CMPE 239 - Web and Data Mining

Project Report

San José State
UNIVERSITY

Professor: Chandrasekar Vuppalapati

Team: Heineken

Ameya Manjrekar	010006334
Chao Nie	009320506
Rajas Hegiste	010006438
Yaopeng Wu	008104902

Date: 4/28/2015

Table of Contents

Project Description
Requirements
    How AskUbuntu.com works
    Data Source
UI Design Principles
High Level Architecture Design
Datasets & Data patterns
        Badges
        Comments
        Posthistory
        Postlink
        Posts
        Tags
        Users
        Votes
Data Flow Diagrams & Architecture
    Data Cleansing
    Data Analysis
    Knowledge Discovery
    Deployment
Data Mining Principles and Algorithms:
    Vector Space Model
        1. Document Indexing-
        2. Term Weighting-
        3. Similarity Coefficients-
    ForceAtlas2 algorithm:
        Force directed graph
        ForceAtlas2 Algorithm
    Settings
    OpenOrd Layout
Data Mining Tools Introduction: R Language and RStudio
    Data Mining Tools Introduction: R
    The distance between nodes: the similarities among questions
    Generate similarity matrix
    Generate term-doc frequency matrix
    Export data to CSV file
KDD Principles
    Data Extraction and Cleaning
        Quality Analysis
        Similarity Analysis

Original
Extracted
Link Analysis
Original
Extracted
Similarity Analysis
Original
Extracted
Result
Link Analysis
Original
Intermediate result
result
Data Mining
300 Questions Visualization
Data import
Edges
Issue
Node
Layout Algorithm
Stats and Result
Discovery
All Questions Visualization
Layout Algorithm
Quality Score (Node Size) Stats
Gephi stats
Discovery
1000 Questions Visualization
Client side design
Testing
Load Testing
Design Pattern Used:
MAPREDUCE
PIG
Hive
Individual Contribution
Ameya Manjrekar
Chao Nie
Rajas Hegiste
Yaopeng Wu

Project Description

This data mining project comprises of analyzing Ask Ubuntu's database to provide a visual representation of the similarity of questions and the quality of answers to those corresponding questions. The database will be inspected through scripts to indicate a better understanding of the data attributes and their values which will aid in selecting only among the relevant data for analysis. Further scrutiny of this selected data will determine the similarity and the quality factors. The similarity of questions will be worked out by applying vector space model to the keywords that will be picked up from the questions and then using cosine distance formula to compute it. The quality of the answers would be based on assessing the upvotes, the comment count, the number of answers and the favorite count. After analyzing all this, the final output will be represented in a visual format resembling different sized bubbles and associations between them.

Many times, Ubuntu users are faced with questions related to the plethora of commands and their actions and various of other operations that help run the open sourced OS. This can be especially common in the case of users who aren't familiar with the terminal centric and hierarchy based directory nature of Ubuntu OS. For each question, on the right side of each page AskUbuntu.com will provide a small list of related questions or questions from hot topics. Even users usually get the desired answer from the most upvoted answers, in some cases when the answers are very outdated or due to other reasons, it wouldn't work and users have to browse in and outside of AskUbuntu.com to find the answers that work for them. In our presentation of visualized relations of questions, users can easily see quite enough numbers of similar questions in our graph, with every node representing a question page in AskUbuntu.com, they pop up the first line of the questions for a hint, and users can click on the node and navigate to the question link in AskUbuntu.com so they will have a clear check of the question.

The original output of this project is to present the most related and qualified questions to user from their input question; however, due to the type of questions varies dramatically, it is not too efficient to calculate the similarity of dynamic input and stored question database, because the processes of this data mining project involves stemming text data and aggregating counts of occurrences of term in each document, which produce outstanding time complexity.

Hance, in our project we have implemented several kinds of data mining with different sets of cleansed data. For example, with the same dataset of questions in askubuntu.com, we applied two types of cosine distance algorithms, which are binary and term frequency–inverse document frequency. We also applied clustering algorithm for larger amount of data and whole dataset, all the generated csv file will be imported to Gephi, the visualization tool, and create the .gexf format file for an advanced view of data mining results. In addition, the different graph will be presented in different web pages.

Requirements

How AskUbuntu.com works

Ask Ubuntu is one of the popular Q&A websites of Stack Exchange network, the website is committed to provide visitors quick and clear answers to their questions as possible. When the types of questions are controlled by the website moderator system, there tends to be actual problems that users have encountered with detailed genre and explanation, and the answers will keep a clean style that prevents generating discussions instead of a straightforward solution.

The website also supports tags for each question to present a better idea of categories it belongs to, by clicking on the tags visitors can navigate to the list of all the related questions that are sorted by the several options of orders such as newest or frequent; however, it doesn't support selection of multiple tags.

One significant feature of Stack Exchange networks is that every question and answer of the website can be voted up or down by any registered user that has a reputation number greater than 15. When answers are voted up, that means a number of visitors thought the answer is helpful or agreeable; on the other hand, questions and answers can also be voted down by users that have a reputation number greater than 125.

At this point, we are ready to deploy our algorithms for this data mining project. The idea is to calculate similarities between an input question and all the other questions in storage. The current attributes we considered to use as factors to calculate the similarities are key words, tags, up/down votes, and key words in the comments. Still, there can be other features to be added in the formula to enhance the accuracy, such as the reputation number of users, or the number of unlocked badges of users, which are both an expression of users that have more experience on this websites.

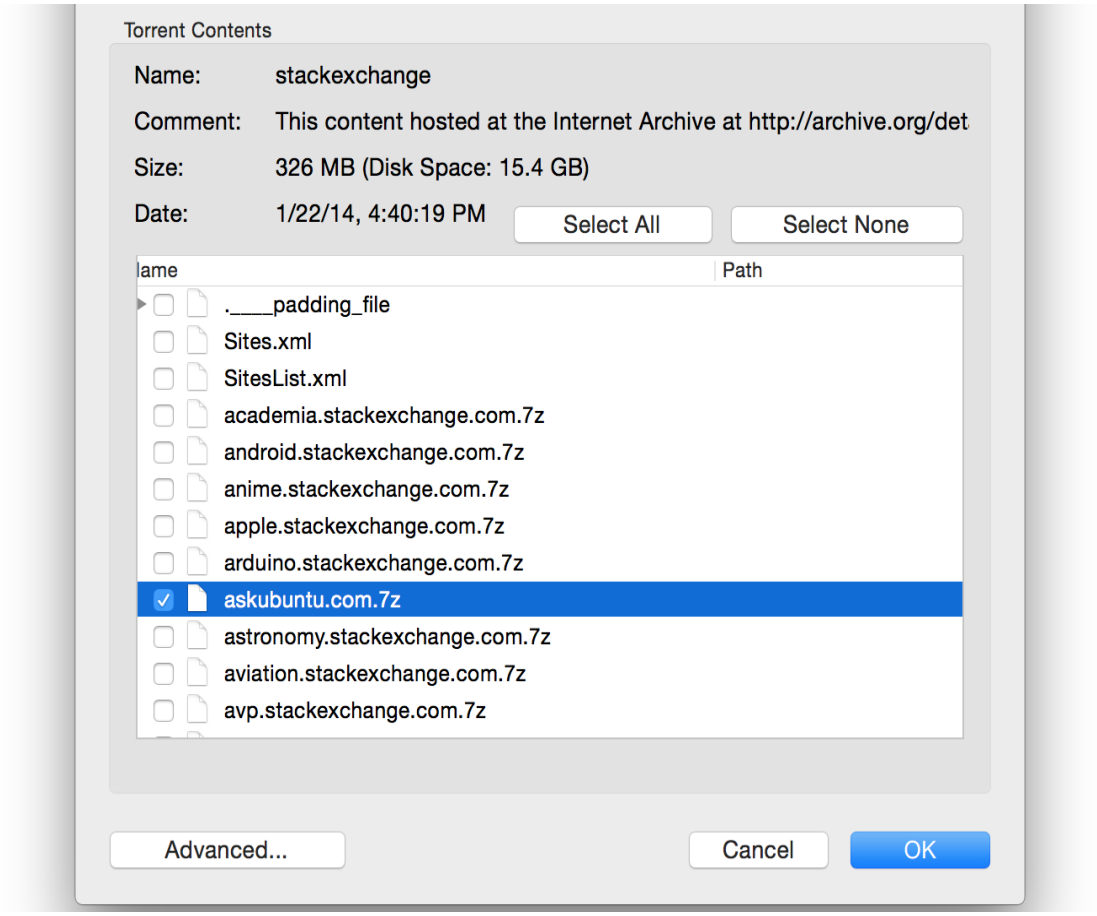
Data Source

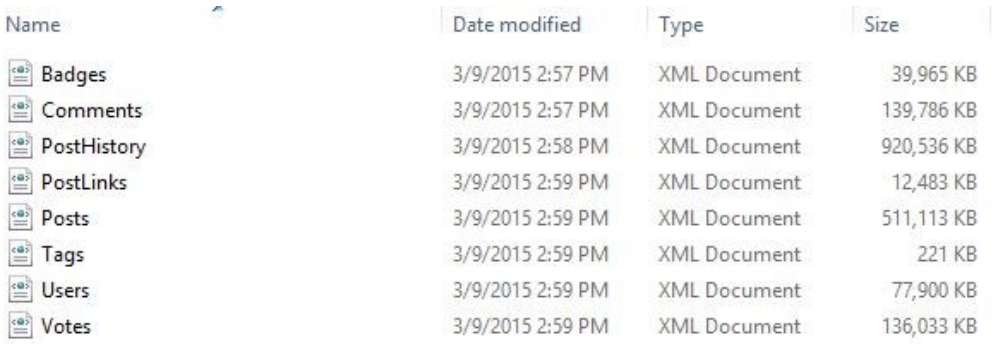
All Stack Exchange Data stacks can be found in this URL
https://archive.org/details/stackexchange

Visitor can simply use click “SEE ALL FILES” and see all the data stacks of different websites. We had four data stack candidates to analyze with; however, the perfect size of unzipped data sets for this project should be around 1 GB, so the left choices are of: *AskUbuntu.com.7z*, *superuser.com*, and *programmers.stackexchange.com*; there are lots of codes in programmers.exchange.com and it is complicated to parse into patterns; and superuser.com discusses every topic related to computer science, for example it also includes questions regarding the usage of Microsoft Excel; finally, the data stack of *AskUbuntu.com* has around 1.7 GB data, which is a good size for this project, and it also has a relatively better

data integration and a specific service domain (questions associated with Ubuntu operating system), so, it's decided to be the date set for this project.

Under all the files listed, navigate to "*askubuntu.com.7z*" and download the expressed file of data set in askubuntu.com. After unzipping the file, the included data attributes are shown as follows:





Name	Date modified	Type	Size
Badges	3/9/2015 2:57 PM	XML Document	39,965 KB
Comments	3/9/2015 2:57 PM	XML Document	139,786 KB
PostHistory	3/9/2015 2:58 PM	XML Document	920,536 KB
PostLinks	3/9/2015 2:59 PM	XML Document	12,483 KB
Posts	3/9/2015 2:59 PM	XML Document	511,113 KB
Tags	3/9/2015 2:59 PM	XML Document	221 KB
Users	3/9/2015 2:59 PM	XML Document	77,900 KB
Votes	3/9/2015 2:59 PM	XML Document	136,033 KB

Figure AskUbuntu.com data attributes

UI Design Principles

The main purpose of our project is visualizing the content of AskUbuntu. Therefore, it was of prime importance that we present the data in highly appealing but at the same time in a simple way that the user can understand.

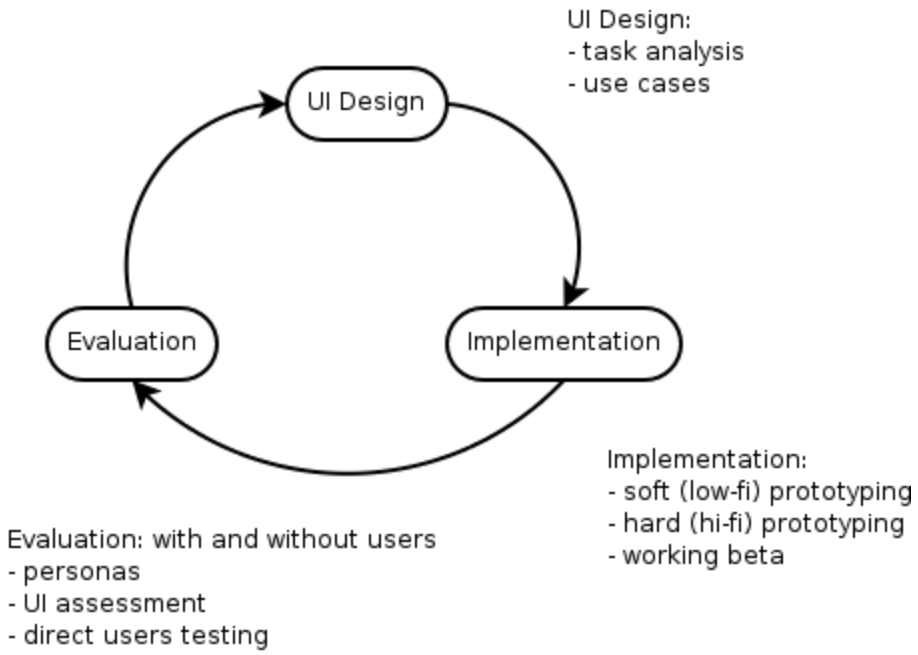


Figure: UI Design Principle

We followed three simple steps in determining the final design of our web app. The first stage involved UI Design where we considered what the final product should look like. In our case, this consisted of displaying the cluster of questions that are similar to each other. As there were four different types of analysis done on the AskUbuntu questions, the web app would contain four different links to each of them.

The next stage was the implementation stage where we created a simple sketch to represent the web page based on the decisions made in the previous step. This wireframe gave us an idea as to how to model the data on the front end and helped us in finalizing sigma.js to be used for the end result. The below diagram depicts one of the visualized graphs developed through gephi as seen after parsing it through sigma.js.

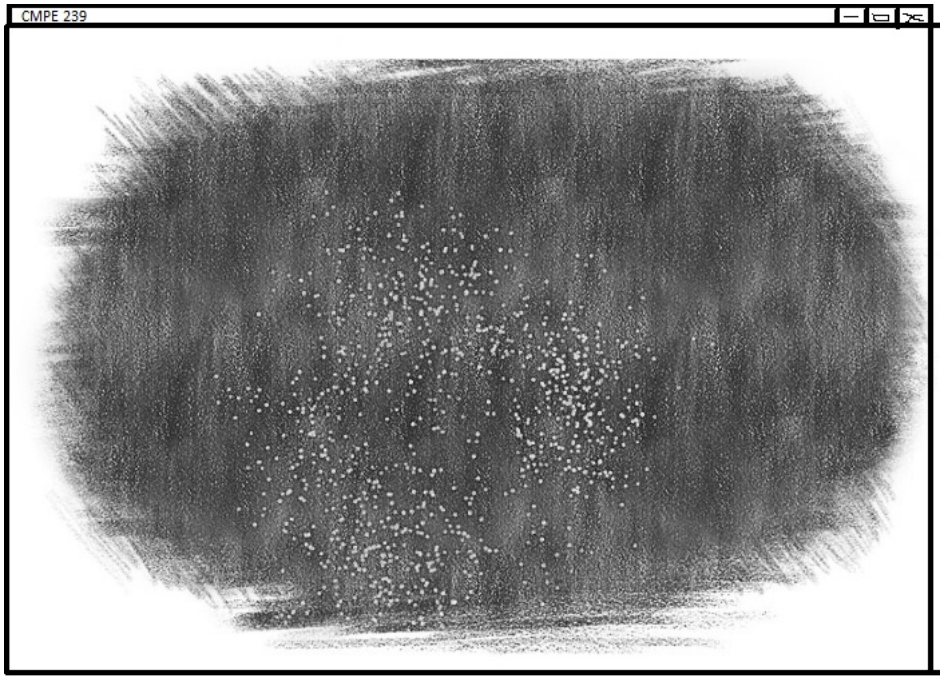


Figure: Simple Wireframe of Web App

The final phase involved deploying the graphs on the node.js server to check out the output from the user's perspective. At first, we planned on showing the links to inter related questions along with the nodes as well. However, as the number of such links were too many, it complicated the UI after which we decided to drop those links and just show the nodes instead.

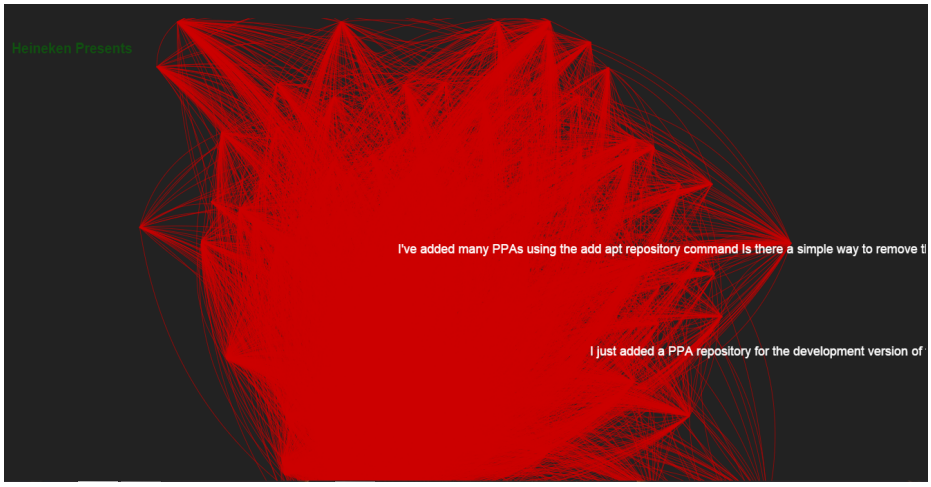


Figure: Cluster as seen with links

High Level Architecture Design

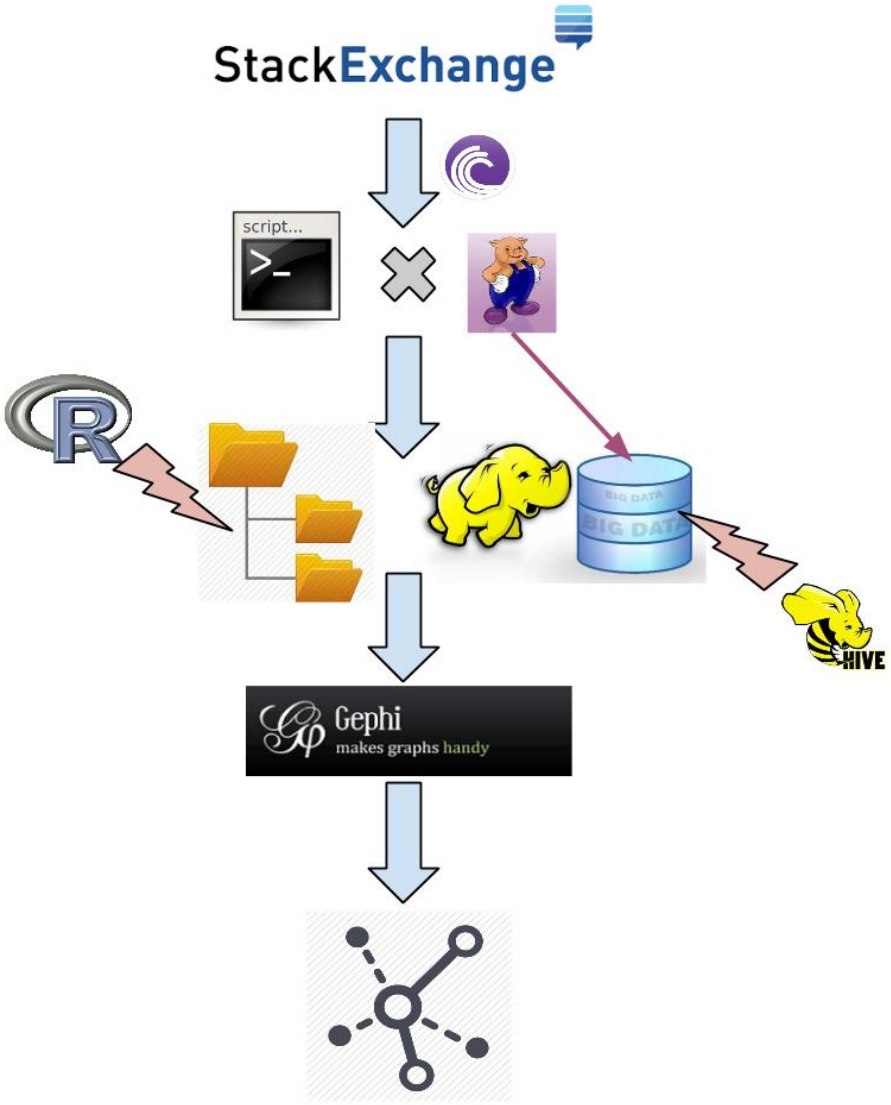


Figure: High level architecture

Datasets & Data patterns

The dataset AskUbuntu comes with eight data: badges, comments, post history,
The explanation for each data is provided below in alphabetical order. A screenshot of a question is attached at the end to help you better understand of some attributes.

Badges

Badges are given to a user with reputation. The name of the badge together with the user ID is included. There are six kids of badges: Question Badges, Answer Badges, Participation Badges, Tag Badges, Moderation Badges, and Other Badges. You can find the details for each type of badge on http://askubuntu.com/help/badges. Since this has less to do with questions, we exclude this data.

Comments

Comments are feedback to a question or an answer. There are four important attributes listed below in this data. You can find the concepts and rules about commnet on http://askubuntu.com/help/privileges/comment. Sometime, a comment contains a hyper reference to another question. As such, this data can be used for association analysis, and thus we selected it.

- PostId: refer to ID of question or answer
- Score: only present if score > 0
- Text: the contents of the comment
- UserDisplayName: the name of the user

Posthistory

This data contains all logs of updates for questions and answers. It has 38 types of updates and they are provided as PostHistoryTypeId. The contents of updates is given as Test attribute along with a timestamp as RevisionGUID. Since we're not interested in the historical view of post, we ignored this data.

Postlink

Often, a question or an answer contains an hyper reference to another question. This data is specifically designed to store that information. As the name suggests, this data is very useful for discovering the relationship between posts. We selected this data to include as a factor of association. They are two major attributes in this data indicating the relationship between posts.

- PostId: contains hyper reference to another question
- RelatedPostId: id of the referenced question

Posts

Post contains the all necessary information to analyze the quality. Since this data is critical to our analysis, we list all the important attributes of this data in the following.

- PostTypeId: indicates the type of the post. Mainly divided into three categories.
    1. Question
    2. Answer
    3 - 8: relating with Tags

  ** Since information about tag has less to do with the quality, we remove all the post with having PostTypeId 3 to 8.
- AcceptedAnswerId: Question referring to its accepted answer Id
- ParentID: Answer post referring to its question Id
- Score: users vote score for the post
- ViewCount: only present in Question type of post
- Body: the contents of the post
- OwnerUserId: Id referring to Id of user Id in users.xml data (present only if user has not been deleted; always -1 for tag wiki entries (i.e., the community user owns them))
- OwnerDisplayName: OwnerUserId or OwnerDisplayName must be present
- Title: the title of a question (only present in Question)
- Tags: only present in Question
- AnswerCount: number of answer posts (only present in Question)
- CommentCount: number of comment posts (appears in both Question and Answer)
- FavoriteCount: users vote favorite for the question (only present in Question)

Tags

Most of the questions are tagged with appropriate key words. The tag data contains Id, tag name, the total count of questions it's tagged with, and the reference Id to its wiki page provided in the post data. Although this data is perfect for clustering questions, it is out of the scope for this project.

