Imperial College London

IMPERIAL COLLEGE LONDON

DEPARTMENT OF COMPUTING

Scientific Survey Auto Creation

Author:
Bingwei Chen

Supervisor:
Wu Chao

Submitted in partial fulfillment of the requirements for the MSc degree in MSc Computing Science of Imperial College London

September 2015

Abstract

To extract useful and related information in certain scientific area becomes more complex and difficult since the amount of scientific paper published is continuous growing. Some academic publications with significant redundant information confuse the user to decide which part is relevant. The existing search engines only return theses based on semantic similarity, without showing inside relationships such as reference between these papers. Therefore, the searching result of current search engine is partially complete due to the missing of citation relation.

This project aims to develop an application which can generate coherence graph of searching query from researchers. By contribute of storyline in previous research, coherence graph will be built by using citation relation instead of semantic similarity. A search system, a Java graph construction system and a web application are constructed to achieve the application. The Java Restful API is used to communicate between web and Java program. Concurrency of Java is applied to improve the speed of the program. The web application based on React is constructed and D3.js is used for visualization. Some tests are conducted to evaluate the result of coherence graph and its concurrency procedure. The parameters of the graph such as number of node, sub-chain length and coherence threshold will be adjusted and analysed. Finally, the limitation of speed and accuracy will be discussed and some ideas of improvement are listed.

This project is uploaded to github and its procedure of construction is given in appendix A. User manual and some surveys conducted by users are displayed in appendix B and C.

Acknowledgments

I would like to thank my supervisors Dr Chao Wu for his invaluable advice and guidance.

I would like to thank Dr Weiming Lu for his support of understanding storyline concept.

I would also like to thanks people who complete the survey and give valuable feedback.

Finally, I would like to thank Imperial College London's Computing Support Group, Communications and Facilities departments for providing us with their software, documents and support.

Declaration

Some algorithms of generating coherence graph is contributed by Lingyu Lu in his report "Application on Unstructured Data for Academic Literature".

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Objectivities . . . 2
1.2.1 Search System . . . 2
1.2.2 Coherence Graph . . . 3
1.2.3 Web Application . . . 3
1.3 Contributions . . . 3
1.4 Report Outline . . . 3

2 Background **5**
2.1 Document Retrieval . . . 5
2.1.1 Overview . . . 5
2.1.2 Stemming and Lemmatization . . . 5
2.1.3 TF-IDF . . . 5
2.2 Storyline . . . 6
2.3 Graph in Java . . . 8
2.4 Solr . . . 8
2.5 React . . . 9
2.6 Ant Design . . . 9
2.7 Spark Framework . . . 10
2.8 D3.js . . . 10
2.9 Related Works . . . 10
2.9.1 Application on Unstructured Data for Academic Literature . . . 10
2.9.2 Metro Map of Science . . . 11

3 Design **12**
3.1 Overview . . . 12
3.2 Search System . . . 12
3.3 Citation Graph . . . 13
3.4 Coherence Graph . . . 13
3.5 Web Interface . . . 14

4 Implementation **16**
4.1 Overview . . . 16
4.2 Search System . . . 17
4.2.1 Overview . . . 17
4.2.2 Pre-process . . . 17
4.2.3 Schema . . . 18
4.2.4 Commands and API . . . 19
4.2.5 Search procedure . . . 19
4.2.6 Cross Domain Problem . . . 20

4.3 Java Graph Construction System . . . 20
4.3.1 Overview . . . 20
4.3.2 Construct Graph . . . 20
4.3.3 Construct Citation Graph . . . 21
4.3.4 Construct Coherence Graph . . . 23
4.4 Concurrency . . . 27
4.4.1 Overview . . . 27
4.4.2 Concurrency of Monte Carlo Simulation . . . 27
4.4.3 Concurrency of Constructing Graph . . . 27
4.4.4 Concurrency of Constructing Citation Graph . . . 27
4.4.5 Asynchronous of concurrency . . . 27
4.5 Java Restful API . . . 28
4.5.1 Spark . . . 28
4.5.2 CORS . . . 28
4.6 Web application . . . 29
4.6.1 Overview . . . 29
4.6.2 Main layout . . . 29
4.6.3 Ant design components . . . 29
4.6.4 D3.js . . . 30

5 Results and Discussion **32**
5.1 Overview . . . 32
5.2 Website . . . 32
5.3 Result of Search . . . 34
5.4 Result of Coherence Graph . . . 34
5.4.1 Parameters Discussion . . . 34
5.4.2 Evaluation of an Example . . . 39
5.5 Speed of Program . . . 39
5.5.1 Time of Each Part . . . 40
5.5.2 Time Spends on Adding Ancestral Labels . . . 41
5.5.3 Time Spends on Adding Citation and Same Author Labels . . . 41

6 Conclusion and Future work **42**
6.1 Limitation . . . 42
6.2 Future work . . . 43
6.3 Conclusion . . . 44

References **46**

A Github **48**

B User Manual **55**

C User Survey **57**

Chapter 1

Introduction

1.1 Motivation

There is an increasingly apparent tendency that information overload has becoming an important problem with the advent of information technology. An unprecedented amount of information is generated by the development of Internet penetration. “IBM estimates that 90% of the world’s online data has been created just in the past two years” [1]. These will lead to decision paralysis and time wasting on redundant information. People unable to make a judgement as they cannot see what is relevant.

In order to solve the problem of information overload, some algorithms of data mining and information retrieval coming forward. Unlike structured data, un-structured data which is more ubiquitous does not have a pre-defined model. And it is of great importance because most information originates in unstructured form[2]. Research paper is a classic example of unstructured data. The amount of academic publication increase rapidly, causing researchers suffer in redundant information. In 2016, the total number of academic publication in IEEE Xplore is 4,011,868. Even for topic “neural network”, it still shows 122,805 results.[3]

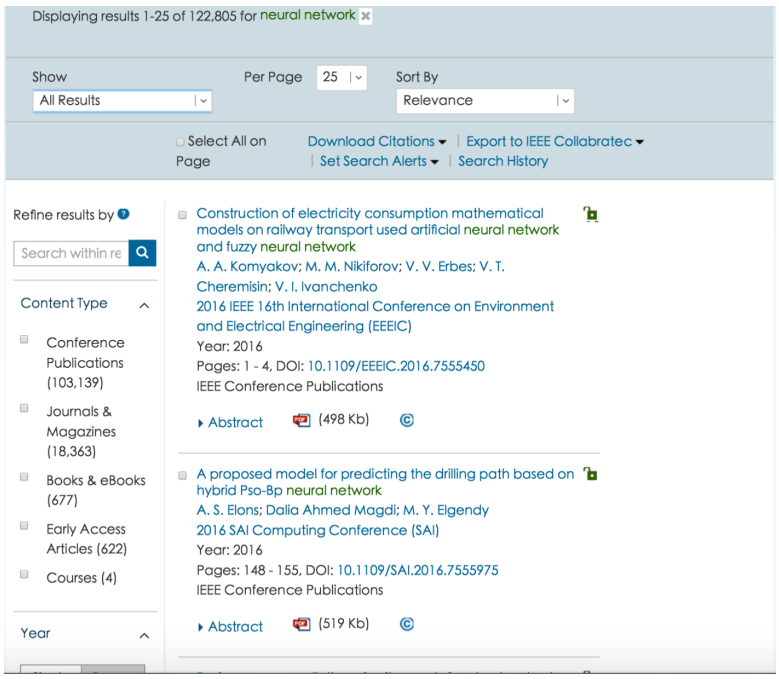


Figure 1.1: Xplore [3]

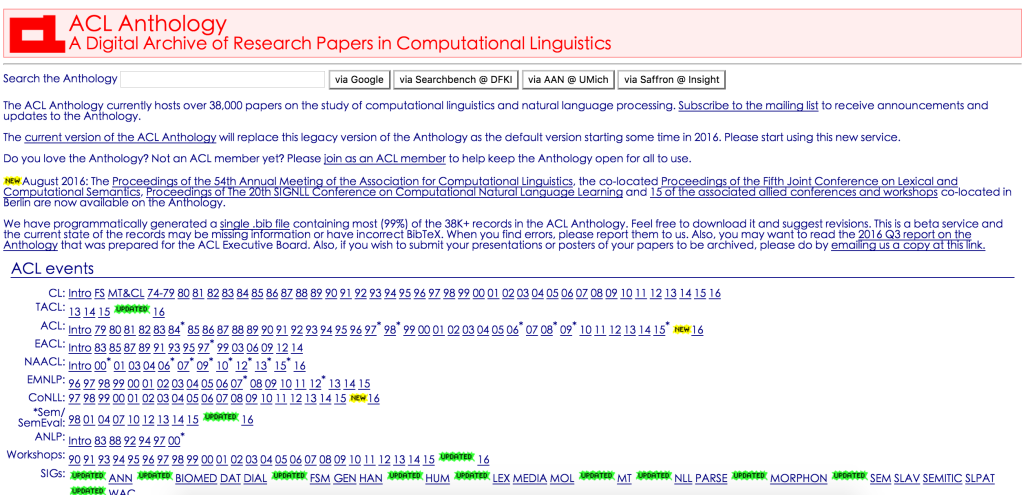


Figure 1.2: ACL Anthology [4]

On the other hand, the existing search engines return papers only rely on semantic similarity, which does not cover the most relationship between papers. This result is not complete accurate because the citation relation will influence the importance of the paper. And in most cases, researchers need to find papers which have relevant connections. Papers along the evolution line or grouped by subtopic is more useful. However, the result of Xplore[1] only list papers according to the search query as figure 1.1 shows. The library of ACL Anthology[2] shown in figure 1.2 even do not have any setting of result.

Base on the situations above, the application which generates a coherence graph by using citation relation of publications is useful. It can show the evolution and overall view of the topic, which helps researchers understand the topic and acquire useful information.

1.2 Objectivities

The aim of this project is to develop an application which can generate storyline graph for the search query collected from users and visualize the graph on website. Firstly, the local search engine should be built which can use search query to get valid papers with the score of relevance. Secondly, using the paper returned by search engine, a program should be created to generate the coherence graph of papers. Finally, the web application should be built to communicate with users and show the coherence graph on website.

1.2.1 Search System

The first task is to convert source paper to machine-readable data. Then words should be extracted and refined from data. After that, the index of words and papers should built. Finally, the relevant score of each paper for each searching should be calculated.

[1]IEEE Xplore: a scholarly research database that indexes, abstracts, and provides full-text for articles and papers on computer science, electrical engineering and electronics.

[2]ACL Anthology: Association for Computational Linguistics to facilitate the sharing of information on all aspects of Computational Linguistics

1.2.2 Coherence Graph

After collect the semantic similarity of each paper, the citation relation need be added by consider the reference, authors, and some hidden information of the paper. Based on the influences output by citation graphs, the storyline should be constructed which represents the development of the search query.

1.2.3 Web Application

Firstly, the web application should collect search query and some custom options from users. Then it need to communicate with local program to get valid papers and the coherence graph. Finally, the coherence graph should be visualized on the website.

1.3 Contributions

This project contribute an storyline auto-generation and visualization application, by building a search system, a Java program and the front end. The search system is based on Solr which will be introduced in background chapter, and it can return the papers contained in JSON[1] for less than one seconds. The Java program which can create the coherence graph need several minutes to generate the output. It will be active when the HTTP request is sent from web application to local Java Restful API server. Programmer uses React[2] as the framework of front end and D3.js[3] to realize the visualization coherence graph. All of concepts mentioned here will be introduced in detail in background chapter.

