained about the Batch processing of addresses, but most worrying factor is the limitation over the request rate. In batch processing, file size and

file formats are taken into consideration and the input file must follow specified guidelines, which suppress the usability here.

Using web-services in custom developmental works not only suffers from restriction imposed by the service provider but, also the dependency factor affects the functionality of the user application. Be cautious over giving unstructured input address to the system, because sub location and locations names are duplicated around the globe. Conversion of address into geographic coordinate's process requires custom database of all available addresses with their corresponding latitude and longitude coordinates. But it's expensive to buy the data from the available sources.

3.4 Sentiment analysis:

Sentiment analysis become significant in today's world to analyse the corpus or bulk texts. It is evident the time constraint, high frequency of data and reports, rapid user feedbacks imposing extra burden on servicing bodies (blogging groups, market analysts, stock boards, portals). Apart from the supervision it needs an automated tool to evaluate the sentiment in a text. There is scope of study by using sentiment analysis tool in ongoing speculation in public life, customer opinion analysis, tracking the reviews of a product and to study the mass sentiment over different issues or aspects. Present it's been prioritised in research and development of certain tools to attain a better analysis over bulk data in growing economies.

Rahman, mukras & nirmalie (2007) in their paper explained that a text or document can be analysed and bifurcated into positive and negatives sentiment, and in order to that they have designed a procedure to evaluate the input data corpus, primary task is part-of speech tagging to each phrase of input text with predefined coding. Rahman, mukras & nirmalie (2007) have defined a secondary task of word/phrase frequency detection in given text, and extracting "bi-Gram" (sentiment rich phrases/words) and assign a score which is predefined for sentiment or emotion words (based on the intensity of the word). Finally by aggregation of positive and negative sets of score, the predictive score of the sentiment in the text get excavated; in this regard an algorithm was derived (Rahman, mukras & nirmalie, 2007). Rudy & Mike (2009) introduced new sentiment analysing tools for implementation and have derived a new combined approach used for single classifier for sentiment analysis. Rudy & Mike (2009) have extended the Rahman, Mukras & Nirmalie (2007) and developed a new approach using distinct classifier a two levels micro- level and macro-level, and averaging the sentiment at both levels.

Let's take the scenario, we have a corpus of files each file will get analysed by using the set of available classifiers and have taken their corresponding average score of sentiment. Rudy & Mike (2009) have measured the accuracy of each classifier on the file and take the highest accuracy score of sentiment which is known as micro level averaging, it is important because one classifier predict a wrong score can affect the entire mechanism. Secondly by choosing the micro averages from a list and average those in macro level get overall

predictive sentiment score of corpus (list of document or files) or datasets. Rudy & Mike have also stated the Rudy & Mike (2009) have made an evaluation of the contemporary available sentiment classifiers and briefed about the implementation procedure, but it was a complex implementation as the response time is high because of the complex procedure (Rahman, mukras & nirmalie, 2007). Rudy & mike (2009) have defined a hybrid system by inducing a lot of rule based test which reduces adaptability and raises complexity, it influence the usability. The implementation efficiency and usability is predominant than complex theoretical procedure in choosing suitable sentiment classifiers in relative to both (Rahman, mukras & nirmalie, 2007) and (Rudy & mike, 2009).

Now it raises the question a simple mechanism reusable sources or readily available resources to make use in sentiment mining. Always there is option to switch on alternatives like sentiment analysis API (Application Programming Interface), for the custom development programming languages like JAVA, .NET etc (libraries or API). Alias- i (2008) Provides JAVA library for semantic analysis and developed a supervised system which need to train on user specific sensitive models. Alias- i (2008) needs to train classifier with user context aggregated dataset's initially to run sentiment application. Limitation over the usability and adoptability to custom application is, (Alias- i, 2008) only operates on corpuses or datasets. No where it's defined about simple text (user argument) processing apart from taking input as corpus. Cunningham et al (2011) and Alias-i, (2008) have made quite similar mechanism in mining the sentiment. A simple and efficient classifier need to get build based on the limitations and constraints laid by, early implementation of sentiment analysers.

3.5 Significant or key phrases extraction:

Phrase is a word or a set words that form meaningful sentence, "significant Phrase" means word or set of words have significance in a statement or text. Significant phrases assist a reader or user to derive partial inference in quick review of article or text. It showcases potential idea behind the text, though highlighting the words that have potential impact on framing the sentences.

Metadata of a document or text present the key information, which elevates prominence of data provided by the document (corpus or text). Now it arises how to detect and extract significant phrases from text. Turney P.D (2000) states relative difference between human generated and machine generated key phrases, as the perspective humans vary by one another also it contradicts the machine generated ones sometimes. Turney P.D, (2000) Proposed an algorithm to extract phrases having significance, by aggregated list of common words and adverbs matches the text and extracted rest and listed separately.

Turney P.D (2000) counted repetitive words and removed the duplicates and listed as final list, in which he also included the number phrases. Experimented and compared the human generated and machine generated significant phrases and concluded in most cases machine generated phrases are valuable. Youngzheng (2005) have defined three key method which are "TFIDF, KEA, and Keyterm" to extract key phrases from the text and also distinguished narrative and plan text. Narrative text is informative (structured detailed information) and

reasoned text, but non-narrative text (plain text) contains some non-sense or noise words. Youngzheng (2005) has evaluated all methods and calculated experimental results on narrative and non-narrative text and concluded that narrative text abet in improvised performance of extraction methods.

Yuan j.Lu (2007) had proposed “KE algorithm” and explained how efficiently domain independent text can be processed through training the machine (machine learning). KE is trained on key “phrases and non-key phrases” to distinguish between key significance of phrases. KE states POS (Parts Of Speech) tagging the input text and filter adjective, noun and verbs apart from stop words in step1.the nouns filtered in text title and “calculated the $TF \times IDF$ ” score to for proper noun in step 2 combine filtered phrases in 1st and 2nd steps scores assigned to the each phrase based on distance calculation, sort the phrases after removing duplicates. Outcome of this procedure called as significant key phrases yuan j.Lu (2007).

After all going through all proposed works related to significant phrase or key phrase extraction from the given input text, it is evident to filter the noise words with the list of stop words and extract specific POS (Parts-Of-Speech) tagged words. Algorithms are defined to meet the accuracy and correct phrase detection. Machine learning (training data) is necessary to assist in extraction phrases, which is unavoidable.

4.0 Project Requirement Specifications:

This artefact is a package used to enable collecting data from twitter and facilitates extracted information from the data collection to the front-end visualisation applications. Textual analysed data with the extracted entities related to annotation, geo-coding the address, content sentiment analysis and significant or important key phrase detection over the entities for each record of data. This package concentrates mainly on back-end processing of data accesses from data sources, in this context twitter Package is an integration of components. the tweet collector to populate accessed tweets in to database, NER (Named Entity Extraction) to classify the text annotations and significant phrase extraction, textual analysis for predicting emotional or sentiment analysis. It's purely back-end implementation. Annotation extraction, significant phrase extraction and sentiment analyser runs over the fetch data from twitter and get store in a database.

4.1 Requirements

4.1.1 Project scope:

This project intended to provide a database for visualisation, based on text analysis of each text record received from social networking database (twitter). It supplies formal data after get processed based on the specification, to front-end visualisation applications. So it needs to be platform independent, user friendly and easy maintenance. To satisfy usability of final outcome of this project, JAVA serves as object oriented programming language with assured platform independence. MYSQL as open source database it provides free accessibility and cost free with optimum performance. All the API's and libraries are of open source with easy access.

4.1.2 Software Requirements:

1. JDK 1.6.
2. MYSQL server 5.0.
3. Twitter API libraries.
4. Apache Tomcat version 6.0.
5. Stanford NER version 1.22.
6. Lingpipe 4.1.0
7. Lucence version 3.0.

Java JDK 1.6 is required because it's a platform independent and does justice to object oriented programming concept, for flexibility in integration with other systems and maximises the reusability of code. The database Mysql server 5.0 is chosen because of open source with vast number vendors and easy to maintain. Apart from this all the other software requirements are based on the prerequisites of the project implementation.

4.1.3 Functional Requirements:

1. Connect to twitter and fetch tweets from a geographic location of greater London.
 - a. Do not duplicate the fetched tweets.
 - b. Retrieve the metadata of each tweet along with text content.
 1. Tweet Id.
 2. Sender Id.
 3. Receiver Id.
 4. Sender Name.
 5. Receiver Name.
 6. Date and time of tweet creation.
 7. Profile Image of Sender.
 8. Geo Coordinates (latitude and longitude).
 9. Sending source.
 10. Sender's place.
 11. Actual text of the tweet.
 - c. Send http request to twitter for every 1 minute interval to fetch the tweets.
2. Find the geo coordinates of tweets which are not geo tagged.
 - a. Using the place of the tweet find out the geo coordinates.
3. Perform NER (Named Entity Extraction) on each tweet text and extract the Annotations including.
 - a. Organisation.
 - b. Person name.
 - c. Date.
 - d. Location.
 - e. Money
 - f. Time.
 - g. Percent.
4. Perform sentiment analysis on each tweet text and find out the sentiment of each tweet, and calculate the score may it is.
 - a. Positive.
 - b. Negative.
 - c. Neutral (zero).
5. Extract significant or key phrases from each tweet and store them in a string.
6. Create Database Table to store meta-data and original tweet text along with extracted information of geo-coordinates, Annotations, sentiment, and significant or key phrases.

4.1.4 Non-Functional Requirements:

1. Register the application with twitter and get the access keys.
2. As a client application to the twitter, we need to provide 'consumer key', 'consumer secret' and 'access tokens'. Update the twitter4j.properties file with consumer key, consumer secret and access tokens, to access twitter through twitter4j.
3. As server side system it needs high performance CPU configuration, requires minimum 2.4 GHz processing speed with a physical memory of 3 GB.
4. Java class path set to external library jars.
5. Global variables of MYSQL Database need to be configured, set 'query_cache_size' to 512 MB and
Set 'read_buffer_size' to 32 MB for performance tuning.
6. Check the query performance and adopt query performance measures, make sure that the Database table are properly index.
7. Study the specifications and configuration setting of external libraries and API's, while integrating with user application.
8. Java heap space should be taken into consideration, set java Run Configuration VM argument (jvm) as "-Xms<memory size>" here memory size is the max allocation size of physical memory. Example
"-Xms512M".

4.2 Use cases:

The above functional and non-functional requirements framed by taking consideration of use case diagram, here twitter is meant as twitter server, the twitter collector is the application (development project) and user the end-user who is focused to use the information in the database.