Users

Every user has his/her individual page containing information about introductions and activities. This data helps identify each user's profile and geological information as well as his/her contribution and reputation in the forum. Even though we perform user based question association analysis, user Id provided in post data is enough to complete this analysis. Since user profile is not needed for our analysis, we ignored this data.

Votes

The vote data is designed to organize all vote to questions and answers. Consisting of four attributes, it keeps track of the reference Id to post, type of vote, reference Id to user, and creation date. Although through this data we can view the change of popularity of a question, it is less relevant to our analysis.

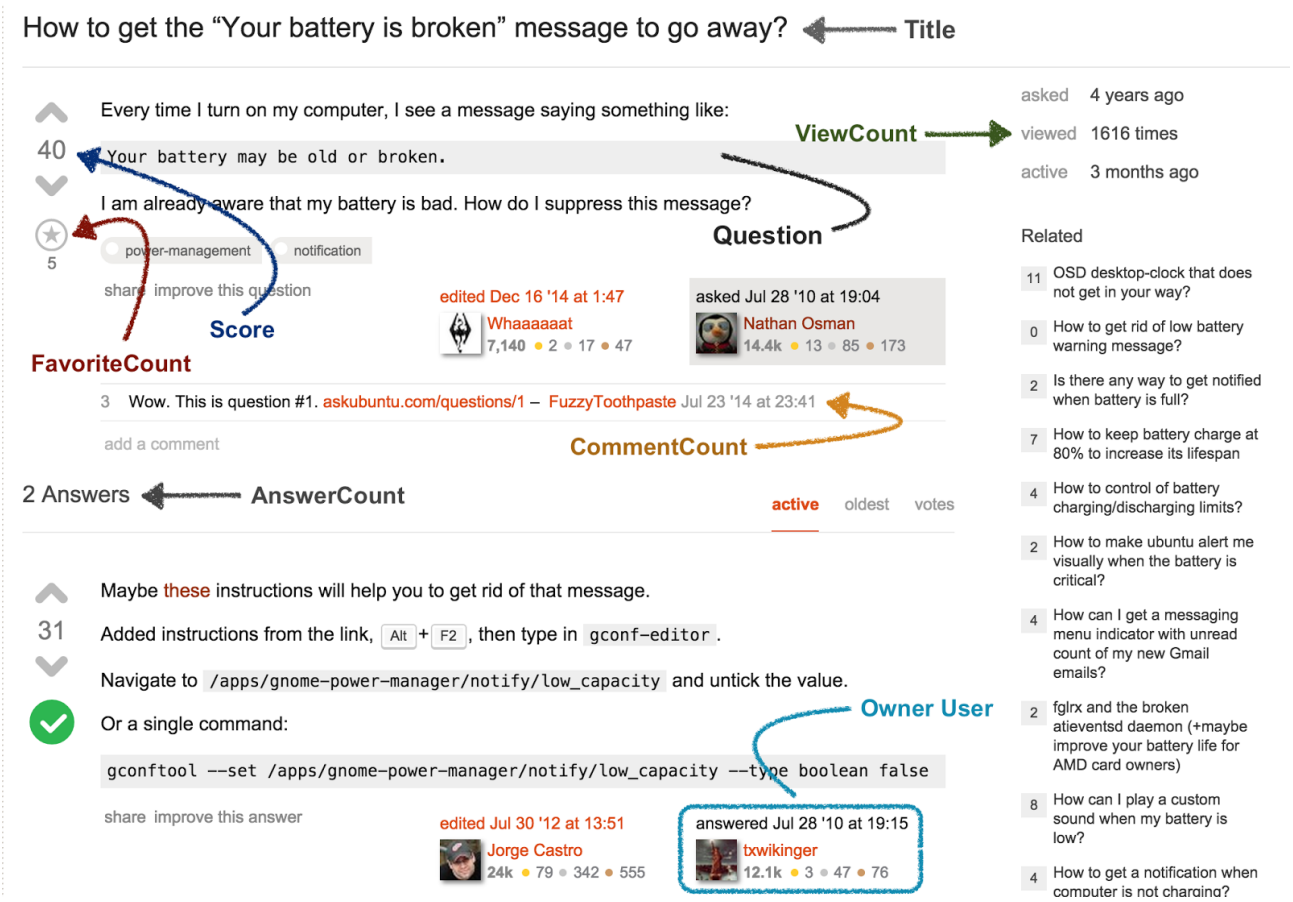
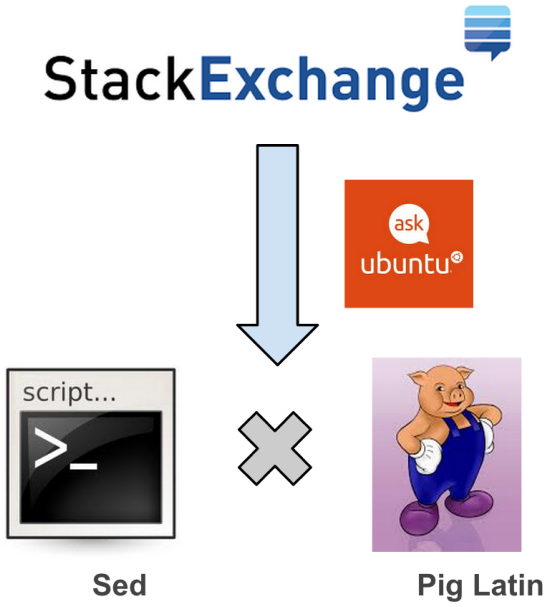


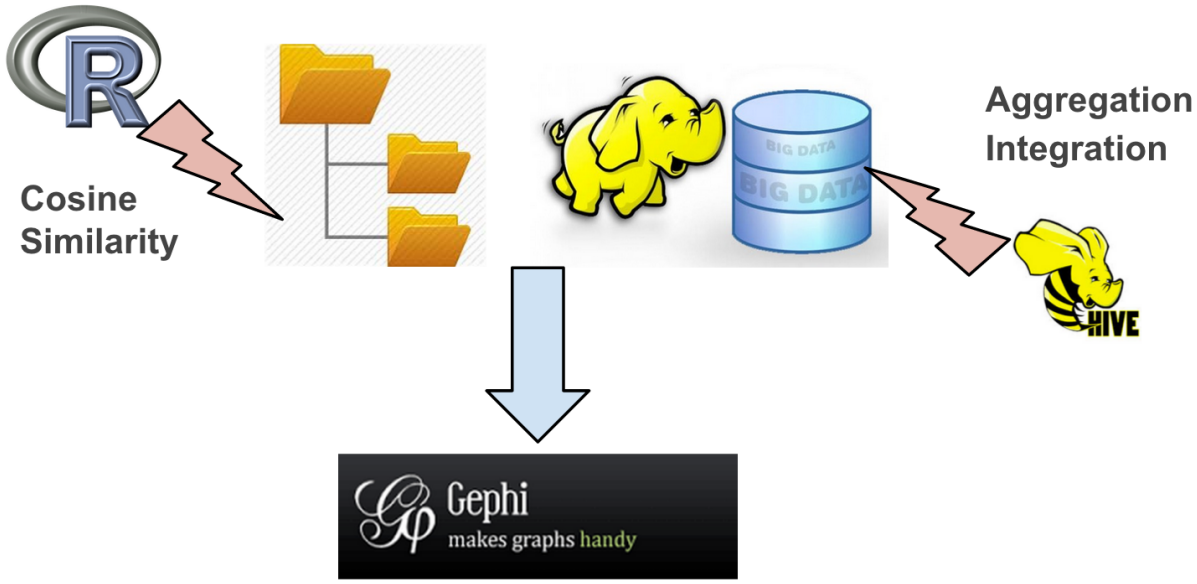
Figure 5: A question in AskUbuntu

Data Flow Diagrams & Architecture

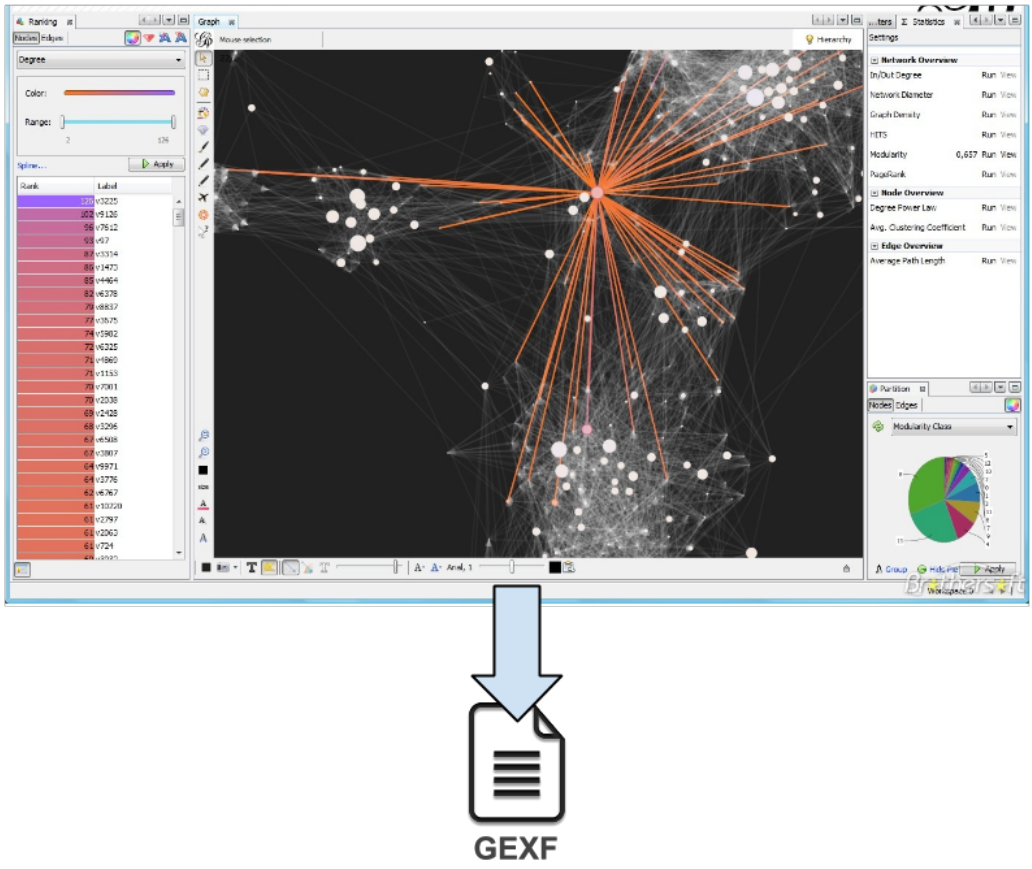
Data Cleansing



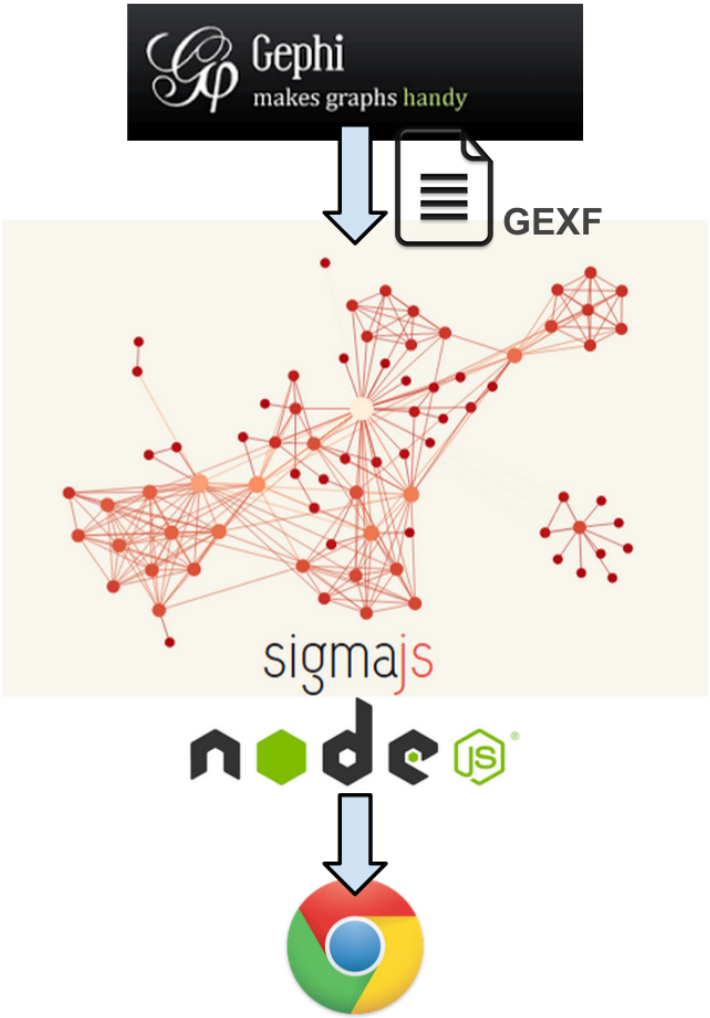
Data Analysis



Knowledge Discovery



Deployment



Data Mining Principles and Algorithms:

Vector Space Model

Vector space model or term vector model is an algebraic model for representing any kind of object in general. The Vector Space Model is a technique usually used to collect similar characteristics from some document or some data. It is a method used for measuring similarities in a complex space for calculating similarities between any kind of queries and any kind of information. The vector space model is a procedure that is divided into three stages. The first stage would be document indexing which would contain extracting the key words from the document or an object in general. The second stage would contain assigning weights for key terms in a document that would be needed by the user. The third and last step would be finding the similarities between the user queries and the document or an object.

1. Document Indexing-

As every document or any object would always contain lots of content that would not be related such as words like the, is. Such type of words would be automatically removed as it would not be needed. The primary step would be to identify the keywords and terms in questions which would be the most important task. Once all these terms are identified then based on these terms they would be used to calculate the weight. Now there have been many methods like serial clustering that would automatically index words that are content bearing.

2. Term Weighting-

The term weighting part is done basically from its frequency. The term frequency is dependent on the content of any object that can be anything in any case which is the basis for any type of weighted document vector. There also a possibility that any such document or object can use a binary or Boolean vector. There are many factors that are considered during term weighting one such factor is document length normalization factor. For every long document that is being used as data there are always going to be a large term set which makes them more prone to be getting retrieved every time. Whereas in the case for a short document it is exactly opposite short documents would not retrieved as much compared to the bigger documents. There have been various weight schemas that have been used to calculate term frequency but some of the best results that are obtained in the case of length normalization.

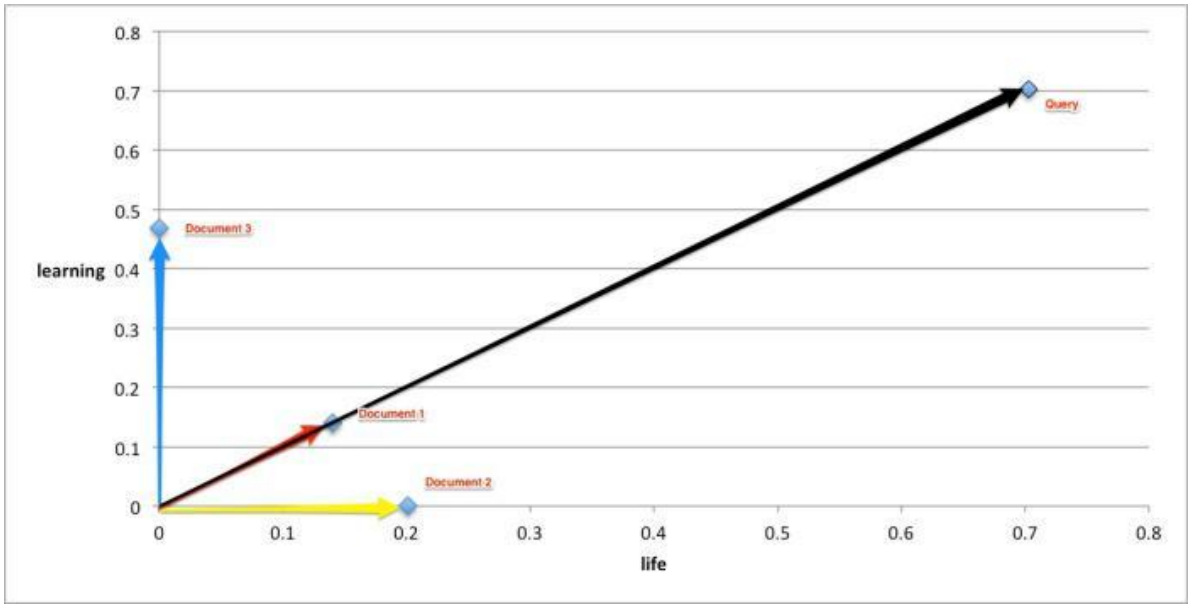
3. Similarity Coefficients-

Similarity Coefficients are used for analysis based on the object and the query that is being compared and would indicate the overlapping similarities between them. The product on which similarity coefficient is used is always normalized in the previous steps. The most popular similarity method that is always used is cosine distance which can be used to calculate the distance between the object and the query in this case. The main reason for using this particular technique is because other methods like Euclidean distance are useless

for calculating similarities as there are so many words between the object and query that are uncommon. Here the attributes ca be used as a vector to obtain a normalized output from the object and query. To get similarities one has to get the cosine of the angle between the objects. The similarity can be calculation is expressed mathematically.

$$similarity(x, y) = cos(\theta) = \frac{x \cdot y}{||x|| * ||y||}$$

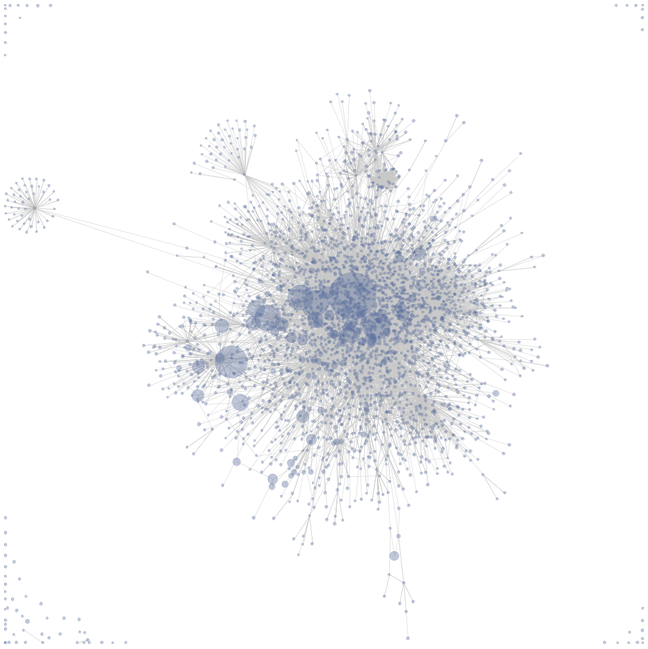
Based on the calculations that are made for similarities graph can be plotted in a 2 dimensional space as shown further. In the image it shows that a document 1 has highest score is most similar whereas document 2 and document 3 are not.



ForceAtlas2 algorithm:

Force directed graph

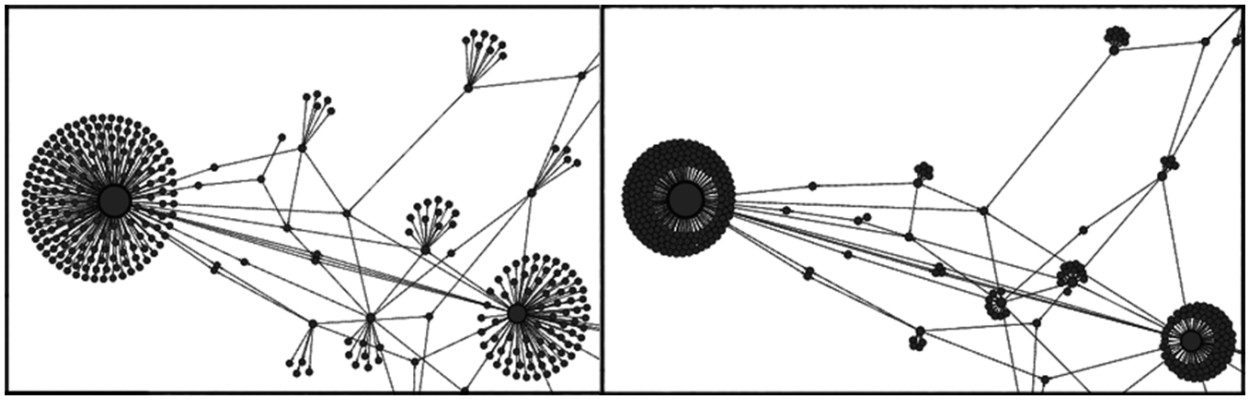
ForceAtlas2 is a force directed graph. Force directed graph are those in which the algorithms are used to draw graph in a two or a three dimensional space and among this space nodes are placed in such a manner such that the edges among then almost equal in length and also there are possible overlapping edges among them.



This image shows us visualization of links between pages on a wiki using a force-directed layout. This image being an example as to how the visualization occurs using such types of algorithms.
Force Directed graph contain nodes and edges which act as if there has been a spring like force between them. The endpoints of this graphs sometimes attract towards each other like some force is pulling them towards each other whereas at the same time there is also a repulsive force that is pushing them away from each other at the very same moment. At the state of equilibrium the edges appear to have same length between them as well as the nodes that don't have any connection between them appear very further away from each other.

ForceAtlas2 Algorithm

ForceAtlas2 is one of the algorithm used by gephi for visualization of data of any kind. It simply lays out projects a physical system in a way to spatialize a network. Spatialize just means projects the graph in a planned or studied space which gives the overall idea of the entire environment and surroundings of the space. These type of graph are generally used for network visualization. The final configuration of this algorithm is to integrate the data in a graphical manner.
In this algorithm, each of the node can contain any of the attribute allotted to it. Each time one of the attribute is allotted to it than all the other attributes are of no use at that moment. At this time each node contain different details of that attribute. After this the edges come in the picture usually all the edges almost have the same length but when the nodes have no similarity they appear very far away from each other. These types of graphs tend to show visual densities which also lead to show lot of structural densities.



The above image shows the difference between the Regular repulsion and repulsion by degree. The left image in the above figure shows how the regular repulsion appears. In regular repulsion the repulsion is normal and all the nodes are away from each other as well as the nodes that are not similar appear very far away from each other. In the figure the image on the right shows the Repulsion by degree. Here it shows that the nodes that are poorly connected are closer to those that are highly connected nodes which gives a clear idea of what we should be looking at. This type of image directly gives a clear view of what we want to look at by directly taking a look at the visualized graph of the data.

