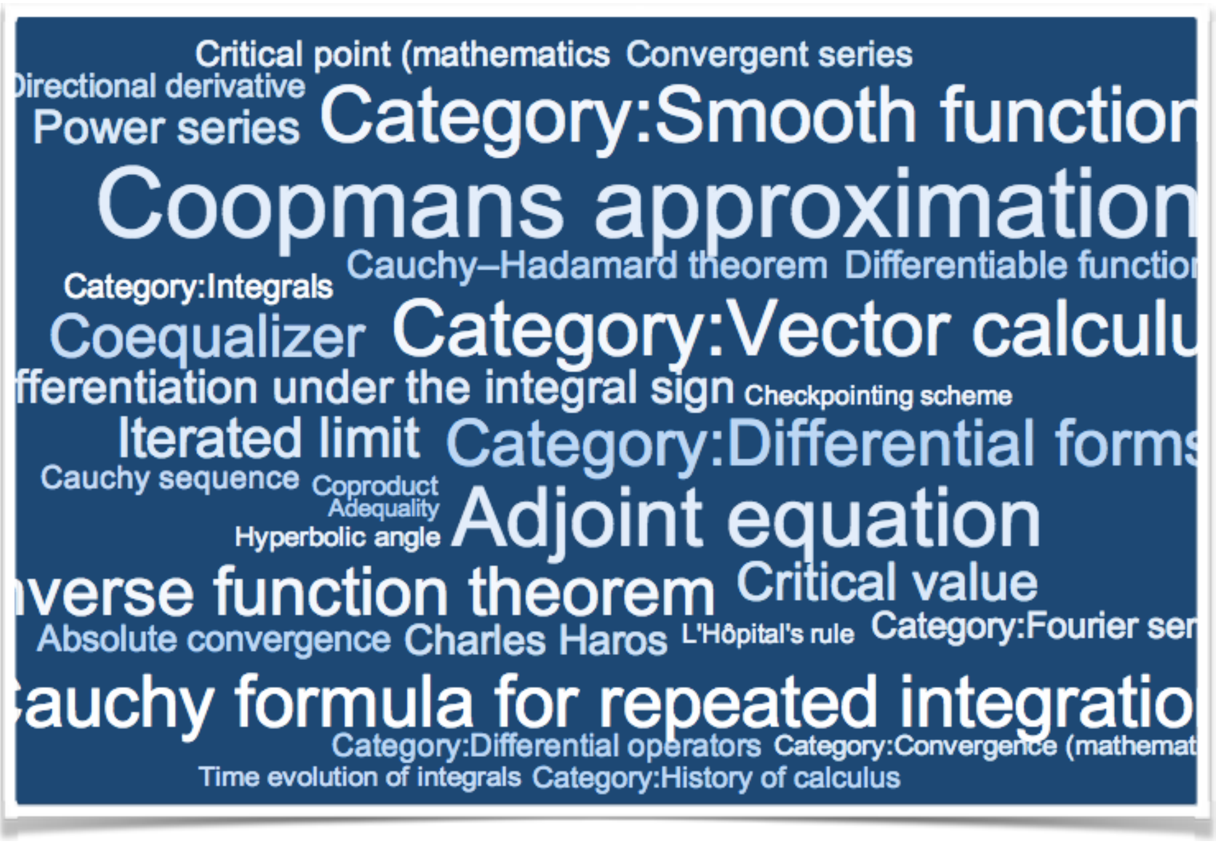
Final Report

Visualisation of knowledge development in wiki



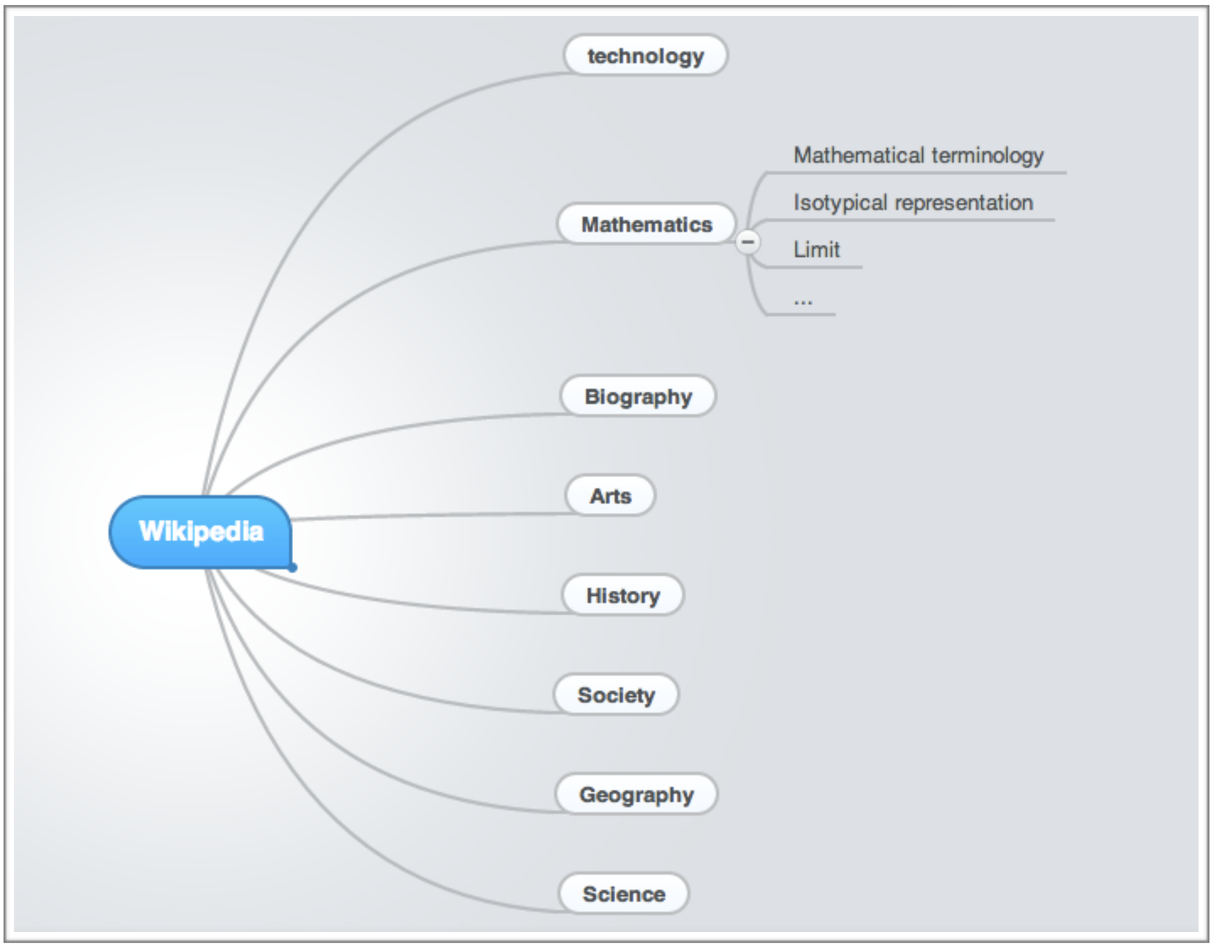
Group members

Ni Fang	2013854450
Zhang Ningyi	2013952484
Liu Liying	2013950254
Fan Neng	2013951375

Visualisation of knowledge development in wiki

1. The Idea Of Wiki Data Visualisation

We get all kinds of information from the internet everyday. The organisation of the content on the internet is a interest topic to be considered. For example, the wikipedia is a public knowledge website that contains millions of terms, and these terms are connected. The structure of the knowledge in Wiki is like a tree or a forest. Specifically, if the category of math is a tree, the sub category is the branch of the tree, and the term "limit" is one of the leaf node of the tree.



However, all these terms and there relationships is created by the users, which means that by analysing the architecture of wiki, we can figure out some knowledge management patterns. The patterns may tell us some interesting things about wiki and even the whole internet.

2. Project Target And Division Of Work

In this project, we are going to analysis the historic edition of the article in Wikipedia. Wikipedia is a public knowledge platform, where 30 million articles in 287 languages are written collaboratively by volunteers around the world. Every article in Wikipedia experiences the processing from fragmentary to complete and related article grows from discrete to integrate.

Thus the historic edition of the article is the record of the development of the maturity of knowledge. Taking the links between different articles into consideration, they also indicate the formation record of a knowledge system.

By visualizing the historic edition of the articles in a certain area, we can observe how a knowledge area is generated and how the articles in the knowledge are linked to others.

Our project aimed at visualizing the process of building knowledge system. We want to display the formation of the knowledge. We choose the knowledge area “limit” as our theme and intend to demonstrate the development in this area from 2001 to 2013.

We decide to visualize the articles as nodes and the links between articles as edges. Users may click on each node to get detail information. As the articles and links change with time, the nodes and edges will change correspondingly. At first, the knowledge is incomplete with several discrete articles. As time goes on, editors add details and link to the articles. The knowledge system develops and finally the articles become interconnected.

As is mentioned in the proposal, there are several tasks to be finished in our project:

1) Find a sub knowledge system to analysis. The whole Wikipedia is too complex for us to analysis. We need to find a featured area to get a better outcome.
2) Get the data of the sub system. The data downloaded from the Wikipedia contains all the terms of the Wikipedia system.
3) Add time label to the data.
4) Preprocessing of the data to make the data acceptable to the software.

5) Try different visualization method and choose the suitable visualization method.
6) Analyse the outcomes and evaluate.

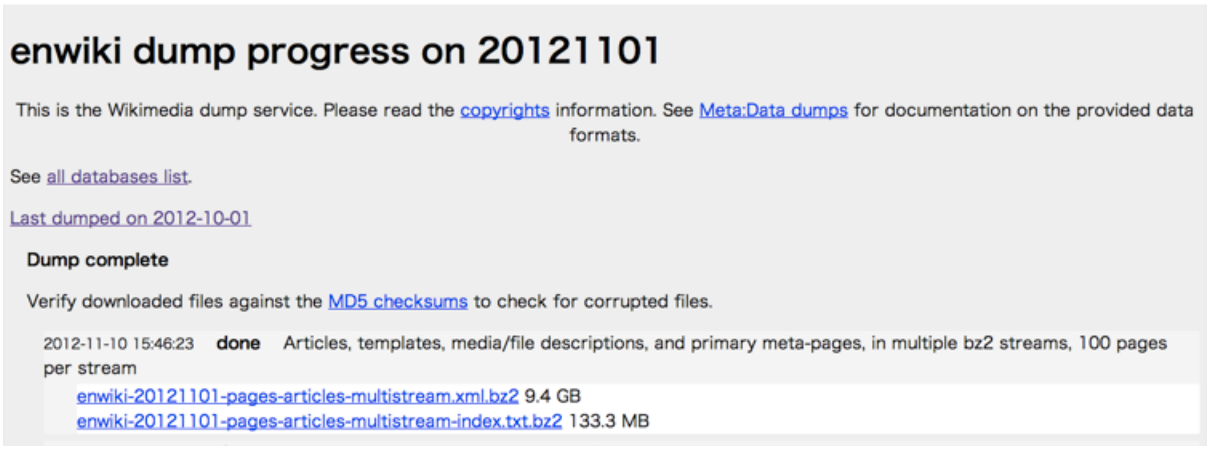
We divided our work into four parts and the division of our work is as follow:
1) Data preprocessing: Ni Fang, Fan Neng and Liu Liying
2) Trying visualization method: Zhang Ningyi and Ni Fang and Fan Neng
3) Evaluation: All
4) Documentary: Liu Liying and Zhang Ningyi

3. Preparations

1) Access to data

Wikipedia offers free copies of all available content to interested users. All text content is multi-licensed under theCreative Commons Attribution-ShareAlike 3.0 License (CC-BY-SA) and the GNU Free Documentation License (GFDL). In past years, Wikipedia used to provide SQL dump for pages, revision history and text. However, in the middle of 2005, they upgraded the Wikipedia sites, which use a very different database layout than earlier versions. Changes to the backend storage are aggressive. As a result, Wikipedia provides XML dump of article histories for forward and backward compatibility without requiring authors of third-party dump processing or statistics tools to reproduce every internal hack.

Although the archived dump is impressive, it is not ideal for use in this research .It can be seen that the archive of current pages including discussion and user pages is is so huge, with up to more than 9 Gigabytes in compressed data . Even the meta-data of all revisions(no page text) reaches 33.3Gigabytes. According to Wikipedia, the size of uncompressing archive could become up to 100 times. The limitation of the data storage makes it quite hard to hold such large volume of data.



enwiki dump progress on 20121101

This is the Wikimedia dump service. Please read the copyrights information. See Meta:Data dumps for documentation on the provided data formats.