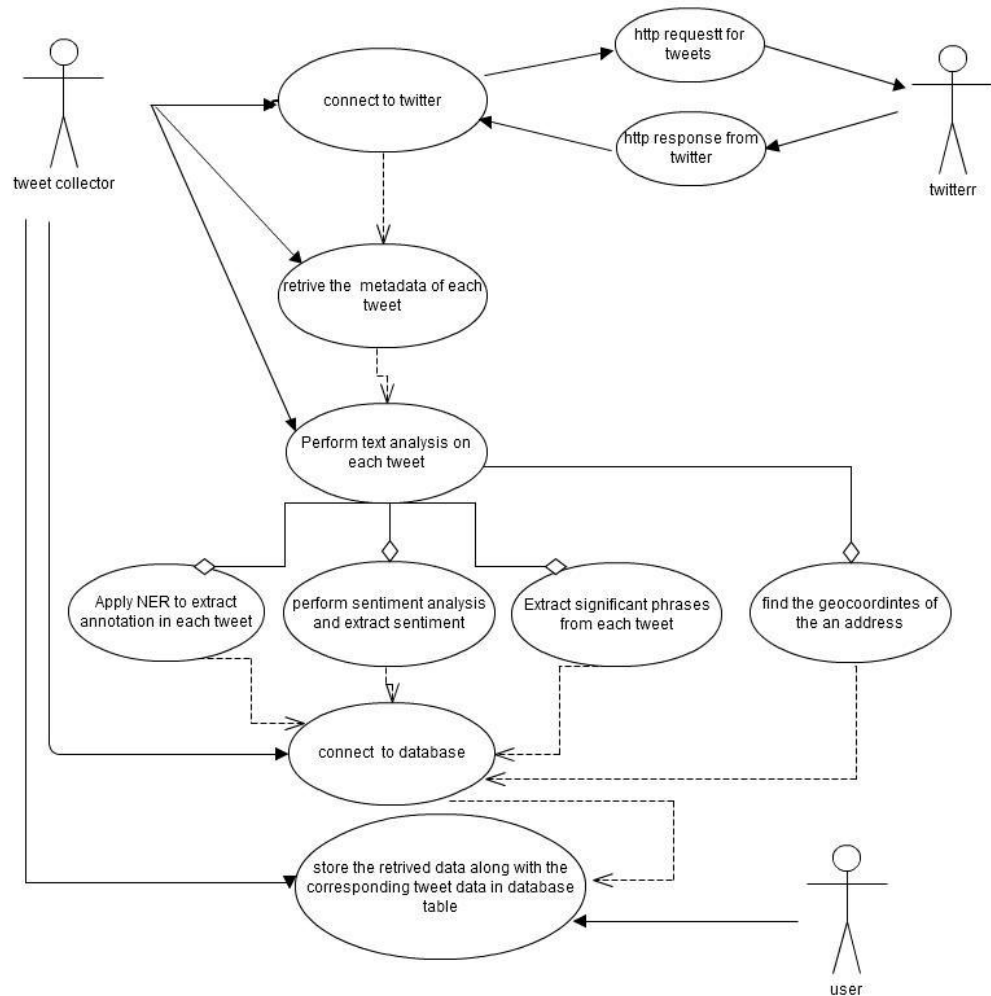


Figure 1: Use Cases

Figure 1 represents the use case diagram of the system. The interaction between different components of the software package and control flows is represented in above figure. Tweet collector is a main component that interacts with the twitter to get access based on search criteria. Tweet collector also is dependent of four modules annotations, sentiment extraction, significant phrases and geocoding. The main component interacts with all the modules and aggregates the data and get stored in the database. In this use case it is focused to facilitate a database of processed tweets fetch from the twitter. The processing involves the information extraction using Text analysis and NER (Named Entity Recognition). So that a user can make use of this information of processed data (processed tweet database) in various scenarios like visualization, and custom

application with requirements satisfies the usability of this database. The extracted data from raw tweet delivers users a clear understanding of dependencies in a tweet; also it simplifies the work of filtering information from data. Data encapsulation plays a role in implementation that end user didn't know the implementation logic of system also the user can only access the final database data. To connect twitter server proposed system requires internet connection, but the user can make use data by standalone applications

5.0 Analysis and Design:

5.1 System Design

The proposed system behaves as client-system to the twitter server while accessing tweets; it communicates through internet satisfying client credential of the twitter server. Another side of the proposed system it acts like server by interacting with Text analysing components, for geocoding, NER (Named entity recognitions for annotations, sentiment analysis and significant phrase extraction.

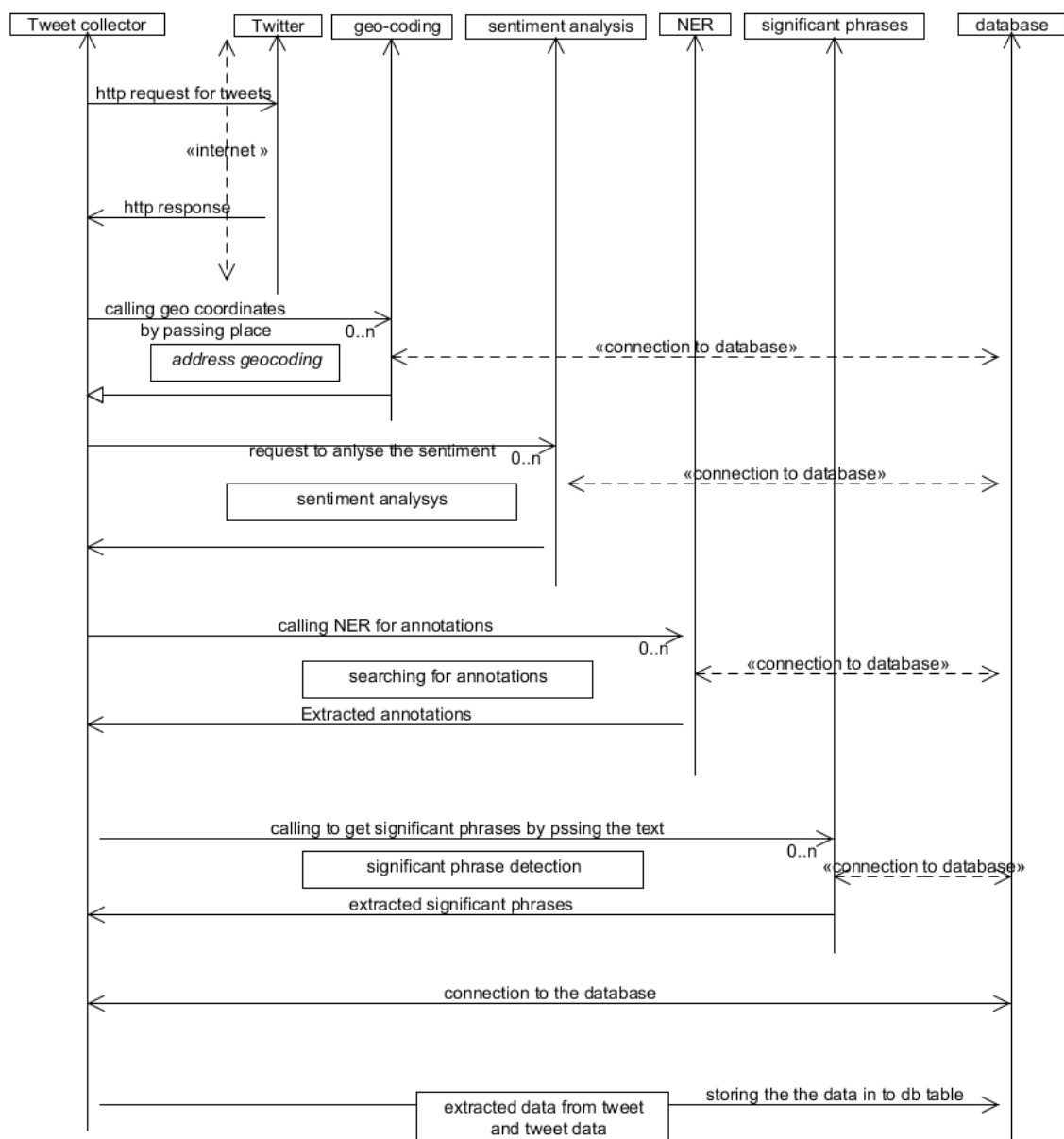


Figure 3 sequence diagram of the proposed system.

The above sequence diagram explains the design of operations, which take part in current proposed work. The CentralTweetcollector in the centre-stage for all operation, it used to integrate text analysis components or classes. CentralTweetcollector communicates with Twitter server through HTTP request and response protocols, and it communicate with text analysis component class locally through objects instance. Java facilitates the object oriented programming is an added advantage in reusability of the classes and embedding the library files with any extra. Twitter API is responsible for interpreting the http request and Responses between Twitter server and CentralTweetcollector, the connection is not long-lived as specified by the API requests are based on the query mechanism to search and fetch.

The text analysis components communicated MYSQL database for dependent training data, through JDBC connection. CentralTweetcollector is responsible for fetching tweets and conducting text analysis over each tweet and populating the database with extracted information from fetched tweets. After fetching it has to extract the metadata of the tweet, means the detailed information of a tweet.

S.No	Metadata	Details
1.	Tweet ID	Unique id of the tweet
2.	Sender ID	Id of the tweet sender account
3.	Sender username	User Name of the tweet sender
4.	Receiver ID	Id of the tweet receiver account
5.	Receiver name	User name of the tweet receiver
6.	Profile Image	Profile image of the sender account
7.	Latitude	Latitude (Geo coordinate) of the sender
8.	Longitude	Longitude (Geo coordinate) of the sender
9.	Place	Address of the sender
10.	Source	From which source application the send used to tweet
11.	Date and Time	Date and time on which the send used to tweet the text
12.	Tweet text	The actual text of the tweet (original content in tweet body)

5.2 Overview of the proposed system design:

CentralTweetcollector is the integration of every action that takes place in the system. it interacts with every component for responsible text analysis. Whereas it needs to connect the twitter server is primary to get access, for the sake of http connection to twitter server the twitter4j API acts as an interface, to reduce the weight on CentralTweetcollector to communicate over searches API of the twitter. The abstract method implementation focuses the re-usability and flexibility. The proposed system implementation in object oriented programming (OOP) concepts, it's preferred java because it satisfies requirement specifications. The individual components of text analysis didn't rely on each other, any dependency factor relative to logic development.

5.2.1NERextraction:

This module component of the application requires the external java library jar files, the class path need to be configured to respective jar files. NERextraction requires the training file to be stored in the corresponding location, to be specified by implementation logic. The extract annotations for a given text, returns string of tab separated annotations to the calling method of the CentralTweetcollector.

5.2.2 Geocode:

After receiving the place, it gives the corresponding geo-coordinates (latitude and longitude coordinates) of the place in return. Geocode need database to create table for dumping address with their corresponding geo-coordinates data. The following columns names should be there in table creation.

postcode	Latitude	longitude	Location	sub location
----------	----------	-----------	----------	--------------

5.2.3 Senticalculate:

This part project calculates the sentiment score of the given text based on the logic defined. In calculating the sentiment polarity whether it is positive or negative or neutral (zero), it needs to get interact with database table to get the scores of phrases. The table should be designed with taken care of the following column names.