Previous related work by Lingyu Lu [5] is also capable of searching and generate coherence graph. The data crawled by it from ACL Anthology will be used continually in this project.

1.4 Report Outline

Chapter 2: This chapter summarises some basic theories and presents the software used. The theory of information retrieval and storyline is introduced. The software of react, Solr, Spark framework and d3.js is introduced. The related works are also analysed.

Chapter 3: This chapter presents the design of the application. The overview of the design will be explained and the each modules will be analysed in detail. It will be divided by 4 section: Search system, citating graph, coherence graph and web interface.

Chapter 4: This chapter shows the detail procedures of entire application. The overview of the implementation will be analysed. The detail procedure of construct search system, Java graph construction system and web application will be presented.

[1] JSON: JavaScript Object Notation
[2] React: an open-source JavaScript library providing a view for data rendered as HTML
[3] D3.js: Data-Driven Documents is a JavaScript library for producing dynamic, interactive data visualizations in web browsers.

Chapter 5: This chapter provides experiment results of this application and evaluate the results of different sections. The achievement of website will be showed and result of coherence will be discussed in different parameters such as numbers of nodes, sub-chain length and coherence threshold. The evaluation of concurrency also provided in this chapter

Chapter 6: This chapter concludes the contributions of this projects. The limitation and future task of improvements are investigated.

Chapter 2

Background

2.1 Document Retrieval

2.1.1 Overview

Document retrieval is the technology to match some stated user query against a set of free-text records which could be unstructured data such as articles, report or scientific publications. User queries can range from multi-sentence to a few words. It is the fundamental basis of all Internet search engines. One database and one classification algorithm to build text index are needed to construct the document retrieval system. The system can find relevant documents to user queries and evaluate the matching results and sort them according to relevance. [6]

In this section, some relevant basic algorithms of document retrieval such as stemming, lemmatization and TF-IDF will be introduced.

2.1.2 Stemming and Lemmatization

Stemming is the process of reducing inflected words to their base form. Many search engines will dealing with vocabulary, synonyms for the same stem as a query expansion. Stemming project generally related to stemming algorithms. [7]

Lemmatization is the process of grouping together the different inflected forms of a word so they can be analysed as a single item and in computational linguistics it may involve complex tasks such as understanding context and determining the part of speech of a word in a sentence [8].

Stemming algorithms are typically easier to implement and run faster. However, Lemmatization algorithms can discriminate between words which have different meanings depending on part of speech [8].

2.1.3 TF-IDF

TF-IDF [1] is a weight often used in information retrieval which evaluate how important a word is to a document in a corpus. The importance increases proportionally to the number of times a word appears in the document but is offset by the frequency of the word in the corpus. TF-IDF weight is the product of TF and IDF. [9]

[1] TF-IDF: term frequency–inverse document frequency

Term Frequency

Term Frequency measures how frequently a term occurs in a document. Since every document is different in length, it is possible that a term would appear much more times in long documents than shorter ones. Thus, the term frequency is often divided by the document length as a way of normalization.

$$Term\ Frequency(t) = \frac{Number\ of\ times\ term\ t\ appears\ in\ a\ document}{Total\ number\ of\ terms\ in\ the\ document} \tag{2.1}$$

Inverse Document Frequency

Inverse Document Frequency measures how important a term is. All terms are considered equally important while computing TF . However it is known that certain terms, such as “is”, “of”, and “that”, may appear a lot of times but have little importance. Thus we need to weigh down the frequent terms while scale up the rare ones. [9]

$$Inverse\ Document\ Frequency(t) = \frac{Total\ number\ of\ documents}{Number\ of\ documents\ with\ term\ t\ in\ it} \tag{2.2}$$

2.2 Storyline

Storyline In News

The storyline in News can help users quickly understand the overall information which should “present a terse and brief summarization about the event by showing different clues of its development, and thus help readers to effectively find out ‘when, where, what, how and why’ at a glance”. [10]

The sentence-based temporal summarization of news topic was proposed by Allan. [11] It collects news stories and output a sequence of extracted sentences. Each sentence is one event on given topic. This system works well when 10% of the original information can be extracted. However, that output is still too large.

Metro Map of Science

The Metro Maps of Science came forward by Dafna Shahaf at 2012. [12] Metrics of influence, coverage, and connectivity for scientific literature are proposed, which can be used to create structured summaries of information. As figure 2.1 shows, multiple lines of research are rendered by different colours and dashed gray lines are used to represent relevant citation relationship.

Generating Chains

The method of formalise the characteristics of a good chain are proposed in “Connecting the Dots Between News Articles” by Dafna Shahaf, and this paper provide an efficient algorithm to connect two fixed endpoints. [13]

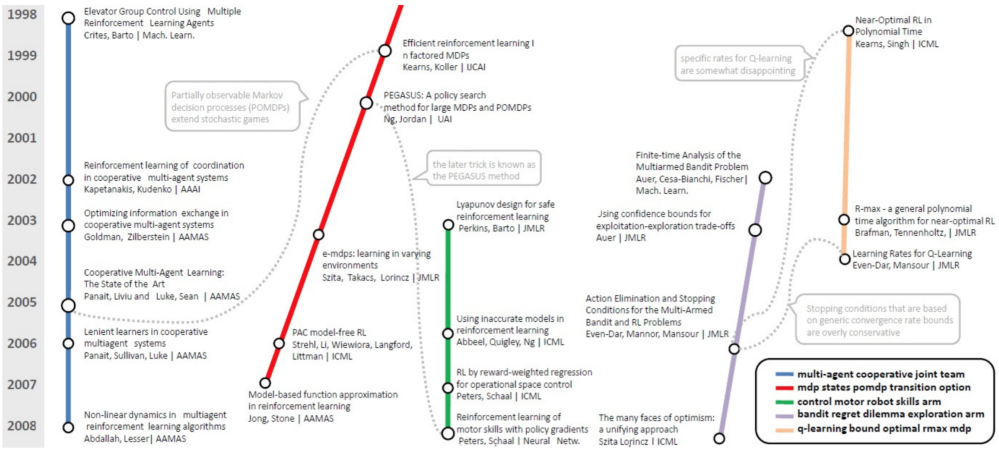


Figure 2.1: Metro Map

Coherence of Chain

The coherence of a chain is proportion to the similarity between each two consecutive papers along the chain. However, the coherence of the entire chain will be destroyed when a single bad transition exist. The strength of the chain should be adjust by the strength of the weakest link. [14]

m-length sub-chain

The combining of two strong chains may result in a much weaker chain. There is no evidence that both chains belong to the same storyline when concatenate (d1, ..., dk) with (dk, ..., d2k-1). However, combining with (d2, ..., dk+1) can assume they are in same storyline. [14] As figure 2.2 shows, the coherence graph can be constructed which contains all m-length sub-chain.

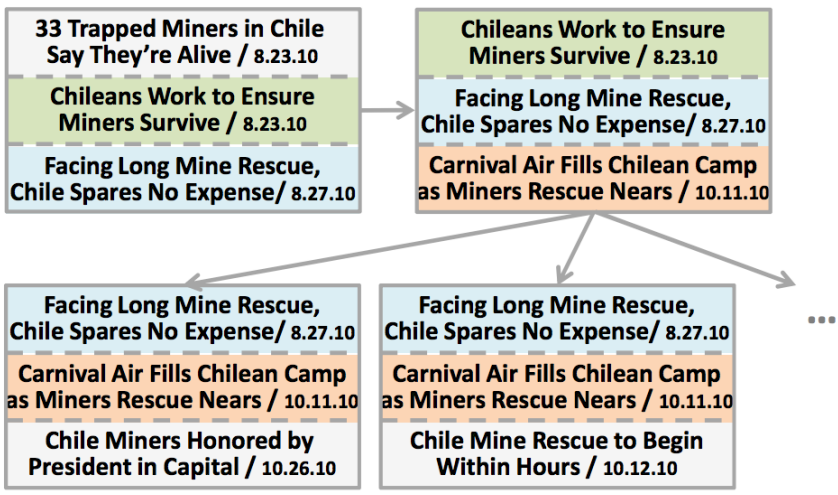


Figure 2.2: coherence graph for m = 3

2.3 Graph in Java

Graph is a structure amounting to a set of objects in which some pairs of the objects are in some sense "related." The objects are correspond to mathematical abstractions called vertices, and each of the related pairs of vertices is called an edge.

The edges of graph in this project are directed because the actions between object A and object B can not reciprocated. The graph can be represent in Java by adjacency matrix or adjacency List. The adjacency List will be used in this project which save a lot of memory and is fast to iterate over all edges. However, it may slower than adjacency matrix when finding the presence or absence specific edge.

2.4 Solr

"Solr is the popular, blazing-fast, open source enterprise search platform built on Apache Lucene." [15] Solr is built on Apache Lucene which is a Java-based library for building an inverted index to text-based documents. It is the second-most popular enterprise search engine and has REST-like HTTP/ XML and JSON APIs that make it usable from most popular programming languages.

Solr already completed the procedures of information retrieval such a tokenzation, stemming and part-of-speech tagging. It also support MapReduce framework which enable multiple servers to process.

Inverted index

As figure 4.2 shows, when the document is indexed, each field will be identified and their frequency will be stored in dictionary with the term of their document id. After this process, the frequency of search query in each document can be return quickly, by finding the field of query in dictionary. In order to increase the speed of search, integer number is used as the id of documents.

Advantages of Solr

- full-text search
- faceted search: enable information collection by applying multiple filters
- near real-time searching: documents are available for search almost immediately after being indexed
- rich document (e.g., Word, PDF) handling
- distributed search and index replication
- scalability: handle a growing amount of work
- fault tolerance

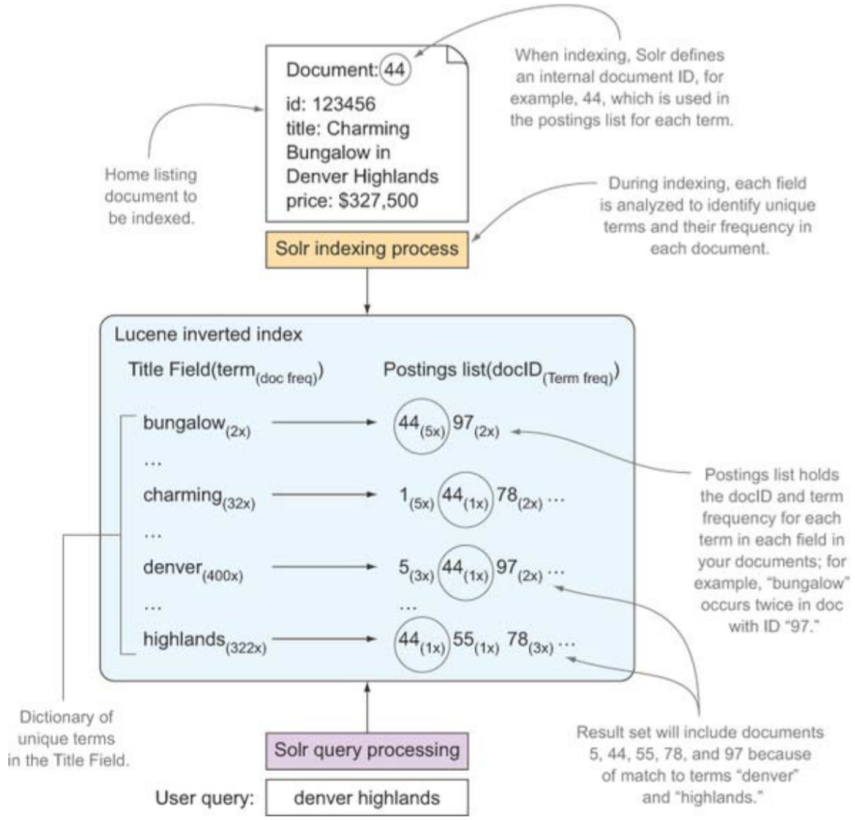


Figure 2.3: Solr inverted index explanation

2.5 React

React is a Javascript library for building user interfaces developed by Facebook. [16] It approaches building user interfaces differently by breaking them into components and really shines when the data changes over time.