See all databases list.

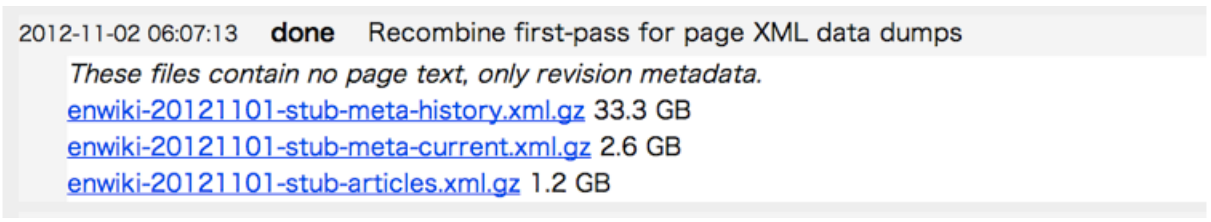
Last dumped on 2012-10-01

Dump complete

Verify downloaded files against the MD5 checksums to check for corrupted files.

2012-11-10 15:46:23 **done** Articles, templates, media/file descriptions, and primary meta-pages, in multiple bz2 streams, 100 pages per stream

enwiki-20121101-pages-articles-multistream.xml.bz2 9.4 GB
enwiki-20121101-pages-articles-multistream-index.txt.bz2 133.3 MB

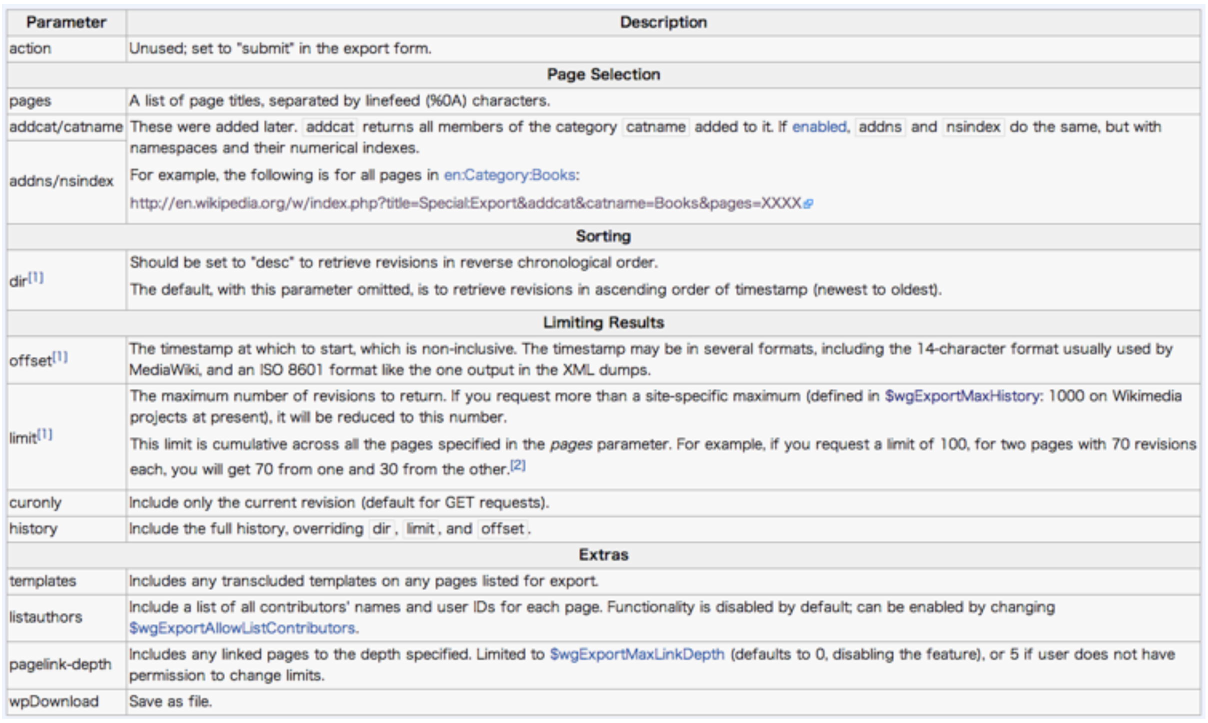


2012-11-02 06:07:13 **done** Recombine first-pass for page XML data dumps

These files contain no page text, only revision metadata.
enwiki-20121101-stub-meta-history.xml.gz 33.3 GB
enwiki-20121101-stub-meta-current.xml.gz 2.6 GB
enwiki-20121101-stub-articles.xml.gz 1.2 GB

One way to download a single article dump from Wikipedia is by issuing a HTTP POST request. For example, one can export the oldest 100 history revisions by issuing the following HTTP POST request:
http://en.wikipedia.org/w/index.php?title=Special:Export&addcat&catname=Limits_(category_theory)

Some key parameters for the exporting are shown in following picture.

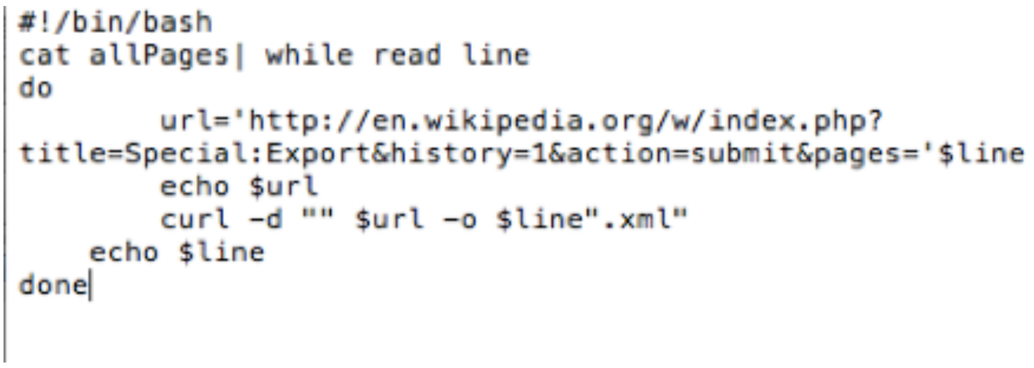


Parameter	Description
action	Unused; set to "submit" in the export form.
Page Selection	
pages	A list of page titles, separated by linefeed (%0A) characters.
addcat/catname addns/nsindex	These were added later. addcat returns all members of the category catname added to it. If enabled, addns and nsindex do the same, but with namespaces and their numerical indexes. For example, the following is for all pages in en:Category:Books: http://en.wikipedia.org/w/index.php?title=Special:Export&addcat&catname=Books&pages=XXXX
Sorting	
dir[1]	Should be set to "desc" to retrieve revisions in reverse chronological order. The default, with this parameter omitted, is to retrieve revisions in ascending order of timestamp (newest to oldest).
Limiting Results	
offset[1]	The timestamp at which to start, which is non-inclusive. The timestamp may be in several formats, including the 14-character format usually used by MediaWiki, and an ISO 8601 format like the one output in the XML dumps.
limit[1]	The maximum number of revisions to return. If you request more than a site-specific maximum (defined in $wgExportMaxHistory: 1000 on Wikimedia projects at present), it will be reduced to this number. This limit is cumulative across all the pages specified in the *pages* parameter. For example, if you request a limit of 100, for two pages with 70 revisions each, you will get 70 from one and 30 from the other.[2]
curonly	Include only the current revision (default for GET requests).
history	Include the full history, overriding dir, limit, and offset.
Extras	
templates	Includes any transcluded templates on any pages listed for export.
listauthors	Include a list of all contributors' names and user IDs for each page. Functionality is disabled by default; can be enabled by changing $wgExportAllowListContributors.
pagelink-depth	Includes any linked pages to the depth specified. Limited to $wgExportMaxLinkDepth (defaults to 0, disabling the feature), or 5 if user does not have permission to change limits.
wpDownload	Save as file.

In order to automatically retrieve dump from Wikipedia website, cURL is selected for crawling revisions from Wikipedia. cURL gives great simplicity to grab data from Wikipedia. For example, one can download the dump for Knowledge Management by issuing the cURL command:

- curl -d "&pages=Main_Page&offset=1&action=submit" http://en.wikipedia.org/w/index.php?title=Special:Export -o "somefilename.txt"

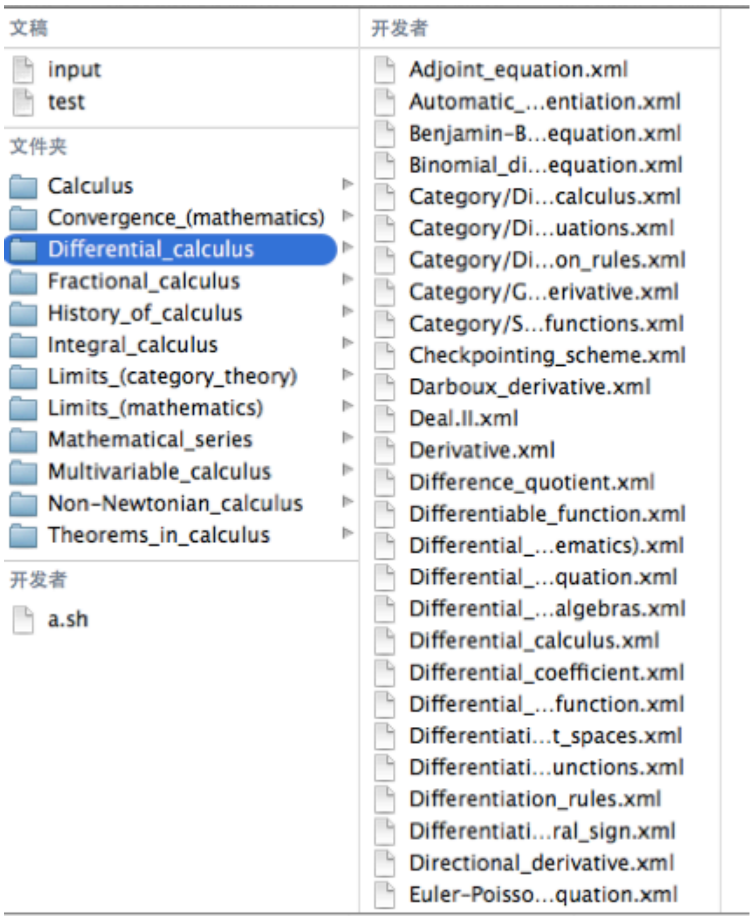
So we wrote the bash program to get data in using cURL. All pages 'title was in the file named "allPages".



```
#!/bin/bash
cat allPages| while read line
do
        url='http://en.wikipedia.org/w/index.php?
title=Special:Export&history=1&action=submit&pages='$line
        echo $url
        curl -d "" $url -o $line".xml"
    echo $line
done
```

In term of big data of all pages, we just use the pages belonging to the category of Calculus for research.

The result as following.


```
文稿                                开发者
  input                               Adjoint_equation.xml
  test                                Automatic_...entiation.xml
文件夹                                Benjamin-B...equation.xml
  Calculus                            Binomial_di...equation.xml
  Convergence_(mathematics)           Category/Di...calculus.xml
  Differential_calculus               Category/Di...uations.xml
  Fractional_calculus                 Category/Di...on_rules.xml
  History_of_calculus                 Category/G...erivative.xml
  Integral_calculus                   Category/S...functions.xml
  Limits_(category_theory)            Checkpointing_scheme.xml
  Limits_(mathematics)                Darboux_derivative.xml
  Mathematical_series                 Deal.II.xml
  Multivariable_calculus              Derivative.xml
  Non-Newtonian_calculus              Difference_quotient.xml
  Theorems_in_calculus                Differentiable_function.xml
开发者                                Differential_...ematics).xml
  a.sh                                Differential_...quation.xml
                                      Differential_...algebras.xml
                                      Differential_calculus.xml
                                      Differential_coefficient.xml
                                      Differential_...function.xml
                                      Differentiati...t_spaces.xml
                                      Differentiati...unctions.xml
                                      Differentiation_rules.xml
                                      Differentiati...ral_sign.xml
                                      Directional_derivative.xml
                                      Euler-Poisso...quation.xml
```

2) Data Processing

The data parsing stage consists of two steps: XML parsing and wiki notation parsing. The XML dump contains multiple revisions of an article. The first step is to retrieve various data from the XML dump. This includes the timestamp, the contributor, the comments as well as the Wikipedia notation for revisions. The second step is to parse the wiki notation and retrieve sections and hyperlinks from each revision. In the following sections, the two steps will be discussed in detail.

a. XML Parsing

Once the XML dump is crawled, it can be processed by any XML parser. There are two typical ways to process XML data, SAX and DOM.