Pos indicator	serial	pos	neg	word
---------------	--------	-----	-----	------

5.2.4 Significant phrases:

This component deals with significant phrase finding in text supplied by the CentralTweetcollector; in return it gives continuous string of extracted significant or key phrases in given text. It interacts with licence API libraries in correspondence with logic care should be taken in configuring the class path. Significant phrase Component demands a reference table to be created in the back-end for operational purpose. The table should contain the following column names.

Serial	phrase
--------	--------

5.3 Security concerns:

The implementation is abstracted from the user, that user can't modify the server side implementation. User has authorisations to query for a data in the database, not permitted for data modification commands. The username and root password is necessary to get read only access of database.

5.4 Databases:

The proposed system needs back-end database support for text analysis component operations, and to store the final outcome of the CentralTweetcollector data. The associated DB (database) tables of each individual operational purpose of text analysis components created with specified design. They are properly indexed for optimal performance database, to reduce the data retrieval time of query. Apart from the metadata extracted from the tweet along with supplied extract information from text analysis component should be populated to dataextraction table with corresponding column names.

5.5 CentralTweetCollector Class Diagram:

This class diagram represents the overall proposed system design without DAO Pattern and relative dependencies.

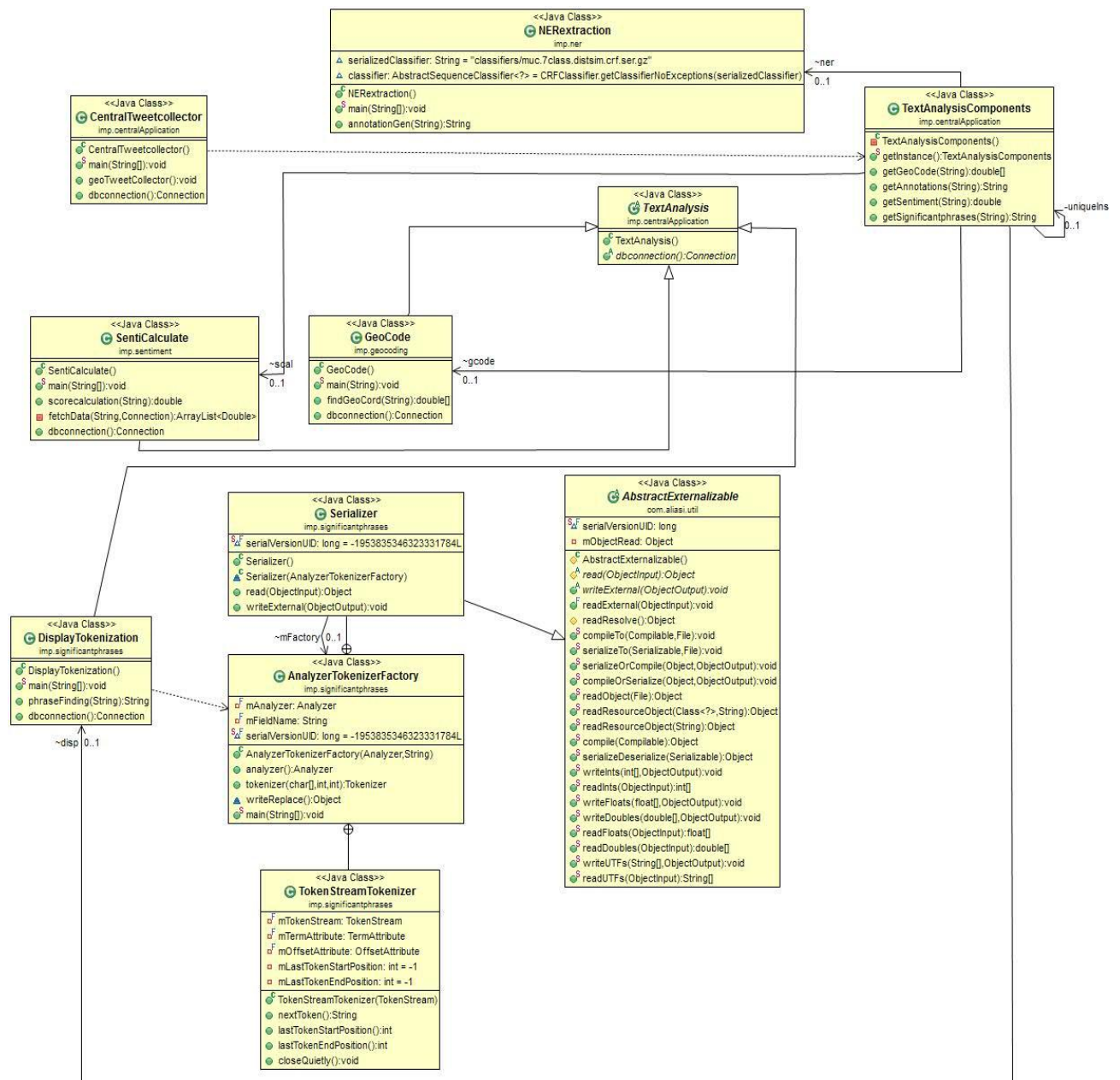


Figure 2 Class Diagram of CentralTweetCollector without DAO design pattern.

This class diagram represents the overall proposed system design with DAO Pattern and related dependencies.

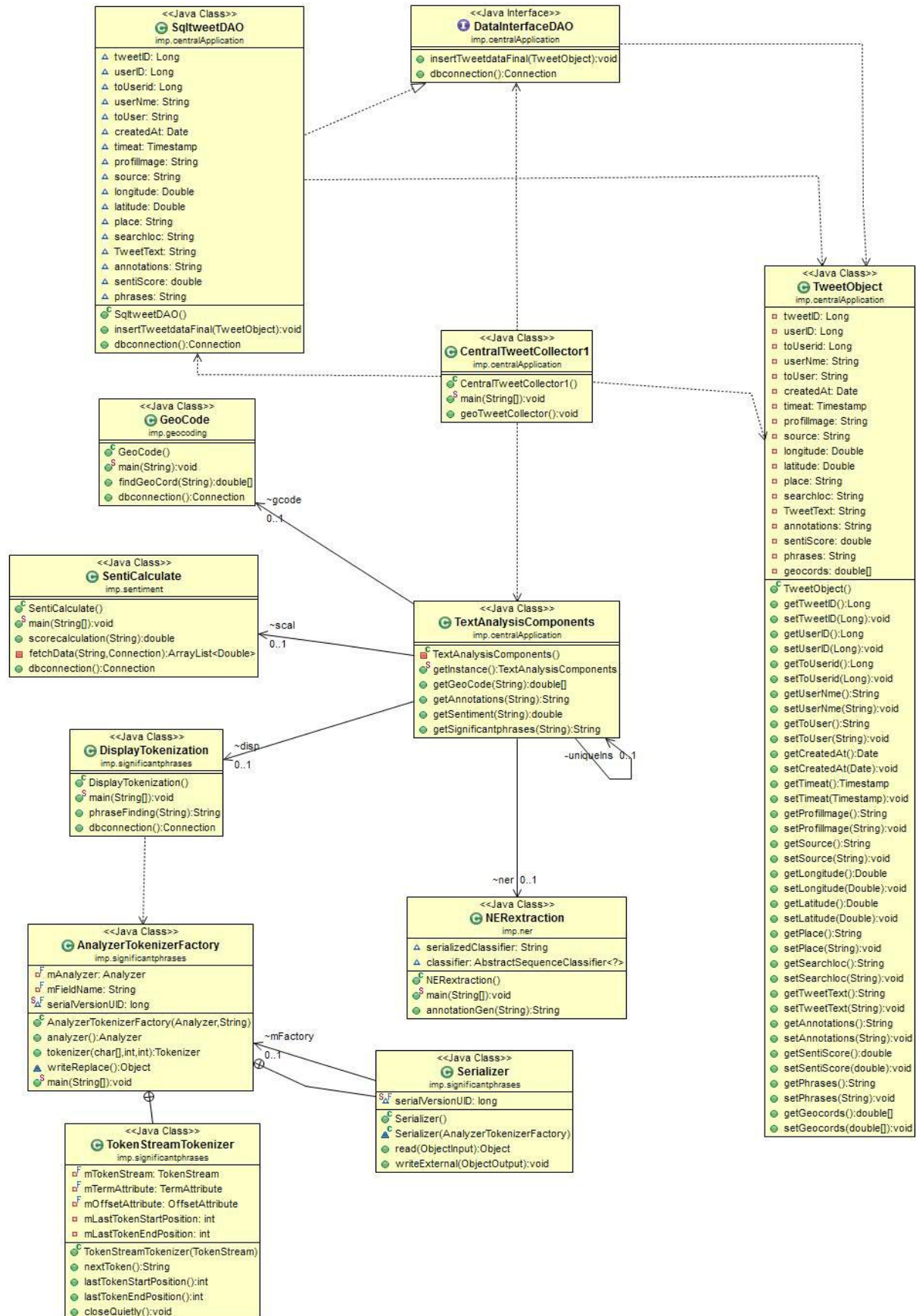


Figure 3. Figure 2 Class Diagram of CentralTweetCollector with DAO design pattern.

6. Implementation and Testing

6.1 Implementation:

The success of an implementation depends mainly on project design. The assumptions of design with respect to project requirements can be made into realities through planning and procedure over software development life cycle. Before starting the implementation care should be taken in reviewing the requirements, and to be cautious over configuring the system with proper understanding based on the requirement specifications of the project. After configuring system hardware and software specification, check the system thoroughly because it will influence to decide the upcoming phases of development.

6.1.1 Planning or approach for implementation

The proposed system is an integration of sub-systems; they are tweetcollector, NERextraction, GeoCode, sentimentcalculation and significant phrases. In this context it's evident to adopt bottom-up and top-down approaches. Top-down carries in implements sub-system based on focus of dividing the system in to sub-system with regards to the peculiarity in functionality and logical frontiers. Individual module components (sub-system) of system is tested and evaluated individually in top-down approach. While integrating the sub-system into a system, starting subsystems to a single system relative to the inter-dependencies between the sub-systems, in this regard bottom approach is appropriate.

6.1.2 Design patterns:

In this application supposed to implement DAO design pattern and Singleton. Have two versions of Tweet Collector implementations, one is intended to make use DAO design pattern and singleton design pattern. And another with only singleton design pattern.