- efficient updating of the HTML document when data changes
- clean separation between components on a modern single-page application
- subcomponents cannot directly affect enclosing components

2.6 Ant Design

Ant Design is an enterprise-class UI design language and React-based implementation. In designing procedure of front end, different specification and implementation bring a lot of confusion, and the duplication of similar pages and components reduce the efficiency of development. Ant Design can avoid these unnecessary cost and use unified specification to improve the experiment and efficiency of design[17].

Feature

- An enterprise-class design language and high quality UI
- Graceful UI components out of the box, base on React Component

- A npm[1] + webpack[2] + babel[3] + dora[4] workflow

2.7 Spark Framework

Spark is an open-source software web application framework and domain-specific language written in Java. It is intended for "quickly creating web-applications in Java with minimal effort". The Java 8's lambda philosophy used in Spark makes it less verbose than other Java web frameworks[18].

2.8 D3.js

D3.js is a JavaScript library for manipulating documents based on data. It can bring data to life using HTML, SVG, and CSS. Arbitrary data will be binded to a Document Object Model (DOM) and then data-driven transformations will be applied to the document[19].

The advantage of D3.js is that you can make your visualizations the way you want rather than bar, line or column. And it supports a large amount of data without limitation. It also can provide zoom or click functions for any graph you want[20].

2.9 Related Works

2.9.1 Application on Unstructured Data for Academic Literature

This report describes how to develop an application to analyzing, grouping academic paper and generating coherent storyline for these papers in a given research topic. The importance of generate storyline was explained and the ways of using storyline in news and scientific publication were discussed. The implementation of this application was achieved by four steps [5].

Dataset Collection

- Crawler4j is used to find correct pages stored in ACL archive and download PDF-format papers from these webpages.
- Jsoup is used to parse and acquire attributes for the paper collected by Crawler4j.
- PDFBox is used to convert pdf to txt
- ParsCit is used to extract citations from research papers

Pre-processing

- Stanford CoreNLP is used to process text in the sequence of sentence splitter, POS tagger and lemma-tizer.
- word2vec is used to cluster words based on distributed representations in vector space.

[1]npm: Default package manager for the JavaScript runtime environment Node.js.
[2]webpack: An open-source JavaScript module bundler.
[3]babel: A JavaScript compiler.
[4]dora: A fully pluggable server for development.

Citation Graph Construction

- find valid paper which contains at least one word in word cluster
- for each valid paper, find its set of citations papers
- for each citation paper, if it contains one word in word cluster, add “CITATION ”labelled edge
- for each valid paper, find its set of authors
- for each authors, find all its papers and add “COMMON_AUTHOR ”labelled edge

Coherence Graph Construction

- calculate influences between any pair of papers with respect to each word cluster

However, this application cost to much time on pre-processing and graph-construction and do not have user interface, which can not meet the requirement of web users. And the graphic image can not be generated automatically.

2.9.2 Metro Map of Science

Dafna Shahaf et al. create structured summaries of information, which called metro maps. It was used to help researchers acquire new knowledge efficiently. Firstly, they investigate methods for automatically connecting the dots. It provides a structured, easy way to navigate within a new topic and discover hidden connections. Their system can automatically find a coherent chain linking two articles[13]. However, it only focus on news domain. Then, they raise the complex stories spaghetti into branches, side stories, and intertwining narratives. The methodology they propose can explore these stories and explicitly show the relations among retrieved pieces and captures story development[14]. Finally, the metro map they created is improved with each level of zoom showing finer details and interactions[12].

Chapter 3

Design

3.1 Overview

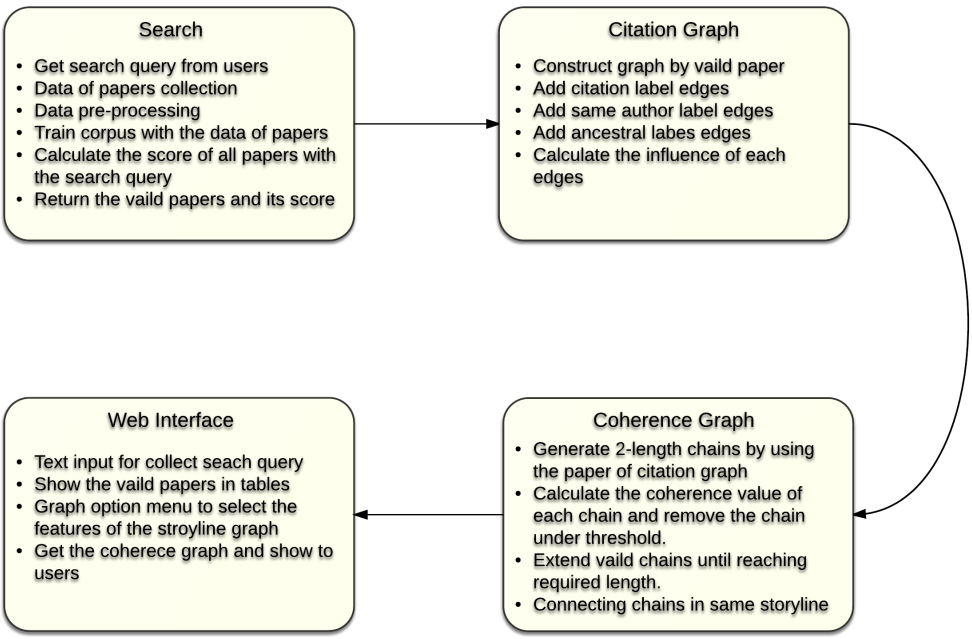


Figure 3.1: Overview of the architecture

The objective of this project is to build an application which can construct coherence graph and display it on web interface. The major work can be divided into 4 sections as figure 3.1 shows. Firstly, we need using search query to get valid papers and its score. Secondly, these data will be used in Java program to construct the citation graph. Then coherence graph will be generated by analysing the relationship of all papers. Finally, the data of the graph will be received by web interface and display to users.

3.2 Search System

The Solr platform which was introduced in background chapter will be used in this section. Initially, the corpus of Solr needs to be trained before searching. Papers collected from ACL Anthology (Association for Computational Linguistics) will be used and processed to standard documents required by Solr. Then, these documents will be indexed to Solr corpus and the

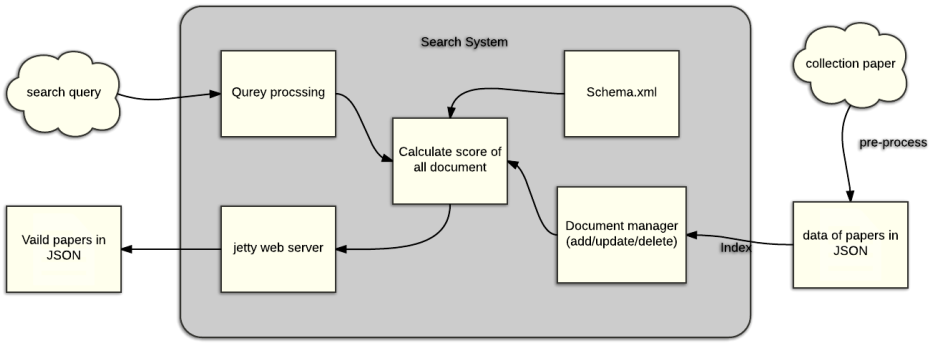


Figure 3.2: Search System

schema is set to define the field of documents. When the search query send to Solr server, the score of all documents will be calculated by inverted index. In the end, the valid papers with score will be returned in JSON form.

Collection Papers

The collection paper was got from Lingyu Lu [5]. It was crawled from ACL antology and converted to XML. There are 30501 papers totally collected in XML form .

3.3 Citation Graph

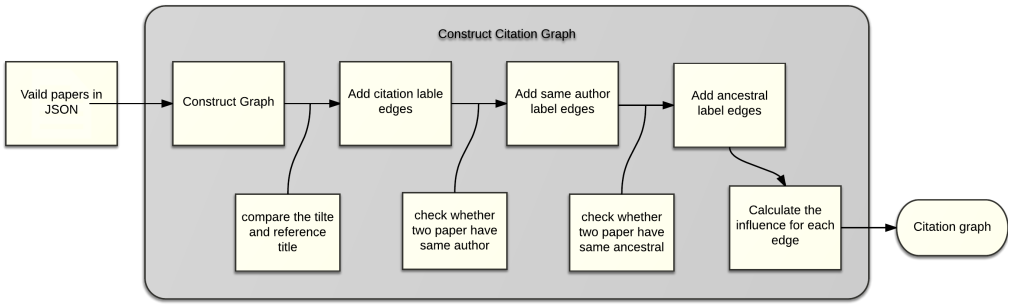


Figure 3.3: Construct Citation Graph

After the valid papers in JSON are got from searching section, programmer will use Java to construct the graph. Each paper should represent one vertex in the graph. The citation label edges can be added by comparing titles with reference titles of the graph. The procedure of adding same author label edges is similar. Monte Carlo Simulation will be used to get ancestral label edges. The algorithm will be designed to calculate the influence of two papers.

3.4 Coherence Graph

Coherence graph can be constructed by picking suitable chains from citation graph. However, representing all chains whose coherence is at least $\mathcal{T}$ is a non-trivial task. [14] Thus, the divide-and conquer algorithm is implemented to realise this function. Primarily, each pair of papers in citation graph can be collected as a chain and its coherence will be calculated.

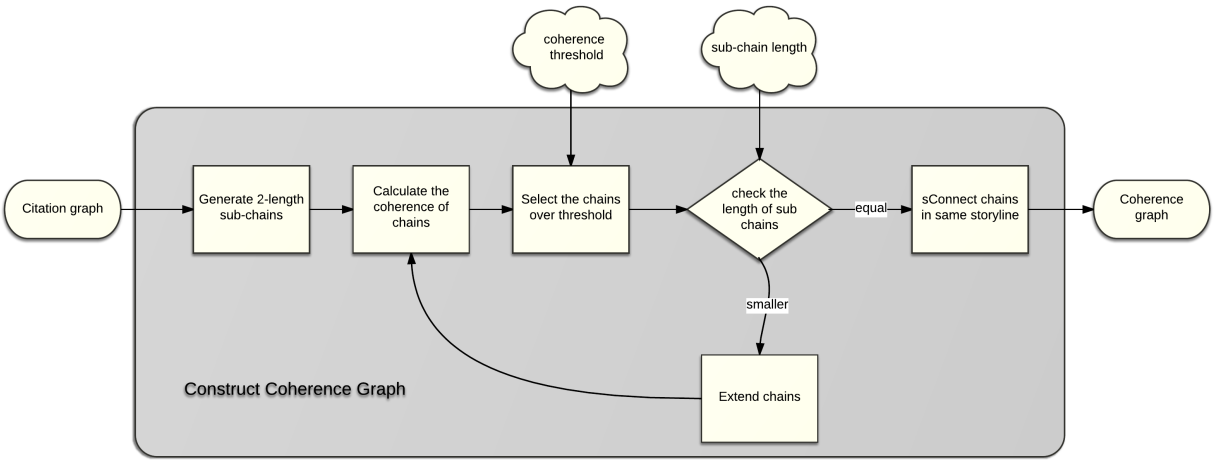


Figure 3.4: Construct Coherence Graph

Then, the chains above the threshold which is got from web interface, will be selected. After that, the system will check whether the length meet the requirement. The requirement of sub-chain length is also collected from web interface. The chains shorter than requirement will be extended until all of them reached the requirement length. At the end of the procedure, certain chains in same storyline will be connected and the coherence graph is constructed.