SAX stands for Simple API for XML (Harold 2002). It is an event based parser that invokes methods when mark-up, such as a start tag or an end tag, is encountered. No tree structure is created - data is passed to the application from the XML document as it is found. SAX parsers are typically used for reading XML documents that will not be modified. SAX-based parsers are available for a variety of programming languages.

The XML DOM (XML Document Object Model) defines a standard way for accessing and manipulating XML documents (Harold 2002). The DOM views XML documents as a tree-structure. All elements can be accessed through the DOM tree. Their content (text and attributes) can be modified or deleted, and new elements can be created. The elements, their text, and their attributes are all known as nodes. DOM parsers are typically used for manipulating and transforming XML documents.

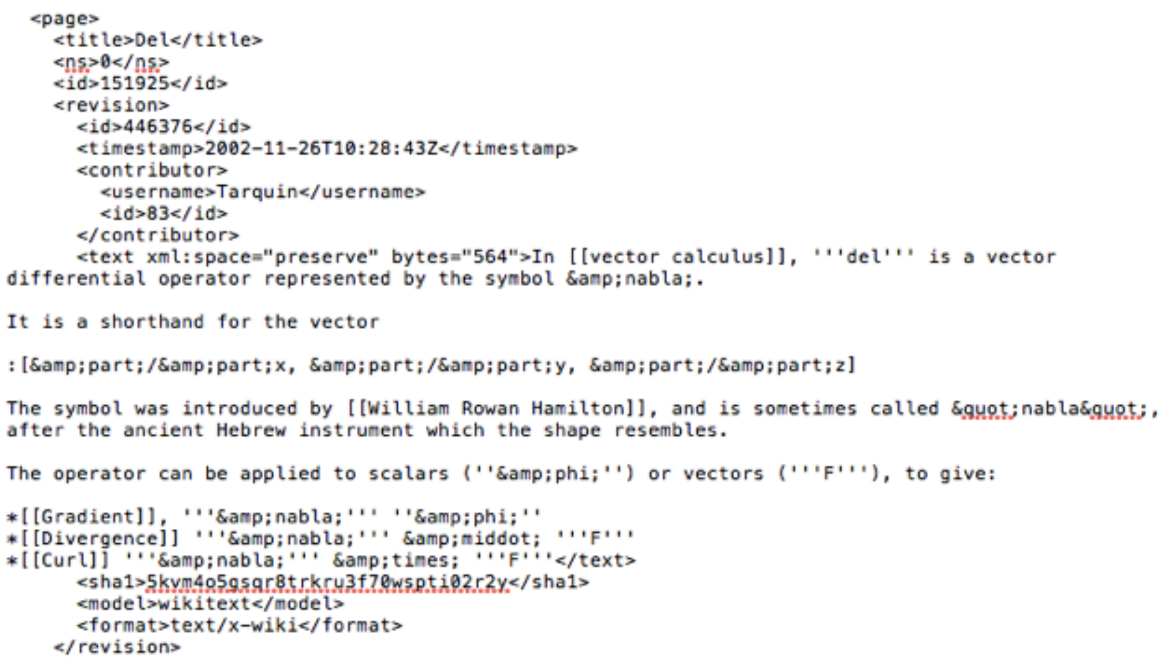
In this research, the SAX parser is selected to process XML data as it has two advantages over DOM parser:

• XML as data source. The data in XML format will be loaded into local database for further processing. It is mainly for read only purpose rather than manipulation. SAX is a perfect candidate to do this job.

• Less resource consumption. The DOM reads all the data and builds an internal tree representation in memory; it consumes much more resource than SAX parser. The returned XML dump can potentially be very large. While SAX only reads a small portion of data at one time, it consumes less resource.

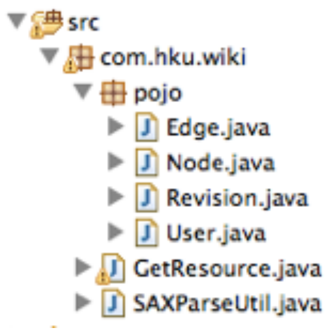
b. wiki notation parsing

Look back the xml dump:



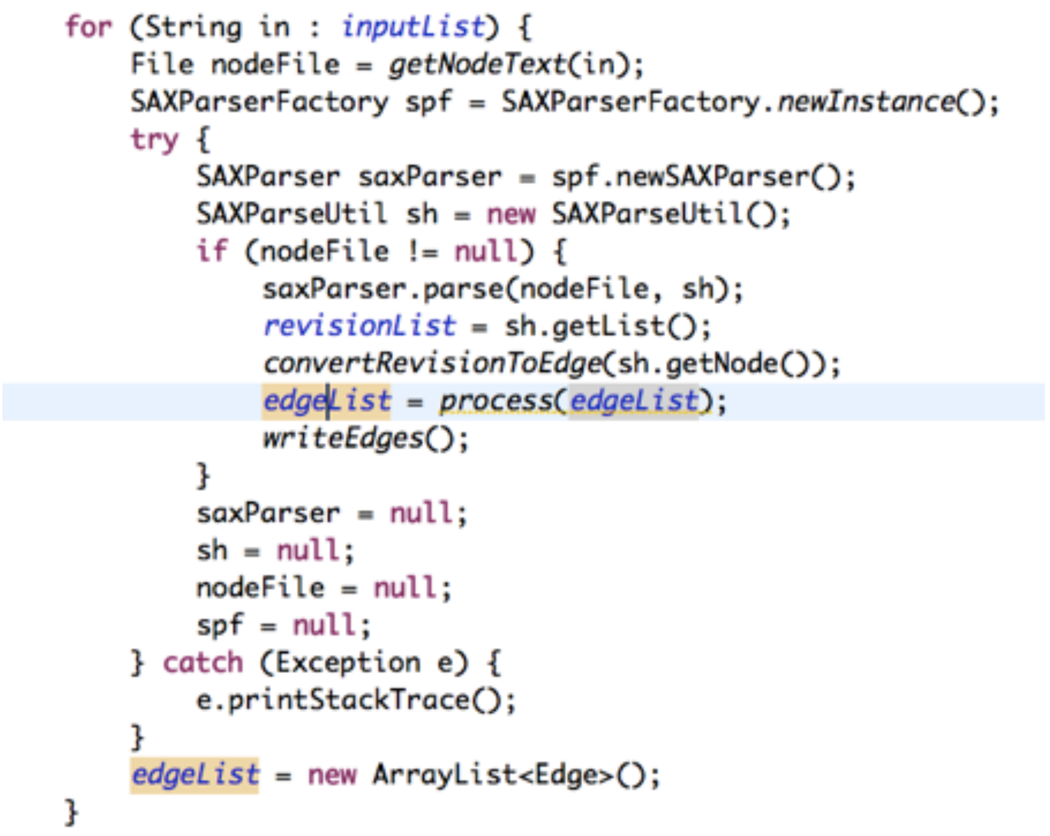
```
  <page>
    <title>Del</title>
    <ns>0</ns>
    <id>151925</id>
    <revision>
      <id>446376</id>
      <timestamp>2002-11-26T10:28:43Z</timestamp>
      <contributor>
        <username>Tarquin</username>
        <id>83</id>
      </contributor>
      <text xml:space="preserve" bytes="564">In [[vector calculus]], '''del''' is a vector
differential operator represented by the symbol &amp;nabla;.

It is a shorthand for the vector

:[&amp;part;/&amp;part;x, &amp;part;/&amp;part;y, &amp;part;/&amp;part;z]

The symbol was introduced by [[William Rowan Hamilton]], and is sometimes called &quot;nabla&quot;,
after the ancient Hebrew instrument which the shape resembles.

The operator can be applied to scalars (''&amp;phi;'') or vectors ('''F'''), to give:

*[[Gradient]], '''&amp;nabla;''' ''&amp;phi;''
*[[Divergence]] '''&amp;nabla;''' &amp;middot; '''F'''
*[[Curl]] '''&amp;nabla;''' &amp;times; '''F'''</text>
      <sha1>5kvm4o5gsqr8trkru3f70wspti02r2y</sha1>
      <model>wikitext</model>
      <format>text/x-wiki</format>
    </revision>
```

This is the program structure.



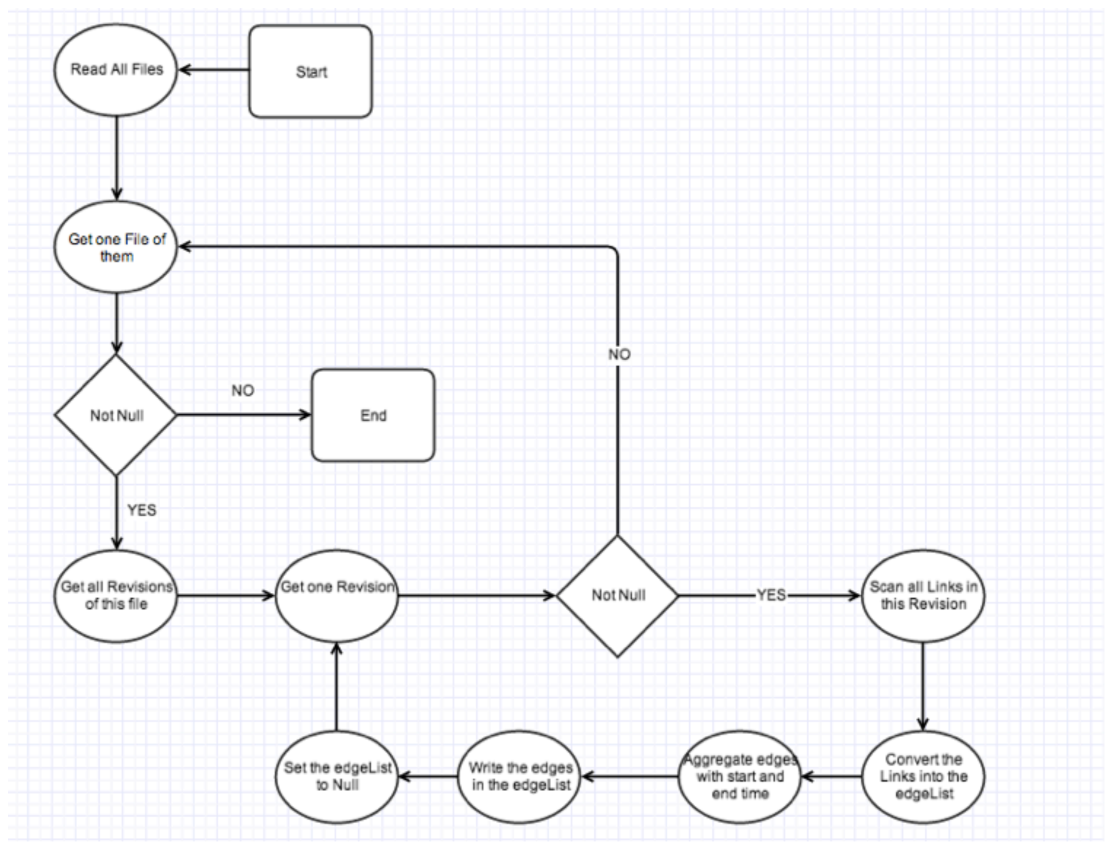
```
▼ src
  ▼ com.hku.wiki
    ▼ pojo
      ▶ Edge.java
      ▶ Node.java
      ▶ Revision.java
      ▶ User.java
    ▶ GetResource.java
    ▶ SAXParseUtil.java
```

Some attributes can be parsed and save into Pojos(Edge.java, Node.java, Revision.java, User.java).
SAXParseUtil.java was the Utilize class that was used to handle the xml to the Pojos.
GetResource.java contains the main function.