6.1.3 TweetCollector:

TweetCollector is implemented in j2ee java and following singleton pattern and DAO design pattern. CentralTweetCollector is the main application class which integrates all the components involved in text analysis of tweets and storing in to database. In CentralTweetCollector the important method is geoTweetCollector () that handles the connection to the twitter server through twitter4j API, and collect the tweets based on the geo search query.

```
//instantiation of the scheduler
ScheduledExecutorService scheduler = new
ScheduledThreadPoolExecutor(1);

/*schedules this method creates and executes at a fixed time interval
delay * until the task get cancelled */
scheduler.scheduleWithFixedDelay(new Runnable() {           @Override

    public void run() {

        /* tweet search and text analysis code for extraction and */

    },1,1,TimeUnit.MINUTES);
```

In `geoTweetCollector ()` the `ScheduledExecutorService` used to schedule and repeat the job for every 1 minute interval between the executions of the code. Because the search API is not a long-lived connection to access tweets so, to continue accessing tweets from twitter server we need to run the code for connecting twitter at repeated intervals Every time at most it access 1500 tweet.

```
// setting the specific geo-coordinates of grates london
    GeoLocation locsearch = new GeoLocation(51.3026, 0.739);

    // creating instance of a twitter4j.TwitterFactory
    Twitter twitter = new TwitterFactory().getInstance();
    try {
        for(int i=1;i<16;i++)
        {
            /* setting Query options to search tweets by geo searching */
            Query qy = new Query();
            qy.setRpp(100);
            qy.setPage(i);
            qy.resultType("Mixed");
            /*specifying filter to search tweet with a radius of 100 miles
            * from london */
            qy.geoCode(locsearch, 100,"mi");
// checking the results perpage, access only if the results more than
10
            if(qy.getRpp()>10)
            {
                // implement the query for search the tweets
                QueryResult result = twitter.search(qy);
                List<Tweet> tweets = result.getTweets();

            }
        }
    }
```

Here search criteria defined for searching the tweets is geocode based query, it is implemented to search the tweets within the boundary 100 miles radius of London city . And the result types are specified as ‘mixed’ (all types of tweets). Collect the search query results in a list, to loop used to iterate each tweet to extract the metadata from the tweet apart from the tweet text (original message text content in a tweet).

We make use of TextAnalysisComponents class object 'tac' to call methods NERextraction::annotationGen(), SentiCalculate::scorecalculation(), DisplayTokenization::phraseFinding() and GeoCode::findGeoCord() through getter and methods of single instance of TextAnalysisComponents. It's evident by specifications of twitter every tweet is not geocoded, in sense it don't have latitude and longitude coordinate values. tweet.Geolocation() is null when the tweet is not geocoded one, in this scenario the alternative to extract through 'place' of the tweet if the sender use to specify the physical address as place name. Geocordinate values get unfolded by calling the getGeocode method.

6.1.3.1 Storage of persistent data using DAO design pattern:

Java platform offers one of the best technique which separates data access logic from the object persistence. Dao design pattern intended for implementation, specially related to data access and persistent data storage system tasks. DAO facilitates separation of business logic and data storage system, DAO provides an interface of abstract methods to interact with database. Grant the access to interact without exposing the database details to the application. Benefits of DAO are concerned for long term application maintenance, flexibility over the future changes in database without affecting the application logic and least bother about the data tier changes which do not need to change the business logic.

In this context of implementation of application, the extracted meta-data from the tweet used to send SqltweetDAO class (which implements DataInterfaceDAO Interface) by making use of simple-bean object called TweetObject. With the help of getter and setter methods of TweetObject class object, data (to store or access persistent data) communicated between CentralTweetCollector and SqltweetDAO. The DataInterfaceDAO interface facilitates data abstraction and polymorphism mechanism so, that CentralTweetCollector not aware of the SqltweetDAO code. This helps if there are any changes to be made in SqltweetDAO as future demands, it won't affect the functionality, but only a minimal changes may needed in CentralTweetCollector class.

6.1.3.2 Storage of persistent data without using DAO design pattern:

If there is no scope using DAO design pattern in data management system (RDBMS, Flat file or object data base), then its manual process of temporary storage in data variable and perform the DDL and DML operations of database with data supplied by data variables. But it's has a constraints tight-coupling and dependency factor over the business logic and data tier. There is no provision of flexibility of code over future changes in data tier, that it effects the entire application and forced to re-engineering the entire application.

6.1.3.3 Storing the persistent data:

The extracted data has to store in a table as persistent data to be accessed by the user. In due of this a table is created with relevant data types assigned to the column fields. And also the table get indexed properly for tuning the performance of a query. The required table 'dataextraction' indexed with tweet_id field. Connecting to the database to access required table is achieved through JDBC connection. To satisfy the functional requirement of 'no duplication of tweets in database', it's necessary to cross check the new tweet_id with the existing tweet_id's in the database. If query returns no values then populate the database with tweet data else skip that tweet.

```
ResultSet res1= stmt.executeQuery("SELECT * from dataextraction where
tweet_id = '" + tweetID + "'");
//checking for duplication of tweet through tweetID
    if(!res1.next()){
/* inserting tweet data along with the information extracted from
 * the text anlysis components of annotations, sentiment, geocoding,
 * and significant phrases */
st = conn.prepareStatement("INSERT INTO
`centraldatabase`.`dataextraction`
(`tweet_id`,`username`,`userid`,`touser`,`touserid`,`createdat`,`time`,
`profile_image_source`,`geolatitude`,`geolongitude`,`place`,`geolocatio
n`,`source`,`tweettext`,`annotations`,`sentiscore`,`phrases`) VALUES
(?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?,?)");

st.setLong(1, tweetID);
st.setString(2, userNme);
st.setLong(3, userID);
st.setString(4, toUser);
st.setLong(5, toUserID);
st.setDate(6, createdAt1 );
st.setTimestamp(7, timeat );
st.setString(8, profilImage);
st.setDouble(9, latitude);
st.setDouble(10, longitude);
st.setString(11,place );
st.setString(12,searchloc);
st.setString(13, source);
st.setString(14,TweetText );
st.setString(15,annotations);
st.setDouble(16,sentiScore);
st.setString(17,phrases);
st.executeUpdate();
```

6.1.4 TextAnalysisComponents:

This TextAnalysisComponents class act's as an interface to interact the text analysis components (annotation, geocoding, significant phrases and sentiment analysis). Singleton design pattern used here in implementation, this class is responsible for providing single instance of text analysis object's the seeking class. Singleton class makes it possible to serve

from a single point of access, so that we can regulate usage of expensive resources by maintaining a global state.

It provides the single instance to called methods of concrete classes' instantiated in the TextAnalysisComponent class. It presents data encapsulation concept, hiding the resources from CentralTweetCollector but, it make use though getter methods of TextAnalysisComponent class.

```
public class TextAnalysisComponents {
    //unique instance variable of TextAnalysisComponents
    private static TextAnalysisComponents uniqueIns;
    // declaration of concrete classes text anlysis components
    NERextraction ner;
    SentiCalculate scal;
    DisplayTokenization disp;
    GeoCode gcode;
```

```
private TextAnalysisComponents() {
    gcode = new GeoCode();
    ner = new NERextraction();
    scal = new SentiCalculate();
    disp = new DisplayTokenization();
}

public static TextAnalysisComponents getInstance() {
    if(uniqueIns == null)
    {
        uniqueIns = new TextAnalysisComponents();
    }
    return uniqueIns;
}

public double[] getGeoCode(String str){

    return gcode.findGeoCord(str);
}
public String getAnnotations(String str1){

    return ner.annotationGen(str1);
}
public double getSentiment(String str3){

    return scal.scorecalculation(str3);
}
public String getSignificantphrases(String str4){

    return disp.phraseFinding(str4);
}
}
```

6.1.5 GeoCode:

GeoCode sub-system is concerned about taking a physical address of variable formats and returns the appropriate geo-coordinate (latitude, longitude pair) values as output. In building this sub-system, due to difficulty of maintaining high frequencies of data comparisons in between input values and available storage persistence data round the globe. Limitations are

laid over finding the geo-coordinate values; by restricting to mechanism to frontier of specific country in this context UK (United Kingdom) is chosen. That means it operates address of locations residing in UK only.

Before starting the implementation of this sub-system, there is every need to examine the variations in coming input data from clients. Sometimes the meta-data variable of a tweet 'place' gives input value other than standard physical formats. After a close observation of inconsistent formats of incoming 'place' data listed out most frequent possible formats which are stated below.

Physical address formats of place	Example values
Region, sub-region or area	Reading, England
ÜT: <latitude>,<longitude>	ÜT: 51.472311,-0.090327
iPhone: <latitude>,<longitude>	iPhone: 51.375107,-1.114642

For operational purpose the sub-system demands a persistent reference data for predicting the geocode, in comparison with all possible matches of existing data available in custom database. The persistence data table 'locationdata' plays a key role in deciding the geo-coordinate values of an address. This is a custom build table with a collection of all possible UK regions and sub-regions or areas, postcodes with their corresponding latitude and longitude values.

Postcode	latitude	longitude	region	Sub-region
----------	----------	-----------	--------	------------

The important method of GeoCode class is findGeoCord() which takes a string of place value. Here it comes to decoding the incoming data formats, to extract latitude and longitude values exclusively if 'place' value is embedded with geo-coordinates.

Pseudo code:

```

If place length > 2 and contains a prefix of 'ÜT:'
Then
    split the place into tokens delimited by ','
    Assign corresponding tokens to latitude and longitude.

If place length >6 and contains a prefix of 'iPhone:'
Then
    split the place into tokens delimited by ','
    Assign corresponding tokens to latitude and longitude.
  
```

If place contain the value apart from the above mentioned patterns, then it need to analyse format and extract and segregate the regional and sub regional identities of address. And query the database for list of possibilities. If there is any appropriate matches in finding the geo-coordinates of place vale then in return it give latitude and longitude coordinates.

```

strToken[0]=strToken[0].trim();
strToken[1]=strToken[1].trim();
strToken[0]= StringEscapeUtils.escapeSql(strToken[0])
strToken[1]= StringEscapeUtils.escapeSql(strToken[1]);
ResultSet res1;
try {
    Statement stmt1;
    stmt1 = conn1.createStatement();
    res1 = stmt1.executeQuery("SELECT * from locationdata where
location like '%" + strToken[0] + "%' || location like '%" + strToken[1] + "%'
|| sublocation like '%" + strToken[0] + "%' || sublocation like
 '%" + strToken[1] + "%'");

if(res1.getFetchSize()==1) {
    if(res1.first()){
        latitude = res1.getDouble(2);
        longitude = res1.getDouble(3);
        locationString = res1.getString(4);
        sublocationString = res1.getString(5);
    }
}
}

```

```

else{
    while(res1.next())
    {
        lat = res1.getDouble(2);
        lon = res1.getDouble(3);
        locationString = res1.getString(4);
        sublocationString = res1.getString(5);

        if(locationString.contains(strToken[0]) &&
locationString.contains(strToken[1]) ){
            latitude =lat;
            longitude =lon;
        }else if(locationString.contains(strToken[0]) ||
locationString.contains(strToken[1])){
            latitude =lat;
            longitude =lon;
        }else if(sublocationString.contains(strToken[0]) &&
sublocationString.contains(strToken[1])){
            latitude =lat;
            longitude =lon;
        }else if(sublocationString.contains(strToken[0]) ||
sublocationString.contains(strToken[1])){
            latitude =lat;
            longitude =lon;
        }else{
            latitude =lat;
            longitude =lon;
        }
    }
} catch (SQLException e) {
// TODO Auto-generated catch block
e.printStackTrace();
}
}

```


6.1.6 NERextraction:

NERextraction is a sub-system responsible for extracting the annotations in a given text input. Here some limitations lay over the annotation types to extract from text of present context. Because of overloading training data to be provided for broad spectrum of annotation extraction of text which influence the system performance in operation, it should be restricted and narrow down to limited annotation types. In view of this problem the sub-system implemented to facilitate, only the specified annotation types to be extracted from the text as specified in requirements section.