Settings

There are various types of settings that need to be looked at before starting. These settings have an important role in finding a clear solution of what we want. There are various aspects that need to be looked at before getting the output that is required as per our needs. The factors that are important in setting the environment are as follows:

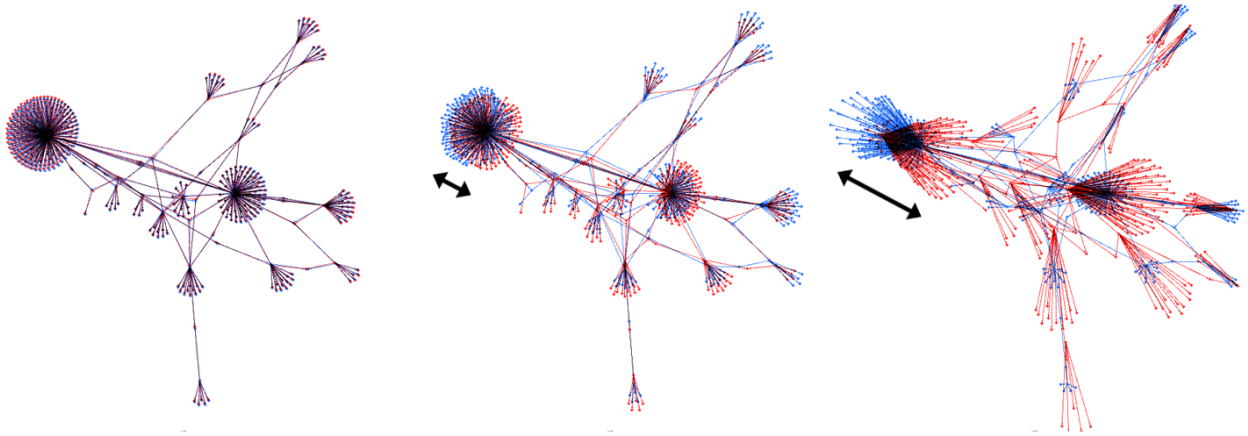
1. Gravity
2. Scaling
3. Edge Weight
4. Dissuade Hubs
5. Prevent Overlapping
6. Approximate Repulsion

Performance

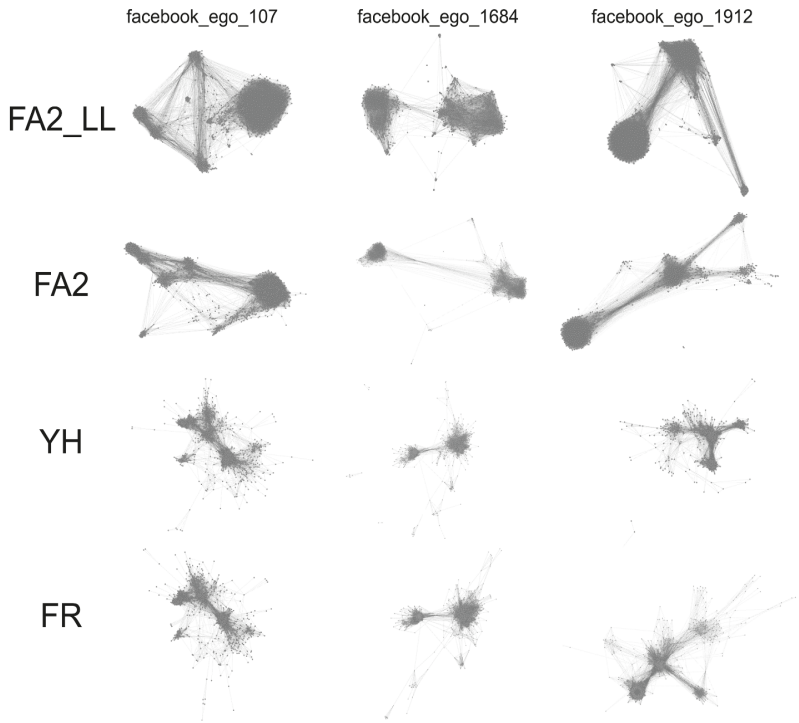
There are factors that come in account when performance of anything comes in picture. In this scenario what happens is the speed and the precision are inversely proportional to each other. When the speed is high the precision of the graph is less or it is decreasing and when the speed is low then the precision increases and the graph appears in a manner that can be very easy to understand just by looking at the visualized version of the data.

In the figure below there are 3 images:

1. The first image shows that the nodes are well defined and precise in a way that is more appealing to the user. Here the speed is low and hence the precision is high.
2. In the second image the speed is increased as compared to first and in that respect we can see that there is variation in the precision.
3. In the third image of the figure the speed is increased ad as a result the precision of the output data that is visualized is low.

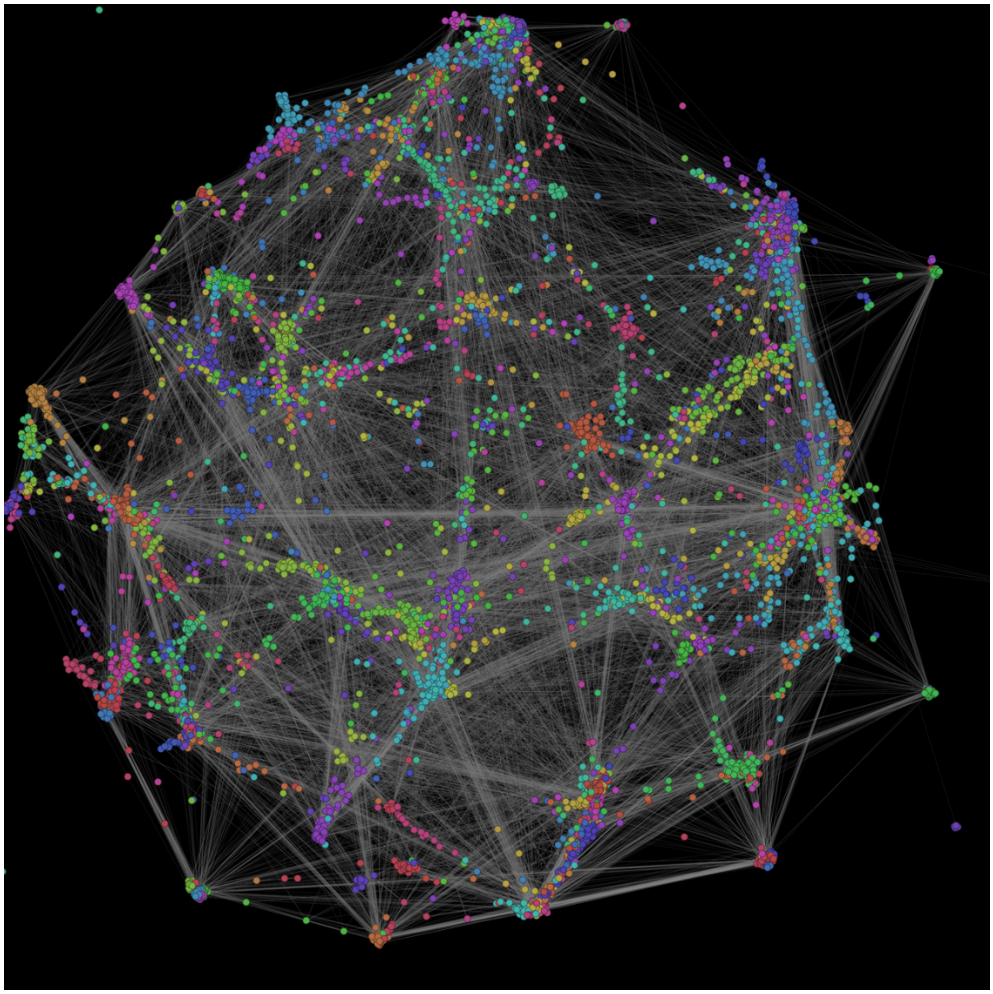


The below image shows that FA2_LL and FA2 are more readable, defined and precise compared to the others. The image shows all the difference that is occurred in the based on the network in different areas.



OpenOrd Layout

This algorithm is used for undirected graph and used for a better understanding of the cluster by visualizing it in a better way. It shows the clusters that are formed and also shows the connection between them by the edges. In this algorithm edge cutting is also supported as there is lot of edge overlapping in this algorithm. The below figure will give the exact idea of what this algorithm does when used for visualization of data.



Data Mining Tools Introduction: R Language and RStudio

Data Mining Tools Introduction: R

1. Background of R

R is the most broadly used programming language and software environment for data miners and statisticians to do data analysis or develop statistical software. It is one of the free software of GNU, a Unix-like operating system.

In association with lexical scoping semantics, the programming language R has implemented S, a statistical programming language designed by Bell Labs; the software environment R was primarily written with C, FORTRAN, and R. The software is based on a CLI environment, however, there are several graphical front-ends available for users or programmers.

R language is an interpreted language like Python, its object can be directly manipulated with Python, C++, or Java, etc. Its functions are also extendable through new packages provided by CRAN networks. R's libraries contain a various types of functions and algorithms such as K-means clustering and SVM classifier. A collection of libraries will be a package, which, for examples, supports model building for classification, generate dynamic graphics, or provides GUI functionality.

Basic Machine Learning Functions

	Function	Library	Description
Cluster	hclust	stats	Hierarchical cluster analysis
	kmeans	stats	Kmeans clustering
Classifiers	glm	stats	Logistic Regression
	rpart	rpart	Recursive partitioning and regression trees
	ksvm	kernlab	Support Vector Machine
	apriori	arules	Rule based classification
Ensemble	ada	ada	Stochastic boosting
	randomForest	randomForest	Random Forests classification and regression

Noteworthy Data Mining Packages

Package	Comment
caret	Well organized and remarkably complete collection of functions to facilitate model building for regression and classification problems
rattle	A very intuitive GUI for data mining that produces useful R code

2. **Usage of R**

There are several GUI tools to make data mining easier. For this project, we will install and use RStudio for data analysis.

In R language, the assign mark for R is '<-', although '=' can be used alternatively. Here is a simple example of creating a function within R that returns the square of a number:

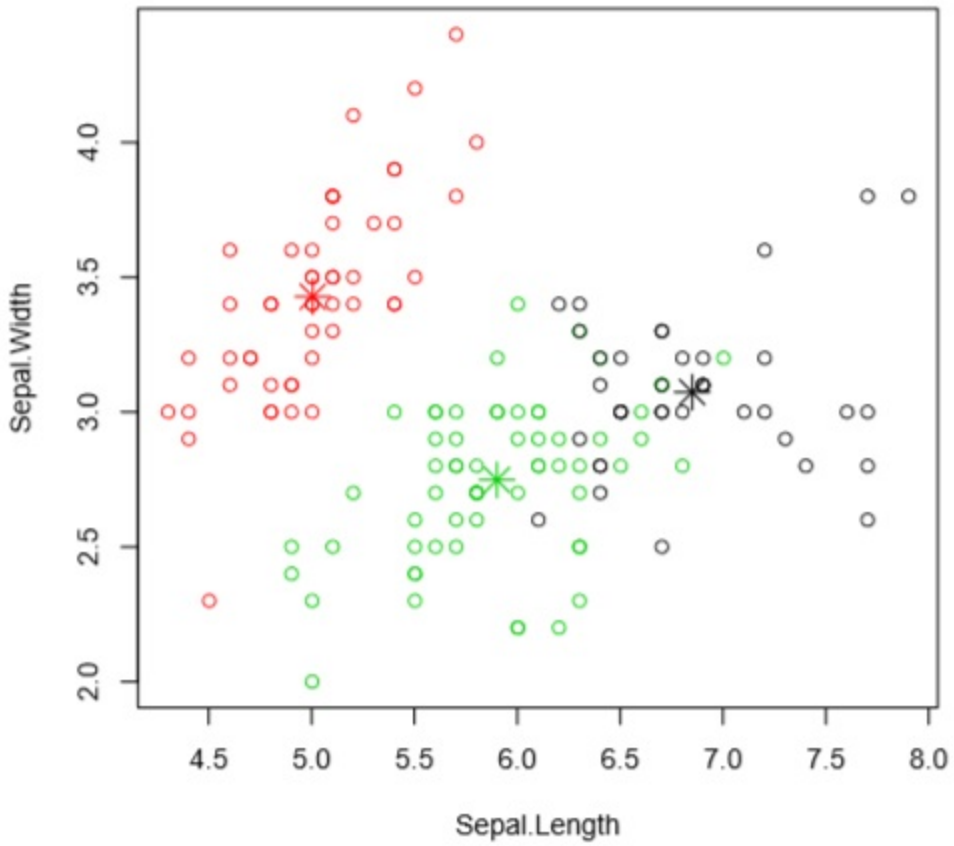
- theFunction <- function(x) { # a user-created function named "theFunction"
- return(sum(x^2)) # return the sum of squares of the elements of vector x
- }

The output of calling the function theFunction(1:4) will be

- > thefunction(1:4)
- [1] 30

That is the sum of the square of number 1 to 4.

In R libraries, there are plenty of functions that run through the imported data, and RStudio can generate graphic results, and it also supports gradually running the testing data so you can have a visual on the processes of creating a final cluster, for example.



Sample result of clustering

For RStudio to analyze data and bring out visible results, original data have to be formatted or converted. With functions like read() or write(), RStudio accepts data types such as

- .r file, which is the script file of RStudio, within .r file, user can process data directly in R runtime environment (terminal)
- .RData, the workspace project file
- .CSV, an extended Microsoft Excel file where every element is separated by comma
- .SAS, statistical analysis software file, stores data in standard ASCII text format
- .txt, RStudio can process data directly from text files
- .xls, RStudio imports and exports data with Excel files, there might be a 65,536 limit of rows to be processes to write to an Excel file

RStudio also provides connections to ODBC database, in library “RODBC”, functions like odbcConnect(), sqlQuery() connects and sends SQL query sentences to database.

The distance between nodes: the similarities among questions

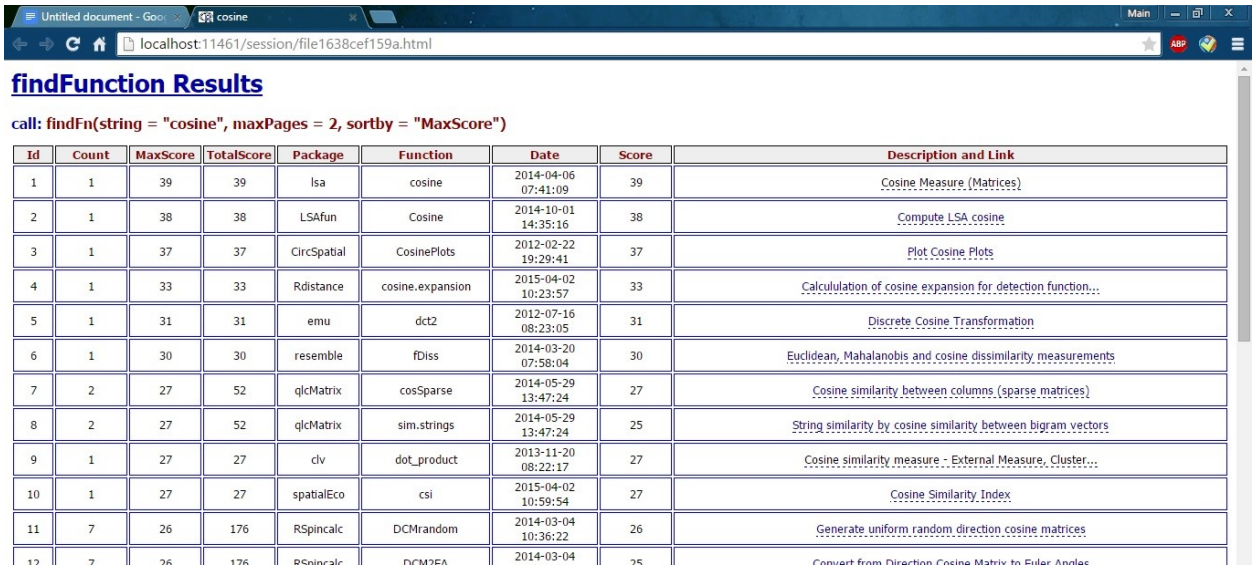
R has many packages of functions, when you are not sure if R already has a function that does the job you want, there is a convenient way to find out if it’s in their libraries. In R console, install package “sos”, and load library in project. For example, we want to find the function related to “cosine” function:

```
install.packages(“sos”)

library(sos)

findFn("cosine", maxPages=2, sortby="MaxScore")
```

The last line indicates that we can use the function “findFn” to search R library, the argument **maxPage** determines only look within 2 pages of R function library, **sortby** determines to list the most related functions first. After running the function, RStudio will download an **html** file and automatically open the file with your system’s default web browser. The result is as follows:



Id	Count	MaxScore	TotalScore	Package	Function	Date	Score	Description and Link
1	1	39	39	lsa	cosine	2014-04-06 07:41:09	39	Cosine Measure (Matrices)
2	1	38	38	LSAfun	Cosine	2014-10-01 14:35:16	38	Compute LSA cosine
3	1	37	37	CircSpatial	CosinePlots	2012-02-22 19:29:41	37	Plot Cosine Plots
4	1	33	33	Rdistance	cosine.expansion	2015-04-02 10:23:57	33	Calcululation of cosine expansion for detection function...
5	1	31	31	emu	dct2	2012-07-16 08:23:05	31	Discrete Cosine Transformation
6	1	30	30	resemble	fDiss	2014-03-20 07:58:04	30	Euclidean, Mahalanobis and cosine dissimilarity measurements
7	2	27	52	qlcMatrix	cosSparse	2014-05-29 13:47:24	27	Cosine similarity between columns (sparse matrices)
8	2	27	52	qlcMatrix	sim.strings	2014-05-29 13:47:24	25	String similarity by cosine similarity between bigram vectors
9	1	27	27	clv	dot_product	2013-11-20 08:22:17	27	Cosine similarity measure - External Measure, Cluster...
10	1	27	27	spatialEco	csi	2015-04-02 10:59:54	27	Cosine Similarity Index
11	7	26	176	RSpincalc	DCMrandom	2014-03-04 10:36:22	26	Generate uniform random direction cosine matrices
12	7	26	176	RSpincalc	DCM2EA	2014-03-04	25	Convert from Direction Cosine Matrix to Euler Angles

Figure findFn result

Generate similarity matrix

- **Import data into RStudio and setup data for R to process**

Besides importing data through function, in RStudio, we can use the provided GUI to import .txt file data, which is formatted to two columns, ID and question body

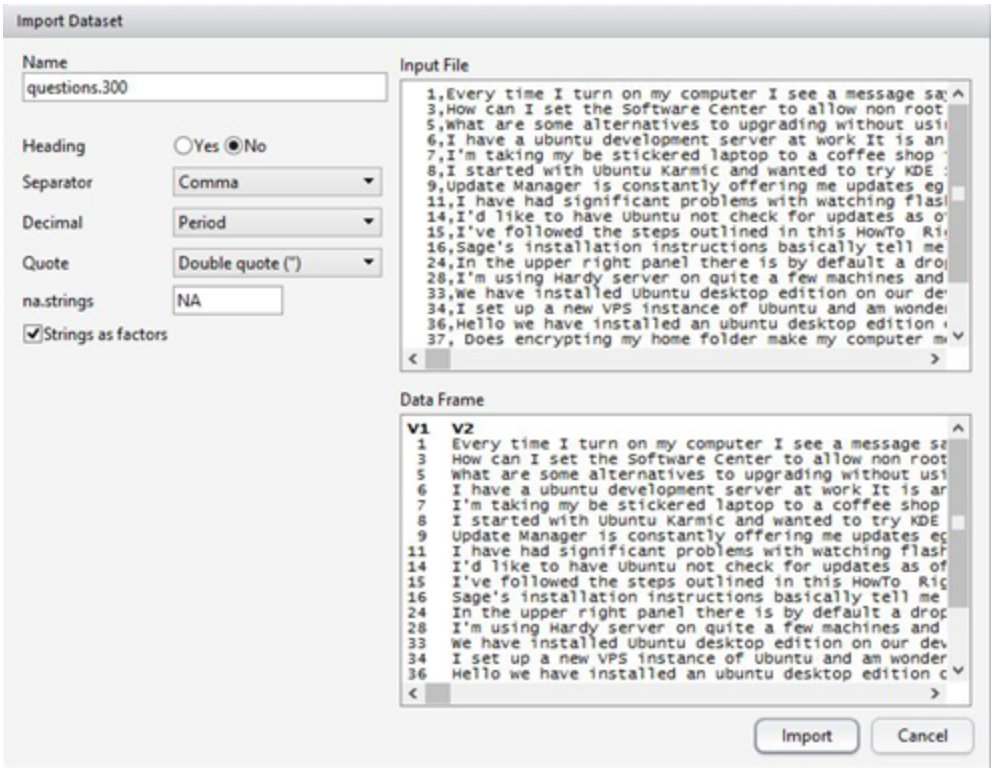


Figure import data with RStudio GUI

Data “questions.300” has two vectors, and the second vector contains the questions data we need. We save the length of the vector (the number of rows) for further use. In RStudio console, type in

```
> N.questions <- length(questions.300[[2]])
```

Install package “tm” and load the library

```
> install.packages(“tm”)
```

```
> install.packages(“SnowballC”)
```
(for word stemming function)

```
> library(“tm”)
```

For now, the imported data are unstructured, for RStudio to understand what it's processing with, we will have to convert the data into a data type called **corpus**, which is the fundamental data type in package "tm".

```
> my.corpus <- Corpus(VectorSource(questions.300[[2]]))
```

We can also name each element of the corpus with the title we want, for example

```
names(my.corpus) <- paste0("Q", c(1:N.questions))
```

so that each element will name after Q1, to output a specific element, type in

```
> my.corpus$content[47]
```

- **Data cleansing with package "tm"**

An important process of data mining is data cleansing. In the case of text mining, we need to eliminate redundant whitespace, convert characters to lowercase, remove stop words such as "the", "a", and most importantly, word stemming.

Word stemming is a process of query expansion (QE), meaning it conflates words into its root format (stem), or a specified format. It is very important and necessary in information retrieval process. Different environment may modify words to different stems. In RStudio, we use the functions in R library to cleanse the questions.

```
> my.corpus <- tm_map(my.corpus, removePunctuation)
> my.corpus <- tm_map(my.corpus, content_transformer(tolower))
> my.corpus <- tm_map(my.corpus, removeWords, stopwords("SMART"))
> my.corpus <- tm_map(my.corpus, stemDocument)
> my.corpus <- tm_map(my.corpus, stripWhitespace)
```

Each function argument controls "**tm_map**" function what to do, and the arguments make the functions understandable. "**removePunctuation**" excludes the punctuation symbols, "**content_transformer(tolower)**" is the new argument that replaces "**tolower**" in the old "tm" package, "**removeWords, stopwords("SMART")**" removes stopwords collected from SMART information retrieval system, which obeys the stopwords list from MC toolkit, the list of words can be found here:

http://jmlr.csail.mit.edu/papers/volume5/lewis04a/a11-smart-stop-list/english.stop

"**stripWhitespace**" eliminate redundant whitespace, finally, "stemDocument" revises every word to the root format agreed by R. Output a random element to see the cleansed questions:

```
> my.corpus$content[7]
```

Output:

```
[[1]]

<<PlainTextDocument (metadata: 7)>>

updat manag constant offer updat secur fix updat ppas ubuntu instal automat download instal updat
```

Original sentence is

Update Manager is constantly offering me updates eg security fixes updates from PPAs How can I tell my Ubuntu installation to automatically download and install updates whenever they become available

- **Stemming Approach in R**

In R text mining package "tm" implemented the stemming algorithm porter stemmer, which strips suffix but prefix. This algorithm appeared in 1979 in Cambridge computer laboratory, and remained not amenable for any further alteration. It is a very applicable algorithm for information retrieval projects encoded in many types of programming languages such as Java, Google Go, PHP, MySQL, and Python, etc.