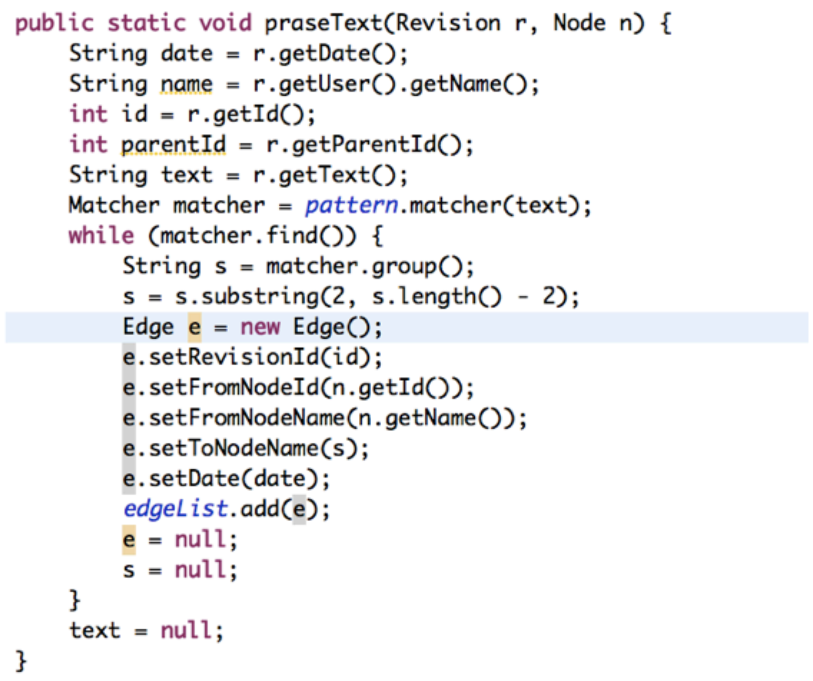
```
for (String in : inputList) {
    File nodeFile = getNodeText(in);
    SAXParserFactory spf = SAXParserFactory.newInstance();
    try {
        SAXParser saxParser = spf.newSAXParser();
        SAXParseUtil sh = new SAXParseUtil();
        if (nodeFile != null) {
            saxParser.parse(nodeFile, sh);
            revisionList = sh.getList();
            convertRevisionToEdge(sh.getNode());
            edgeList = process(edgeList);
            writeEdges();
        }
        saxParser = null;
        sh = null;
        nodeFile = null;
        spf = null;
    } catch (Exception e) {
        e.printStackTrace();
    }
    edgeList = new ArrayList<Edge>();
}
```

As the requirement, Two files contained nodes and edges respectively should be produced as the basic data source.


Start
Read All Files
Get one File of them
Not Null
NO
End
YES
Get all Revisions of this file
Get one Revision
Not Null
NO
YES
Scan all Links in this Revision
Convert the Links into the edgeList
Aggregate edges with start and end time
Write the edges in the edgeList
Set the edgeList to Null

The apart how to create Edges is more complex so that I attach the code and paint a flow chart to explain that.

Following is the key codes. The edge in our program means the link. So I must



```
public static void praseText(Revision r, Node n) {
    String date = r.getDate();
    String name = r.getUser().getName();
    int id = r.getId();
    int parentId = r.getParentId();
    String text = r.getText();
    Matcher matcher = pattern.matcher(text);
    while (matcher.find()) {
        String s = matcher.group();
        s = s.substring(2, s.length() - 2);
        Edge e = new Edge();
        e.setRevisionId(id);
        e.setFromNodeId(n.getId());
        e.setFromNodeName(n.getName());
        e.setToNodeName(s);
        e.setDate(date);
        edgeList.add(e);
        e = null;
        s = null;
    }
    text = null;
}
```

scan all links in the context using the Regular Expression.

4. Difficulties Encountered

1) Difficulty in data accessing and processing

i. Big Data and Low Memory
ii. We have to retrieve dump manually. It takes time.

2) Difficulty in data visualization

i. The dynamic effect of Gephi.
ii. Choose the proper layout algorithm and plug-in.
iii. Combine the different visualization together.

5. Visualization Techniques Used

1) Gephi

We use Gephi's basic function: net work display. By changing the format of the raw data, we enable our data table to be recognised by Gephi. We use nodes to represent the article in Wiki and edges to represent the link between articles. Layout algorithm is used to adjust the distribution of the points. Through this adjusting ,more information ,such as the relationship or clusters among the nodes will be exposed.

As we want to show the growth of the knowledge system, the ' merge column' and 'create time interval' function is used to create dynamic graph. These two functions transfer our time data from 'start' and 'end' into time interval and enable the graph to display the data in years.

The 'give color to nodes' plug-in is used to color nodes with different categories attribute corresponding colors. Therefore the nodes categories can be distinguished easily and more information will be showed.

2) Tableau

Tableau is used to analysis the nodes and edges separately, and display the data in a static way.

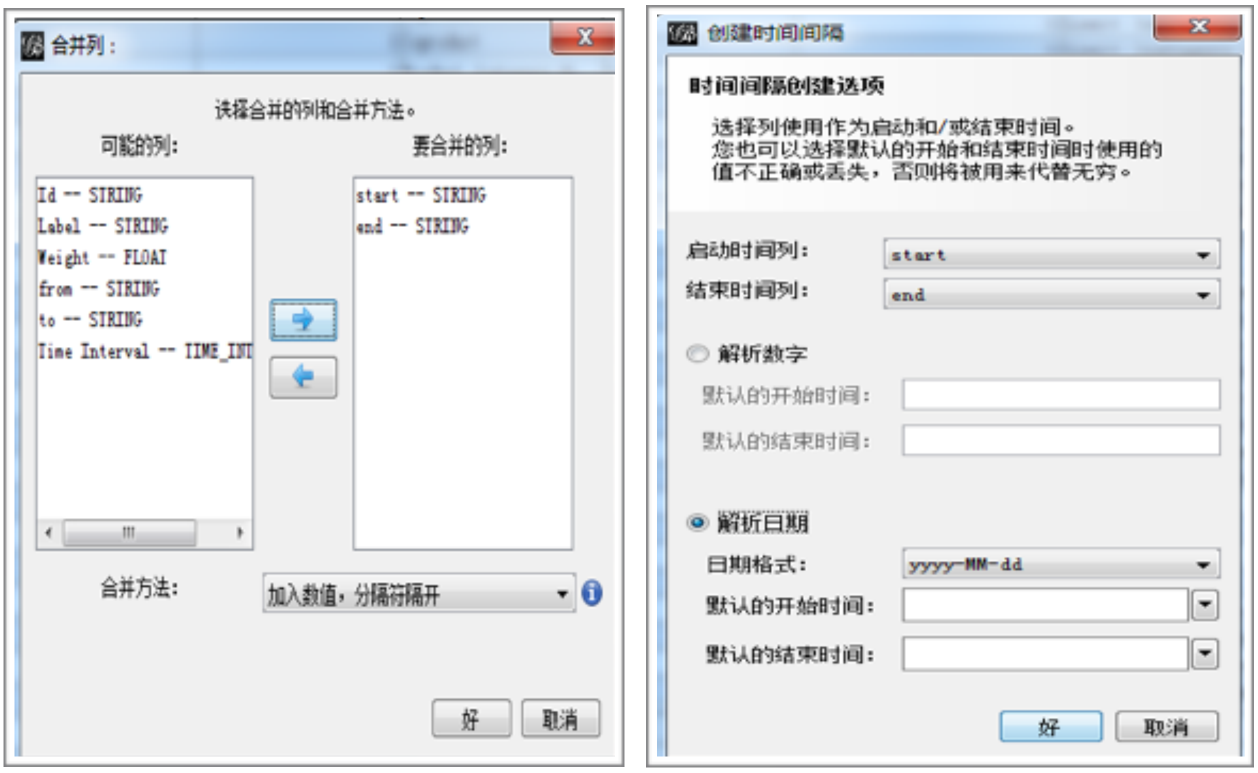
Instead of exploring the interrelation between nodes and edges, we can find certain regulations and patterns from the visualizations in Tableau with the help of different sorts of layout like bars, lines and packed bubbles.
After generate the charts, we save it to the web with the registered account for further use.

As shown before, in Tableau charts we use different color to represent the class attribute of articles. We also use the filter tableau offers to provide easy and simple interact function.

6. The Iterative Processing Of Visualization

The data visualization is dividend into two parts. The first is the visualization of the dynamic growth of articles and the links between them. The other part is the visualization of analytical data extracted from the raw data.

Gephi is used to visualize the dynamic growth process of the articles and links in the area “mathematics”. To demonstrate the dynamic process, time information is added into data table. There is a “date” attribute in the origin nodes data table indicate the time of occurrence of the article. In the edge table we use “start” and “end” attributes to mark the life cycle of each link. We use “merge column” function to create time interval of nodes and edges. In nodes table, we set the date attribute as start time and set the end date as unlimited. In edge table, just set the start time as start and end time as end.



合并列：

选择合并的列和合并方法。

可能的列:
- Id -- STRING
- Label -- STRING
- Weight -- FLOAT
- from -- STRING
- to -- STRING
- Time Interval -- TIME_INT

要合并的列:
- start -- STRING
- end -- STRING

合并方法: 加入数值，分隔符隔开

创建时间间隔

时间间隔创建选项

选择列使用作为启动和/或结束时间。
您也可以选择默认的开始和结束时间时使用的值不正确或丢失，否则将被用来代替无穷。

启动时间列: start
结束时间列: end

[ ] 解析数字
默认的开始时间:
默认的结束时间:

[✓ 解析日期]
日期格式: yyyy-MM-dd
默认的开始时间:
默认的结束时间:

Figure 1 Merge columns to create time interval

We tried several layout algorithms to adjust the layout of the nodes and edges. Some the former version of our visualization is as follow:

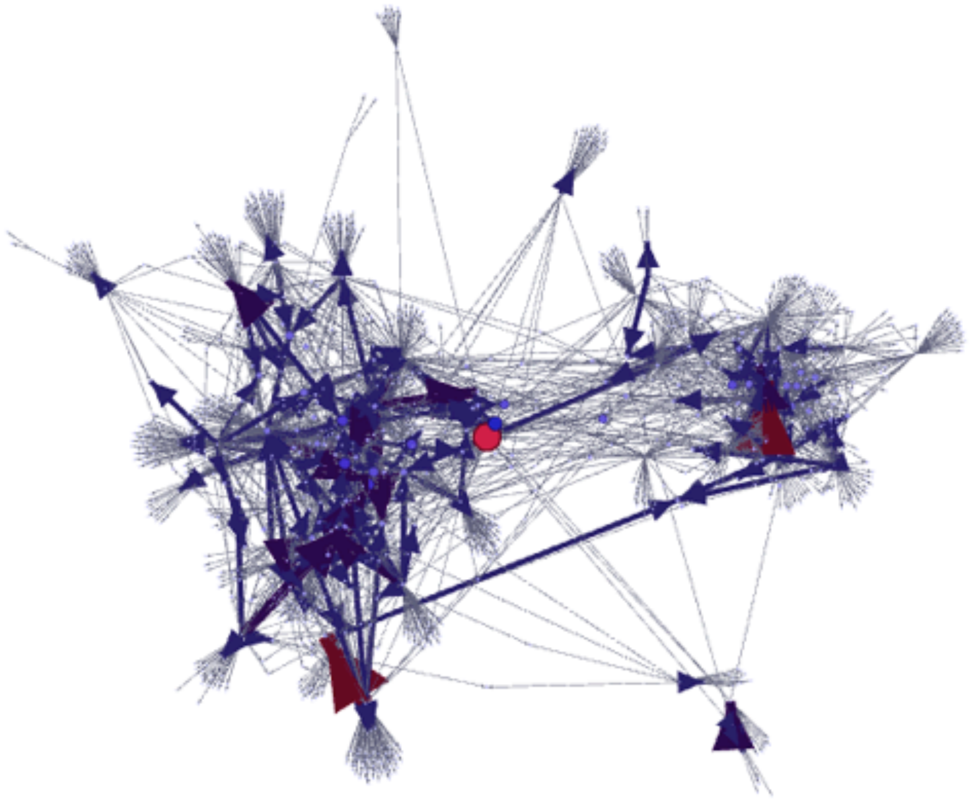


Figure 2 Former Version 1

The former version 1 is a static version. The category of data is very clear in the graph and it seems that the information it displays is quite enough. But our goal is to create a visualization which can show the growth process. From this perspective, version 1 is far from enough.