This sub-system makes use of Stanford NER java libraries to classify the annotation types after comparing the given text by the classifier, and extracts the annotations. Linear chain CRF (conditional random field) sequence model implementation used in development of this Stanford NER classifier libraries for the annotation extraction. The training data is very important for declared functionality, the file 'muc.7class.distsim.crf.ser.gz' is necessary to be assigned to serializedClassifier.

```
String serializedClassifier =
"classifiers/muc.7class.distsim.crf.ser.gz";
AbstractSequenceClassifier<?> classifier =
CRFClassifier.getClassifierNoExceptions(serializedClassifier);
```

The NERextractor class takes the input text string through findGeoCord () method by passing the called parameter input into stTweet. For classification of sequential text need to be supplied to the classifyToCharacterOffsets() of the classifier class.

classifier.classifyToCharacterOffsets (stTweet);

```
public String annotationGen(String stTweet)
{
    //listing the triple's of extracted annotation information
    List<Triple<String, Integer, Integer>> l1 =
    classifier.classifyToCharacterOffsets(stTweet);
    //creating iterater listance to iterate the list
    ListIterator<Triple<String,Integer,Integer>> li=
    l1.listIterator();
    String st[][] = new String[15][2];
    int i = 0;
    while(li.hasNext()){

        Triple<String, Integer, Integer> tst =li.next();
        st[i][0]=tst.first();// annotation type
        st[i][1]=stTweet.substring(tst.second(),
        tst.third());// annotation value
        i++;
    }
    String str2 = " ";
    for(int j=0;j<st.length;j++){
        if(j< (st.length -1)){
            if(st[j][0]!=null)
                str2 = str2 +
        st[j][0]+"\\t"+st[j][1]+"\\n";
        }
    }
}
```

classifyToCharacterOffsets() return type is list of triples, which in return gives the extracted offset value, annotation type and annotation value in a list. Iterate the through these lists and access type and name pair to formulate into a string and returns continues string of annotations.

6.1.7 SentiCalculate:

As a sub-system SentiCalculate used to perform the sentiment analysis over the given text data, and calculate the accurate polarity score of the sentiment that elevated in sense whether it is positive or negative if else neutral zero. Before stepping forward into coding, it demands a training data for operational purpose so the sentiwordnet table is defined with training data. This table data is adapted from sentiwordnet3.0 in research of sentiment analysis. The table structure has the following column values parts of speech, serial no, positive score, negative score and phrase. SentiCalculate class have the method scoreCalculate that takes input argument of text to analysed, and tokenize the given sequence of sentences as tokens. For each token it query and fetch the positive and negative scores from table sentiwordnet, through the fetchData method.

```
public double scorecalculation(String st){
String line= st;
StringTokenizer token = new StringTokenizer(line," ");
double Score =0.0;
int count = 0;
String value = null;
Connection conn= dbconnection();
while(token.hasMoreElements())
{
    double posScore =0.0;
    double negScore =0.0;
    value = token.nextToken();
    //fetching the individual score values of each token in a arraylist
    ArrayList<Double> al = fetchData(value, conn);
    posScore = (Double) al.get(0);//positvie score of each token
    negScore = -1*(Double) al.get(1);//negative score of each token
    count+=1;// count of no of tokens wit hspecific score's assigned
    Score += posScore+negScore;// sum of posive and negative score
    al.clear();
}
// average of the sum of sscore and selected score values number
double scoreCatch = Score/count;
try {
    conn.close();
} catch (SQLException e) {
    // TODO Auto-generated catch block
    e.printStackTrace();
}
return scoreCatch;
}
```

The fetchData method receives tokens one at a time and queries the sentiwordnet db table after connecting, for positive and negative scores based where condition with a predefined

Reg-expression for pattern matching the word. “Where word like '%"+StringEscapeUtils.escapeSql(val)+"#%" ” After successful querying the qualified positive and negative scores of the phrase are returned to calling method.

```
// if the value contains the special character like "'" ,building the
specific sql query
if(val.contains("'")){
sql= "SELECT pos,neg,word FROM sentiwordnet where word like
'%" +StringEscapeUtils.escapeSql(val)+"#%'";
}
else{
sql="SELECT pos,neg,word FROM sentiwordnet where word like
'%" +StringHelper.escapeSQL(val)+"#%'";
}
ResultSet rs = stmt.executeQuery(sql);
double posScore =0.0;
double negScore =0.0;
while(rs.next())
{
String strch = rs.getString(3);
String strch1 = val.substring(0, 1);
if(strch.startsWith(strch1))
{
posScore = rs.getDouble(1);
negScore = rs.getDouble(2);
}
}
}
```

After getting the each individual positive and negative score, aggregate and average those by using word count in a sentence. The final score is the polarity of the sentiment of the given text and return the sentiment score to calling method.

6.1.8 DisplayTokennization (significant phrases/words)

The DisplayTokennization meant for extracting significant phrases from a given text. As a Primary task before starting implementation it's very much necessary to present the theoretical concept. The extraction of significant phrases from a text is carried out by stopping a list adverb's and noise words from a given text, after the given input got tokenized. For tokenisation process lucence (Bob, Mitzi & Breck, 2011) makes it possible to break in to token of alpha-numeric's, numerics and common word constructs based on indo-European languages. This is following list of stop words may it will get increased in coming-days.

```

Set<String> stopSet =
CollectionUtils.asSet("a","able","about","across","after","all","almost",
",","also","am","among","an","and","any","are","as","at","be","because","
been","but","by","can","cannot","could","dear","did","do","does","eithe
r","else","ever","every","for","from","get","got","had","has","have","h
e","her","hers","him","his","how","however","i","if","in","into","is","
it","its","just","least","let","like","likely","may","me","might","most
","must","my","neither",
"no","nor","not","of","off","often","on","only","or","other","our","own
","rather","said","say","says","she","should","since","so","some","than
","that","the","their","them","then","there","these","they","this","tis
","to","too","twas","us","wants","was","we","were","what","when","where
","which","while","who","whom","why","will","with","would","yet","you",
"your","can't","want","do","did","went","go","might","may","be","should
","would","get","move","shall","will","knows","know","on","below","top"
,"side","to.till","untill","at","on","for","ago","past","present","by",
"since","on","under","over","across","through","into","beside","next","
towards","onto","after","already","during","finally","just","last","lat
er","next","soon","now","always","every","never","often","rarely","usua
lly","sometimes.except","like","between","as","around","among","times",
"off","save","outside","unlike","via","witj","without","during","but","
plus","per","among","behind","before","following","along","inside","out
side","round","much","some","thinks","makes","up","down","being","http"
,"a","b","c","d","e","f","g","h","i","j","k","l","m","n","o","p","q","r

```

For operational purpose it demands for persistence reference database table, here the question comes what is the actual data that is populated by the table. It contains the dictionary words which are not so common, verbs, adjectives and nouns. This data help in finding the complex and significant terminology in a text. Now it comes to the actual implementation of DisplayTokennization class, phraseFinding method takes input string and gets tokenised by using IndoEuropeanTokenizerFactory class of lucence 3.0 (Bob, Mitzi & Breck, 2011) and StopTokenizerFactory() is used to remove the noise words from the list of tokens by sending IndoEuropeanTokenizerFactory instance and stop word list as parameters.

```

StandardAnalyzer analyzer = new StandardAnalyzer(Version.LUCENE_30);
AnalyzerTokenizerFactory atok = new
AnalyzerTokenizerFactory(analyzer,"foo");
TokenizerFactory stok= new StopTokenizerFactory(new
IndoEuropeanTokenizerFactory(),stopSet);
Tokenization tokenization = new Tokenization(text,stok );

```

Now the deciding factor is to check the each tokens presence in 'phrases' table. If the query to phrases' results a match then, the token get appended to the string else it get it not get appended and continues to consecutive one.

```

for (int n = 0; n < tokenization.numTokens(); ++n) {

    String token = tokenization.token(n);
    Statement stmt1;
    try {
        stmt1 = conn1.createStatement();
        ResultSet res1 = stmt1.executeQuery("SELECT
list1 from phrases where list1 =
'" + StringEscapeUtils.escapeSql(token) + "'");
        if (res1.next()) {
            strRes = strRes + token + " ";
        }
    } catch (SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
    }
}

```

6.2 Testing:

To inspect the project implementation satisfies the requirement specifications or not, project testing phase is crucial and fundamental. Apart from system functionality verification, also focused to uncover the coding errors in implementation of logical design. Validating the software developed and verifying the system performance. In this project testing priority is given to focus on consistency (correctness) and performance of code. For checking code correctness testing is carried out in 3 levels, at first level after each unit of a sub-system was developed and it will be tested then only reach the second unit as a developer this is called unit testing. At the second level after a successful implementation of a sub-system by binding all the unit's together, the functionality is tested it's called functional testing. At the third level all the well developed sub-system integrated together and tested called as integration testing.

Were as for implemented code performance is tested individual sub-systems and a system as a whole. Project will be successful only after successfully completion of three levels of testing along with performance testing in this scenario.

At level one:

In this level a particular purposeful unit of code being tested individually after development. Unit testing is a testing of independently each unit to check developed code is doing what suppose to do or not. CentralTweetCollector is an integrated system of different modules each of projected sub-system with specific purpose. As preliminary level of testing small units of code behaviour and state need verification, to exercise the variation between implementation design and code developed. For developer's fusibility Junit provides the framework for unit testing. The TextAnalysisComponent contains number of individual sub-system; in this context java classes have specific state and behaviours of each GeoCode, Sentalculate, Display Tokenization and NERextraction are tested individually to confirm code development implements project design.

At level two:

After successful completion of level code unit testing, Functional testing is recommended to be carried out to verify the code is satisfying the functional requirements of the project or not. This level of testing is intended for relative functionality of small units together in a sub-systems. In this context of project GeoCode, Senticalculate, Display Tokenization and NERextraction are independent sub-systems, it required to carry out functional testing individually on each sub-system.

Without integrating the CentralTweetCollector with Text Analysis components functional testing will be conducted, testing should be performed by providing input to generate output in view of expected results. Now CentralTweetCollector is tested for HTTP request for tweet and response from the twitter server is tested as a unit of development, because it's a primary data which is fundamental for project.

After configuring the twitter4j.properties with access tokens and keys provided twitter authorisation, execute the CentralTweetCollector for connecting to twitter to request user query. Repetitive schedule of execution with a specified delay time has to test.

Test case	steps	Results expected	Result
1.	Prerequisites : a. Configure the twitter4j. Properties with access tokens and key Execute the CentralTweetCollector to call geoTweetCollector method	In response: tweet with metadata Or Server busy with error message	Same as expected
2.	Set the delay to 1 min And execute the geoTweetCollector method	Executes the geotweetcollector to connect twitter recessively with 1min delay	Same as expected

The database connection is tested after a db table is properly structured. The code implementation dbconnection method which request for a driver connection to the corresponding database needed to be tested.