Porter Stemmer applies 6 steps of stemming processes:

- Step 1: Gets rid of plurals and -ed or -ing suffixes
- Step 2: Turns terminal y to i when there is another vowel in the stem
- Step 3: Maps double suffixes to single ones: -ization, -ational, etc.
- Step 4: Deals with suffixes, -full, -ness etc.
- Step 5: Takes off -ant, -ence, etc.
- Step 6: Removes a final -e

Generate term-doc frequency matrix

Now that we have fewer and cleaner words to process with, R can easily create a two dimensional matrix, which contains the words on rows and the number of the document on columns, the elements in the matrix is the number of occurrence of words.

In R console, use function “TermDocumentMatrix(x)”:

```
> term.q.matrix <- TermDocumentMatrix(my.corpus)
```

To view limited number of the elements, for example, to see the 301st to 400th terms and the occurrence of the first 5 documents, type in

```
> inspect(term.q.matrix[300:400, 0:5])
```

Output will indicate the term-frequency table and the corresponding information

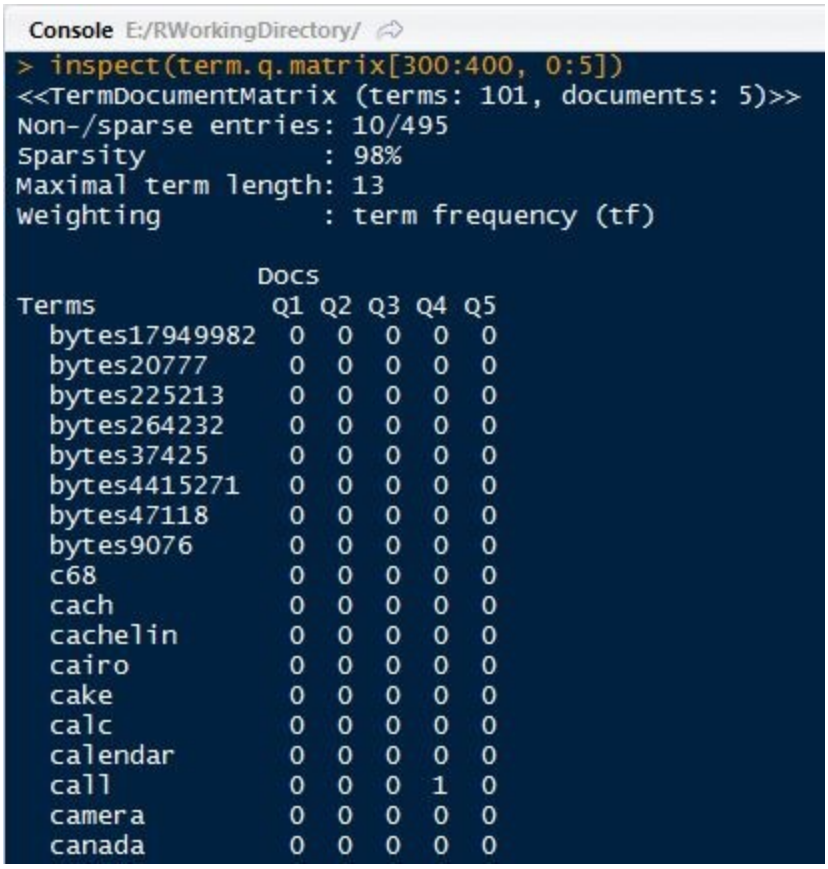


Figure inspect term-frequency table

The matrix object in package “tm” stores non-zero values elements with triplets with three variables: row, col, and value (i, j, value), the type of the matrix is called “**Simple Triplet Matrix**”. To directly operate with these object, we need to install and load “**slam**” package in library. As we can see, the output table contains lots of zero values, resulting a large sparsity

of the matrix; in addition, it's reasonable to reduce the sparsity and increase storage cost. In order to do so, simply type in

```
> term.q.matrix.den <- as.matrix(term.q.matrix)
```

We can check the sizes of the original matrix and the dense matrix:

```
> cat("Original:", object.size(term.q.matrix),"bytes.\n", "dense matrix:",
object.size(term.q.matrix.den),"bytes.\n")
```

Output:

```
original: 224704 bytes.

 dense matrix: 4989880 bytes.
```

- **Implement TF-IDF formula, generate vector space model matrix**

There are different ways to normalize term frequency, in this case, we will apply Cornell SMART system formula, which is

$$TF(d,t) = \begin{cases} 0 & if\, freq(d,t) = 0 \\ 1 + \log(1 + \log(freq(d,t))) & otherwise. \end{cases}$$

$$IDF(t) = \log \frac{1 + |d|}{|d_t|},$$

$$TF\text{-}IDF(d,t) = TF(d,t) \times IDF(t).$$

Figure term frequency normalization

We have to create our own function in R environment to finish the TF(d, t) formula, as well as to generate the dot production of TF and IDF, then we will get the weighted matrix. The function receives a term-frequency vector for a certain term, and an int number that means the number of questions this term occurred in; when the number of term-frequency vector is greater than 0, the function calculates the TF-IDF weights. The created function is shown below:

```
get.tf.idf.weights <- function(tf.vec, dt){
```

```
# Computes tf-idf weights from a term frequency vector and a set of document #containing the term

weight = rep(0, length(tf.vec))

weight[tf.vec > 0] = (1 + log(1 + log(tf.vec[tf.vec > 0]))) * log2((1 + N.questions)/dt)

weight

}
```

Then we create a function to count the number of questions in which a certain term occurred, the time of one term occurred in one question will be accumulated:

```
get.weights.per.term.vec <- function(tfidf.row) {

term.df <- sum(tfidf.row > 0)

tf.idf.vec <- get.tf.idf.weights(tfidf.row, term.df)

return(tf.idf.vec)

}
```

Now we use R built-in function “apply” to apply our created function on the vectors (every row).

```
> tfidf.matrix <- t(apply(term.q.matrix.den, c(1), FUN = get.weights.per.term.vec))
```

The argument c(1) means to apply the function “get.weights.per.term.vec” on every rows of matrix “term.q.matrix.den”. We can also assign the column name of “term.q.matrix.den” to “tfidf.matrix” so that there will be a clear presentation.

```
> colnames(tfidf.matrix) <- colnames(term.q.matrix.den)
```

To print a small part of “tfidf.matrix”, for example, to see the TF-IDF weights of the 401st to 405th terms in the 17th to 47th questions, type in:

```
> tfidf.matrix[401:405, 17:47]
```

The output in console looks like this:

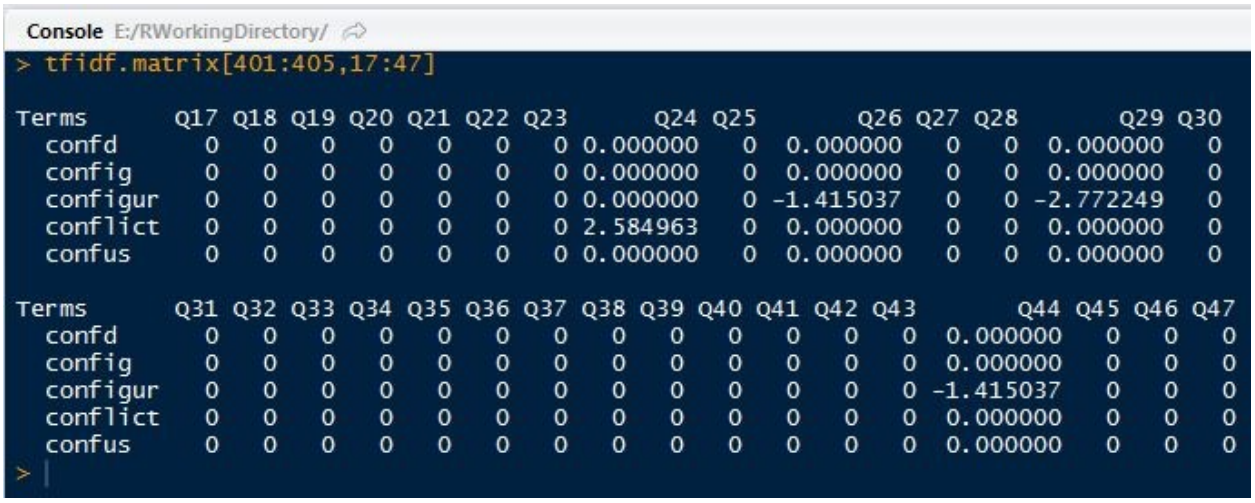


Figure inspect certain part of TF-IDF matrix

- **Normalize the vector and calculate cosine distance**

Before we work out the cosine distance among the questions, we have to normalize the column vectors of “tfidf.matrix” to 1 so that calculating **cosθ** will only be a dot product of two vectors.

```
> tfidf.matrix <- scale(tfidf.matrix, center = FALSE, scale = sqrt(colSums(tfidf.matrix^2)))
```

The argument “scale” should equal to either a logical value or a numeric vector of length equal to the number of columns of “tfidf.matrix”. Now we print the same part of the matrix, the weights will be slightly different:

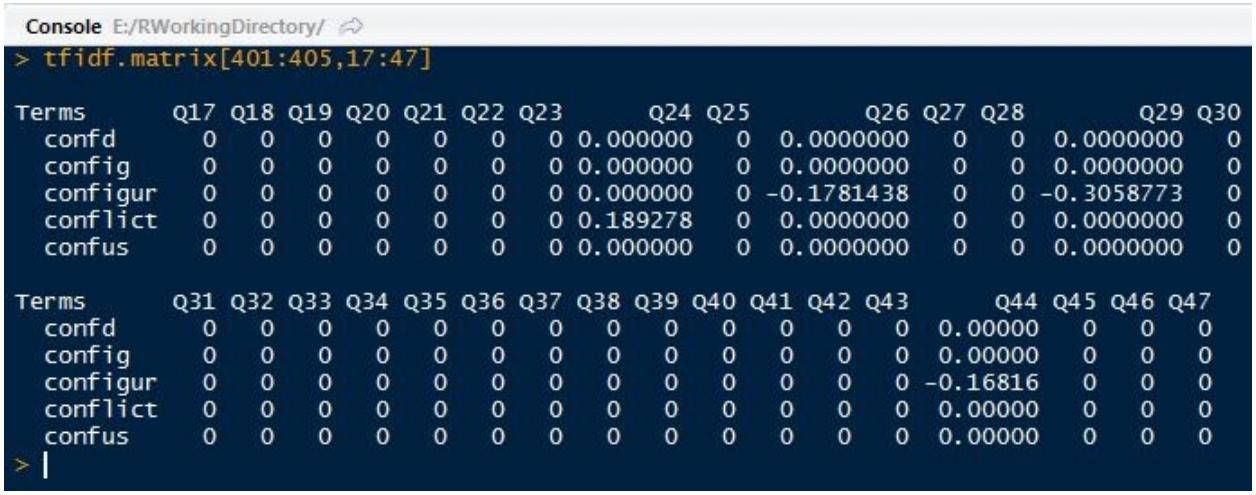


Figure inspect part of normalized TF-IDF matrix

Since we have normalized the “tfidf.matrix” vector, we can simply get every question’s similarity by getting the dot products of the “tfidf.matrix” with itself:

```
> q.scores <- t(tfidf.matrix) %*% tfidf.matrix
```

Then we convert “q.scores” to R data frame:

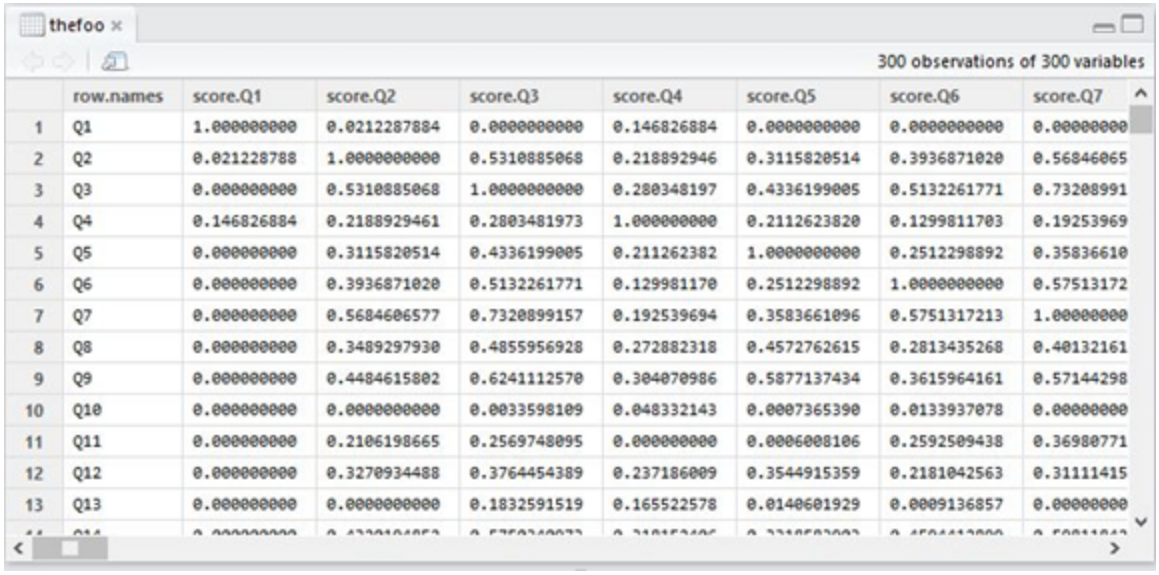
```
> results.df <- data.frame(score = t(q.scores))
```

Export data to CSV file

The R built-in function “write.table” takes care of data export:

```
> write.table(results.df, file = "thefoo.csv")
```

We can find the file “thefoo.csv” in the R project directory. When we re-import the data, the file looks like this, there are questions that are nothing in common with each other, and notice that when a question is compared with itself, the cosine distance result is 1, which means it’s completely the same:



thefoo ×

300 observations of 300 variables

	row.names	score.Q1	score.Q2	score.Q3	score.Q4	score.Q5	score.Q6	score.Q7
1	Q1	1.000000000	0.0212287884	0.0000000000	0.146826884	0.0000000000	0.0000000000	0.00000000
2	Q2	0.021228788	1.0000000000	0.5310885068	0.218892946	0.3115820514	0.3936871020	0.56846065
3	Q3	0.000000000	0.5310885068	1.0000000000	0.280348197	0.4336199005	0.5132261771	0.73208991
4	Q4	0.146826884	0.2188929461	0.2803481973	1.000000000	0.2112623820	0.1299811703	0.19253969
5	Q5	0.000000000	0.3115820514	0.4336199005	0.211262382	1.0000000000	0.2512298892	0.35836610
6	Q6	0.000000000	0.3936871020	0.5132261771	0.129981170	0.2512298892	1.0000000000	0.57513172
7	Q7	0.000000000	0.5684606577	0.7320899157	0.192539694	0.3583661096	0.5751317213	1.00000000
8	Q8	0.000000000	0.3489297930	0.4855956928	0.272882318	0.4572762615	0.2813435268	0.40132161
9	Q9	0.000000000	0.4484615802	0.6241112570	0.304070986	0.5877137434	0.3615964161	0.57144298
10	Q10	0.000000000	0.0000000000	0.0033598109	0.048332143	0.0007365390	0.0133937078	0.00000000
11	Q11	0.000000000	0.2106198665	0.2569748095	0.000000000	0.0006008106	0.2592509438	0.36980771
12	Q12	0.000000000	0.3270934488	0.3764454389	0.237186009	0.3544915359	0.2181042563	0.31111415
13	Q13	0.000000000	0.0000000000	0.1832591519	0.165522578	0.0140601929	0.0009136857	0.00000000

Figure Cosine Distance result

There is further manipulation of the table could be done, for example, half of the figures are redundant, and also we don't need the compared number of any question with itself. In R language, there is a convenient way to remove an element in a matrix. For example, if you want to remove the 10^{th} element in a matrix, simply type in:

```
> your.matrix <- your.matrix[ 1: 9, 11: length.of.your.matrix]
```

KDD Principles

Data Extraction and Cleaning

Quality Analysis

To calculate the overall score of a question, we need to extract certain attributes in the *posts.xml* in order to work with the data. In order to parse the XML data format unique to this dataset, we use Pig with Piggybank to perform the job. After parsing and extracting the necessary attributes, Pig stores the result data into the predefined table in shared datatable, HCatalog, for Hive to access the data.

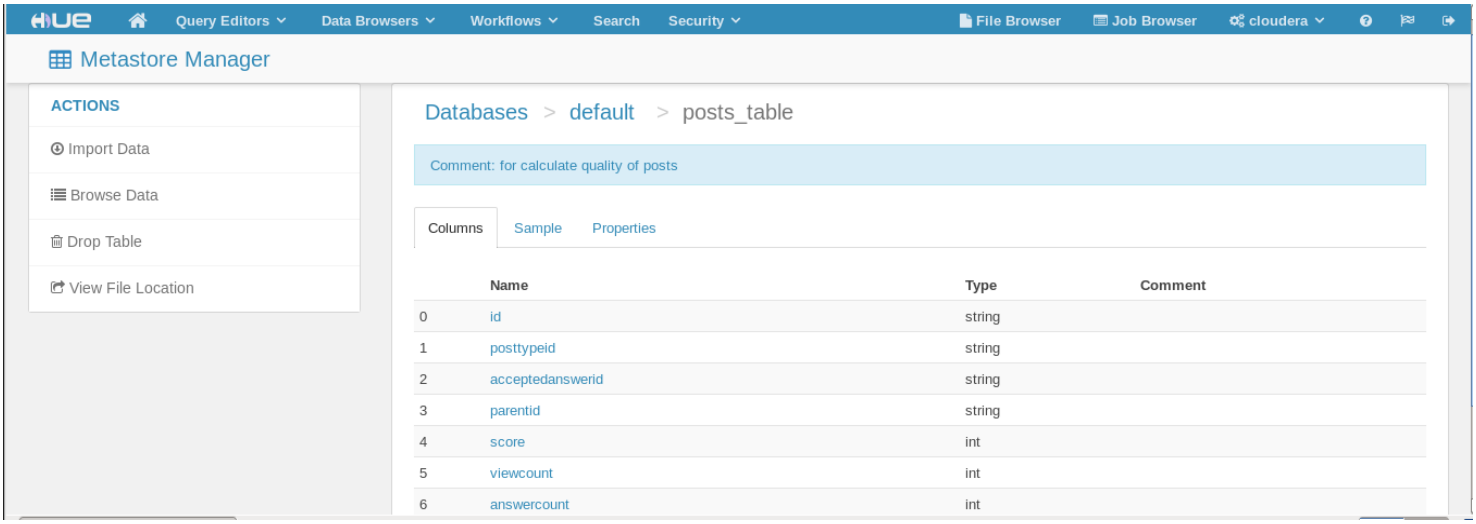


Figure: Table in HCatalog

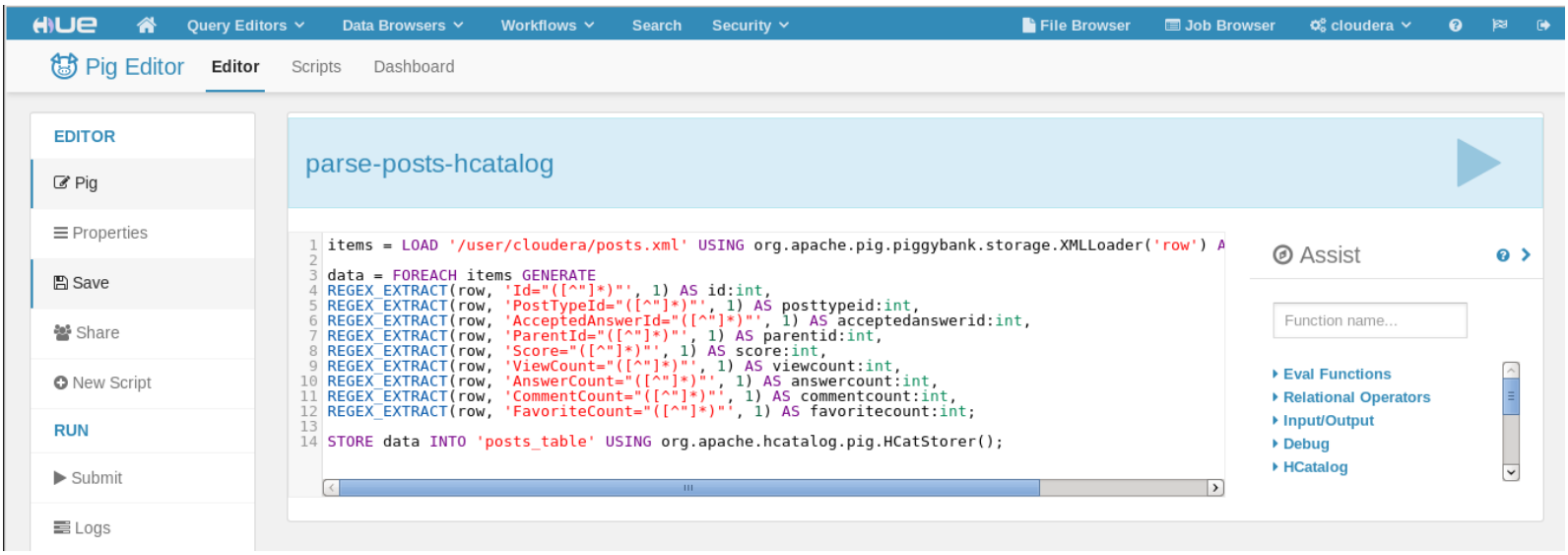


Figure: Quality attributes Extraction

Similarity Analysis

To perform the content based analysis on the questions, we must extract the Body attribute in the post data. Since post data contains not only questions but all posts data including answers and tags, we have to choose only question type by specifying “PostTypeId = 1.” Additionally, cleaning the question’s content by removing HTML tags, meaningless symbols, and formatting for import to R is must. Because of the convenience and extensibility, we mostly used ‘grep’ and ‘sed’ to extract the desired data. For complete steps and a detailed explanation, please refer to the comments in source code: clean_posts.sh.

Original

<row Id="1" PostTypeId="1" Body="<p>Every time I turn on my computer, I see a message saying something like:</p>

<pre><code>Your battery may be old or broken.
</code></pre>

<p>I am already aware that my battery is bad. How do I suppress this message?</p>
" ... />

Extracted

1,Every time I turn on my computer I see a message saying something like Your battery may be old or broken I am already aware that my battery is bad How do I suppress this message

Link Analysis

There are two data providing link information among questions. As mentioned above, a comment sometimes contains a hyper reference to another question. We must include this data into the existing Postlink data in the source data. To extract only the links among all comments, we first used the grep to grab only the comments having an hyer reference part in their body. Since ever URL follows Restful convention, the number in the URL refers to a question id: *http://askubuntu.com/questions/quesiton_id.* Because we are only interested in the question ids, we then replace the entire URL with only its question id. For complete steps and a detailed explanation, please find the comments in source code: clean_comments.sh. Now, the two post data, Postlink and CommentLink, are ready in XML format. We use Pig to extract the certain attributes and convert them into CSV format; source code is available in pig_latin.sh.