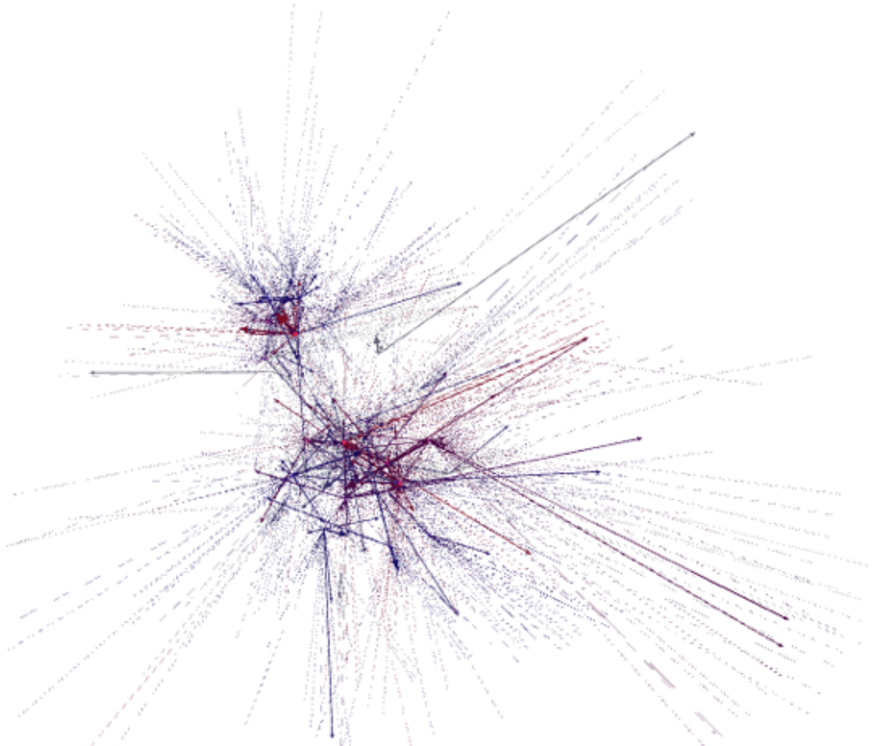


Figure 3   Former Version 2

Former version 2 is our first dynamic version. We can distinguish two main classes in the graph but the whole graph seems disordered and little information can be extracted from the graph.

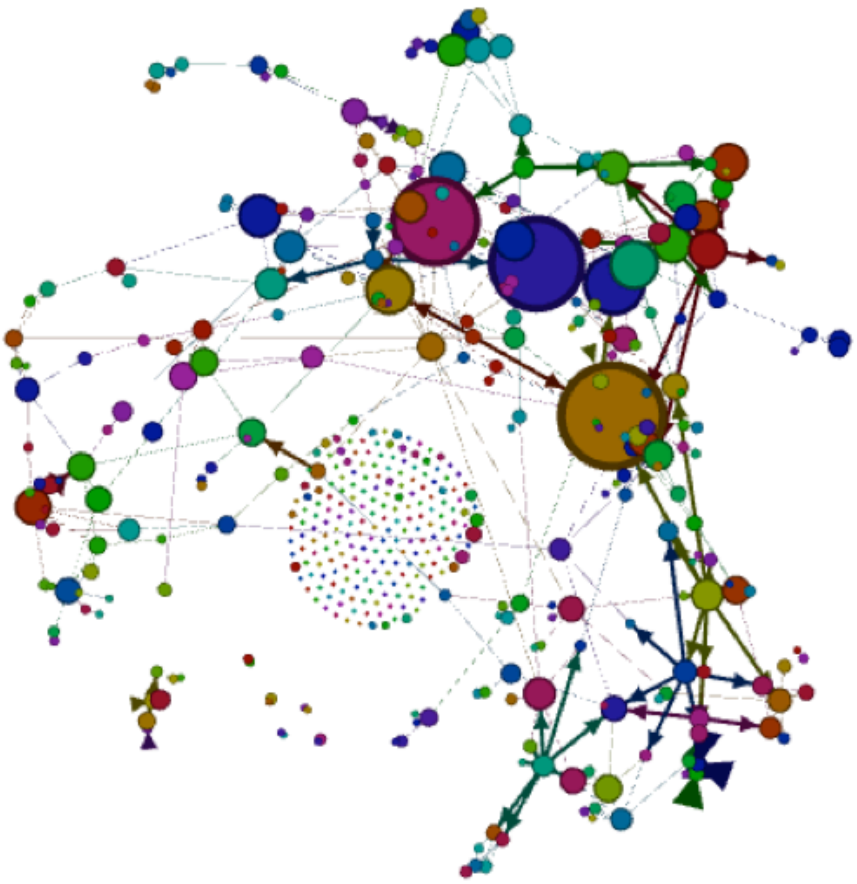
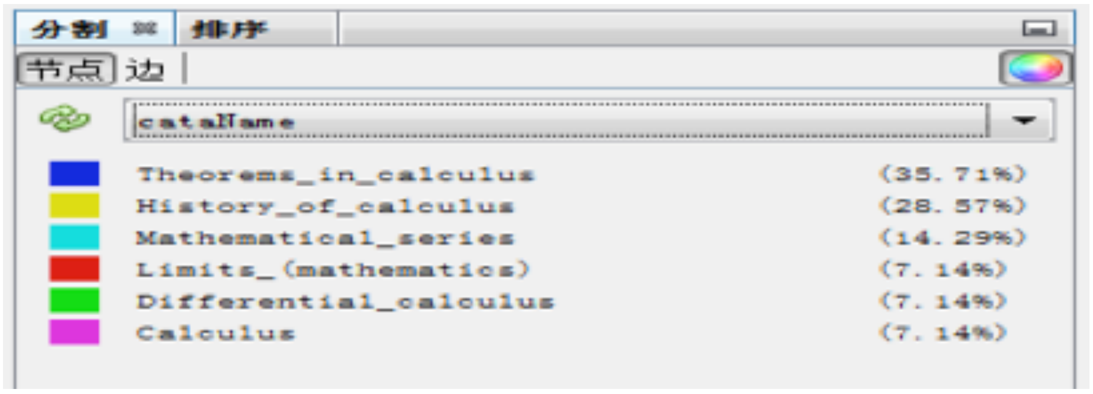


Figure 4 Former Version 3

Former version 3 is an improved dynamic version. The nodes are colored based on the occurrence time of the article the node stands for. Through discussion we evaluate this version as acceptable but not perfect. The color of the nodes indicates the time attribute of the article and the node's appearance in dynamic graph indicate the same attribute. Therefore there is redundancy in the graph. Through comparative analysis, we improved our graph several times and complete the final version. In the final version Force Atlas layout is used as it may distinguish the data into several clusters and form a neat graph with quite enough information. We also split the data with split algorithm and the parameter we used is cataName which stands for the class label. The nodes are colored into different colors correspond to their category to avoid the redundancy in the former version.



🟦	Theorems_in_calculus	(35.71%)
🟨	History_of_calculus	(28.57%)
🟦	Mathematical_series	(14.29%)
🟥	Limits_(mathematics)	(7.14%)
🟩	Differential_calculus	(7.14%)
🟪	Calculus	(7.14%)

Figure5 Split Algorithm

After tries and tests, we chose two algorithm to organise our data. The first one is Force Atlas, and the second one is Fruchterman Reingold.

Using Force Atlas, we can see the inter relationship of these nodes or terms. The final version of our Gephi graph is shown below. As it is a dynamic graph, we just put the first and the last frame here.

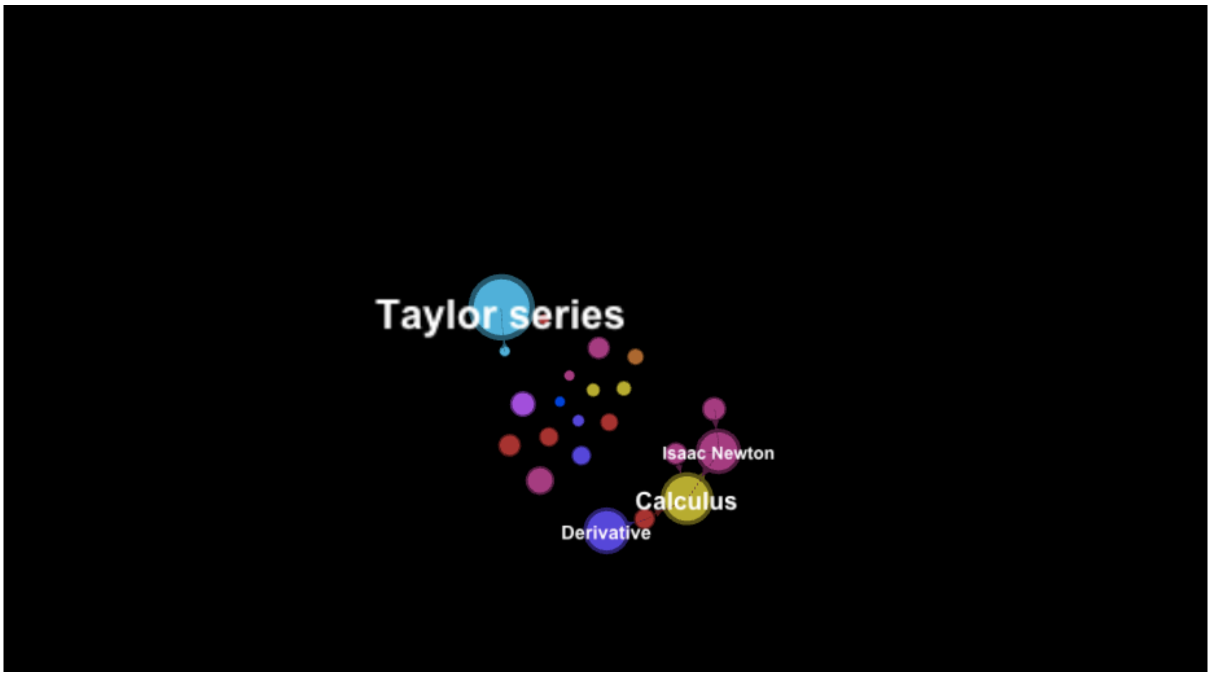


Figure 6 First frame of the final graph

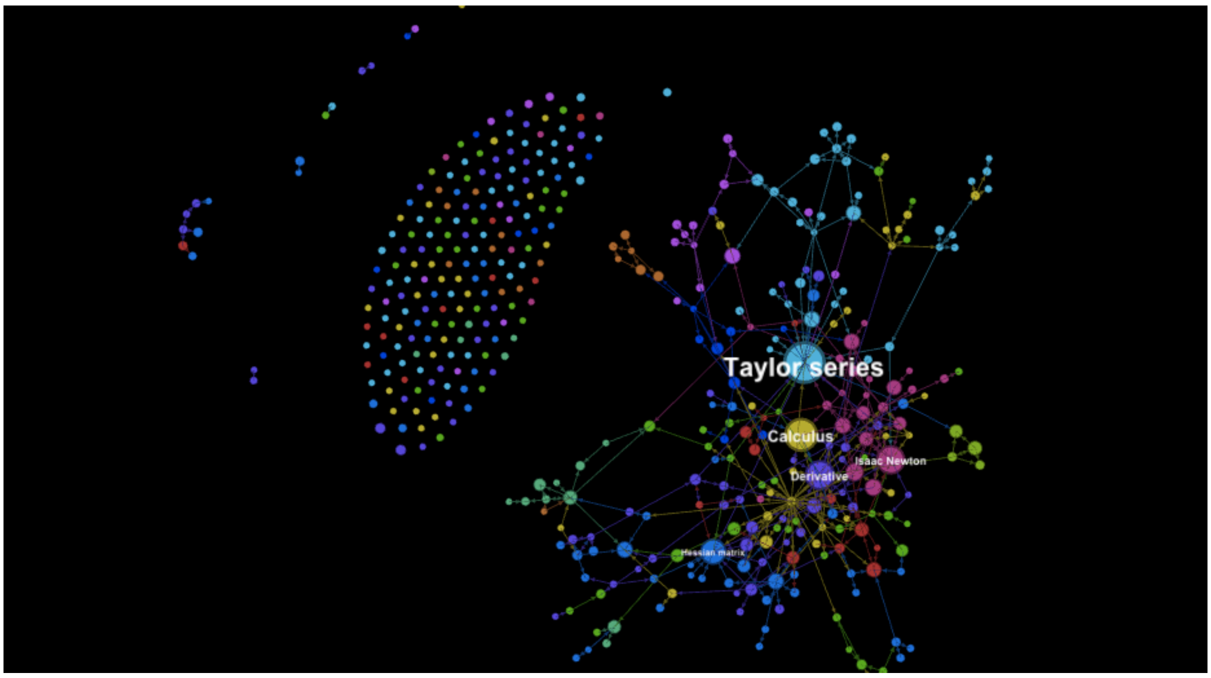


Figure 7 Last frame of final graph

The other algorithm Fruchterman Reingold make the nodes display like a forest. And the dynamic process shows the growth of the forest.