3.5 Web Interface

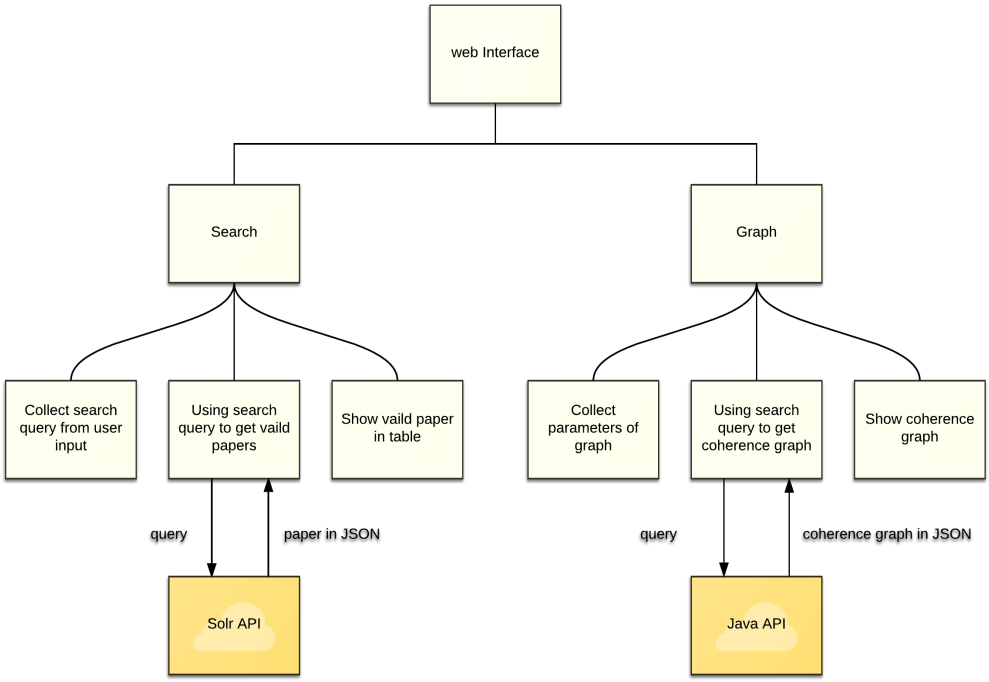


Figure 3.5: Web Interface

The web interface in the project should complete follow tasks:

- Obtain users' search query
- Use the search query to get valid papers from our search system

- Show papers ordered by score of relevance
- Display papers' title, author, URL, date and score in table.
- Collect parameters of graph from users such as the threshold of coherence and sub-chain length
- Use the search query and parameters of graph to get coherence graph from Java graph construction system
- Display coherence graph using D3.js.

The structure of the website is shown in figure 3.5. All functions will be achieved in search and graph pages. The search page can get the search query and show the table of valid graph. The graph pages will collect graph parameters and display coherence graph.

Chapter 4

Implementation

4.1 Overview

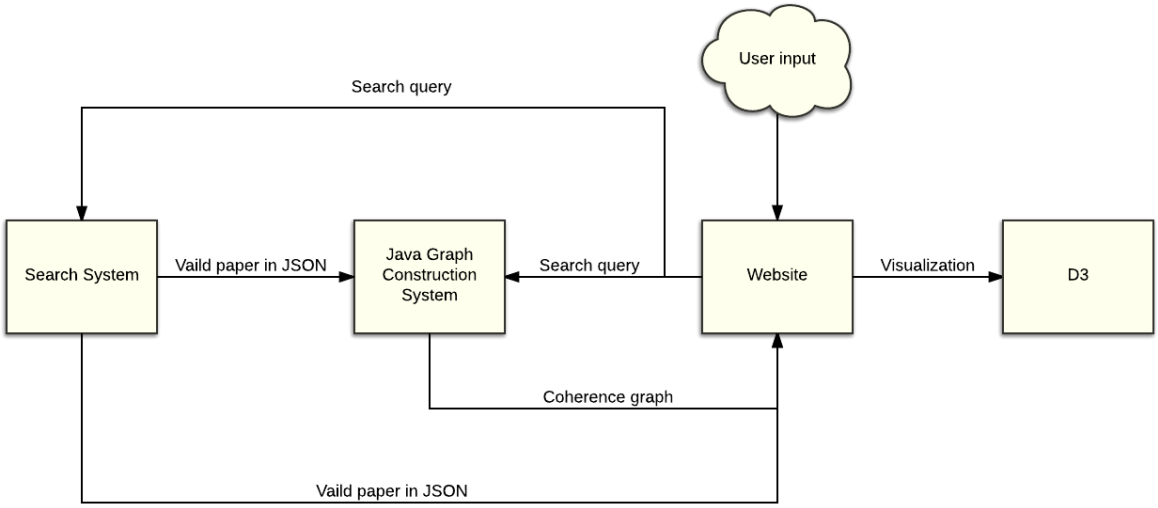


Figure 4.1: Implementation Overview

The aim of this project is to build a web application which can show the storylines graph on the given search query inputed by users. The procedures of the application are shown in figure 4.1. Firstly, the website will get the search query from users and using HTTP Request to get valid paper data from the search system. This data in JSON format contains papers have semantic similarity of the query. All papers will be showed on website ordered by score of relevant. At the same time, the web application also sent HTTP Request to the Java Graph Construction System. The Java program will be called to generate coherence graph of the vaild papers and this procedure may cost several minutes. Then coherence graph will be returned to web application with the content of nodes and edges. Finally, the graph will be visualized by D3.js. In this chapter, the implementation will be divided into 4 sections as following:

- Search System : Return valid papers based on search query
- Java Graph Construction system: Using data of valid papers to construct coherence graph which shows the storylines of the search query
- Java Restful API : Call Java program and return the coherence graph to web application

- Web application : Get search query and show storylines graph to users

4.2 Search System

4.2.1 Overview

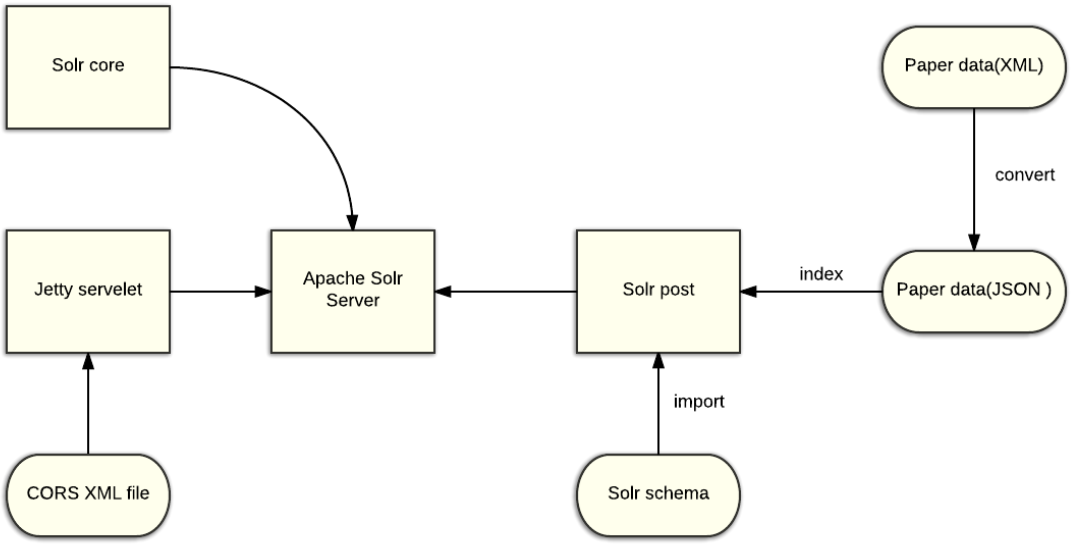


Figure 4.2: Solr Overview

Programmer uses Apache Solr to build search system. Construction of Solr API server need to be accomplished by three steps: index documents, configure schema, configure jetty server. As the figure 4.2 shows, the documents will be got by processing data of paper in XML to JSON. Then it need to be upload to Solr Server by Solr POST commands. The schema also need to be set and post to Solr Server. Finally, the Jetty server need be configured and the cross domain problem need to be fixed.

4.2.2 Pre-process

In last year, the data of ACL Anthology(Association for Computational Linguistics) was crawled and processed to XML by Lingyu Lu [5]. The XML files contains 30501 articles on ACL and three algorithms are used to parse the paper as table 4.3 shows. The three algorithm is implemented to label all section, abstract and citation respectively.

Get information from XML

In this step, Java DOM Parser will be used to get required information which is highlight in table 4.3. The result in parsed XML is not guaranteed, which may causes the information missing in some algorithms. Choosing information appear in different algorithms again will improve the quality of the data, such as title and author will be chosen from either SectLabel or ParsHed.

Get ID, URL and Date

Each paper contains a unique paper ID and it was stored in data's file name. After the traversal of file path, all papers' ID can be got by using recursive function. Some other important

SectLabel	ParsHed	ParsCit
title	title	citation title
authors	authors	citation authors
content	abstract	citation date
email	email	citation journal
reference content		citation raw string
affiliation		citation marker
		citation book title

Table 4.1: Table of XML

information of the crawled papers are stored in the database of Lu's[5] work. Experimenter installs MySQL[1] and imports the academic.sql file which contains ID and URL of each paper. The URL of given paper can be selected refer to the papers' ID which is the primary key of the table. JDBC[2] is used to connect the Java program and MySQL database. After that, the URL of each paper can be got in the Java program.

URL	Numbers	Date
http://www.lrec-conf.org/proceedings/lrec2012/pdf/649_Paper.pdf	2012649	20120649
http://www.aclweb.org/anthology/D/D09/D09-1049.pdf	091049	19911049
http://www.aclweb.org/anthology/A/A92/A92-1047.pdf	921047	19921047
http://www.lrec-conf.org/proceedings/lrec2014/pdf/425_Paper.pdf	2014425	20140425

Table 4.2: Extract date from URL

The date of each paper is not contained in the XML files. However, it can be extracted from URL. As table 4.2 shows, the data extracted is consisted of two part. The first part represents the year of publication and the second is its publish number. The completion of them will be conducted by Java program and output date of 8 numbers.

Construct JSON file

Java JSON Object and Java JSON Array are used to construct JSON file. The JSON Object contains following field as table 4.3 shows.

4.2.3 Schema

The schema file of Solr are used to decide which fields could be contained and define their properties. All fields such as title, date and authors should be configured before adding documents to the index. The properties of each field is similar to JSON files shown in table 4.3.

[1]MySQL: an open-source relational database management system

[2]JDBC: Java Database Connectivity is an application programming interface (API) for the programming language Java, which defines how a client may access a database.