Original

<row Id="5590" PostId="5378" Score="1" Text=" … http://askubuntu.com/questions/3359/ ... />
<row Id="5590" PostId="5378" Score="1" Text="3359" />

Extracted

postId,relatedPostId
3,258

Data Aggregation

Similarity Analysis

Since every answer is belong to a question, we must also include the total score of answers into the question. After all, the total score of a question is defined by following aggregation.

- ➔ overall question score: (score + answercount + commentcount + favoritecount)
- ➔ overall answer score of the question: (score + commentcount)

Because every answer contains the foreign key to its parent question, namely 'ParentID' we used the key to perform join. Moreover, after the join we have to union the those questions which don't have any answer. For complete steps and a detailed explanation, please find the comments in source code: qualityStats.sql.

Original

<row Id="1" PostTypeId="1"... Score="40" … AnswerCount="2" CommentCount="1" FavoriteCount="5" />
<row Id="2" PostTypeId="2" ParentId="1" … Score="31" ... CommentCount="0" />

Extracted

id,posttypeid,acceptedanswerid,parentid,score,viewcount,answercount,commentcount,favoritecount
1,1,2,,40,1602,2,1,5
2,2,,1,31,NULL,NULL,0,NULL

Result

Id,Size
1,82

Link Analysis

Because comments can appear in both questions and answers, we need to integrate them to the parent it belongs to. If it's parent is happened to be an answer, we have to further refer back to its parent as the intention is to obtain the connection between questions. The intermediate query and step by step instructions is available in source code: commentLinkStats.sql. Finally, we combined the resulted data of comment links with the existing links obtained in the extraction step, and form the complete set of links between a question to another question. For the detailed explanation, please refer to the comment in the source code: postLinkStats.sql.

Original

postId,relatedPostId
3,258

Intermediate result

postId, relatedPostId, posttypeid, parentid
5378,3359,2,5363

result

Source,Target
3,258

Data Mining

As in the mission statement, we heavily used graph visualization tool to run several statistics. Because of the limitation of memory available in commodity computer, we were not able to compute the Cosine similarity distances among all questions. Nevertheless, even we somehow managed to handle it, the result data must be too large to fit into the graph application. As such, we divided the analysis into three part.

300 Questions Visualization

Being not able to visualize the entire network, we took the first 300 questions to substitute our goal. This analysis runs with the all information we obtained from the previous steps: Quality overall score, Cosine similarity distance, and Links. Moreover, in order to compare the result between the cosine distance scores of TfIdf and Binary implementation in document term matrix, we carried out the analysis for both.

Data import

Edges

The result data of cosine similarity score is in symmetric matrix format. Although we converted the matrix into individual score with question pairwise, Gephi is smart enough to take the input in the original symmetric matrix format.

score.Q1 "score.Q2" "score.Q3" "score.Q4" "score.Q5"
Q1 1 0.0212287883900368 0 0.146826883546878 0 (
Q2 0.0212287883900368 1 0.531088506771671 0.21
Q3 0 0.531088506771671 1 0.2803481973238 0.433(
Q4 0.146826883546878 0.218892946142563 0.2803(
Q5 0 0.311582051426774 0.433619900495222 0.211
Q6 0 0.393687101968351 0.513226177102197 0.129
Q7 0 0.568460657692586 0.732089915651293 0.192
Q8 0 0.34892979299723 0.485595692776403 0.2728
Q9 0 0.448461580204388 0.624111257030664 0.304
Q10 0 0 0.0033598109168657 0.0483321433489837

Figure: gephi similarity matrix import format

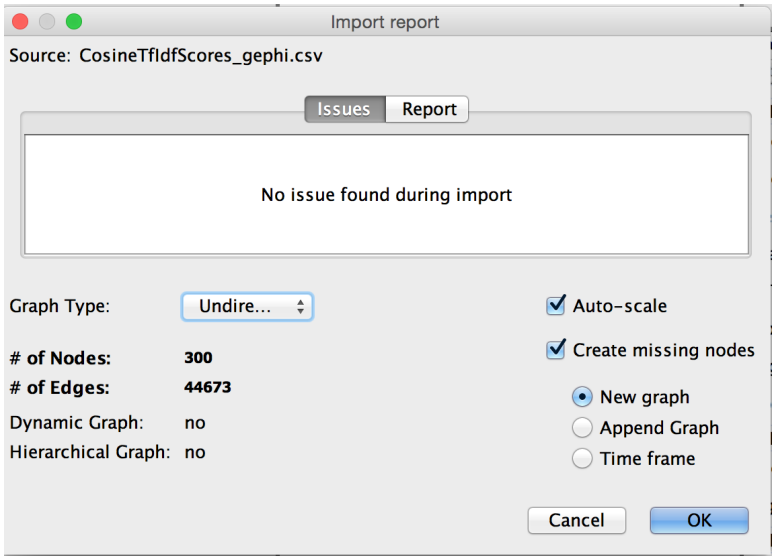
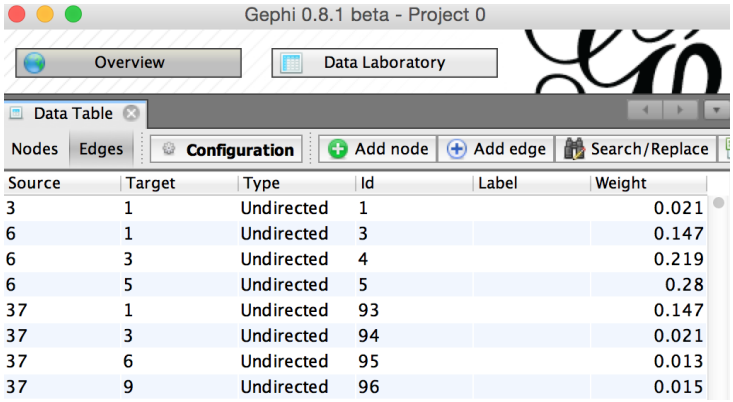


figure: Choose “Undirected” for the symmetric matrix



Source	Target	Type	Id	Label	Weight
3	1	Undirected	1		0.021
6	1	Undirected	3		0.147
6	3	Undirected	4		0.219
6	5	Undirected	5		0.28
37	1	Undirected	93		0.147
37	3	Undirected	94		0.021
37	6	Undirected	95		0.013
37	9	Undirected	96		0.015

figure: result edges overview

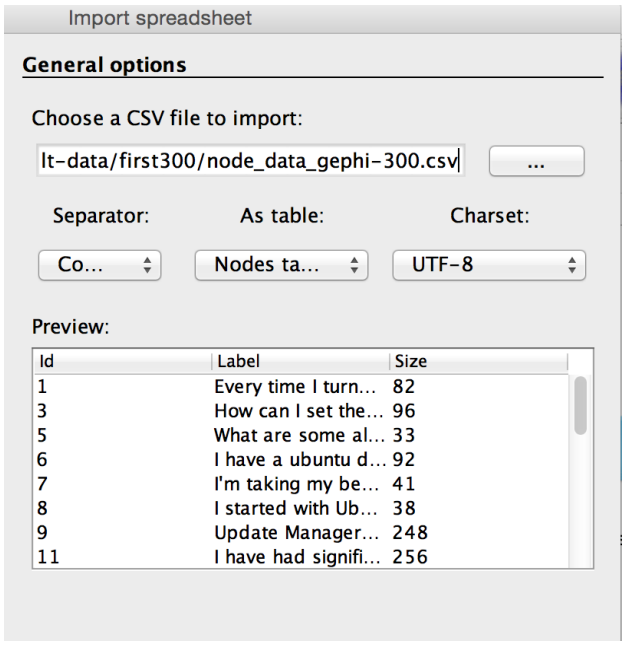
Issue

During data import, we found node #1004 has no connection to the other nodes. As we checked the cosine similarity result, we found the entire scores for the node is missing. We solved this issue simply removed the node.

258	0.000000000	0.4144308052	0.6007145355	0.166252697	0.3213361332	0.4448737529	0.5791538433	0.35985305	0.462500678
259	0.000000000	0.4082768688	0.5681873343	0.276824494	0.5350512452	0.3291953181	0.4695796578	0.59918509	0.770101891
260	0.000000000	0.3118525329	0.4339963220	0.211445777	0.4086861118	0.2514479796	0.3586772039	0.45767322	0.588223932
261	NA	NA	NA	NA	NA	NA	NA	NA	NA
262	0.000000000	0.0000000000	0.0000000000	0.011156471	0.0000000000	0.0000000000	0.0102784124	0.00000000	0.000000000
263	0.026184821	0.2865443818	0.3764900539	0.000000000	0.0000000000	0.3307510490	0.4820442763	0.12000397	0.000000000
264	0.000000000	0.4771008472	0.6171989206	0.245643398	0.3021256969	0.4848730600	0.6916456982	0.33833995	0.434851003

figure: missing cosine similarity data

Node



Layout Algorithm

To resize the nodes, we can apply the Ranking method and change the every node size according to the imported quality score in Size attribute above.

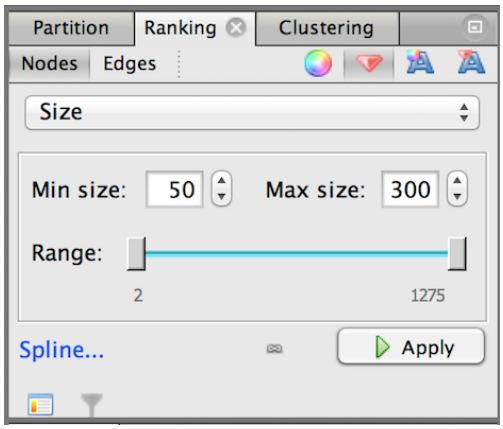


Figure: apply the ranking method to node size

To take the node distance into account, we apply the ForceAtlas2 layout algorithm so that it will figure out the best layout according to the all edge weights which we imported as cosine similarity scores.

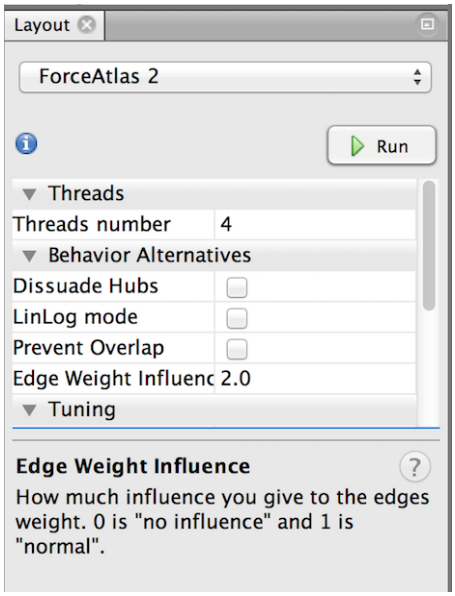


Figure: ForceAtlas2 configuration

Furthermore, we showed the link between the questions if there is any by displaying the edge between the nodes. However, because of the graph density most edges are often hidden behind the nodes. Because there is only 5 links whose source and target are within the first 300 questions, we decided to assign a color to the pair of nodes which have a link.

Stats and Result

The full screen image for the two result is available in the following link:
TfIdf version: https://www.dropbox.com/s/7wrk8q9q87yry8w/similarity300TfIdfGraph.png?dl=0
Binary version: https://www.dropbox.com/s/2ofjq8kuo1veunp/similarity300BinGraph.png?dl=0

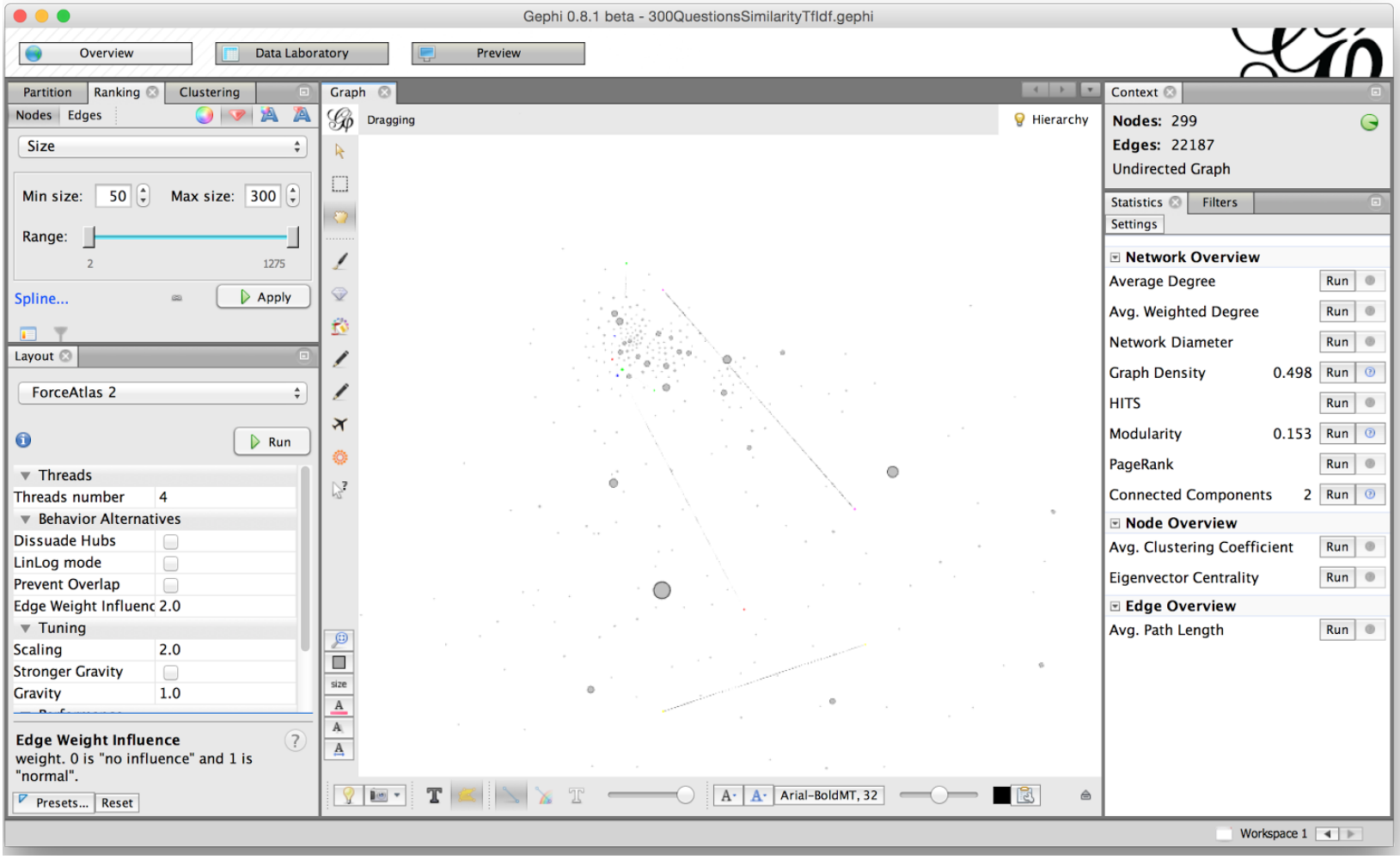


Figure: First 300 Quesions TfIdf Term Document Matrix Cosine Score

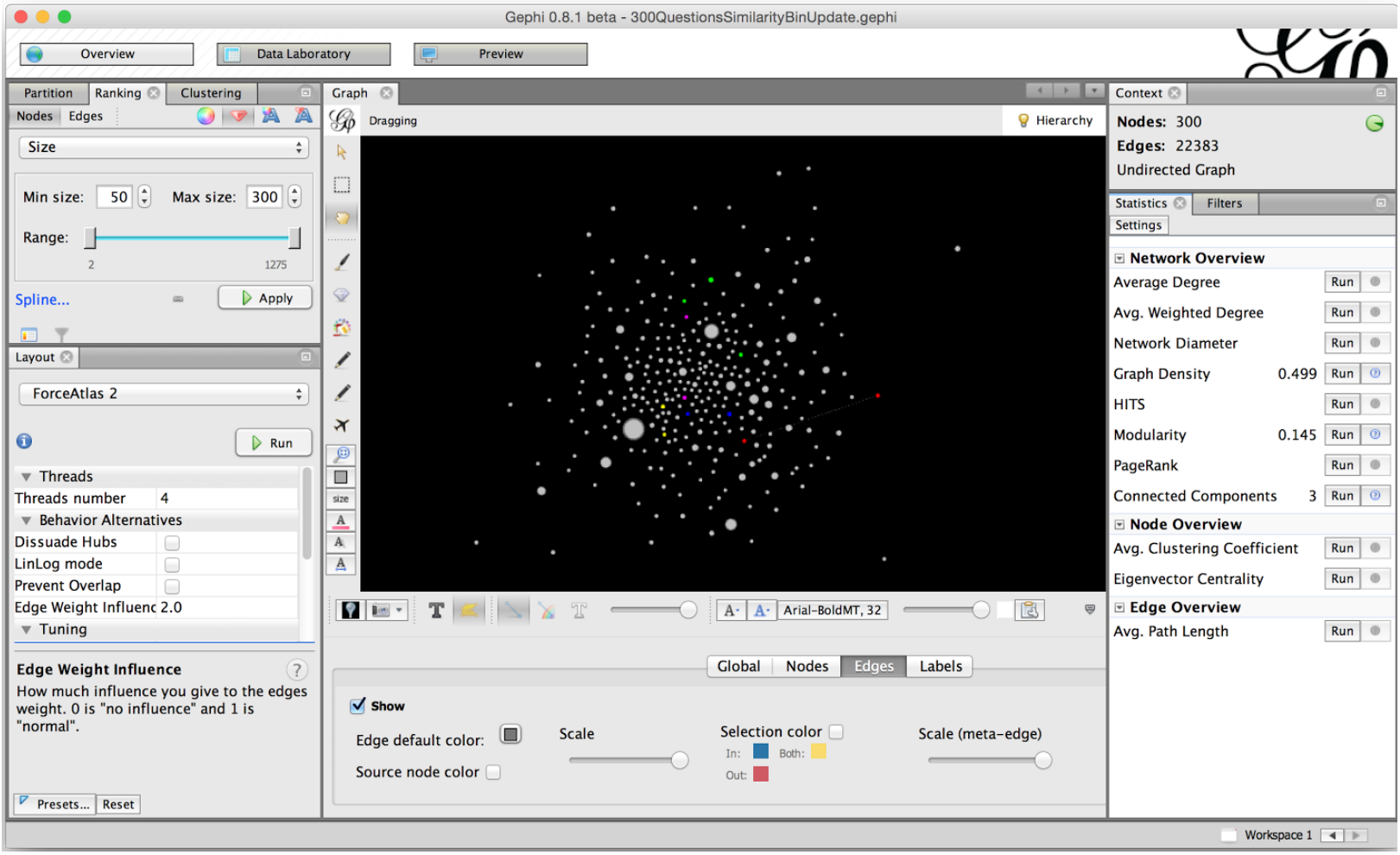


Figure: First 300 Questions Binary Term Document Matrix Cosine Score

Discovery

With the help of graph visualization, we are able to find the following two interesting facts.

1. Compared to binary term frequency, Tfldf provides more accurate and distinct result.
2. Contrary to our expectation, links tend to connect two dissimilar questions rather than similar questions.

All Questions Visualization

As we were not able to calculate the cosine distance for all pairs of questions due to memory and CPU limitation, we only include the quality and links for the whole graph visualization. Thus, the purpose of this analysis is to visualize the correlation between quality of a question and number of links to the question.

Layout Algorithm

After importing the necessary data, we apply the Ranking method to change the noe color based on their quality and then run ForceAtlas2 and OpenOrd layout algorithm to cluster the questions appropriately. The two layout algorithms cluster the nodes by number of its indegree. Nodes with higher indegree will be positioned more center of the graph. In other words, questions having more links to it will appear in the core. To facilitate the view of the graph, we further apply the Ranking method to edges by coloring the node based on the number of reference it receives.