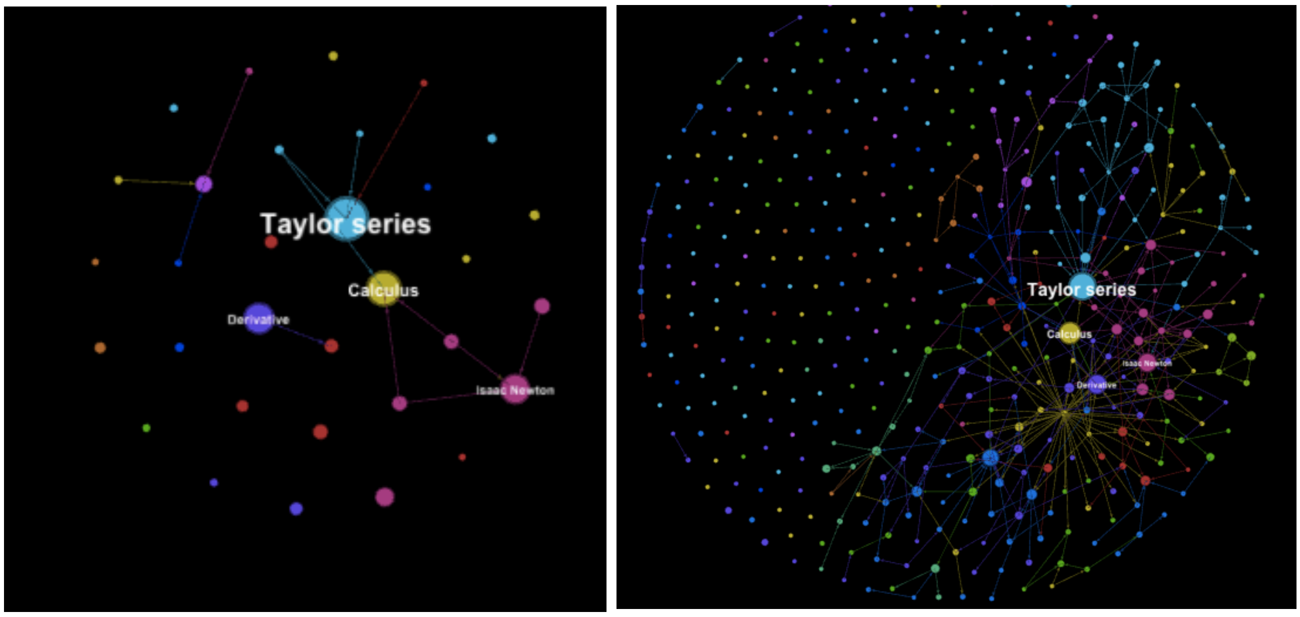


Figure 8 The groth of the forest

We also considered to some export plug-ins to export the visualization. Such as SeaDragon Web plug-in. But through exporting the graph will loss the dynamic feature. So we give up this version at last.

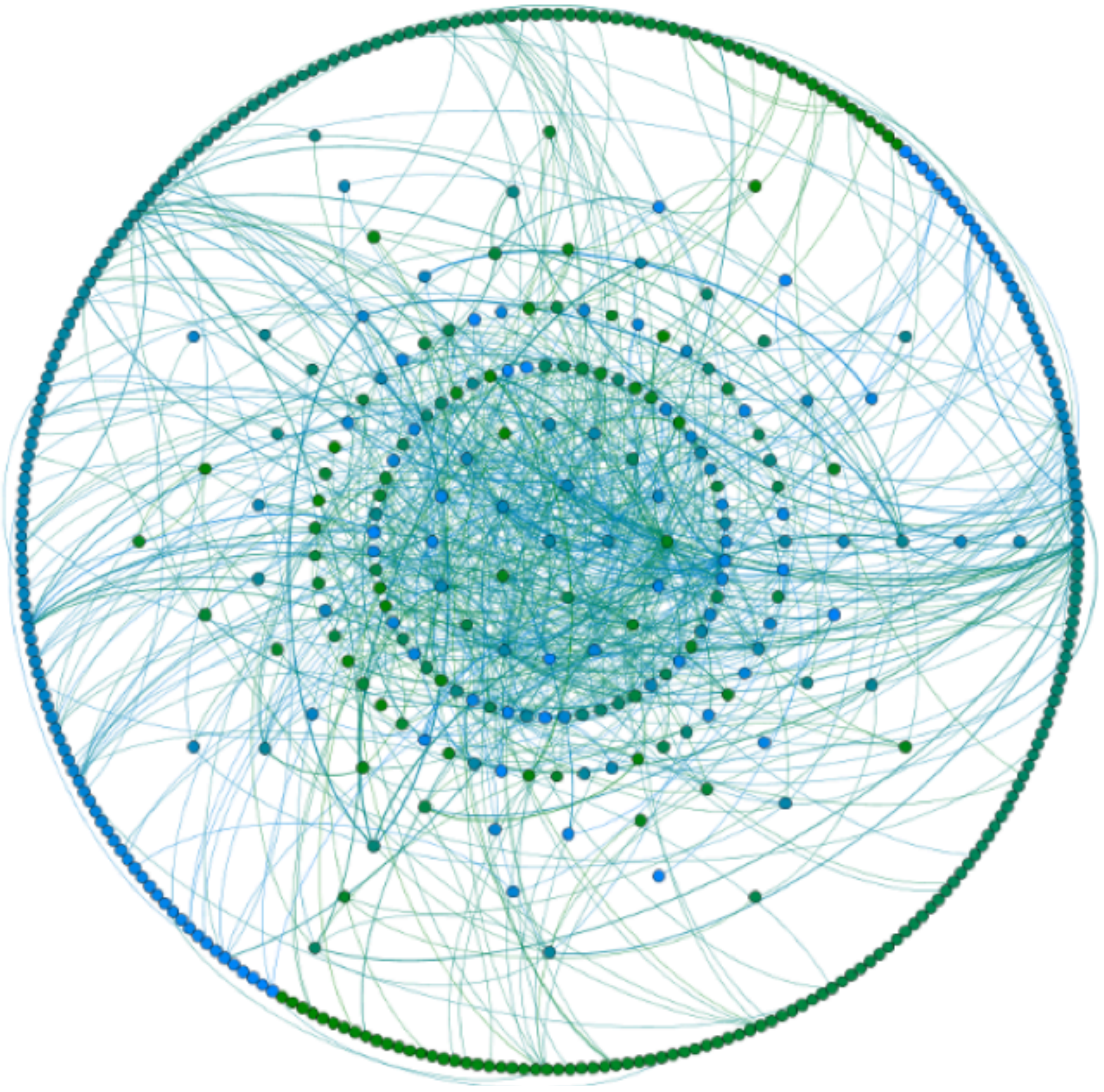


Figure 9 Web export version

7. Visualization Optimization(Using Tableau)

In the process of out project, we felt that Gephi cannot meet our full requirement. Gephi can display network data well. But it lacks statistical analysis functions. Further display and mining of the data need new software.

We choose Tableau as our second tool to demonstrate statistical data and make discovery in data. By drawing charts more characteristics of data are reveled. The following is the charts drew with Tableau:

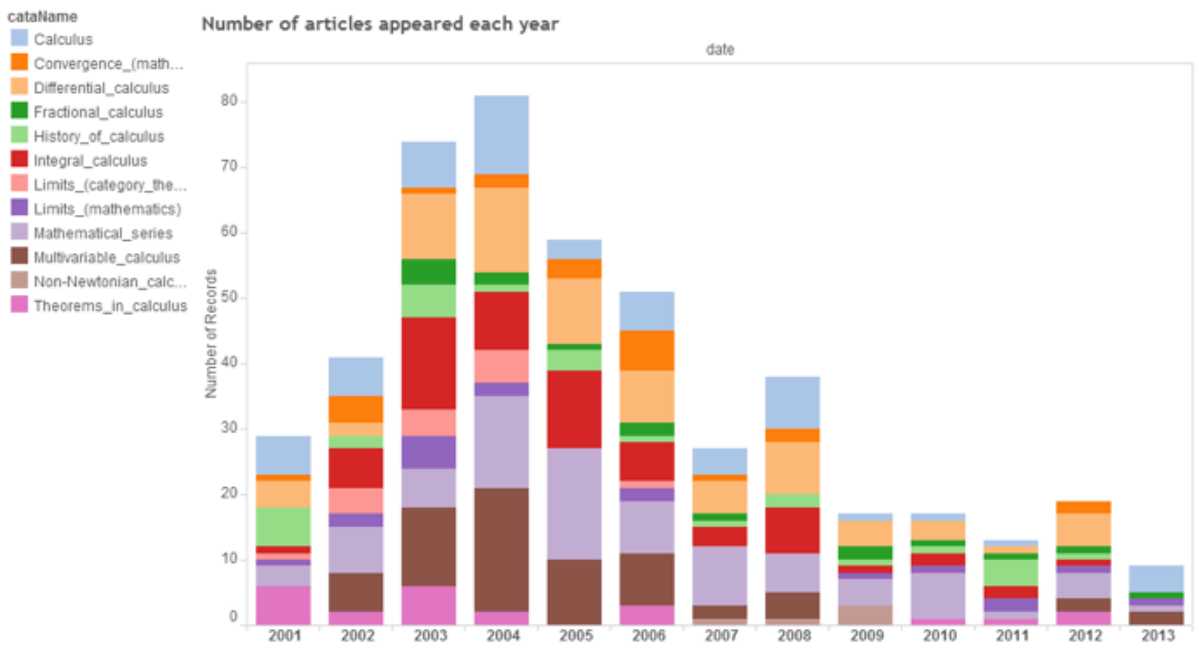


Figure 1 Number of articles appeared each year-Bar Chart

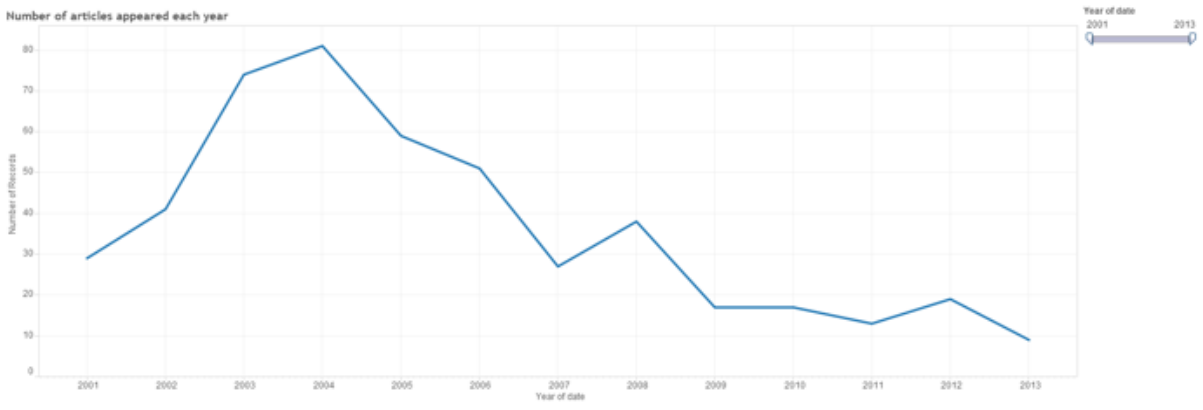


Figure 2 Number of articles appeared each year -Line Chart

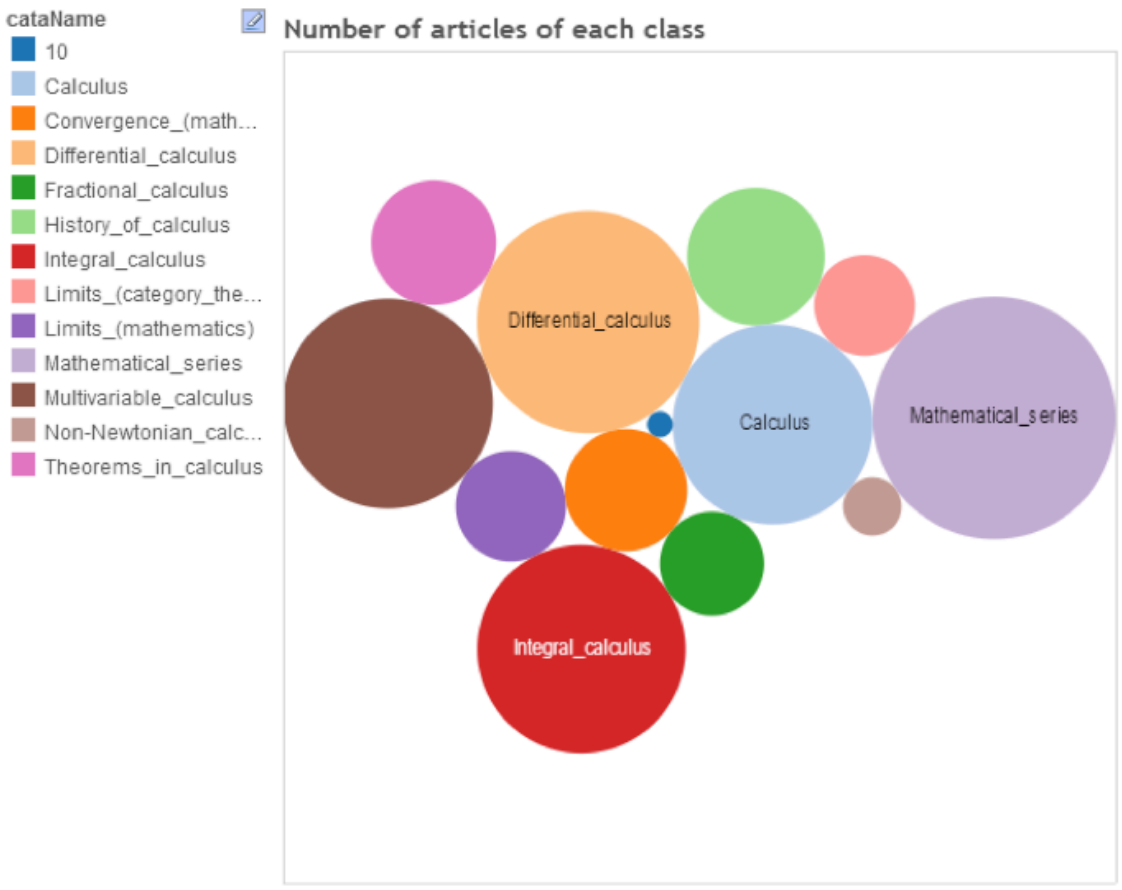


Figure 3 Number of articles of each year-Bubble Chart

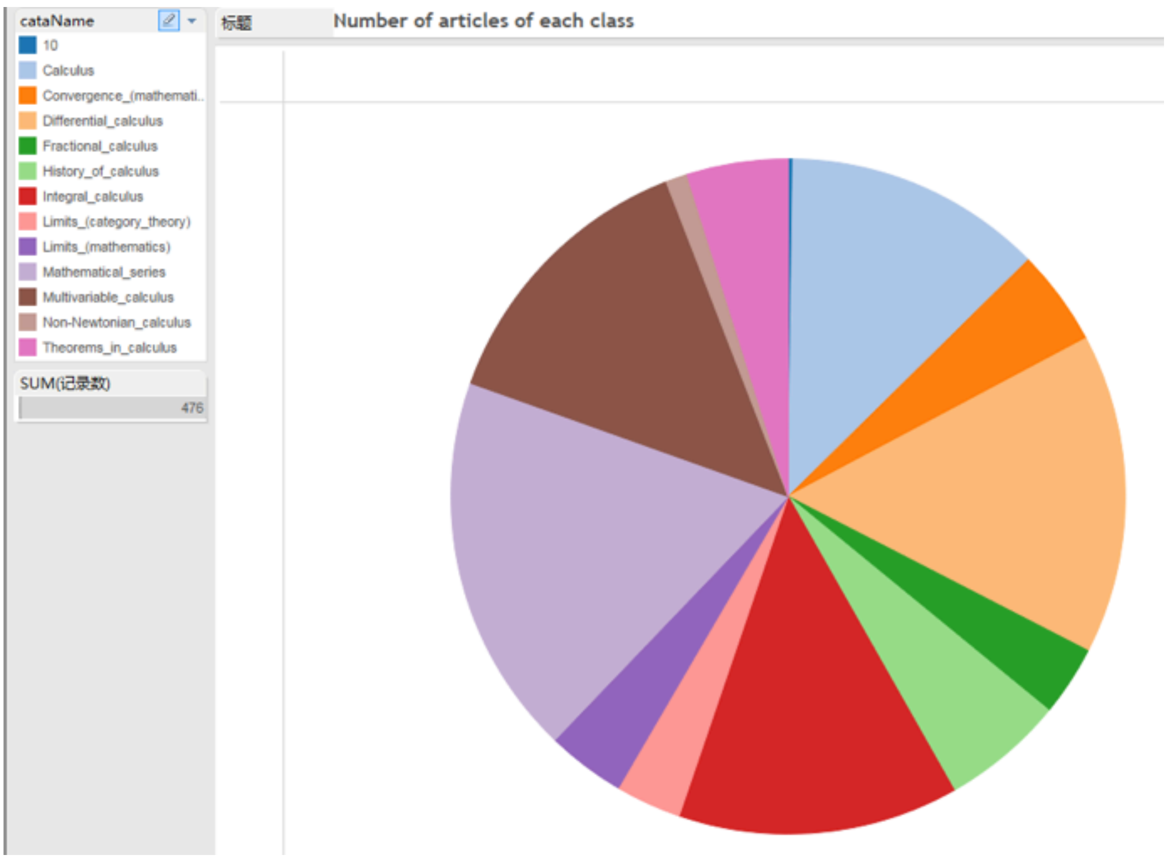


Figure 4 Number of articles of each class -pie chart

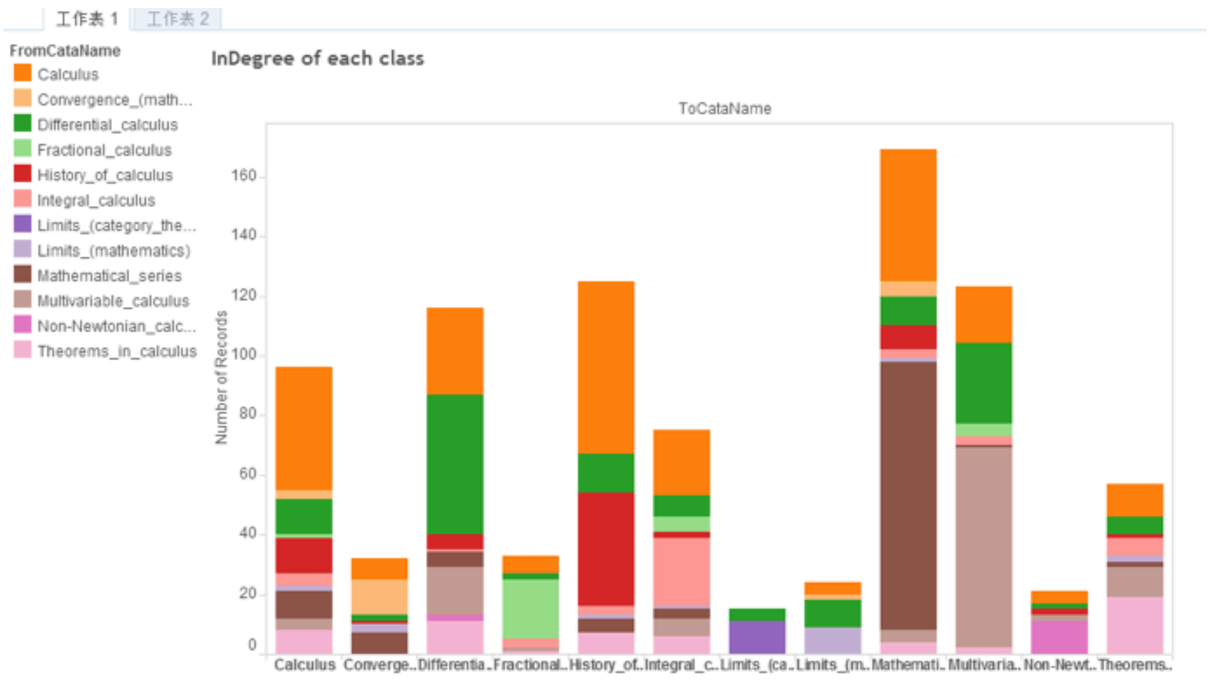


Figure5 In-degree of each class

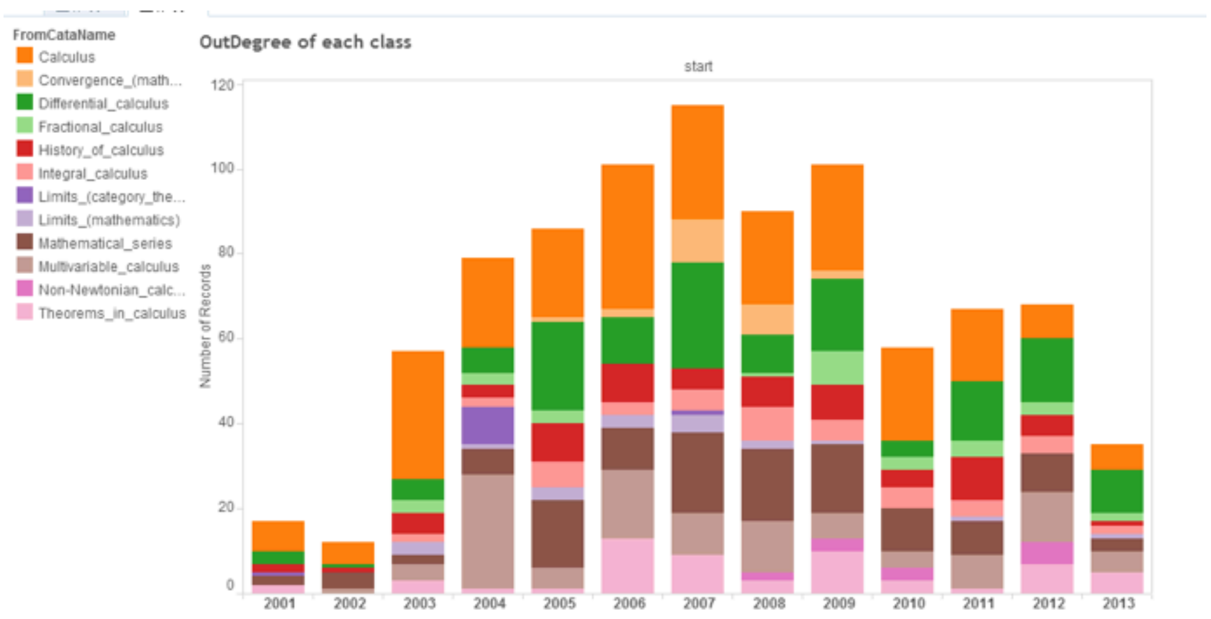


Figure 6 Out-Degree of each class

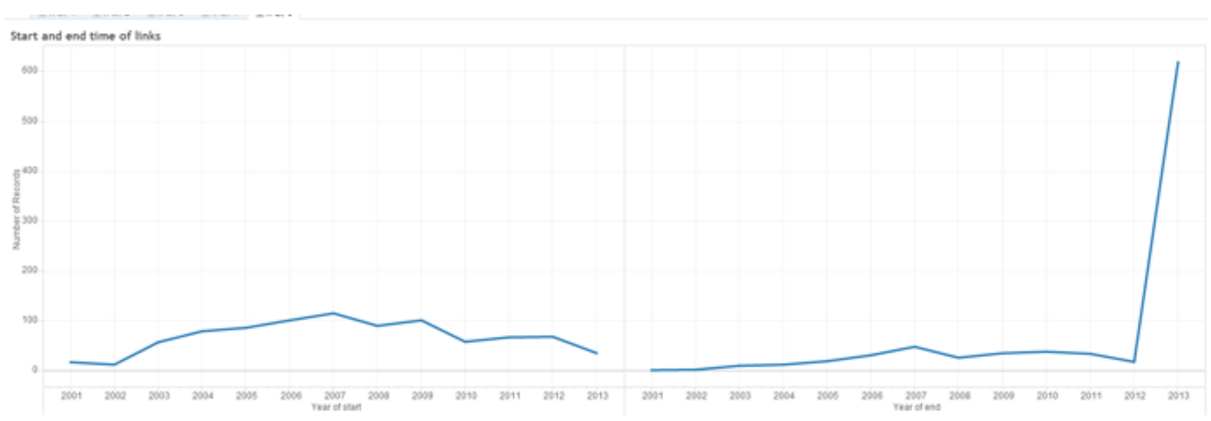
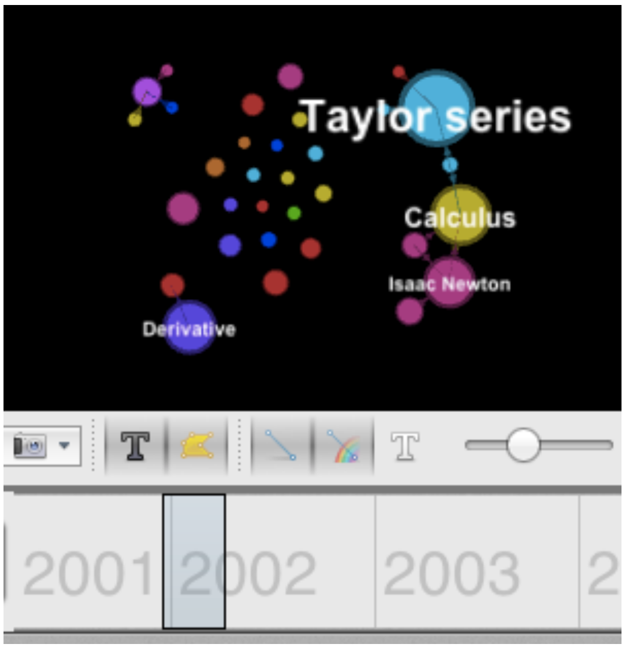