Test case	steps	Results expected	Result
1.	Prerequisites : a. Start the Database server b. Check the database table whether properly structured or not execute the dbconnection method	Output : Connected to database Or Connection failed	Same as expected

The GeoCode sub-system functionality of finding the latitude and longitude coordinates of a place to be tested, by functional testing of the findGeoCord behaviour long with dbconnection behaviour. To verify the resulting outcome with expected one.

Test case	steps	Results expected	Result
1.	Prerequisites: a. Check the db table location data is properly structured and populated with training data. b. dbconnection to the geodata call the findGeoCord method through the instance of GeoCode	In response it gives the latitude and longitude coordinates Or Null value	Same as expected
2.	Prerequisites : a. Start the Database server b. Check the database table whether properly structured or not execute the dbconnection method	Output : Connected to database Or Connection failed	Same as expected

The sentiCalculate sub-system is responsible for calculating the score of sentiment of the given text. The unit tested code of scorecalculation and fetchData of sentiCalculate functionality has to be at level 2 by functional testing. It does suppose to verify the functionality and behaviour of sentiCalculate sub-system.

Test case	steps	Results expected	Result
1.	Prerequisites : a. Check the database connection to the senwordnet database table. b. Sentiwordnet table is get populated with reference data for sentiment calculation. Execute SentiCalculate class and create instance and call scoreCalculation behaviour.	Output: gives the sentiment of the input text	Same as expected
2.	Run dbconnection method of SentiCalculate to get connection to database sentiwordnet.	Connected to database or Database Connection failed	Same as expected

NERextraction is a sub-system implemented to extract annotations from the text, to check whether the developed code which is unit tested supposed to behave according to the design specified or not. Functionality of the annotationGen implementation to be verified and

validated does it satisfies the requirements specified for implementation. Before going to run this annotationGen, taken care in setting the path to 'muc.7class.distsim.crf.ser.gz' file, which is a training data file meant for annotationGen reference to annotation types.

Test case	steps	Results expected	Result
1.	Prerequisites : a. Configure the Stanford NER library jar file by setting class path. b. Training file should be specified to 'serializedClassifier'. Run annotationGen for functional testing	Output: Extract Person, Location, Organisation, Date, Money, Percentage, Time from the text Or Null values	Same as expected

The concept if Significant or key phrase extraction sub-system is implemented through DisplayTokenization class. The unit tested code in building the phraseFinding method along with dbconnection functionally tested by functional testing. Lucence library jar files have to be set at class path before phraseFinding method functionally tested.

Test case	steps	Results expected	Result
1.	Prerequisites : a. Identify and list out the stop words for functional purpose. b. Phrases table is get populated with reference data for comparison and identification of phrases in text. Execute DisplayTokenization class and create instance and call phraseFinding.	Output: gives the significant /key phrases/words of the input text	Same as expected
2.	Run dbconnection method of DisplayTokenization to get connection to database Phrases.	Connected to database or Database Connection failed	Same as expected

At Level Three:

After completing sub-systems functional testing it comes to integration, the final system composed of sub-systems as stated in the beginning. At this level all the sub-system need to integrate after functional validation and inconsistencies in code has been verified.

The final system comes in two versions first one is implement using singleton design pattern, and the second a combination of DAO and Singleton pattern used while implementing. So both system integrations are to be tested. The final system CentralTweetCollector integrated with sub-systems NERextraction, GeoCode, DisplayTokenization and SentiCalculate. The test cases written at level two needs slight modification to run on final system integration, necessary changes has to made and run the test cases a part from new test cases. By doing this the level 3 testing procedure get completed on complete system.

7.0 Project Evaluation:

Project evaluation table:

Evaluation Question to be addressed	Type of Data needed.	Data Collection method.	Field of data collection.	Tested at level	Analysis Method	Information Consent needed?
Do the CentralTweetCollector connect to twitter by HTTP request and response mechanism?	Quantitative	After registering application, need to send the http request to twitter server through twitter4j API, and receive the response	CentralTweetCollector or and Twitter server	Unit, Functional and Integration testing.	Verifying The response from twitter servers and checking the data in response to the http request.	No
Does the no data duplication criteria implemented?	Qualitative	Checking the dataextraction database table based on unique tweet id	CentralTweetCollector or	Unit testing	Verifying the database table for duplicates for a tweet id	No
Does Central TweetCollector request twitter for every 1 minute interval?	Quantitative	Scheduling the code to connect the twitter server every 1 min interval between requests	CentralTweetCollector or	Unit testing	Validating the code through unit testing results.	No
Does the connection to database work fine and delivery their maximum performance?	Quantitative	Check the database connection through dbconnection method	MySQL database and CentralTweetCollector or	Functional and Integration testing	Examine the transaction rate in between application and database.	No
Does the application (project) get registered and get access key for authentication of twitter?	Quantitative	Check the http request doesn't throw any error while requesting for search tweets.	Twitter server and twitter.properties.	Functional testing	Results of functional test cases for the application	No
Class path has been properly configure to referencing java library jars	Quantitative	Check the class path jar files.	Project configuration	Unit testing	Manual checking of the class path destination.	No
Are the system hardware is configure based on non-functional requirements of project?	Quantitative	Manual checking of the system hardware configuration.	System hardware settings.	Unit and functional testing	Configuration setting and resulted output's	No
Are the database table properly structured and populated with training and reference data?	Qualitative	Manually checking the database table indexes and structure.	MySQL database tables.	Unit , functional and Integration testing	Results of test cases executed for positive test scenario.	No
The GeoCord sub-system extracted the latitude and longitude relatively to the given place?	Quantitative	Data collected through findGeoCord method for a given test case.	CentralTweetCollector or and Geocode sub-system.	Unit, Functional and Integration testing	Results of test cases executed for negative test scenario.	No
Does the annotations generated for given text?	Quantitative	Checking annotationGen method based on test cases defined.	CentralTweetCollector or and NERextraction sub-system.	Unit, Functional and Integration testing	Results of the test cases executed to check.	NO
Does the sentiment in the text is analysed to predict the sentiment score?	Quantitative	Collect the predicted scores of text through scoreCalculation and fetchData behaviours and states	CentralTweetCollector or and SentiCalculate sub-system.	Unit, Functional and Integration testing.	Test Results of the functional test cases for the scoreCalculation and fetchData	NO
Do the key/significant phrases get extracted?	Quantitative	Collect the extracted significant phrases of phraseFinding method by defined test case.	CentralTweetCollector or and DisplayTokenization sub-system.	Unit ,Functional and Integration testing	Test result of the unit and functional testing of DisplayTokenisation.	NO
Does the performance of the application justify the system requirements?	Qualitative	Collect the execution and response times of sub-systems and the final integrated system, by interpreting run-time response time based on processing the tweets.	CentralTweetCollector and all sub-system integration.	Functional and Integration testing	Results of performance testing, analysed with time and processing rate optimisation.	NO

In this project duration, we evaluated multiple text analysis components along integration system that can possibly facilitate in text engineering. This extracts viral information from the raw data supplied. Based on review of contemporary technologies and detailed study of available application existing or built so far, we defined a list of functional and non-functional requirements for a projected system building and implemented a system according to the specifications and using upfront technologies.

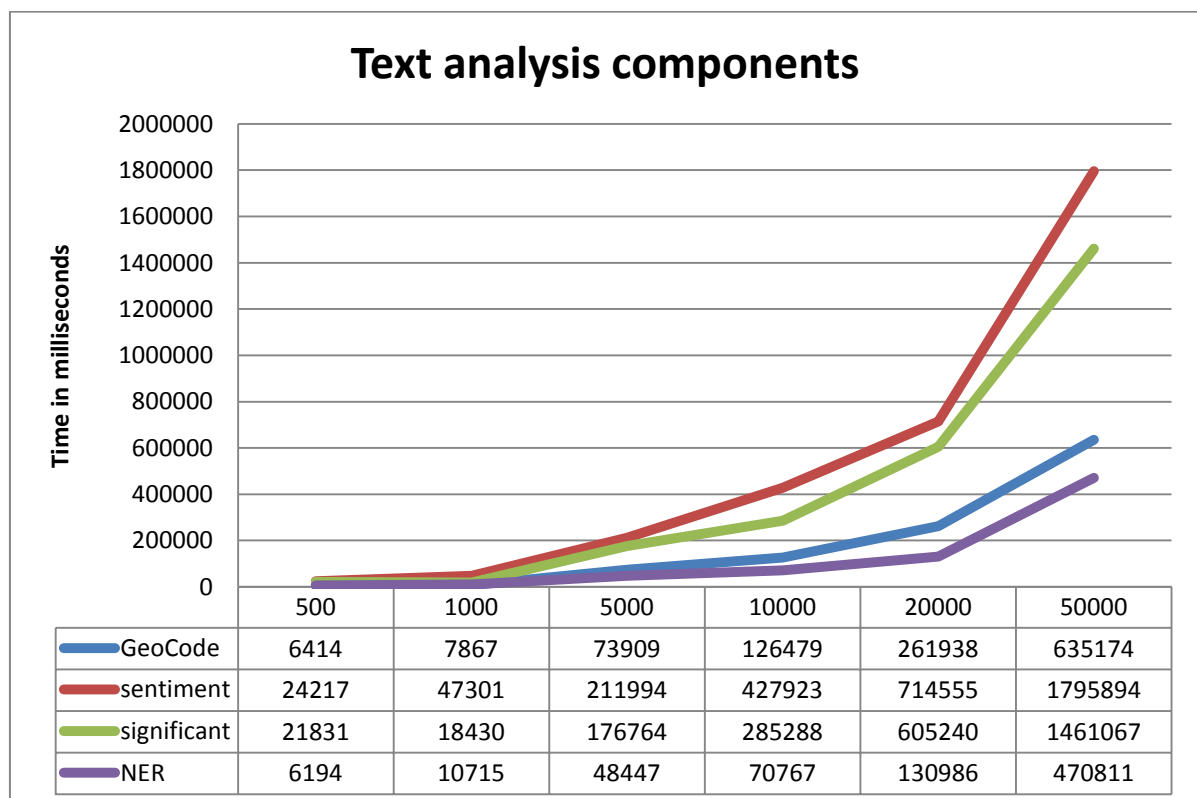
The specifications and technologies made it possible to develop system and proved the system is supposed to do so. The final system outcome do justifies most of the requirements outlined in the project document.

7.1 Requirement specification evaluation:

Based on the evaluation of test results, it is apparent that we met all software, functional and non-functional requirements of project listed in 4.1.2 to 4.1.3 sections. This project implementation code was tested on a laptop with built-in configuration of corei-3 2.5 GHz processor with 2 GB RAM. This is an affordable and optimum configuration in usability. All the tests conducted for this project evaluation are successful and satisfactory performance is achieved.