Steps and Validation

Quality Score (Node Size) Stats

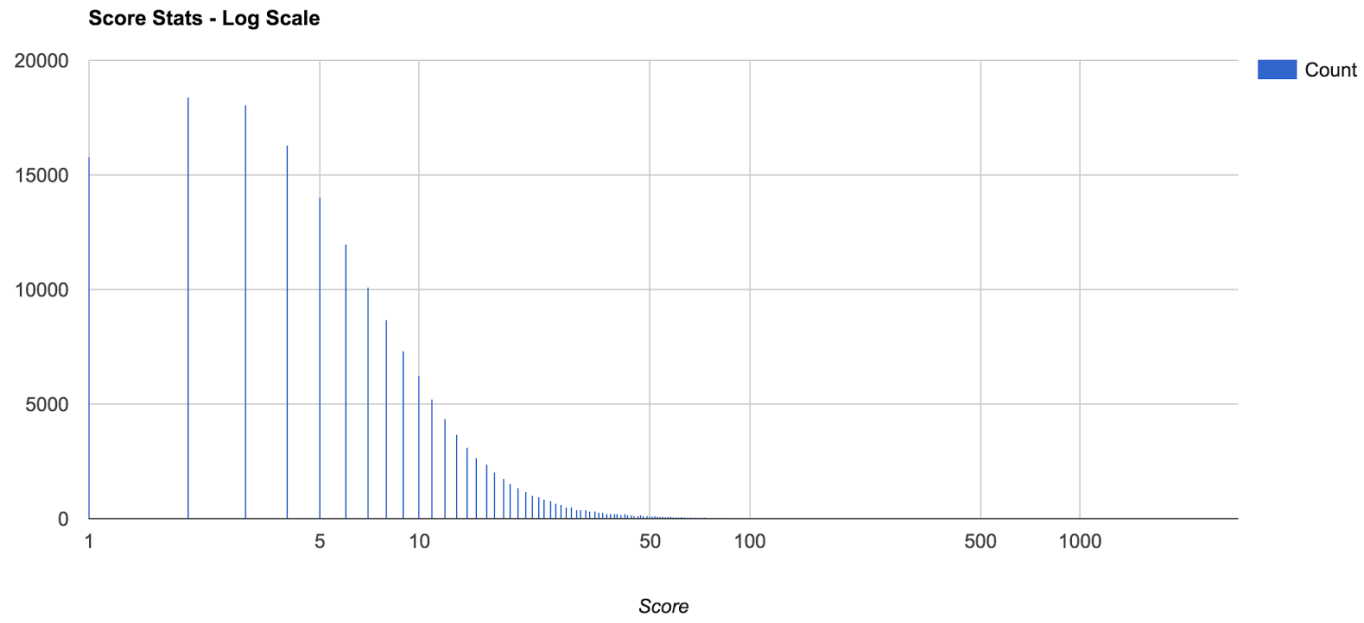


figure: overall score distribution

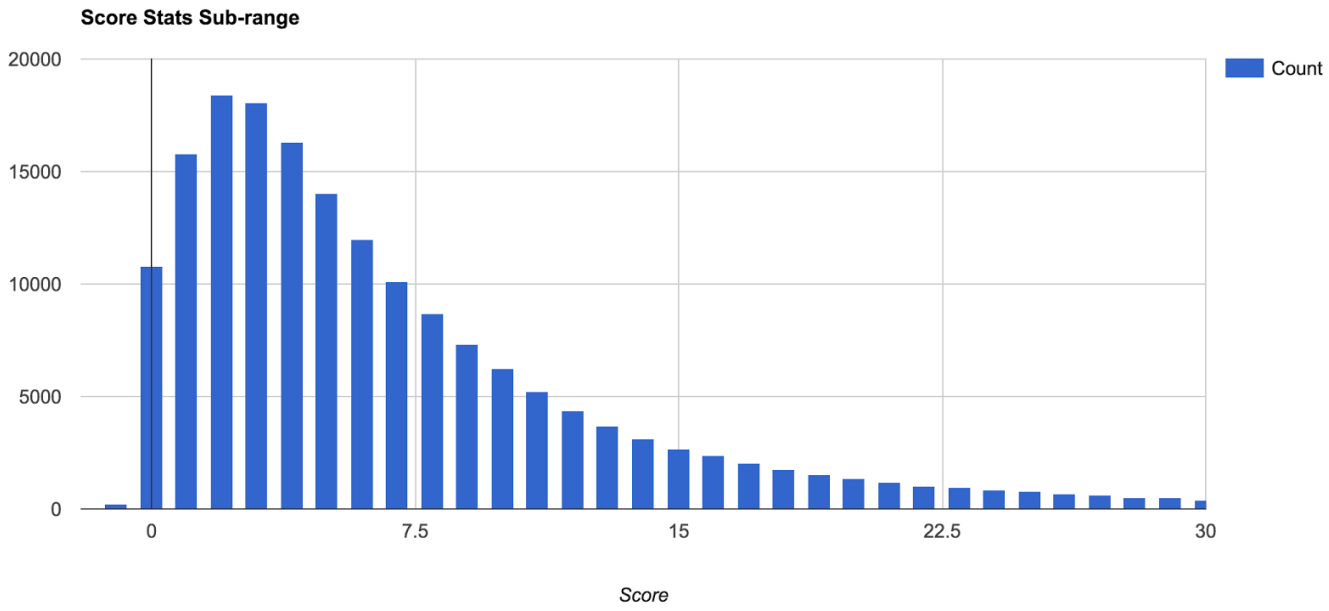


figure: score distribution subrange

Gephi stats

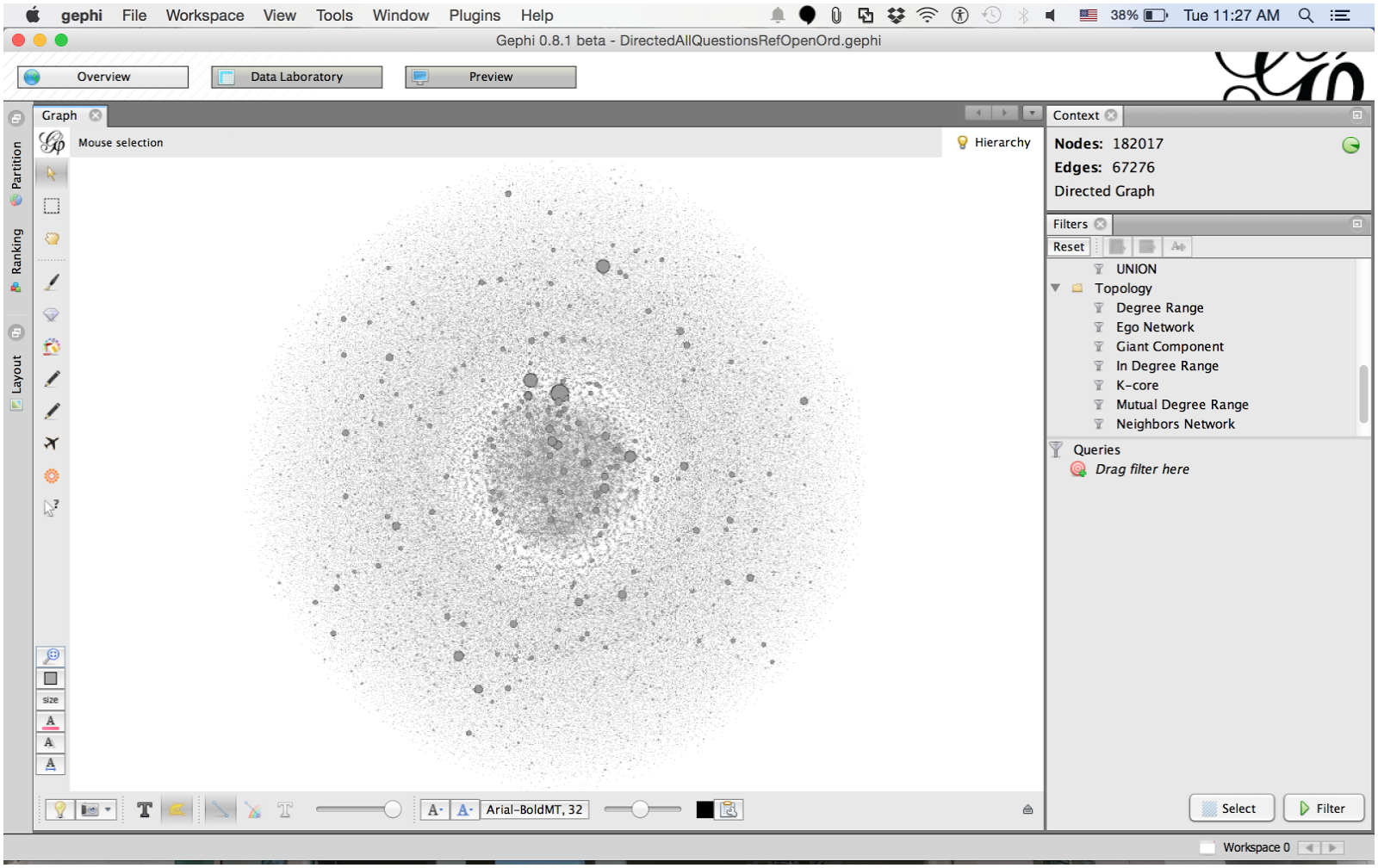


figure: graph after applying ForceAtlas2

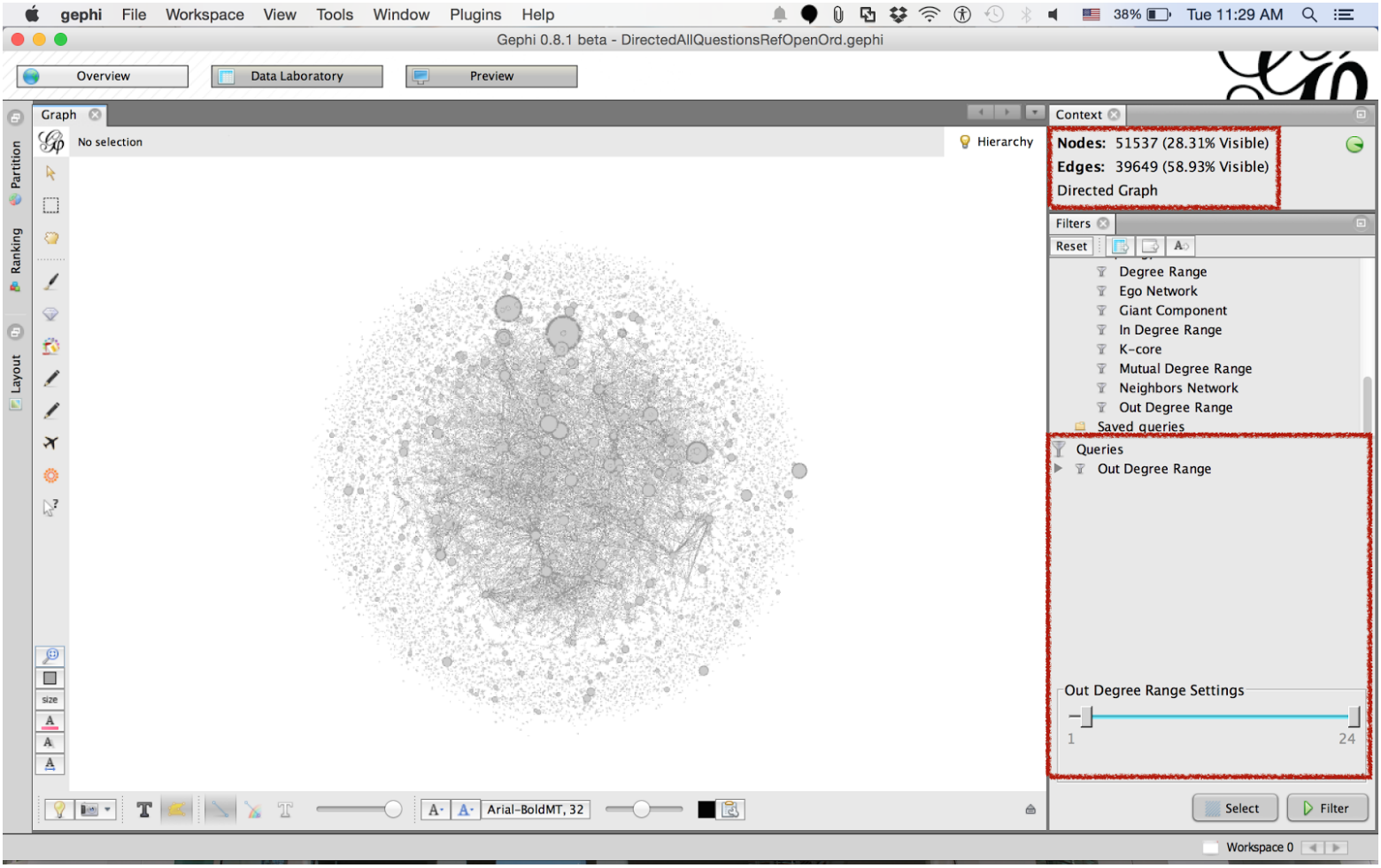


figure: outdegree stats view

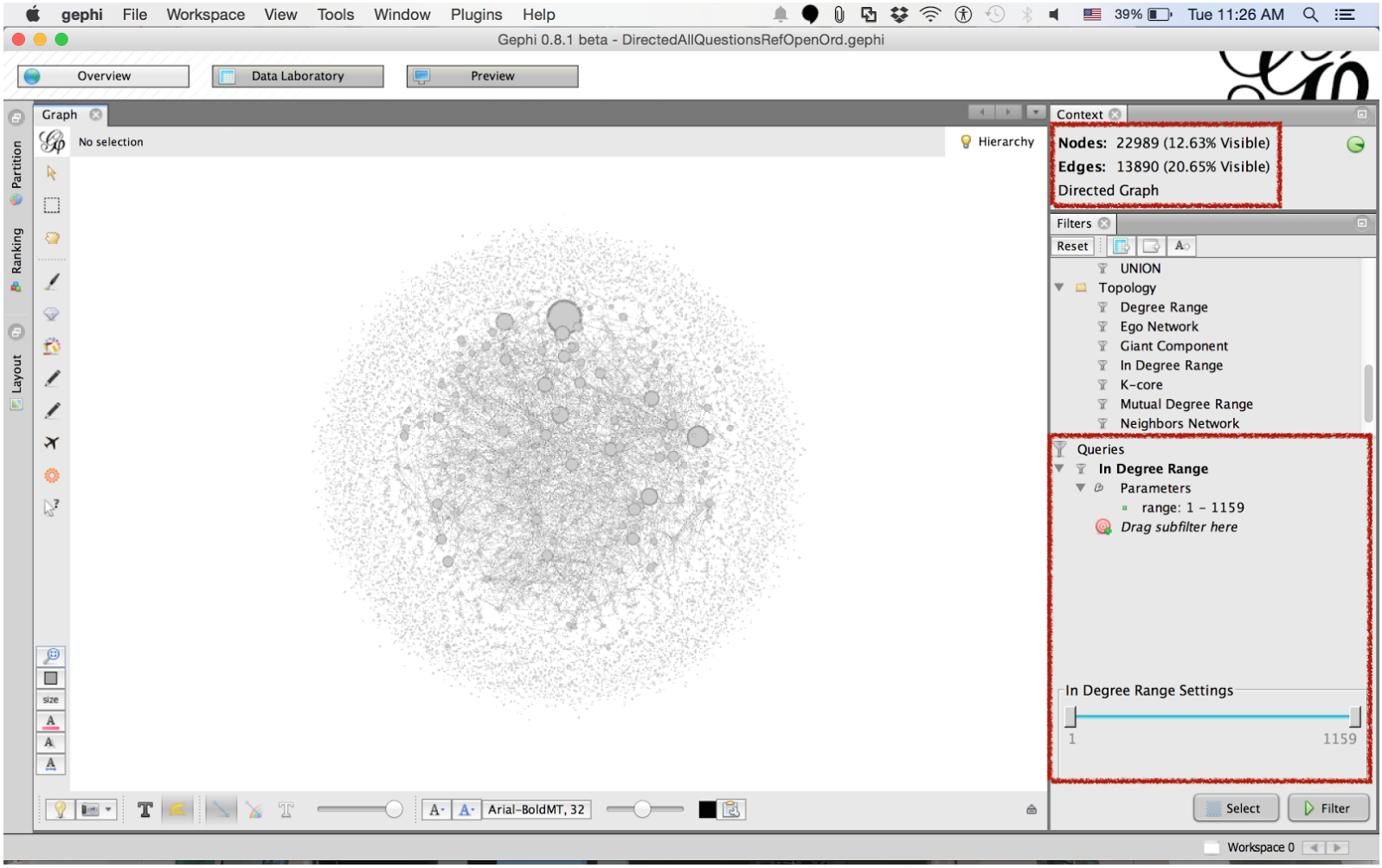


figure: indegree stats view

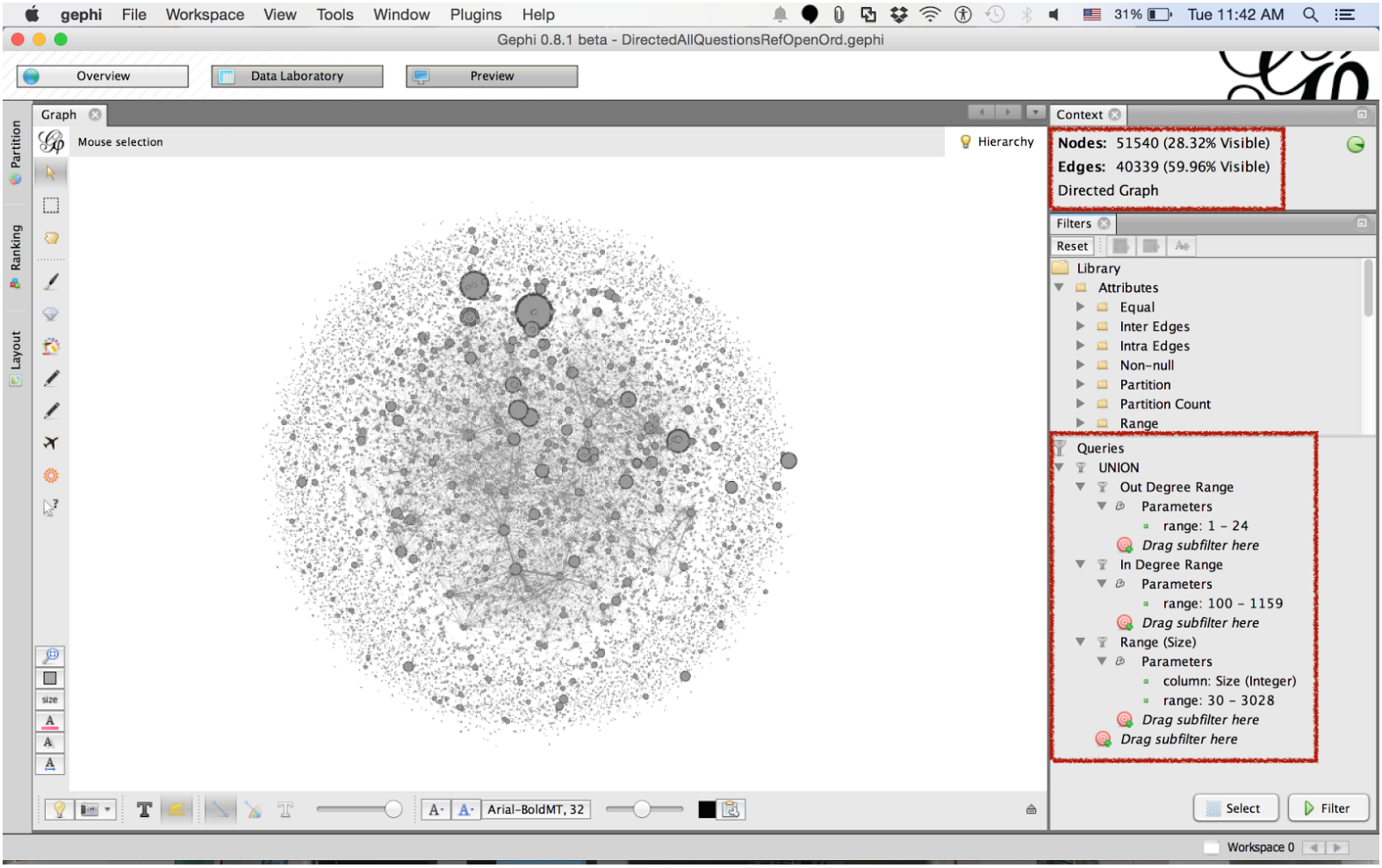


figure: graph after applying a filter

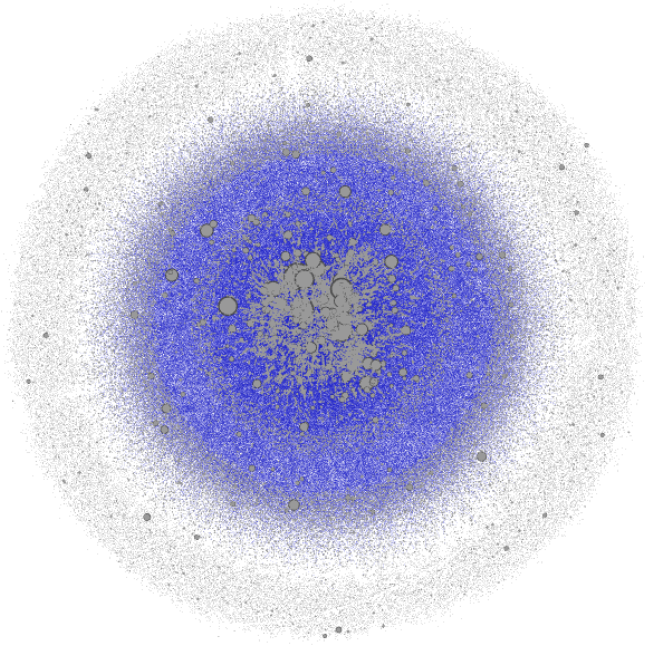


figure: graph after running OpenOrd with edges colored in blue

Result and Stats

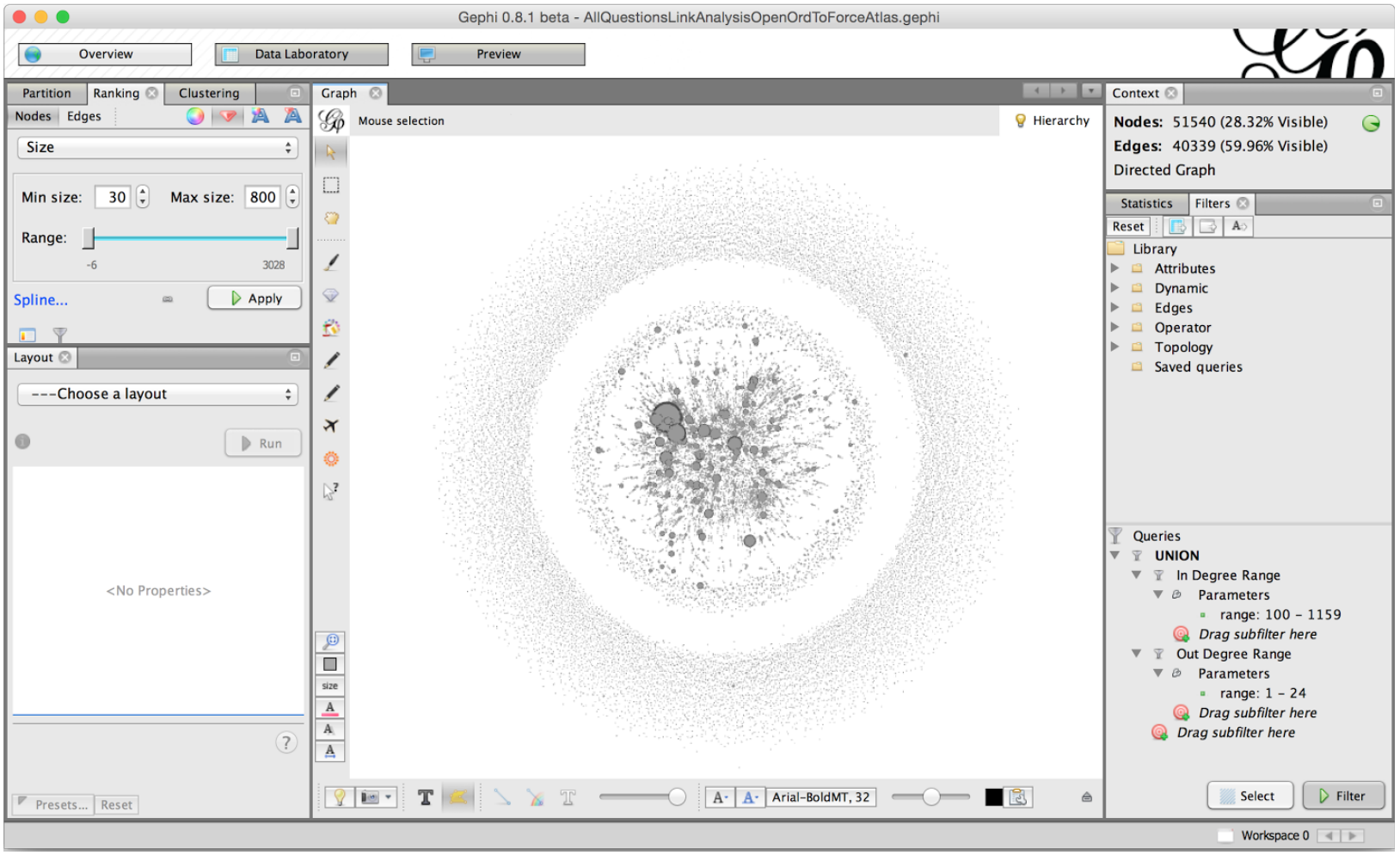


figure: result graph with filter

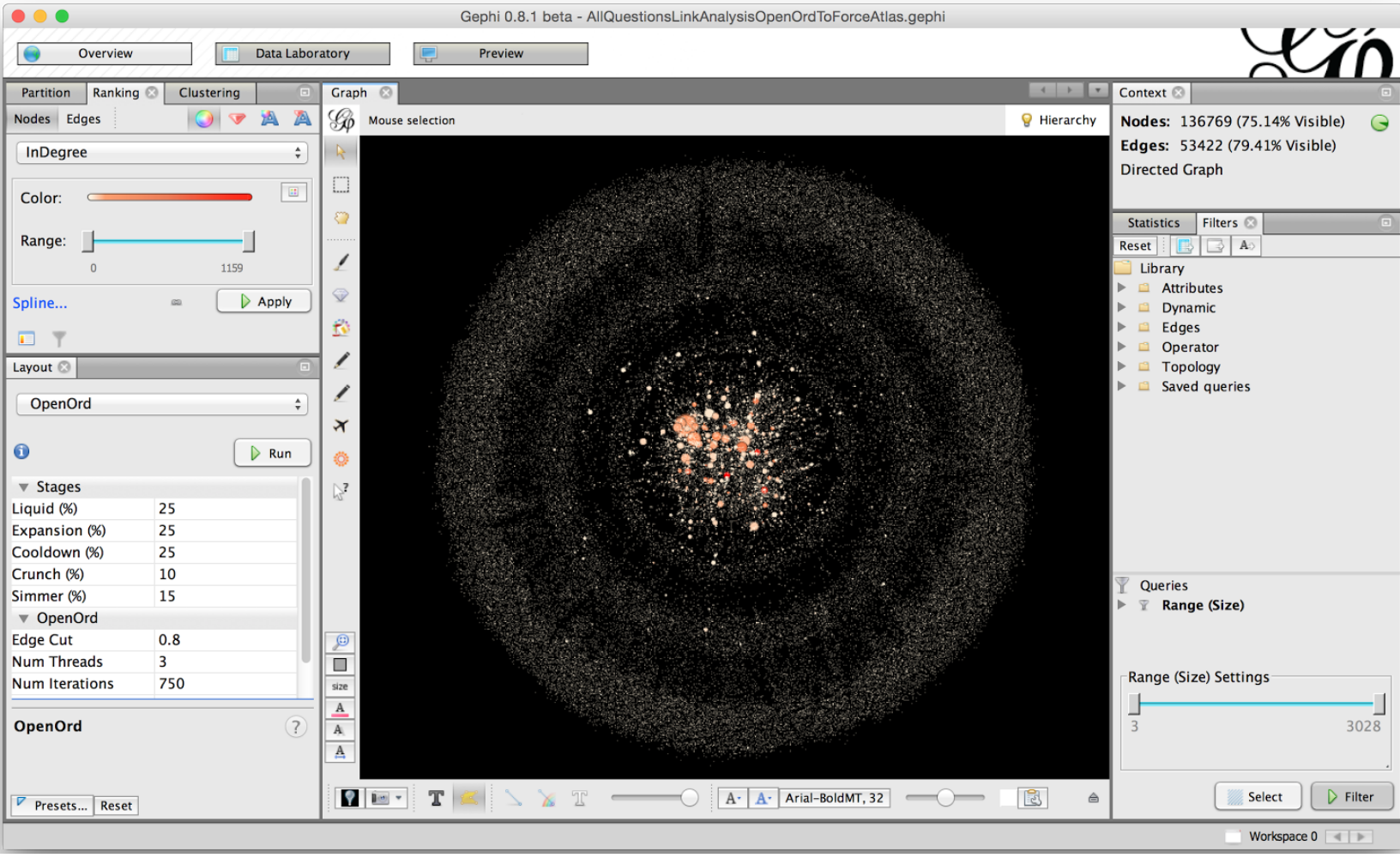


figure: final result graph

Discovery

The final result graph indicates, the more links a question is referred to, the higher quality it tends to have. This result fairly validate the Pagerank algorithm in the lecture. However, it's worth to point out that the most referenced question is not always the question with most highest quality.

1000 Questions Visualization

Lately, we did cluster analysis on the first 1000 questions. By using the Cosine similarity score calculated based on Tfldf on the 1000 questions, we managed to push the memory limitation of available Gephi to the edge. After importing the large symmetric matrix which worths 132MB, we run the Modularity Statistics to get the classification index for every node. Then, we apply the Partition method to color all nodes based on its assigned index. Finally, we run the OpenOrd layout algorithm to position the nodes properly.