Name	Properties	Description
id	integer	identification of the paper
title	string	title of the paper
authors []	array of string	array of authors
url	string	the url of the paper
date	string	year of public and the number(20090011)
content	string	content of the paper
paper_abstract	string	abstract of the paper
ref_title []	array of string	array of references

Table 4.3: Table of JSON

4.2.4 Commands and API

The Solr contains commands and API for GET, POST, UPDATE and DELETE. POST command are used in this project to upload local data to search engine. Developers also can upload paper by POST API with HTTP Request or Postman[1]. ALL commands and API are listed in appendix A.

4.2.5 Search procedure

After the Solr API server is constructed, search API can be used to get required papers in JSON format. As figure 4.3 shows, the search system return with search query "computer science". The search API also support functions to set the number of paper returned and the which field is required.


```
{
  "responseHeader":{
    "zkConnected":true,
    "status":0,
    "QTime":28,
    "params":{
      "q":"computer science",
      "indent":"on",
      "wt":"json",
      "_":"1473349061422"}},
  "response":{"numFound":8141,"start":0,"maxScore":6.1201153,"docs":[
      {
        "date":[20080204],
        "ref_title":["Computing Curricula 2001: Computer Science Volume. Association for Computing Machinery.",
          "Building a search engine to drive problem-based learning.",
          "Natural Language Processing in Python.",
          "Multidisciplinary instruction with the Natural Language Toolkit.",
          "Not seeing the forest for the trees: novice programmers and the SOLO taxonomy.",
          "A multi-national, multiinstitutional study of assessment of programming skills of first-year CS stude
          "The ability to articulate strategy as a predictor of programming skill.",
          "Language Files: Materials for an Introduction to Language and Linguistics. Ohio State",
          "An ontology-based knowledge portal for language technology.",
          "English Language: Victorian Certicate of Education Study Design. Victorian Curriculum and Assessment
        "paper_abstract":["Discourse in and about computational linguistics depends on a shared body of knowledg
        "author":["Steven Bird"],
        "id":"6aec1a98-e0bc-3ee5-8c2e-d54d12acf0e5",
        "title":["\nDefining a Core Body of Knowledge for the\nIntroductory Computational Linguistics Curriculum
        "url":["http://www.aclweb.org/anthology/W/W08/W08-0204.pdf"],
```


Figure 4.3: Search returned from Solr

[1]Postman: A software to help users be more efficient while working with APIs.

4.2.6 Cross Domain Problem

Browsers restrict cross-origin HTTP requests initiated from within scripts. For example, the web application running on http://localhost:8986 can not make HTTP requests to the search system running on http://localhost:8989. In order to solve this problem, Cross-origin resource sharing (CORS) will be applied, which is a mechanism that allows restricted resources on a web page to be requested from another domain outside the domain from which the resource originated[21]. After edit the web configure file of Jetty server and add lastest Jetty servelets and util jar files, our web application can make make HTTP requests to Solr search server[22].

4.3 Java Graph Construction System

4.3.1 Overview

In this section, the coherence graph of given query will be constructed by Java program. Firstly, the Java program will using HTTP Request to get the data of valid papers from search system and stores it in a JSON object. Then, each paper will be extracted to generate the vertex of the graph. After that, the citation relations of these papers will be analysed to construct citation graph. Finally, the coherence graph is constructed by using divide-and-conquer algorithm and will be output as JSON file.

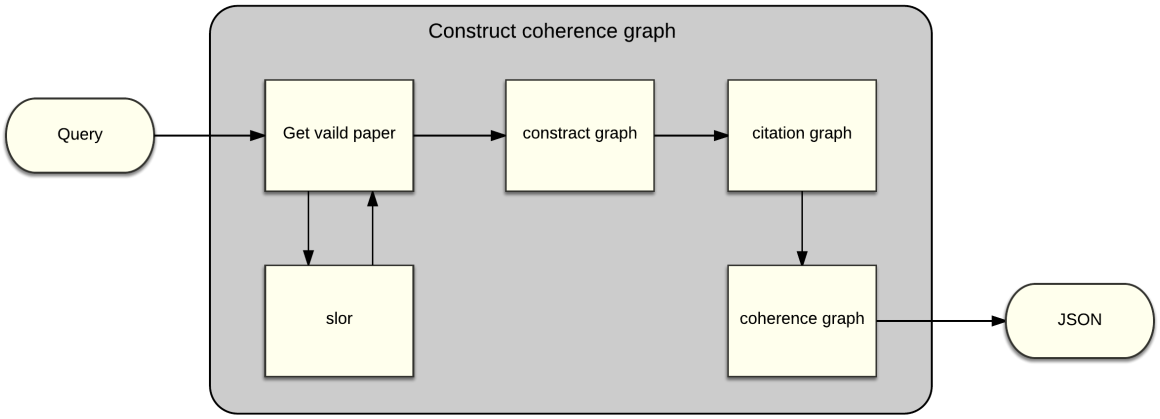


Figure 4.4: Overview of constructing coherence graph

4.3.2 Construct Graph

Graph theory

The graph G = (V, E) contains a set of vertices and edges. Each vertex represents a paper and each edge represents a relationship between two papers. In this graph, the edges have orientations and they are stored in the LinkedList which should be attributes of their vertices.

Construct graph in Java

A graph class needs to be created to contain Vertex and Edge showed table4.4 and table4.4. Each paper will be stored in one vertex and its information will be stored into the attribute

of vertex as a JSON object. The vvalue of each vertex is used for identification. The evalue of each edge represents the vertex it points to.

Name	Type	Description
value	object	the information of the paper
vvalue	int	identification
list	LinkedList<Edge>	store edges of this vertex

Table 4.4: Table of Vertex

Name	Type	Description
evalue	int	the vertex of this edge points to
weight	double	the weight of the edge
label	string	the label of the edge

Table 4.5: Table of Edge

4.3.3 Construct Citation Graph

Advantages of citation graph

"For a given collection of articles, ideas travel from cited papers to citing papers and from earlier to subsequent papers by same author. "Citation graph contains the influences between any pair of paper. The influence was calculated by citation and same author.

Citation label edges and same author label edges

In last procedure, we got a graph of papers without edges between them. The citation label edges will be added when the title of one paper is same as the reference title of another paper. The direction of this edge is from citing paper to cited paper and its weight depend on the score of paper cited. The procedure of adding same author label edges is similar. Same author label edge will be added if two papers have at least one same author. The direction of same author edge is from earlier paper to latest and the weight this edge depends on the score of paper published earlier.

Word segmentation

This procedure is same as the tokenization of information retrival. The titles and reference title of papers must be converted into array of words to compare. After all punctuations are replaced to spaces, the string can be split to array of words by spaces. The pseudocode of this function is shown as fellow.

Compare Two String

The data in parsed XML is not guaranteed, which will result in errors between two same title. The comparison algorithm is used to avoid this error which was shown in Algorithm

Algorithm 1 wordSplit

```
function WORDSPLIT(StringA)                    ▷ Where A - String of a sentence
    create B - an array of String
    convert all letters in A to lower case
    replaced all punctuation in A to white space
    split A into array of words b by space and store in B
     return B
end function
```

2. Firstly, the titles used to compare will be split into words. The similarity of titles will be calculated by checking the words belong to them one by one. The similarity of the two titles will be returned as a number between 0 to 100 and Sum of their length times checking will be conducted at most. We can assume two titles are same when the score of returned is above threshold.

This algorithm is tested for a lot of titles in our database. Some different titles are assumed as same when the threshold is small than 90 while some same titles missed when the threshold is bigger than 90. It is suggested that set threshold to 90 can get the most accurate result.

Algorithm 2 Compare

```
function COMPARE(String s1, String s2)▷ Where s1,s2 - String of a sentences to compare
    String[] a=wordSplit(s1)
    String[] b=wordSplit(s2)
    sort a and b
    s=a.length + b.length, z=0,x=0,y=0
    while a[x] and b[y] are vaild do
        if a[x]===b[y] then
            z++,x++,y++
        else if a[x]<b[y] then
            x++
        else
            y++
        end if
    end while
     return z/s * 200
end function
```

Normalise

As the formula shown below, Z is the normalisation constant and l is the number of paper from same author which published earlier. The contribution of all prior work of authors is equivalent to that of the the citation relation that is why we need a l for common author[23].

$$influence(c1 \rightarrow x) = \frac{a}{Z} \tag{4.1}$$

$$influence(a1 \rightarrow x) = \frac{c}{Z \times l} \tag{4.2}$$

$$Z = \sum_i c + \frac{1}{l} \sum_j a \tag{4.3}$$

As figure 4.5 shows, paper c1 and c2 are cited by x. Paper a1, a2 and x have same author ,and both a1 and a2 are published earlier. So weight of c1 to x equals to "a" divide by Z and weigth of a1 to x equals to "c" divide by 2Z. The normalisation constant of this example is equals to a+b+(c+d)/2.

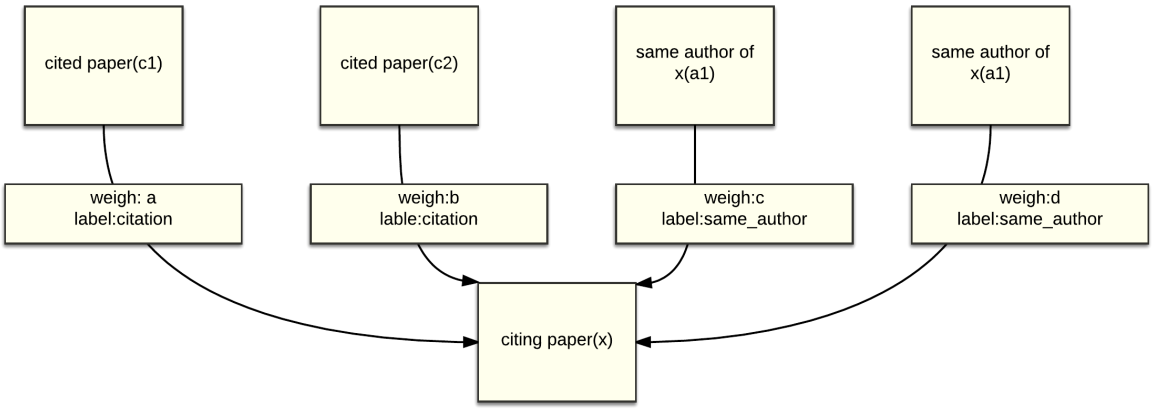


Figure 4.5: Normalize

Ancestral label edges

Direct influence cannot fully account for the coherence among papers. Two papers cite the same article but there is no citation or common authorship relation between them. In order to consider this influence, ancestral relationship is used [14].

Monte Carlo simulation will be used to calculate the influence of every two vertex of the graph in ancestral relationship.[23] In each simulation, a new random sub-graph will generated. Its' procedure is shown in Algorithm 3. The weight of edges in original graph will be converted into the probability of the edge appearing in the new graph. After that, the procedure of Monte Carlo simulation will be used as Algorithm 4 shows. In order to include the cases of 2 papers connected directly, the common ancestor can also be one of them. So firstly, the directly edges of the given two papers will be checked. Then, N times of simulation will be conducted and the the number of times they have same ancestral will be recorded. Finally, the probability of the two papers contains same ancestral in simulation will be returned as the weight of ancestral label edges.

4.3.4 Construct Coherence Graph

In order to pick good chains, all possible candidates needs to be listed. However, representing all chains whose coherence is at least $\mathcal{T}$ is a non-trivial task. The number of possible chains may be exponential, and therefore it is infeasible to enumerate all of them.