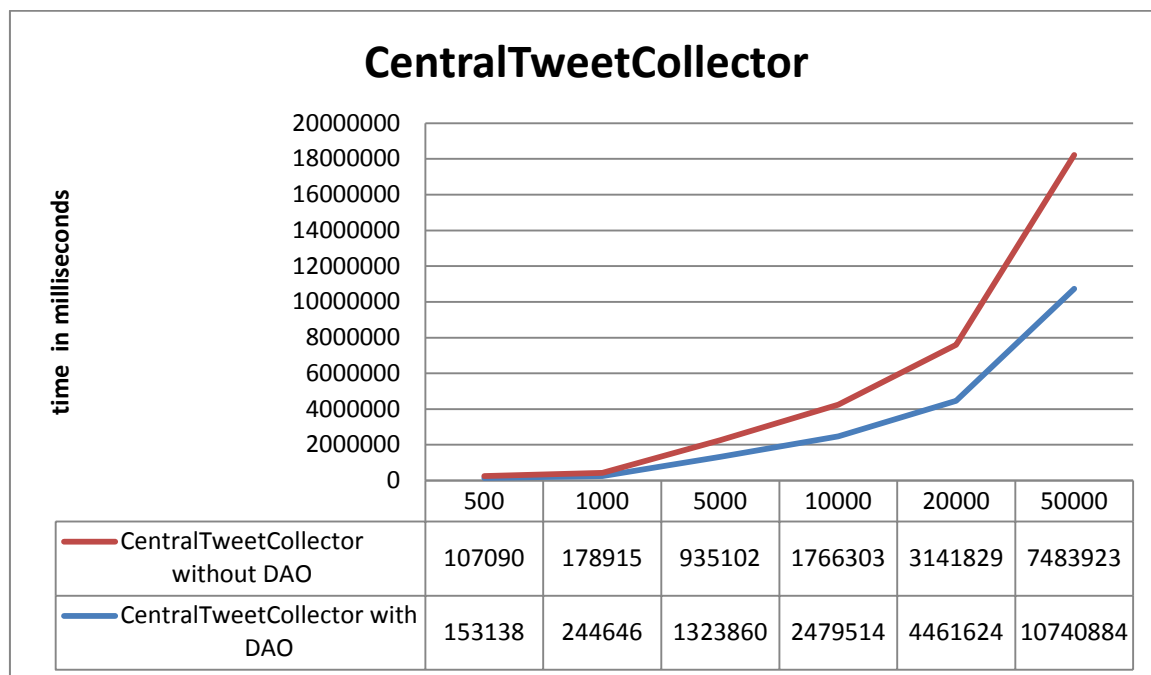
7.2 Performance testing results:

Once the code and design validation get completed on developed system, now performance of implemented system needs to be validated. Time is taken into consideration in performance testing; here time for processing the incoming data from twitter and populating the information into database is important to test application efficiency.



The above chart shows relation between number of tweet processed and time in milliseconds. Here x-axis is taken as the no.of.Tweets to be processed the scale is taken for 500, 1000, 5000, 10000, 20000,500000. And y-axis is taken as times to get processed the tweets time is taken as milliseconds. Comparatively sentiment extractor takes more processing time and NER takes least processing time than the remaining sub-systems. It shows how the sub-systems individual performance based on the execution time and response time.

The integrated system of CentralTweetCollector of both versions has to be tested for performance analysis. It resulted the processing time CentralTweetCollector system is on averages 500 tweets per 107090ms where as CentralTweetCollector implemented on DAO design pattern is 500 tweets per 153138ms.



There is drastic variation in performance between implementation with DAO Design Pattern and without DAO design pattern. These testes are based on the configuration specified in non-functional requirements specification; this may vary with hardware configurations of the system. It needs to review both implemented versions based performance tests, which will be discussed in evaluation section.

7.3 Performance evaluation:

Performance of implemented system tested as whole and also tested sub-systems individually the test results are satisfactory. As mentioned earlier the proposed project implementation was developed in two versions, one in which developed without Using DAO pattern and the second with DAO pattern. Both the system implementation are tested, the test results are varying in between the two system. The one which implements DAO is not giving satisfactory performance compared to other implemented without DAO. The reason

for this is system hardware configuration, to achieve better performance of system implemented DAO need higher configuration than specified.

7.4 Project Demonstration:

This project work in action connects twitter through http request protocol and gets access to tweets based on the search query defined. Each tweet comes along with meta-data, means details of the tweet like sender and receiver. Let us take a single tweet, the following are the meta-data u can extract from a tweet u get accessed from twitter.

Data fields	Expected data	Response data of single tweet
Tweet ID	Unique tweet id	Same as expected '89812375716372481'
Sender ID	Tweet sender id	Same as expected 'ZorkiefromSpace'
Sender username	Username of sender	Same as expected '330008142'
Receiver ID	Tweet receiver id or null value	Same as expected
Receiver name	Tweet receiver user name or null value	Same as expected 'iToolan'
Profile Image	url to download profile image	Same as expected 'http://a2.twimg.com/profile_images/1425172234/IMG00233-20110703-2143_normal.jpg'
Latitude	Null or latitude value	null
Longitude	Null or longitude value	null
Place	Physical address or urban tweet coordinates or iphone geo-coordinates	Same as expected 'ÜT: 51.7491724,-0.3091564'
Geo location	Null or geo-coordinate pair	null
Source	url of the source	Same as expected 'ÜberSocial for BlackBerry'
Date and Time	Date and time of format "day, date month year HH:MM:SS" example : "Wed, 19 Jan 2011 21:16:37 +0000"	Same as expected "Sat, 9th jul 2011 22:45:13"
Tweet text	Actual text message	'Its been a productive day #timeforbed '

For a single http search request to twitter we can access 1500 tweet in response, so the system can continues send request to twitter server with an interval 1 min in between request for tweets. After u gets access the tweets, next step is to process the tweets for text analysis to extract information. Geo-coding the place, the subsystem GeoCode used to find out latitude and longitude coordinates for a given place extracted from the tweet. We identified the formats of place most probably in the following

Sub-system	Probable place data formats	(Latitude, longitude) results
GeoCode	<ul style="list-style-type: none"> • ÜT: 51.472311,-0.090327 • Reading, England 	<ul style="list-style-type: none"> • 51.472311, -0.090327

The remaining sub-systems in the CentralTweetCollector, which NERextraction, SentiCalulate and DisplayTokenization used to give's output for given text as follow's

Sub-system	Input text data(tweet text)	Resulted output (extracted information).
NERextraction	' hi Am good, got job in Microsoft offered \$40000 and relocating to California, peter is coming with me ,return back on next month'	ORGANIZATION Microsoft MONEY \$40000 LOCATION California PERSON Mr.peter
SentiCalulate	' hi Am good, got job in Microsoft offered \$40000 and relocating to California, peter is coming with me ,return back on next month'	0.03409090909090909
DisplayTokenisaton	' hi Am good, got job in Microsoft offered \$40000 and relocating to California, peter is coming with me ,return back on next month'	job offered relocating California peter

The integrated system CentralTweetCollector designed to populate the table with metadata of tweet along with information extracted by text analysis. Some of the accessed tweets don't have latitude and longitude coordinated, for that GeoCode sub-system used to serve the latitude and longitude for specified place of tweet.

serial	tweet_id	username	userid	touser	touserid	createdat	time	profile_image_source	geolatitude	geolongitude	place	geolocation	source
1	898123820192768	predatron	243045742	IDrawBones	6056821	2011-07-09 00:00:00	2011-07-09 22:45:15	http://a0.twimg.com/profile_images/1415024192/5010...	53.4692	-2.07711	Kent, UK	NULL	<a href="http://twitter.com/#!/down re..."
2	8981238243949965	annecupcake	4206530	IDrawBones	6056821	2011-07-09 00:00:00	2011-07-09 22:45:15	http://a2.twimg.com/profile_images/1422180235/avat...	51.500162	-0.126236	London	NULL	<a href="http://twitter.com/#!/down re..."
3	8981238242308872	ItsCharlotte_xx	254946149	IDrawBones	6056821	2011-07-09 00:00:00	2011-07-09 22:45:15	http://a2.twimg.com/profile_images/1423135033/imag...	53.3392	-1.39888	Kent, England	NULL	<a href="http://twitter.com/#!/down re..."
4	89812382328603778	marraistamshiri	11434174	BabaPixie	9956888	2011-07-09 00:00:00	2011-07-09 22:45:15	http://a3.twimg.com/profile_images/1384307948/1108...	51.500162	-0.126236	London	NULL	Ube...
5	89812381907383808	RAW_fashionBTCH	318545283	__JawanHassan	293714240	2011-07-09 00:00:00	2011-07-09 22:45:15	http://a1.twimg.com/profile_images/1415091288/RAW_...	51.500162	-0.126236	London	NULL	we
6	89812380942471168	doctorkanayo	87675487	__JawanHassan	293714240	2011-07-09 00:00:00	2011-07-09 22:45:14	http://a2.twimg.com/profile_images/1419491412/imag...	51.5209	-0.126968	Croydon, London	NULL	we
7	89812380506271744	CB_YNWA	273418761	danielpond1995	209747621	2011-07-09 00:00:00	2011-07-09 22:45:14	http://a2.twimg.com/profile_images/1312268884/2838...	51.4502	-0.0442	51.450200, -0.044200	NULL	<a href="http://twitter.com/#!/down re..."
8	89812380472721408	DonSevensi	81626019	AJamaze	7015783	2011-07-09 00:00:00	2011-07-09 22:45:14	http://a3.twimg.com/profile_images/1432970637/3161...	51.472311	-0.090327	Ut:	NULL	Ube...

k All / Uncheck All With selected:

Show: 8 row(s) starting from record # 8

initial (rotated headers) mode and repeat headers after 100 cells

Page number: 1

Figure 4 shows Database table 'dataextraction' populated with the final system out-come.








Options			
			
			
serial	31	32	
tweet_id	89812377226330113	89812377150832640	
username	charli_watson	livbarryx	Maxine1994babyy
userid	255010702	47149725	
touser	JimboStudios	JimboStudios	tiffaaaani
touserid	4276228	4276228	
createdat	2011-07-09 00:00:00	2011-07-09 00:00:00	2011-07-09 00:00:00
time	2011-07-09 22:45:14	2011-07-09 22:45:14	2011-07-09 22:45:14
profile_image_source	http://a1.twimg.com/profile_images/1325437881/1800...	http://a3.twimg.com/profile_images/1420470679/prom...	http://a1.twimg.com/
geolatitude	51.6856	0	
geolongitude	-0.300488	0	
place	Bishop's Stortford, Herts, UK	upminster Essex	London
geolocation	NULL	NULL	NULL
source	<a href="http://blackberry.com/twitter" rel="nofollow..."	<a href="http://twitter.com/#!/download/iphone" re...	<a href="http://black
tweettext	Hahaha that really reminds me of my ex #themarria...	when u'r sister meets @JoeyEssex_ before you #peak	@tiffaaaani noo I air
annotations	null	null	null
sentiscore	0	0.03125	
phrases	really reminds ex	sister meets peak	time
Check All / Uncheck All With selected:   			

Figure 5 shows Final outcome of individual tweet.