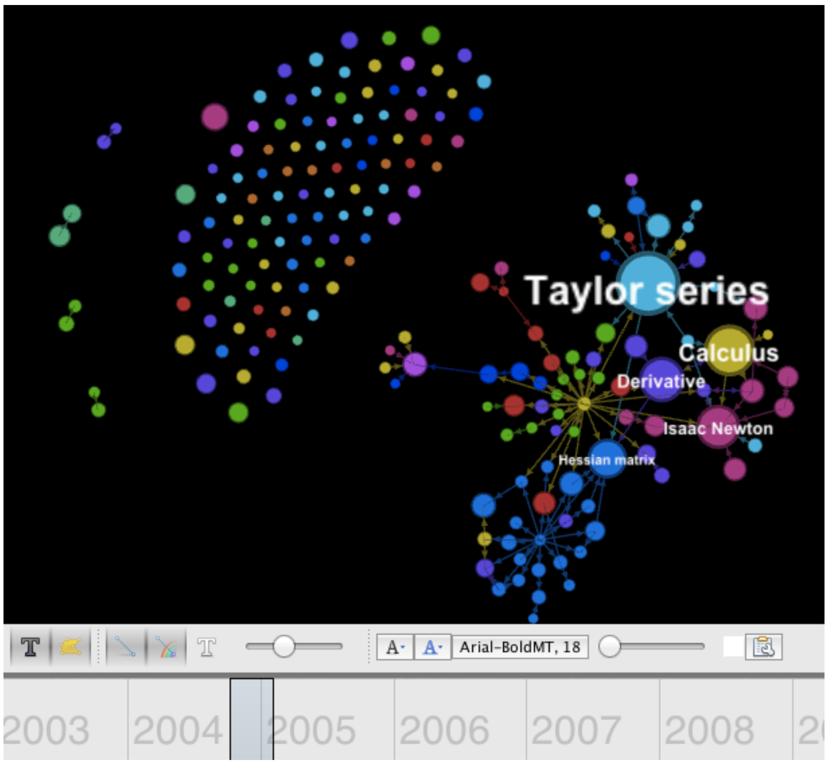
Figure 7 Start and end time of links

8. The Discovery In The Data

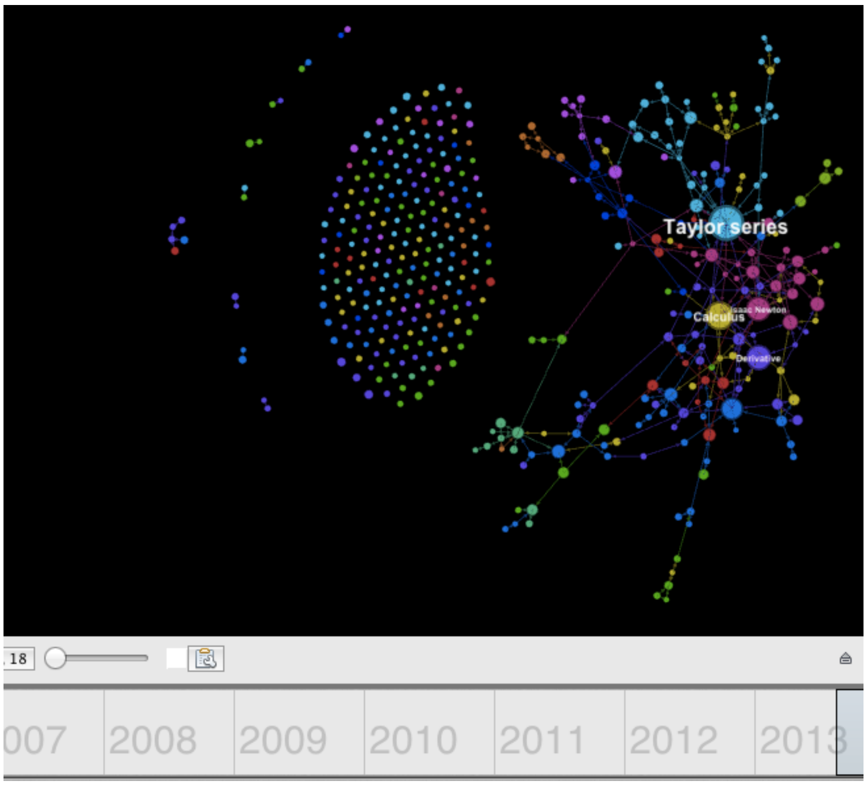
On one hand, as you can see in the attached video, we can find that, in early years, e.g. 2002, there are not too many knowledge been created on wiki. But the most basic knowledge like Taylor Series or Calculus are already been created.



As time goes on, more terms are created by users. The basic terms are referred by the new terms, and many new terms haven't been referred yet.



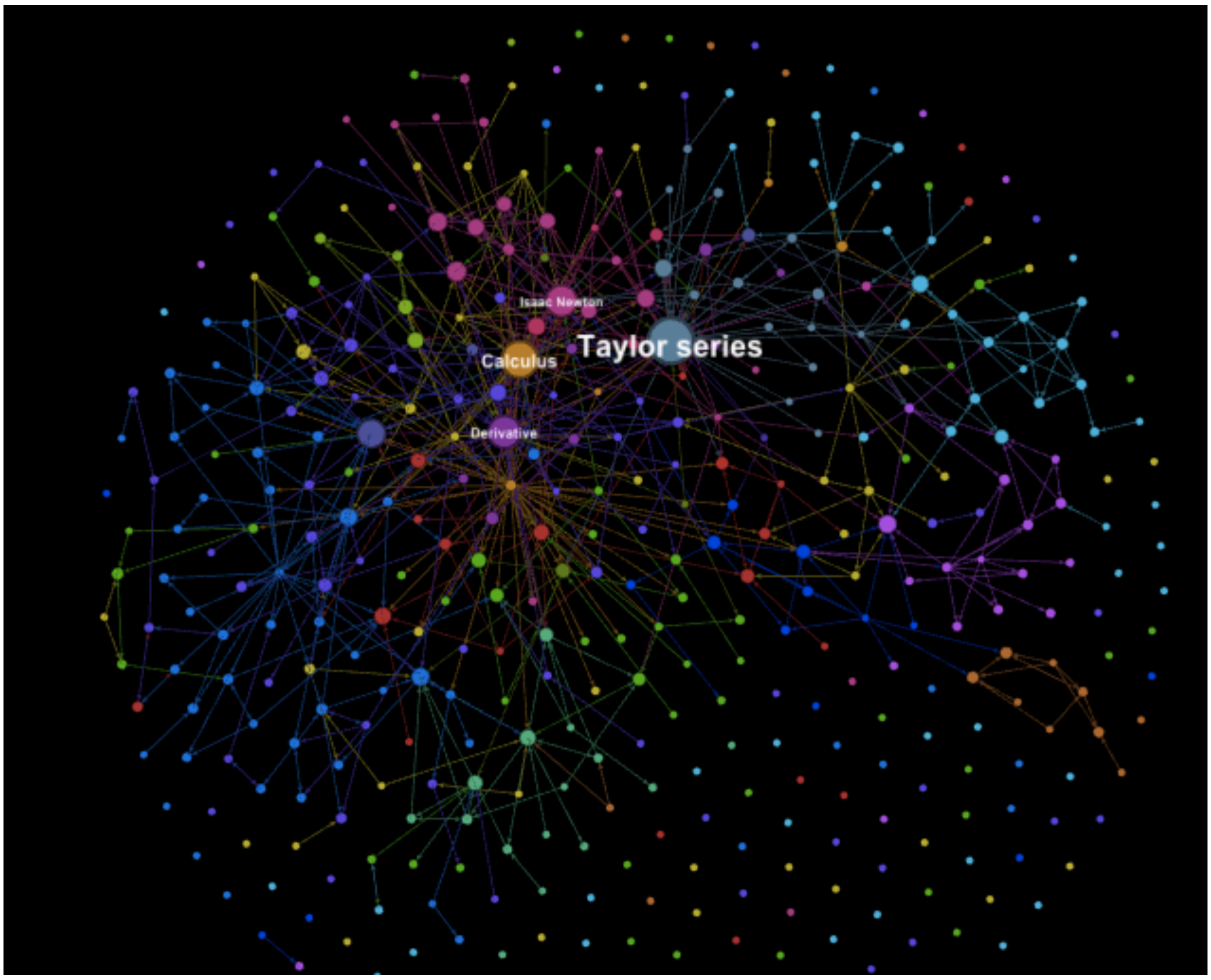
The final frame shows the latest structure in wiki. We can observe from the frame that, the more terms in the structure, the more reference there is. However, the group of none reference terms is due to the amount limit of the data.



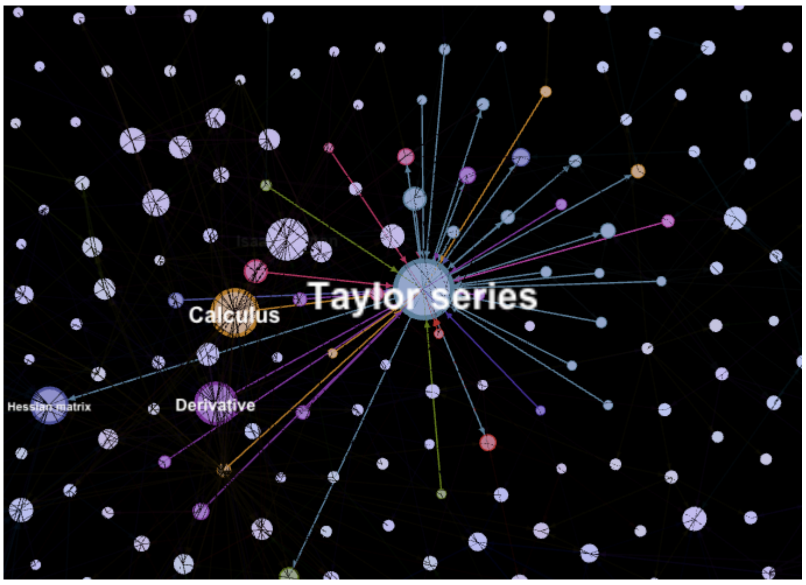
Regardless of the none reference terms in the frame, we can find that, a knowledge system is highly inter connected. Wiki, or even the whole internet is a big knowledge system. The system may be small and without two much nodes at first, but it grows with the action of the system users. Term in wiki is referred by many other terms, which is similar to a webpage on the internet is referred by other web pages.

The colour of the nodes in the frame shows its direct area. There are about 12 areas in this project. coloured by 12 different colours. The visualization shows that different areas are also connected, which means that, a term will not only refer the terms in its own area, but also refer to other areas. This feature makes the sub knowledge systems combined to a big knowledge.

On the other hand, by observing the overview output of Fruchterman Reingold, we can regard the knowledge system as a forest. The isolated nodes are like small trees, the connected nodes are big trees with many branches.



For example, if we regard Taylor Series as root, and other terms who refer to Taylor Series as its branches or leaves, we will get a tree as a result.



In summary, suppose we can visualization all the data in wiki, there may be no isolated nodes. The wiki may be a big knowledge forest with the sub category as its trees.

9. Planned But Not Yet Implemented Functions

As we initially expected, the whole Wiki database will be visualized by different categories and discover the growth of the Wiki knowledge system.

However we were restricted by the memory and storage of the software we used. Gephi is not so suitable for massive data unless the computer has tons of RAM. After we forfeit to download the original 55G wiki data, we chose to complete a small part of the data about Mathematics first.

D3.js, a JavaScript library for manipulating documents based on data, is one of the methods by which we planned to do visualization showcase which would be incredibly stunning if we could finish. Unluckily, considering the time limitation, we made our work a little bit simpler by using 2 generally-used tools.