This problem can be solved by divide-and-conquer algorithm [14], which constructs long chains from shorter ones. After that, the new graph will be constructed, whose vertex represents a short chain. Finally, the edges of the graph will be added which indicate chains which can be concatenated and still maintain coherence.

Algorithm 3 Generate Random Sub-graph

```
function GENERATERANDOMSUBGRAPH(myGraph Gc)        ▷ Where Gc - Graph object
    myGraph Grc=new myGraph()
    add all vertices of Gc to Grc
    for edge in Gc do
        generate random number x in[0,1]
        if weigh of edge > x then
            dd edge to Grc
        end if
    end for
     return Grc
end function
```

Algorithm 4 Monte Carlo Simulation

```
function MONTECARLOSIMULATION(myGraph Gc, Vertex pi, Vertex pj, int N)   ▷ Where
Gc - Graph object pi,pj - vertex of the Graph, N - number of samples
    k=0,count=0
    for edge in pi do
        if the edge points to pj then return the weigh of pi− >pj
        end if
    end for
    for edge in pj do
        if the edge points to pi then return the weigh of pj− >pi
        end if
    end for
    while k < N do
        Grc=GenerateRandomSubgraph(Gc);
        if exist x− >pi and x− >pj then
            count++
        end if
        k++
    end while
     return count/N
end function
```

Preparation

The class of chain which can contain a list of papers should be created. The vertices of graph can be stored into an Arraylist of this class and a coherence calculator is embed. The coherence of chain depends on similarity between each pair of consecutive articles along the chain. Since a single poor transition can destroy the coherence of the entire chain, the strength of chain should depends on its weakest link.[14]

Get All Valid Pairs

The first thing of divide-and-conquer algorithm is to get short chains. Each two vertices of the graph can be collected to combine a chain if there is an edge between them. The order of the papers in the chains should be same as the influence direction of the edges. After filters the chains whose coherence bellow threshold, all the pairs of paper stored in chains will be put into a priority queue.

Generate Extensions of Short Chains

Algorithm 5 generateExtensions

```
function GENERATEEXTENSIONS(chain c1)                ▷ Where c1 - chain object
    for current_chain in pq do                       ▷ Where pq - priority queue
        if tail of c1 == head of current_chain then
            extension_chain= c1 + tail of current_chain
            if length < sub_length_threshold then
                add extension_chain to pq
            end if
        end if
        if head of c1 == tail of current_chain then
            extension_chain= head of current_chain + c1
            if length < sub_length_threshold then
                add extension_chain to pq
            end if
        end if
    end for
end function
```

Secondly, the chains which is shorter than sub_length threshold should be extended. As Algorithm 5 shows, programmer uses the given chain and all valid chains in priority queue to assemble extension chains whose length meet the requirement. At last, these extension chains will be added to priority queue.

Generate Coherence Graph

Finally, all required chains which should be stored as vertices of the new graph can be collected from the priority queue. The comparator of the priority queue are also created to compare the chains depends on coherence value. While the priority queue is not empty, the chain with highest coherence value will be polled. If this chain meet the length of requirement, it will be treat as vertex and added to our new graph. If its length smaller than the

Algorithm 6 Generate Coherence Graph

```
function GENERATE_COHERENCE_GRAPH(Priorityqueue pq)   ▷ Where pq - priority queue contains all pairs
    initial new_graph
    while !pq.isEmpty() do
        highestChain = pq.poll();
        if highestChain.length== sub_length_threshold then
            add highestChain as vertex to the new_graph
        else
            generateExtensions(highestChain)
        end if
    end while
    for v_i in new_graph do
        for v_j in new_graph do
            if v_i==v_j then
                continue
            end if
            if v_i.sublist(1,sub_length_threshold)==v_j.sublist(0,sub_length_threshold-1) then
                add edge j->i
            end if
        end for
    end for
    return new_graph
end function
```

requirement, the extensions of this chain will be generated and added to priority queue.

After all vertices are added, the edges which represent the connection of chains should be generated. The simple way of connect two chains is check the shared papers belong to them. Sometimes two strong chains may result in a much weaker chain. when we concatenate (d1, ..., dk) with (dk, ..., d2k-1), there is no evidence that both chains belong to the same storyline, despite having a shared paper dk. However, (d2, ..., dk+1) may not be, whose first k papers are coherent.

4.4 Concurrency

4.4.1 Overview

The Java program of constructing coherence graph cost too much time. Concurrency programing will be used to improve the speed of process. Java high-level concurrency APIs in the java.util.concurrent packages will be used to implement concurrency to this project. synchronous concurrency programming and asynchronous concurrency programming will be applied to procedures of constructing graph, constructing citation graph and Monte Carlo Simulation.

4.4.2 Concurrency of Monte Carlo Simulation

Firstly, the work needs be divided for different threads to execute. In single thread mode, Monte Carlo Simulation will be executed N times and return the value of Z/N which represent the probability of ancestral. After applied to concurrency mode, The N times of simulation works can be allocated to several threads. Then the return value of each threads should be added by using the concurrency interface Future. Finally, the sum of each threads should be divided by the number of threads. The detail procedures are showed in figure 4.6.

4.4.3 Concurrency of Constructing Graph

Concurrency procedure of constructing graph is similar. The only difference is the same graph will be used for different treads. The information of papers will be divided and added to graph. The identification number of paper will be different for each construction, because the the order of adding procedure by multiple threads is random.

4.4.4 Concurrency of Constructing Citation Graph

The graph will be divided into several part to construct its citation graph separately. However the whole graph will be transmitted as parameters to each part, since the reference title of the given paper need be compared to all titles of the graph.

4.4.5 Asynchronous of concurrency

The asynchronous concurrency can be applied by using ExecutorCompletionService interface. The task of each thread will be summed immediately when it was finished instead of sum of all threads' value when all of them are completed.

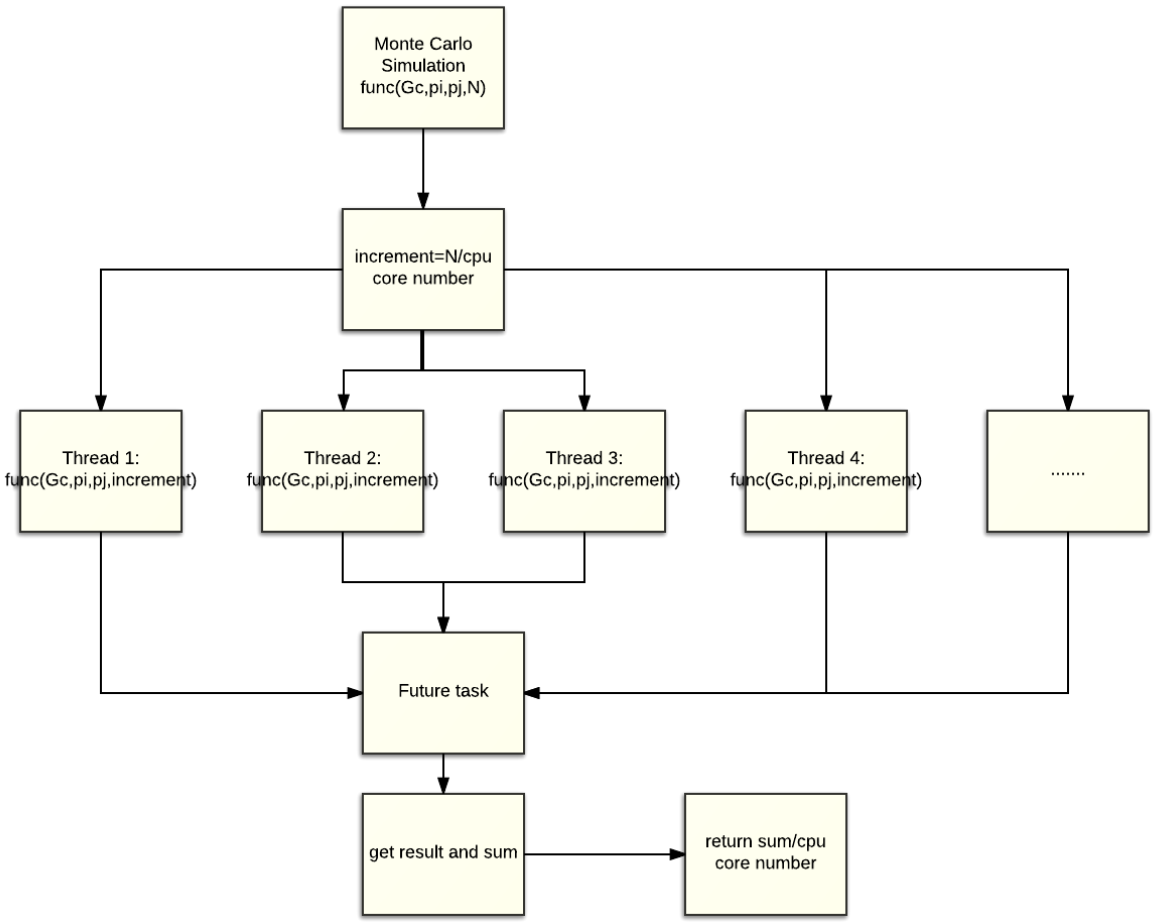


Figure 4.6: Concurrency of Mento Carlo Simulation

4.5 Java Restful API

4.5.1 Spark

Spark framework was used to construct the Java Restful API server of this project. Programmer using Maven to set up the environment of Spark[24]. One Get route was created for get search query and parameters of coherence threshold, sub_chain length and number of nodes. The search query will be split and reconfigured by Java program. All information got will be transmit as parameters to our storyline program. Eventually the coherence graph will be returned output as a JSON file.

4.5.2 CORS

CORS also need be Implemented in Spark to make it can be accessed by our web application. The requests will be completed by browser when the server allows it via the specific headers in the request and response of each cross domain call[25].

4.6 Web application

4.6.1 Overview

The React introduced in background chapter will be used to build our web application. Its applications are built by components and it use JSX Syntax to embed HTML into JavaScript. Some components packaged by Ant Design will be used to compose the user interface of our website. Finally, programmer will use D3.js to visualise the coherence graph. The architecture of file in web application is shown in figure 4.7. The router is composed by search and graph components and each of them contains the main layout. Three ant design components are used to build search page and D3.js is used in graph page.