8.0 Critical Evaluation of project and self reflections:

It was a great experience from starting to end of this project. At the beginning while drafting the proposal am not sure how am going to implement system. Because this is a new area of study to find a complex solution, which is not came across earlier. While reviewing the literature the literature I come to know, what actually I have to do. Assumptions are confirmed while analysing the literature of existing work on text analysis and text engineering. I framed my Aims and objectives to move ahead in building the system.

I learned many aspects in the course of implementation of a system, i learned from my mistakes and resolve issues as quickly as possible with in time. Developing this project is challenging for me sometimes I experienced frustration but, the joyful movement is when i find a solution to the problem. I step ahead and improved y Skill set in java programming and MYSQL database administration, and by doing this project i came across the concept of text analysis, information extraction, NER (Named Entity Recognition).

I do justice to this project with all possible ways. The implementation is satisfied all the functional and non-functional requirements. Test results showed better performance based on the built-in hardware configuration, on which i implemented this system. But, as a critic I can say there are some areas, which need to re-engineer for review to second level. Significant Phrases sub-system is the one which is not fully implemented due to time constraint. Apart from this everything do completed successfully in time.

9.0 Conclusion:

In this Project we aim to serve a processed twitter tweet database to frontend third party visualization applications. Text analysis focused on processing the tweets to extract information from the raw data of tweet, which can benefit the frontend application in projecting more information to the user, in terms of usability and exploring text-engineered data. In delivering a quality solution we conducted early tests on the application proto-type, which is very useful to learn from mistakes and resolved the issues. The project delivers a mechanism to facilitate NER (Name Entity Recognition) in extracting the annotation, Sentiment analysis over the text, significant phrase identification in a text and converting the physical address into Geographic coordinates (latitude and longitude) on tweets accessed from twitter in building a large twitter database, which is Readily available to use for various visualization tools and application back-end data. Am here to conclude we fulfilled the methodologies of possible information extraction by processing the data, to provide in-depth details of social networking data.

Future work:

The scope of future improvements lies in Geographical coding, significant phrases. Due to time constraint we focused to implement a system that finds latitude and longitude of address that resides in uk only. We can upgrade it to global address geocoding. Secondly there is vast scope of research in significant phrase detection for a given text, the possibility of finding word frequency for period of time interval of accessed tweets, comparison word frequencies of group of tweets accessed in a specific period and list them out. We do process one social network there are various social networks available in the market, so at next level i can propose extracting information by processing the data from heterogeneous and multiple social networks together and built a single point of access to database. Further study and research is necessary in building robust system which we are unable to discuss in this work.

10.0 References:

Alias-i (2008). *Named Entity Recognition*. Available: <http://alias-i.com/lingpipe>. Last accessed 4th May 2011

Bob Carpenter, Mitzi Morris, Breck Baldwin. (2011). The Lucene Search Library. In: Bob Carpenter, *Text Processing with Java 6*. <http://alias-i.com/lingpipe-book/index.html>: LingPipe Inc. p1-22.

Cunningham, et al. *Text Processing with GATE (Version 6)*. University of Sheffield Department of Computer Science. 15 April 2011. ISBN 0956599311

Ela Dramowicz. (2004). *Three Standard Geocoding Methods*. Available: <http://www.directionsmag.com/articles/three-standard-geocoding-methods/123627>. Last accessed 8th Jul 2011.

Goldberg DW, Wilson JP. USC WebGIS Services. Available online at <https://webgis.usc.edu>. Last accessed 7th Jul 2011.

Jenny Finkel. (2006). *Stanford Named Entity Recognizer (NER)*. Available: <http://nlp.stanford.edu/software/CRF-NER.shtml>. Last accessed 19th Jun 2011.

Kwak Haewoon, Changhyun Lee, Hosung Park (2010). What is Twitter, a social network or a news media?. *In the 19th international conference on World wide web*. Raleigh USA, April 20. USA: ACM.

Mano Marks. (2010). *Geocoding Strategies*. Available: <http://code.google.com/apis/maps/articles/geocodestrat.html#intro>.

Nadeau, D. Turney, P. & Matwin, S. (2006), Unsupervised Named-Entity Recognition: Generating Gazetteers and Resolving Ambiguity, pp. 266-277.

Rahman Mukras, Nirmalie Wiratunga, and Robert Lothian (2007). Selecting Bi-Tags for Sentiment Analysis of Text. *In SGAI International Conference on Artificial Intelligence*. Cambridge, 2007.

Release 1.0. (2010). *Yahoo! PlaceFinder Guide*. Available: <http://developer.yahoo.com/geo/placefinder/guide/>.

Rudy Prabowo, Mike Thelwall. (2009). Sentiment Analysis: A Combined Approach. *Journal of Informetrics*. 3 (2), p143-157.

Turney, P.D. (2000), Learning algorithms for key phrase extraction, *Information Retrieval*, 2 (4), 303-336.

Twitter developers (2011). *Documentation*. Available: <https://dev.twitter.com/docs>. Last accessed 2nd May 2011.

Twitter4j Community members (2011). API Support matrix. Available: <http://twitter4j.org/en/api-support.html>. Last accessed 2nd May 2011.

Yongzheng Zhang, Nur ZincirHeywood, Evangelos Milios (2005). Narrative text classification for automatic key phrase extraction in web document corpora. *In 7th annual ACM international workshop on Web information and data management*. Germany, 2005. NY, USA: ACM.

Yuan J. Lu. (2007). Extraction of Significant Phrases from Text. *International Journal of Electrical and Computer Engineering*. 2 (2), p101–109.

Appendices

1. Database Table structures:

dataextraction

Field	Type	Null	Default	Comments	MIME
serial	int(11)	No			
tweet_id	bigint(30)	No			
username	varchar(40)	No			
userid	int(11)	No			
touser	varchar(40)	Yes	NULL		
touserid	int(11)	Yes	NULL		
createdat	datetime	No			
time	timestamp	No	CURRENT_TIMESTAMP		
profile_image_source	varchar(150)	No			
geolatitude	double	No	0		
geolongitude	double	No	0		
place	varchar(200)	Yes	NULL		
geolocation	varchar(150)	Yes	NULL		
source	varchar(400)	No			
tweettext	text	No			
annotations	text	No			
sentiscore	double	No			
phrases	varchar(200)	No			

Indexes: ⑦

Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
PRIMARY	BTREE	Yes	No	serial	2655030	A		

Screen shot1. The above Table structure represents dataextraction Table structure, which store the persistence data of CentralTweetCollector system's final outcome.

locationdata

Field	Type	Null	Default	Comments	MIME
postcode	varchar(10)	No			
latitude	double	No			
longitude	double	No			
location	varchar(200)	No			
sublocation	varchar(200)	No			

Indexes: ⑦

Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
sublocation_2	BTREE	No	No	sublocation	872	A		
location_2	BTREE	No	No	location	1841	A		

Screen short 2. The above Table structure represents the location data table structure of GeoCode sub-system's reference database table.

phrases

Field	Type	Null	Default	Comments	MIME
list1	varchar(200)	No			
pointer	int(11)	No			

Indexes: ②

Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
list1	BTREE	No	No	list1	240931	A		

Screen short 3. The above Table Structure represents the phrases Table structure of DisplayTokenisation (significant phrases) sub-system's training data Table.

sentiwordnet

Field	Type	Null	Default	Comments	MIME
charecter	char(2)	No			
serial	varchar(8)	No			
pos	double	No			
neg	double	No			
word	varchar(150)	No			

Indexes: ②

Keyname	Type	Unique	Packed	Field	Cardinality	Collation	Null	Comment
charecter	BTREE	No	No	charecter	2	A		
serial	BTREE	No	No	serial	19541	A		
word	BTREE	No	No	word	19541	A		

Screen short 4. The above Table structure represents the sentiwordnet structure of SentiCalculate sub-system's reference database Table.

2. Plagiarism report:

project_document_version1

ORIGINALITY REPORT

1%

SIMILARITY INDEX

1%

INTERNET SOURCES

0%

PUBLICATIONS

%

STUDENT PAPERS

PRIMARY SOURCES

1	lists.wald.intevation.org <i>Internet Source</i>	< 1%
2	www.chroniclejournal.com <i>Internet Source</i>	< 1%
3	Raymond Kehoe. "Purchaser Acceptance", ISO 9000-3, 1996 <i>Publication</i>	< 1%
4	P. Chhabra. "The Agile Delivery of Service Chain Management Solutions", S. <i>Publication</i>	< 1%
5	innovexpo.itee.uq.edu.au <i>Internet Source</i>	< 1%
6	edu.xvna.com <i>Internet Source</i>	< 1%
7	www.takeda-rika.co.jp <i>Internet Source</i>	< 1%
8	kst.kde.org <i>Internet Source</i>	< 1%

CMT4141 Development Project - CMT4141 Development Project

ASSIGNMENT INBOX & PORTFOLIO

	START	DUE	POST	STATUS	ACTIONS
Turnitin MSc CMT projects dropbox					Collapse this assignment
PAPER	06-Oct-2011 3:00PM	14-Oct-2011 5:00PM	14-Nov-2011 5:00PM	● Submission for this assignment is complete.	Hide details
<p>assignment title: Turnitin MSc CMT projects dropbox</p> <p>assignment instructions:</p> <p>search criteria: internet, TurnitinUK student paper database, periodicals, journals, & publications</p> <p>allow late submissions: no</p>					
Paper Title					Report
project_document_version1					1% 
				GradeMark	Download
				Pending	

The screenshot shows a Turnitin Document Viewer interface. The main document is titled "2010/11 Text analysis for the visualisation of large twitter data". The document is labeled "project_document_version1" by "HARISH MUPPALLA". The Turnitin interface shows a similarity index of 1% and a grade of --. A sidebar on the right titled "Match Overview" lists seven sources with similarity percentages all below 1%:

Rank	Source	Similarity
1	lists.wald.intevation.org (Internet source)	<1%
2	www.chroniclejournal.com (Internet source)	<1%
3	Raymond Kehoe (Publication)	<1%
4	P. Chhabra, "The Agile..." (Publication)	<1%
5	innovexpo.itee.uq.edu.au (Internet source)	<1%
6	edu.xvna.com (Internet source)	<1%
7	www.takeda-rika.co.jp (Internet source)	<1%

The main document content includes a "Table of Contents:" section with the following entries:

- Table of Contents: 1
- 1.0 ABSTRACT: 3
- 2.0 Introduction: 4
- 3.0 Literature Review: 5
 - 3.1 Accessing tweets from the Twitter: 5
 - 3.2 Annotations Extraction: 7
 - 3.3 Geo-Coding of a location: 8

3. Contents of the supplied software in the CD.

- Project document.
- Application source code.
- External Java jar files.
- Code documentation.
- Mysql database files.