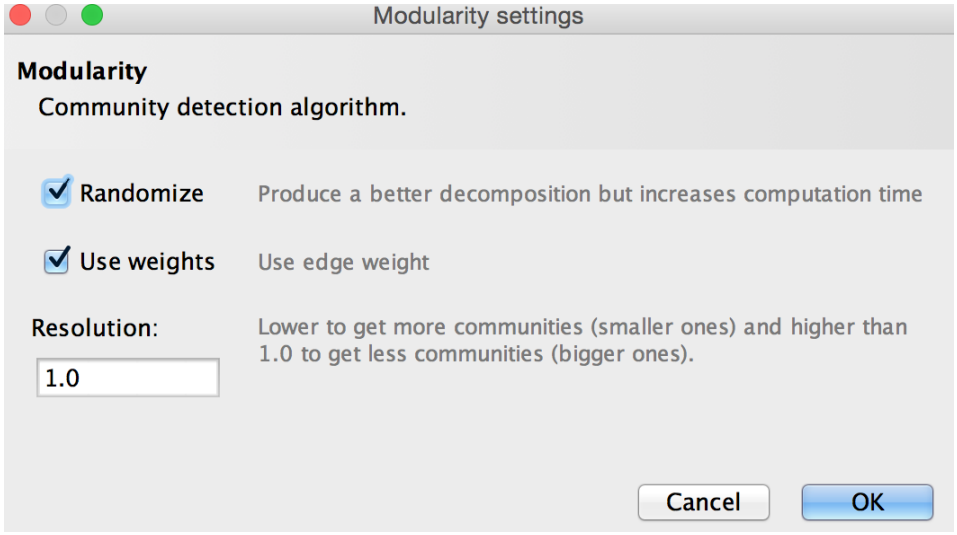


figure: Modularity stats settings

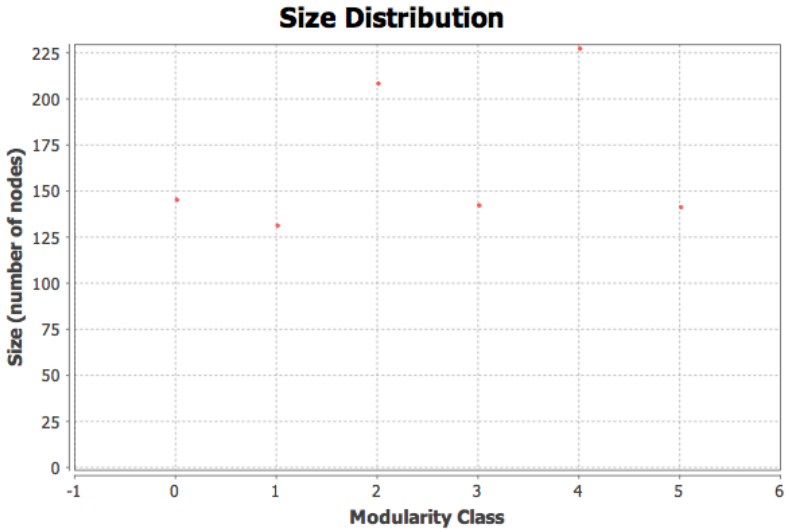


figure: edge weight clustering distribution

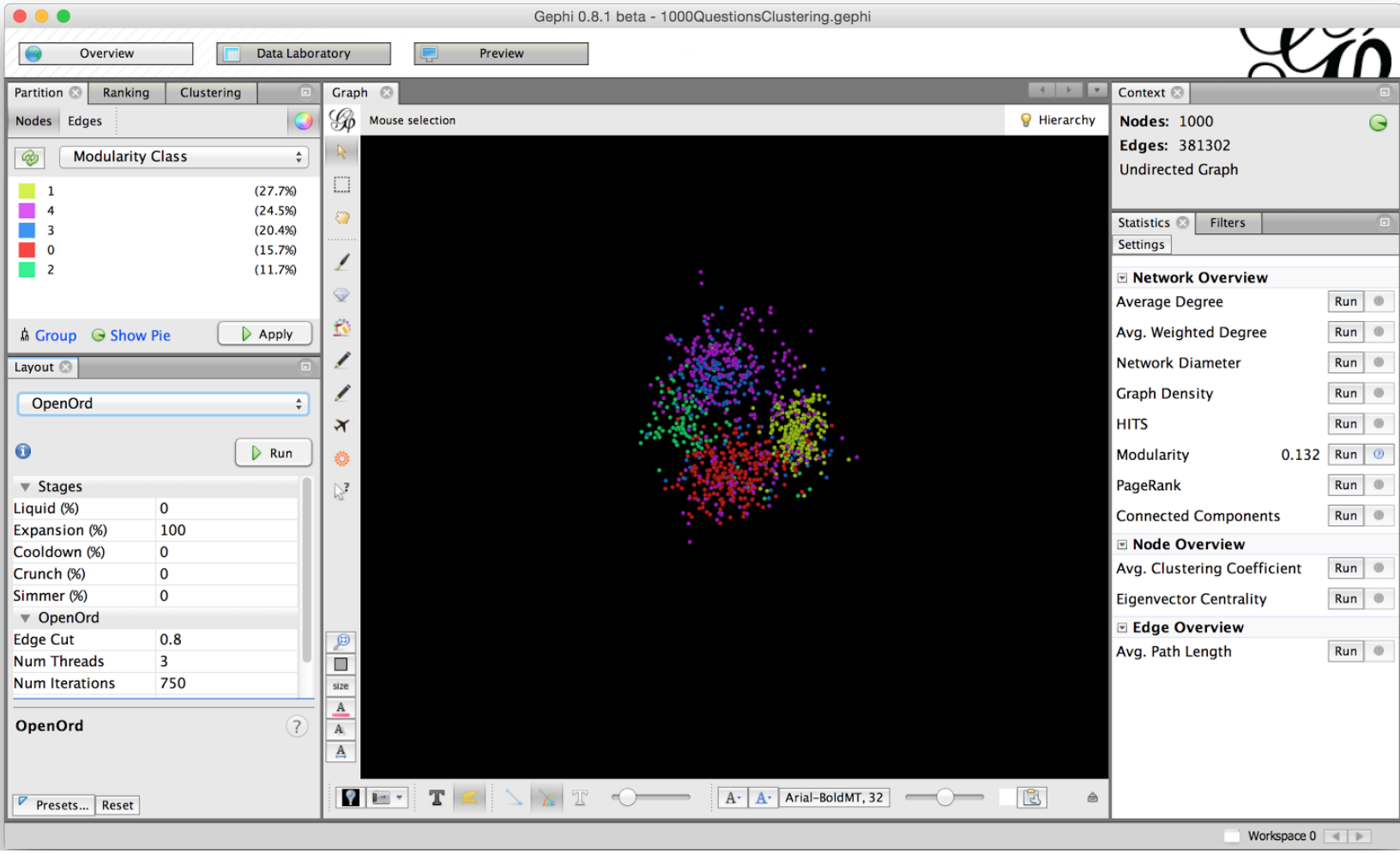
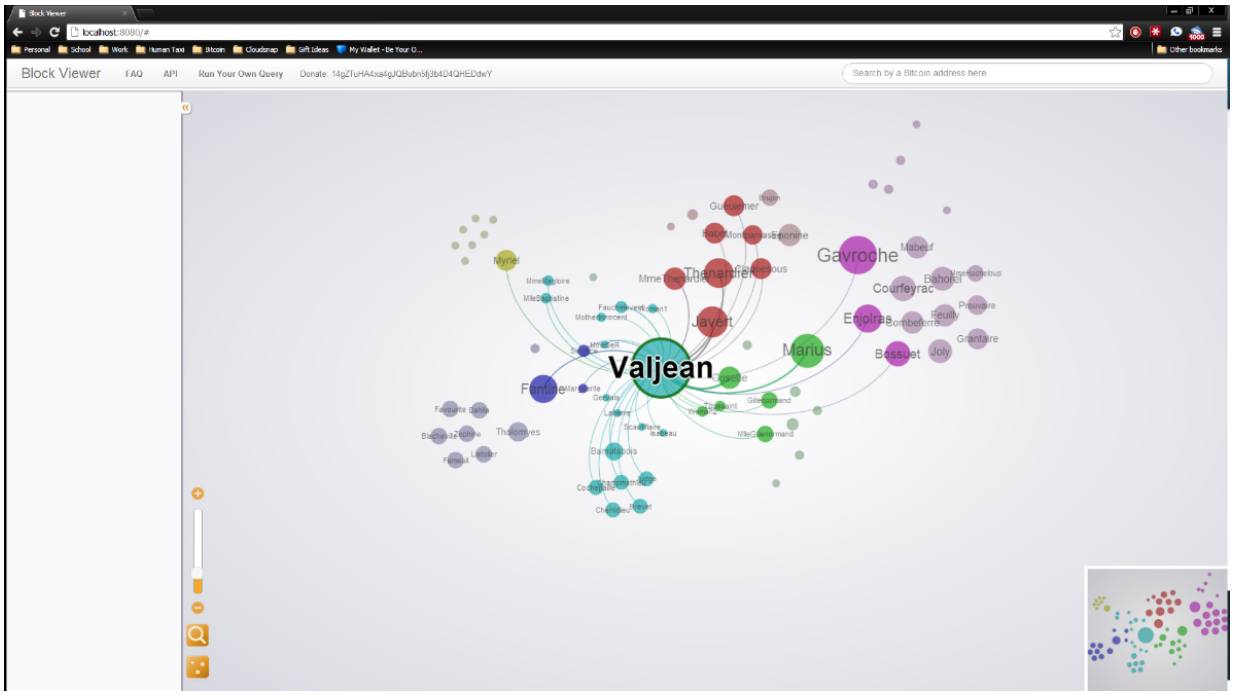


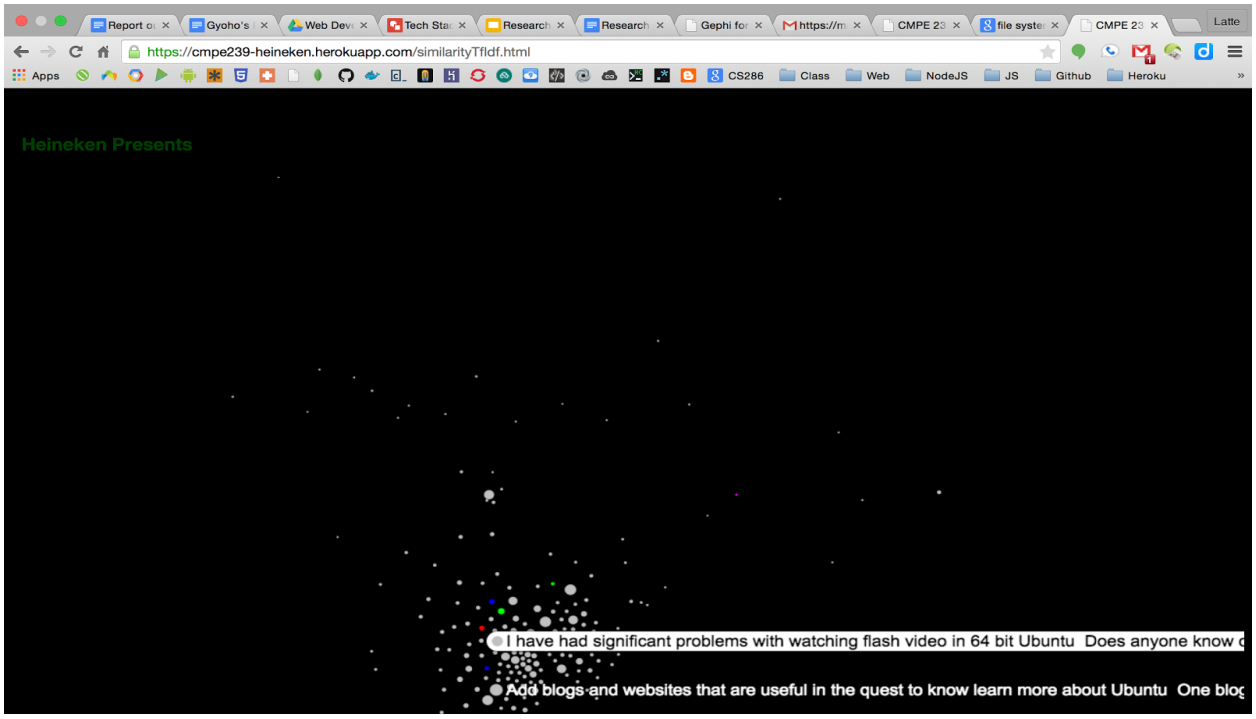
figure: final result graph

Client side design

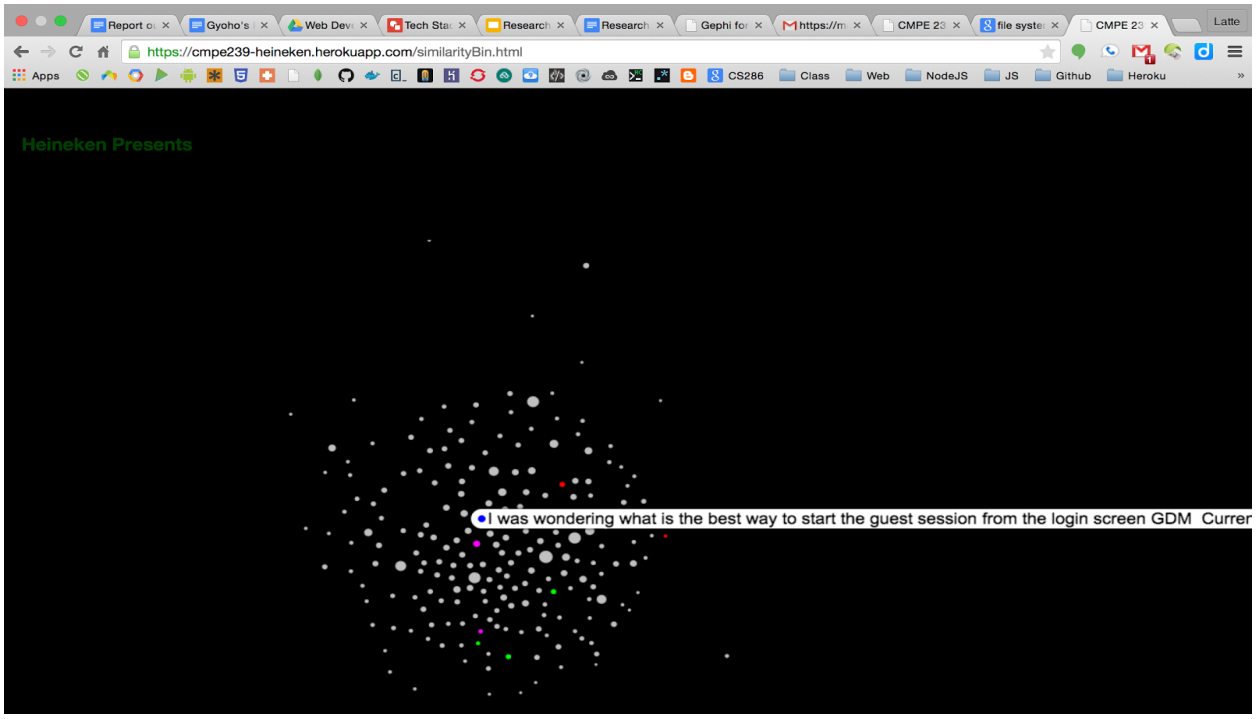
Client Side Scripting helps in building an interactive environment for client to enable him to browse the application. The client requests pages from the server and displays them in the browser for the given user that has requested it. All the web pages that we used on internet are client side pages. All the pages that we access are examples of client side design.
Client side design is basically an architecture that would be show how the client side would be working. In this project the client would be accessing a base url or a home page of the project. This page would contain the links to all the different links that would show the visualization graph for different scenarios. All these different scenarios are different only because all have them show the graph based on different attributes in them.
Below is an example of a gephi application running on localhost would look like.



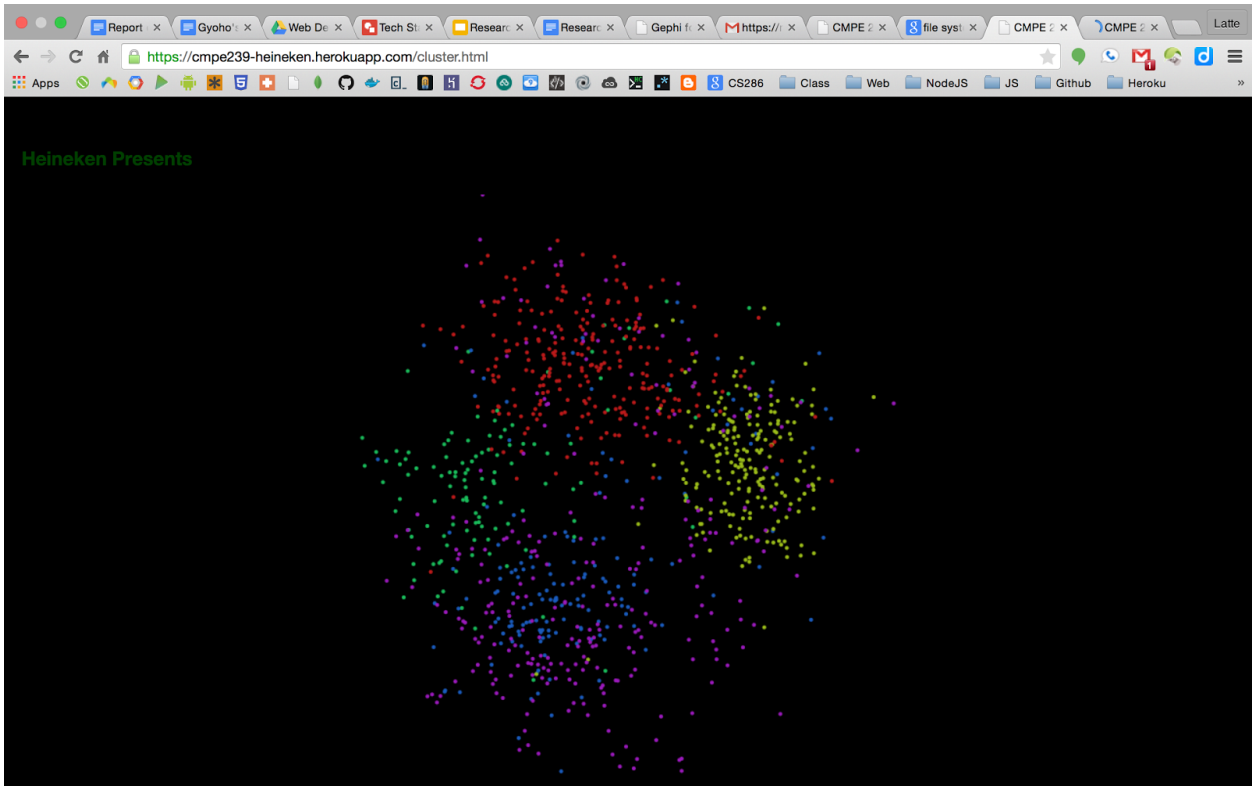
Below are the UI pages which would be visualizing the graphs for different scenarios.
This is the link of the UI page showing output for 300 questions.