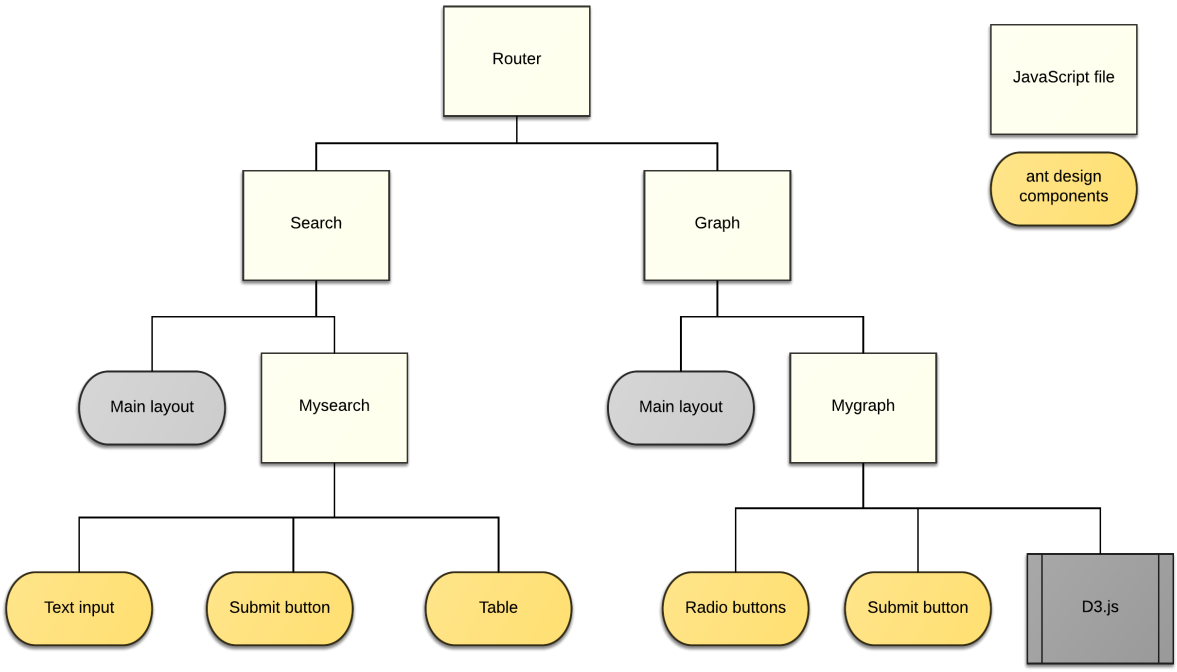


Figure 4.7: Architecture of file in web application

4.6.2 Main layout

The main layout will be used for each page of the website. It uses Sidebar navigation to switch between pages. The main layout also contains an area for content which should be transmitted by child components of Search and Graph.

4.6.3 Ant design components

Mysearch

The text input component will be used to collect users' search query. After using the callback function of Mysearch, the search query will be sent to Mysearch components. When the submit button is pressed, an HTTP Request will be sent to Solr and the data in JSON will be collected by Mysearch components. Then the data will be transmitted to Table components as its props. Finally, the Table components will receive the data and show the information of papers. The papers will be order by score which is return by search system. Only ten

papers will be rendered in one page by a paging component, which can avoid information overloading.

Mygraph

Programmer adds some radio buttons to got parameters of graph from users. The data transmitted in this part is similar as Mysearch. The only difference is the HTTP Request will be sent to Java Restul API server to get the coherence graph in JSON. D3.js will collects the data and achieves the visualization.

4.6.4 D3.js

Firstly, programmer uses d3.select() function to select element in HTML. Secondly, an area of SVG image is appended to the element. Then, D3's selection.data() method is used to bind the data of papers to DOM elements. After that, scale and axis of the graph are created. Finally, the visualization of nodes and edges are achieved by apply Force-Directed Graph.

Scales

The values in any data set are unlikely to correspond exactly to pixel measurements for use in visualization. Scales provide a convenient way to map those data values to new values useful for visualization purposes. Programmer uses linear scale which can returns a scaled output value by passing data value. Once it is created, it can be called anywhere. The input domain is set from smallest date to biggest date of the coherence graph. The output range is set from 0 to 1000.

Axis

The axis element does not contained in SVG and it is complex to using other SVG elements to assemble the axis. However, D3 provide an components d3.svg.axis() which can achieve the work of this section. Experimenter should set the orientation and the number of ticks of the components. The scales defined in last section also should be added to axis. Finally, the axis of date will be showed at the bottom of SVG graph.

Force-Directed Graph

Force-Directed Graph algorithm will be used in this project to draw the coherence graph. The Force-Directed Graph can arrange Nodes and Edges in two-dimension by calculating the force based on relative position of nodes and its connected edges. The edges of the graph will not intersect as far as possible. The trajectory of nodes and edges will be calculated to reduce the their energy continually. Finally, the graph will reach a stable low energy state.

After applied Force-Directed Graph, the nodes of coherence graph will be placed in random position. However, as mentioned in Design chapter, the nodes should be placed from earlier date to latest date in x-dimension. Therefore, the x position of each node should be update by the value of its date returned from our linear scale.

The mouseover event will be added to each node to show the title of its paper. The mouseout event will be added to each node to remove the title by set the text style to fill-opacity. The mouse click event will be added to each node to show the titles of all papers. The double click event will be added to each node to open a new window of the paper's detail information.

The nodes of graph may overlap in some special situation. In order to avoid this, the tick function of Force-Directed Graph will be used to make the position of nodes can be moved by users. The position of all nodes and edges need be update when one of nodes is moved according to force calculation.

Chapter 5

Results and Discussion

5.1 Overview

In this chapter, the result produced by the project will be elucidate and discussed. In the beginning, the whole web application will be presented. Then, the result of search and coherence graph will be analysed. Since there is no standard criterion for storyline, an example of experiment result will be explained in detail and the parameters will be adjusted and analysed with real situation. At the end, the time cost by the application will be estimated. In an attempt to improve the speed of the program, some strategies of concurrency will be explored and evaluated.

5.2 Website

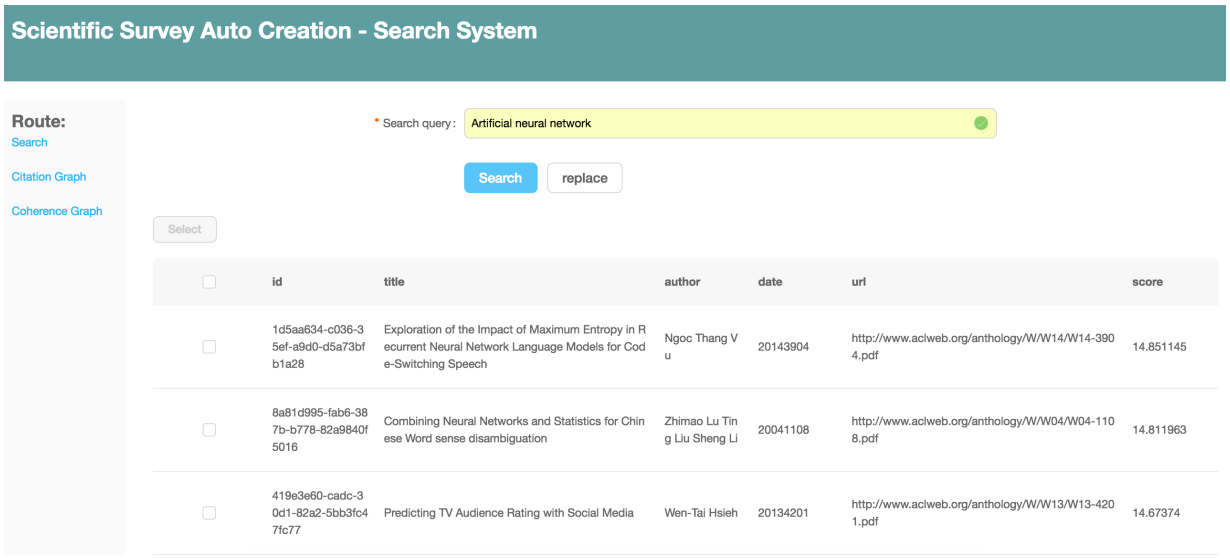


Figure 5.1: Website of Search

The homepage of our website is constituted by a sidebar navigation and a content area. The sidebar navigation can switch between pages and the content area displays the result of search and graph. As figure 5.1 shows, users can find valid papers by entering search query. The detailed information of papers is presented in the table order by score. The custom function is to select part of papers to generate graph which is not implemented and will be described in future work.

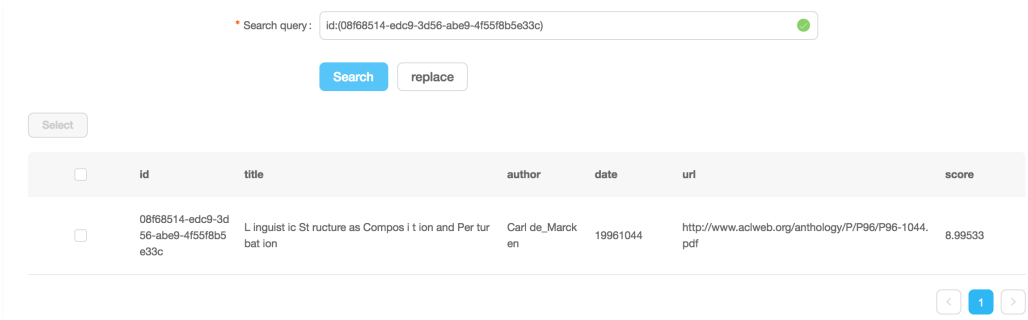


Figure 5.2: Search by ID

The system also provides functions to search specific papers. It supports fields of id, author, title, date and reference title. An search example is illustrated in figure 5.2, which searches detailed information of the given id.

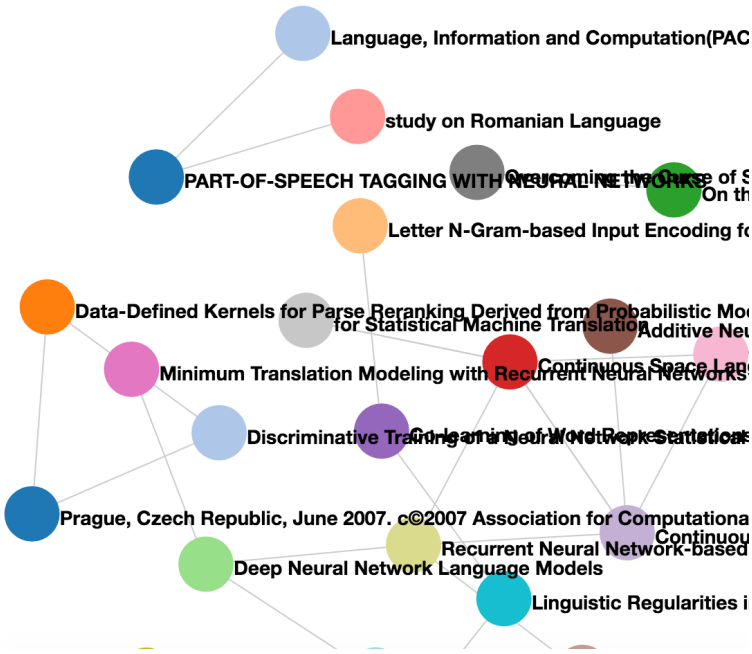


Figure 5.3: Simple citation graph generated by JavaScript

After obtained valid papers, the simple citation graph can be generated by navigating to citation graph page as figure 5.3 shows. Programmer used JavaScript instead of using self-built Java API server to generate previously mentioned citation graph, since that graph is simply used for adjusting the parameters of coherence graph. The whole procedure only costs several seconds and the result is not readable. Users need to employ generated citation graph to check whether there are enough citation relations to construct the coherence graph, otherwise the numbers of nodes should be added to get a more satisfy result.

Eventually, the website can be navigated to coherence graph page. There are three parameters could be adjusted to gain a new graph with various complexity. In most cases, the coherence graph is created in default setting displayed in figure 5.4. Users also could generate a better coherence graph or featured storyline by adjusting different parameters and detailed information will be explained in section 5.4.1 Parameters Discussion.