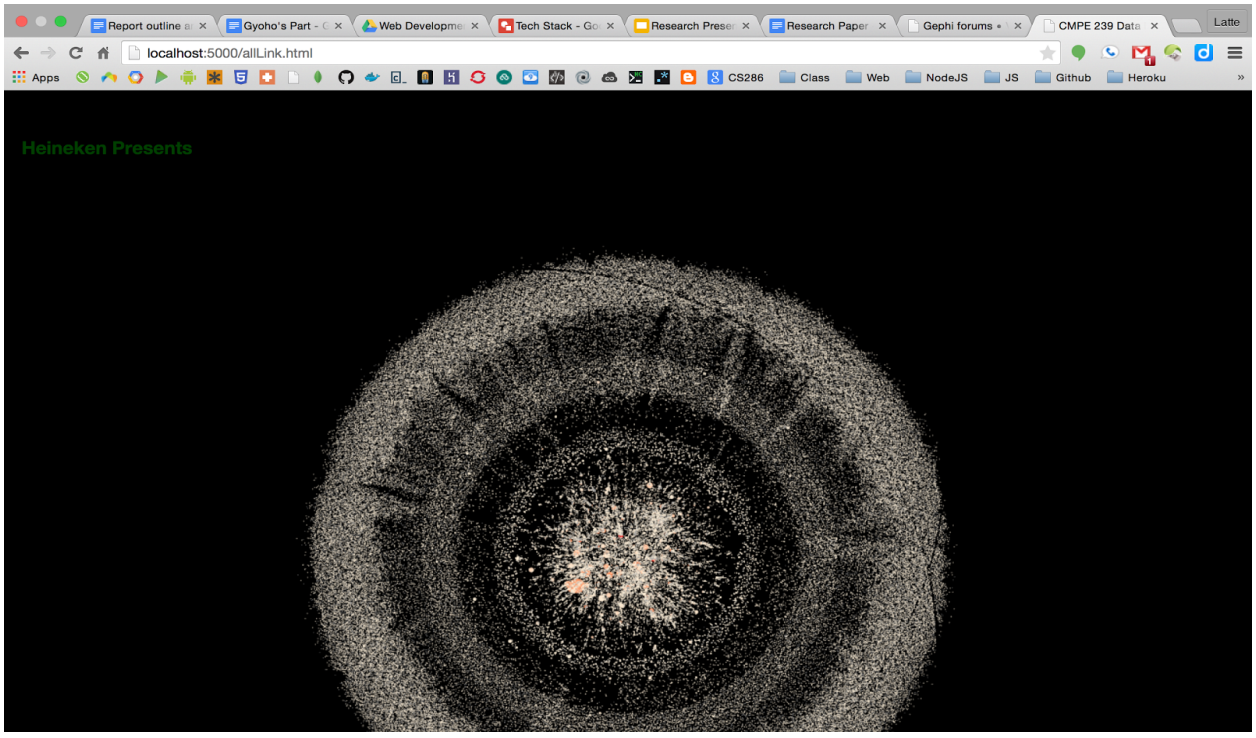
This link shows the UI showing visualized output for 1000 questions.



The below link shows output for 3000 questions that are visualized.



The last link shows output UI for all the question.



Testing

Load Testing

Load testing involves applying stress that is generic in nature to an application for the purpose of verifying that it performs as expected under normal conditions. It is most often used to determine whether a particular frontend software, a website for example, can handle loads that it's likely to come across during regular usage. This is slightly different as compared to stress testing, which is considered to be the next stage of load testing. In stress testing, the system is applied load that is more than it can handle, causing it to break down. It is effective in readying the system for unexpected circumstances and determining exactly how far a given system can be pushed.

In our project, we performed load testing by changing the number of AskUbuntu questions for analysis and the final visualization. We started with a relative small bunch of 300 questions, then gradually moving on to a 1000 questions, then 3000 questions and ultimately finishing off with all of the AskUbuntu questions available. Each one of these data sets were put through RStudio for analysis. The similarity matrix was then generated, the data was then cleaned using stemming and the term frequency was calculated which was then used to determine the cosine distance. Each of those was then exported to CSV format to be imported in Gephi for generating the graphs.

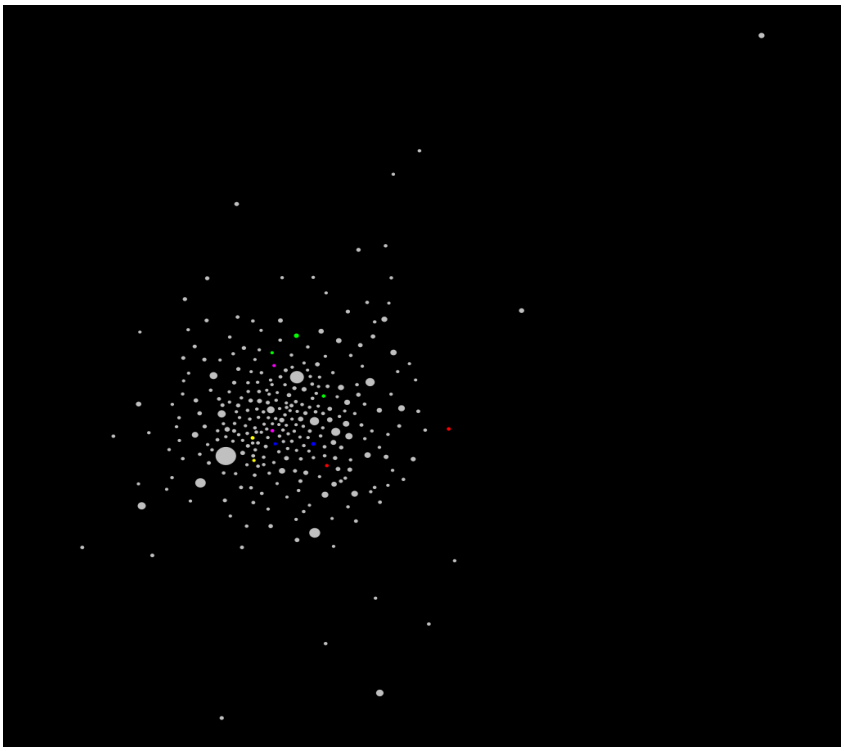


Figure: Cluster of 300 questions

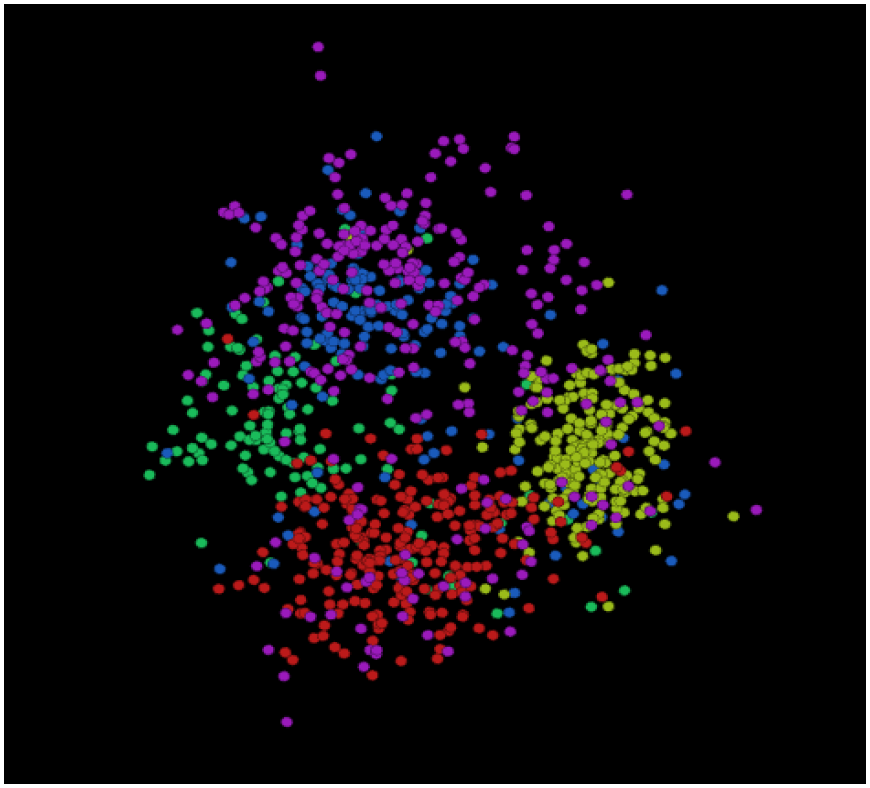


Figure: Cluster of 1000 questions

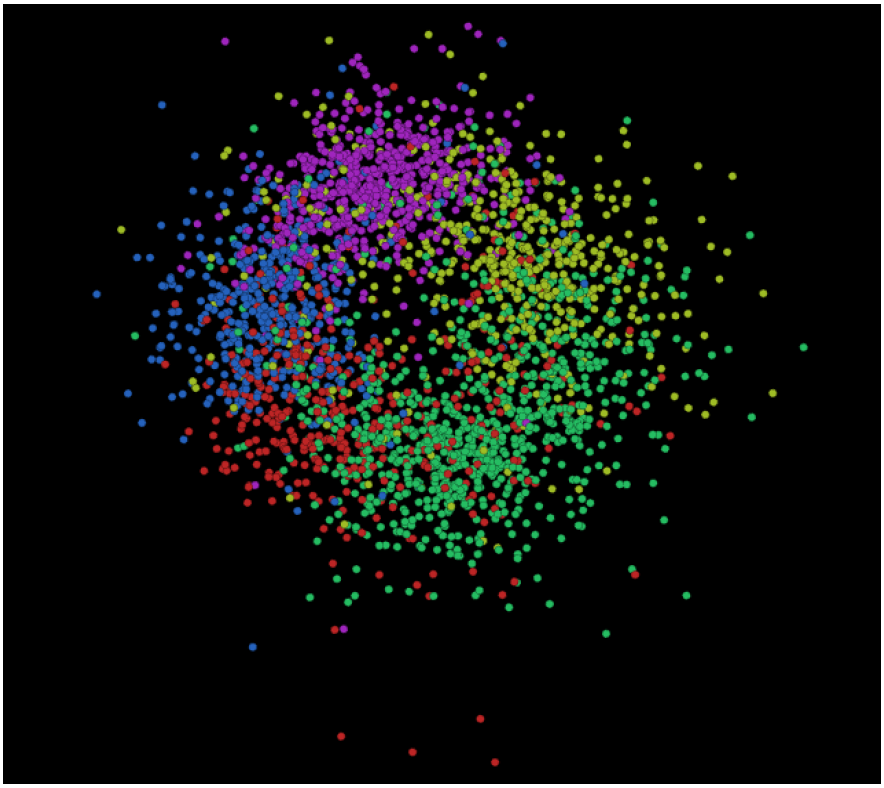


Figure: Cluster of 3000 questions

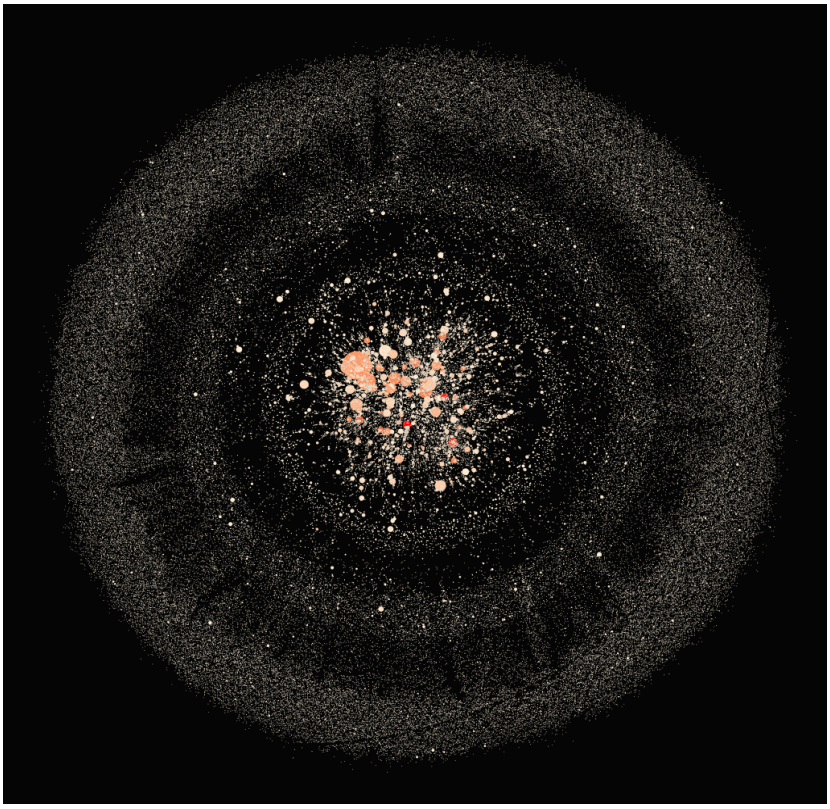
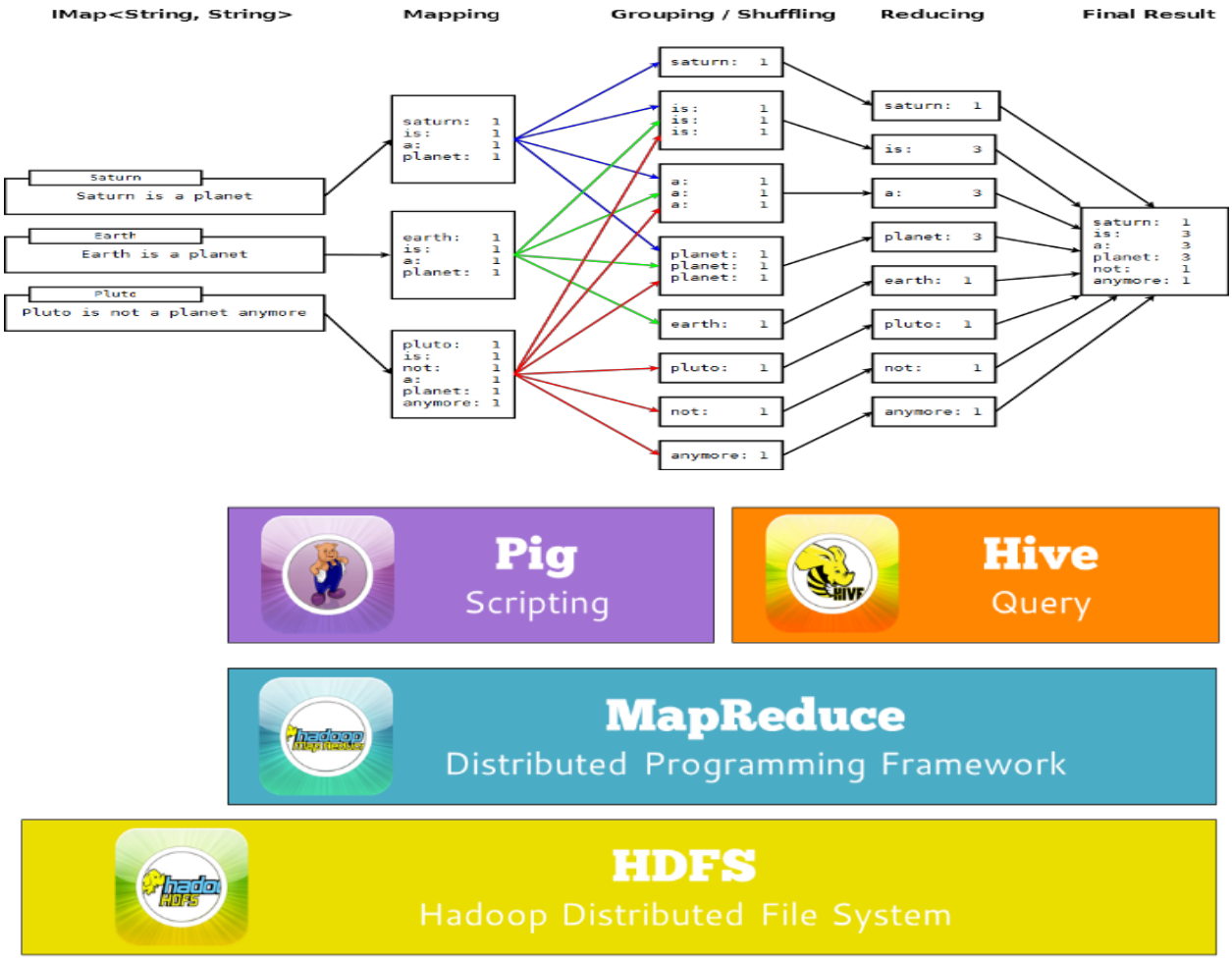


Figure: Cluster of all questions

Design Pattern Used:

MAPREDUCE

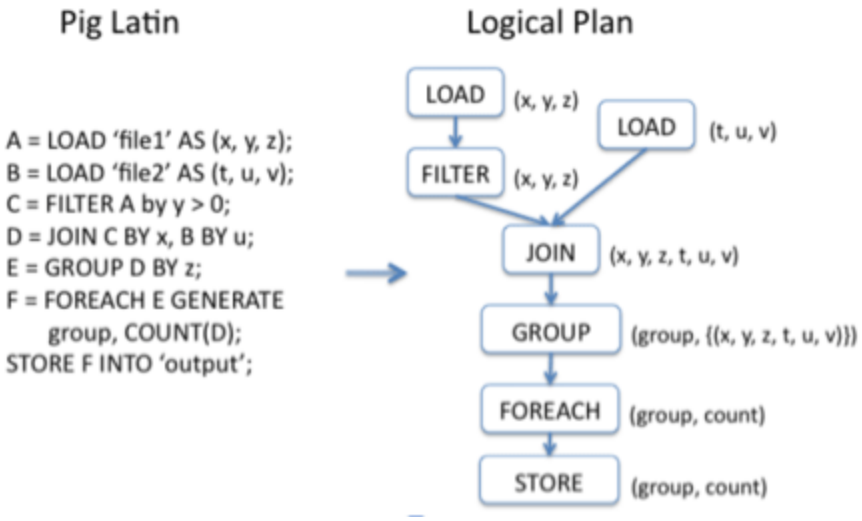
MapReduce is a model in programming used for processing and generating a large set of data by computing it on a parallel or a distributed system. The Map and Reduce function are defined in a data structure format such as (key, value).
Mapper function takes input a key and value for eg. Map (k1, v1). This mapper function generates an output (k2, v2). This output is then passed to the Reducer function. The Reducer function eg. Reduce (k2, v2) then generates an output v3.



The above figure shows how pig and hive are used in MapReduce.

PIG

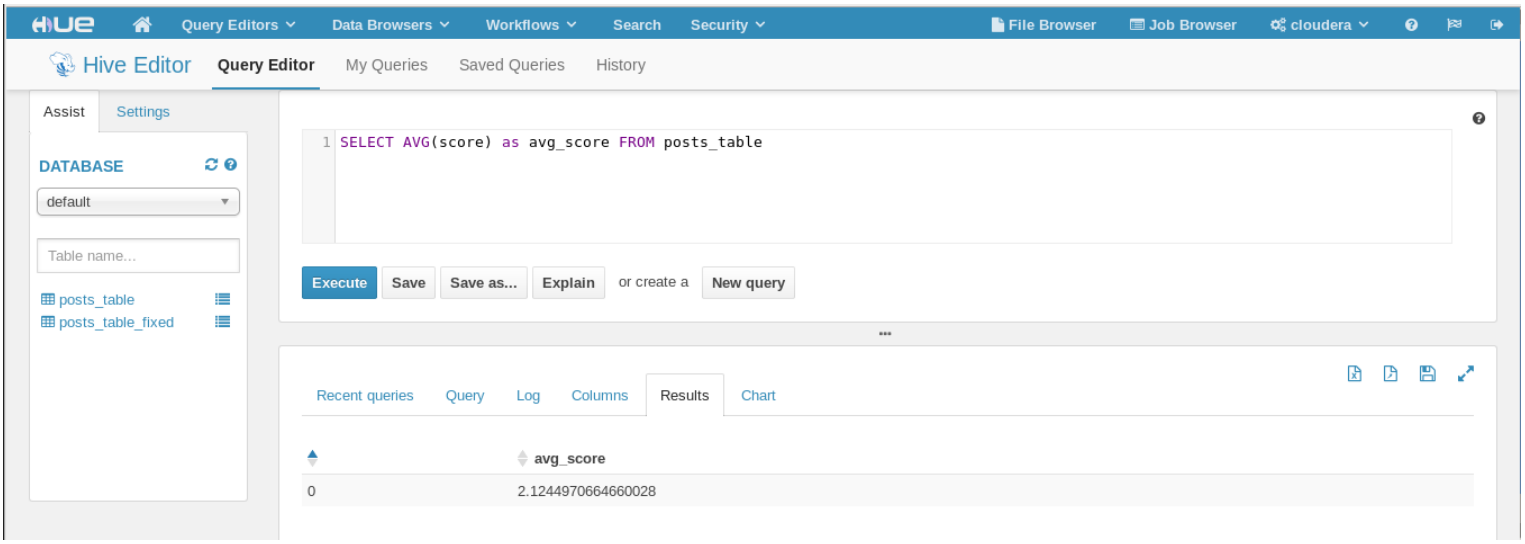
Pig is a high level platform used for large scale analysis of data. Pig uses a language known as Pig Latin. Pig is usually used for data analysis program which are related with high level of evaluation. The most important property of Pig is to enable parallelization of computing which is main reason it is used in problems involving large sets of data in them. Pig has an inbuilt compiler which already has a number of programs in it which include parallel implementation of the functions.
The scripting for the data is done using Pig.



Hive

Hive data warehouse provides facilities to query and manage large scale data that is residing across a distributed storage. It is built on top of Hadoop and is used to run queries that help in reduce function. The queries used in hive are very similar to SQL queries but they are just called as QL. Hive allows users who are familiar with the Map Reduce function to process their analysis that may not be supported by normal language or may require lot of computational power. Hive is mainly used for jobs that contain large set of data like logs, records etc. Hive provides various features like

1. Provides tools for performing operations on data like extraction, transformation etc.
2. Allows user to impose a structure regardless of the format of the data.
3. The data can be accessed directly by Apache HDFS or It is in other types of storage systems like Apache HBase.
4. The main function it does is executes a given queries for mapreduce function.



Individual Contribution

Ameya Manjrekar

Logistic Decision
Dataset Analysis
UI Design - Node.js
Rigid Testing
Data Mining Algorithms - Vector Spcae Model

Chao Nie

Logistic Decision
Analysis on R
Usage of RStudio - methods for TF-IDF cosine similarity
Visualization - Sigma.js
Sigma visual effect plugin and method implementation - fisheye plugin, question link to AskUbuntu.com
Beta Deployment - iPage web server
Dataset Analysis

Rajas Hegiste

Logistic Decision
UI Design - Node.js
Data Mining Algorithms - Vector Space Model.
Design Patterns
Analysis of ForceAtlas2 and OpenOrd Algorithm.
Gephi Setup instructions

Yaopeng Wu

Logistic Decision
Data Flow and Architecture Decision
Data Retrieval
Data Cleaning - Bash and Pig
Data Aggregation - Hive and R
Statistics and Discovery - Gephi
Visualization - Sigma.js
Deployment - Node.js and Heroku