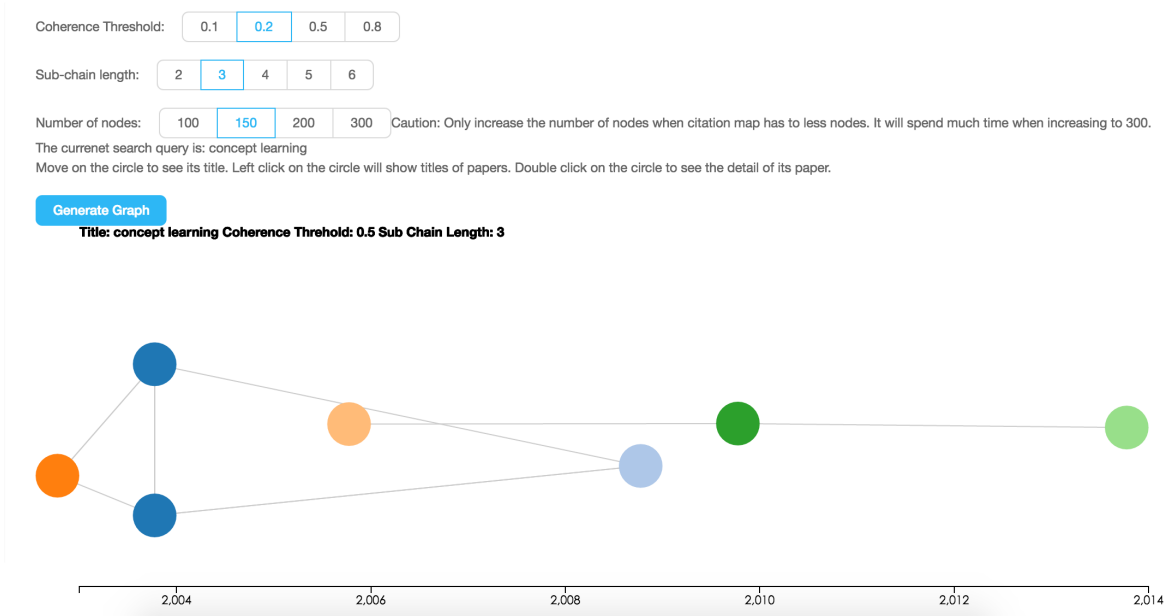


Figure 5.4: Coherence Graph

5.3 Result of Search

To check the search result, an instance is shown in figure 5.5. Satisfied papers are ordered by score with the search query "wireless sensor network"(WSN). Experimenters did some research under the topic, which is using spatially distributed sensors to monitor physical or environmental conditions. The topic also involves adopting wireless devices of transmitting and has development tendency in body sensors, Wearable Devices and Smart Home[26]. The result shows that most papers are relevant to the topic. The article which gains the highest score talking about using low-power Wireless Acoustic Sensor Network (WASN) for the observation of clinically relevant activities of daily living from elderly[27]. The second highest score one also directly involve to sensor network and the rest papers mention body area network, sensor of Human-Computer and monitoring platform which are relevant to WSN as well. The search results are partly concordant with the results in other scholarly research database.

5.4 Result of Coherence Graph

This section will implement some pilot study on generating coherence graph with different parameters. The influence of nodes numbers, coherence threshold and sub-chain length will be analysed and the adjusting procedure of them will be evaluated. To determine the correctness of coherence graph, the query result of "Artificial neural network" is estimated.

5.4.1 Parameters Discussion

The complexity of coherence graph generated under various topic exists obvious difference. The citation relationships of some search query are complex, causing the coherence graph contains too many nodes and edges. In order to generate proper coherence graph, parameters of the graph need be justified. Tests are running on different search query with various parameters to obtain the most appropriate value. The default setting of each parameter is showed in table 5.1. All of them perform well in the test of this section.

id	title	author	date	url	score
0ab49c8c-c49b-306c-bc76-6d56e058d812	Automatic Monitoring of Activities of Daily Living based on Real-life Acoustic	Peter Karsmakers	20133918	http://www.aclweb.org/anthology/W/W13/W13-3918.pdf	13.508835
88570ed0-cc8e-3c5e-87fb-5c779d9a102d	InproTKS: A Toolkit for Incremental Situated Processing	Casey Kennington	20144312	http://www.aclweb.org/anthology/W/W14/W14-4312.pdf	12.974485
056e005f-7e54-3844-9ab2-eecb5f5b9955	Collaborative Exploration in Human-Robot Teams: What's in Their Corpora of Dialog, Video, & LIDAR Messages?	Clare R Voss∗ Taylor Cassidy †∗ Douglas Summers-Stay∗	20140207	http://www.aclweb.org/anthology/W/W14/W14-0207.pdf	12.418663
99c14e91-4c98-3c32-8c46-483748d1e61b	Joint Navigation in Commander/Robot Teams: Dialog & Task Performance When Vision is Bandwidth-Limited	Douglas Summers-Stay	20145402	http://www.aclweb.org/anthology/W/W14/W14-5402.pdf	12.175178
2c948027-f0e8-3282-8f8d-f62b1bf66e87	Investigating speaker gaze and pointing behaviour in human-computer interaction with the mint.tools collection	Spyros Kousidis Casey Kennington David Schlangen	20134048	http://www.aclweb.org/anthology/W/W13/W13-4048.pdf	11.899412
9f3dff5d-c9ec-3948-8467-f1a63568849b	Towards NLG for Physiological Data Monitoring with Body Area Networks	Hadi Banaee	20132127	http://www.aclweb.org/anthology/W/W13/W13-2127.pdf	11.295275

Figure 5.5: The table of papers with search query "wireless sensor networks"

Parameter	Default value
Number of Nodes	100
Coherence Threshold	0.5
Sub-Chain Length	3
Numbers of Monte Carlo Simulation	100

Table 5.1: Default parameters of constructing coherence graph

Number of Nodes

The valid data obtained from search section may contains thousands of papers. There are 5981 papers is satisfied when searching "Artificial Neural Network". However, the score of papers ranked after 200 is much smaller than the score of papers in top 10. It means that the paper ranked after 200 has little influence in the citation map.

The complexity of the coherence graph is acceptable when choosing the first 100 nodes. As figure 5.6 shows, the coherence graph for the search query "Artificial Neural Network" is in proper sizes, with other settings in default. However, when the search query "concept learning" is tested, the coherence graph generated does not contains enough nodes to present the complexity even with the minimising of the parameters of coherence threshold and sub-chain length. The coherence graph of search query "concept learning" with 150 nodes is generated as figure 5.7 shows. The figure 5.8 is the one with 200 nodes which has more storylines. The graph may become too complex when adding more nodes. After testing all options, the graph with 200 nodes is more appropriate.

However,the time spend of generating coherence graph will increase exponentially when

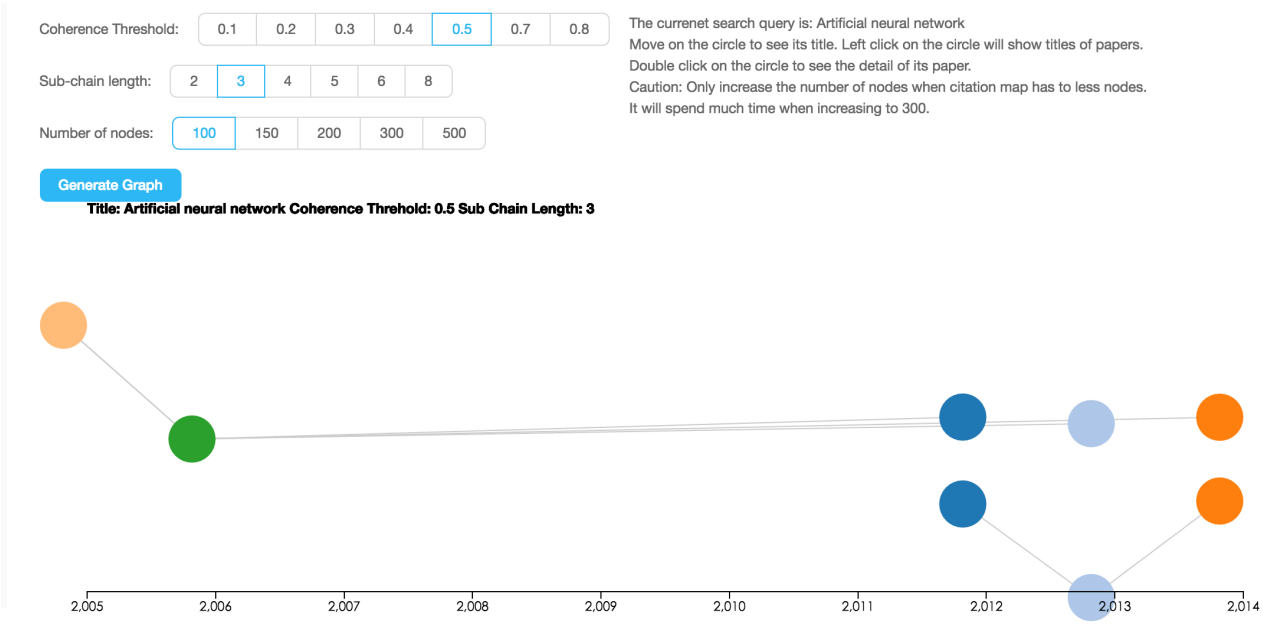


Figure 5.6: The coherence graph of search query "Artificial Neural Network" with default parameters

Figure 5.7: The coherence graph of search query "concept learning" with 150 nodes

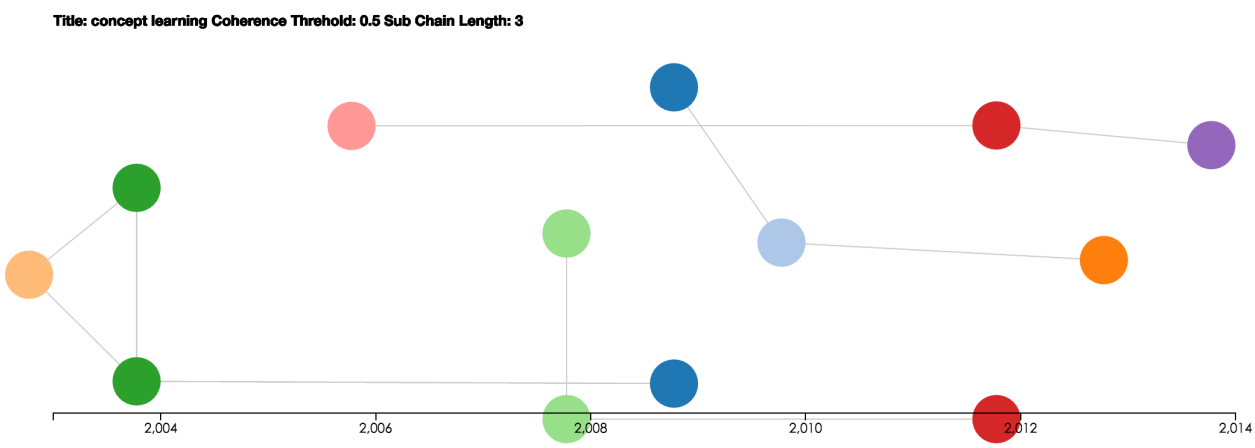


Figure 5.8: The coherence graph of search query "concept learning" with 200 nodes

number of nodes is added. Therefore, it is not convenient to test all options of the node parameter. The citation graph can be generated quickly and it indicates the complexity of coherence graph. After got enough nodes in citation graph, the coherence graph is in appro-

priate size in most case. The detailed information of time spending in different number of nodes will be mentioned in the section 5.5 speed of program.

Coherence Threshold

The coherence threshold is the minimum strength of connection of the storyline. Some storylines will be removed by enlarging this parameter. As figure 5.9 shows, the coherence graph for search query "Artificial Neural Network" with 0.7 coherence threshold is more simple than figure 5.10 whose coherence threshold is 0.4 .

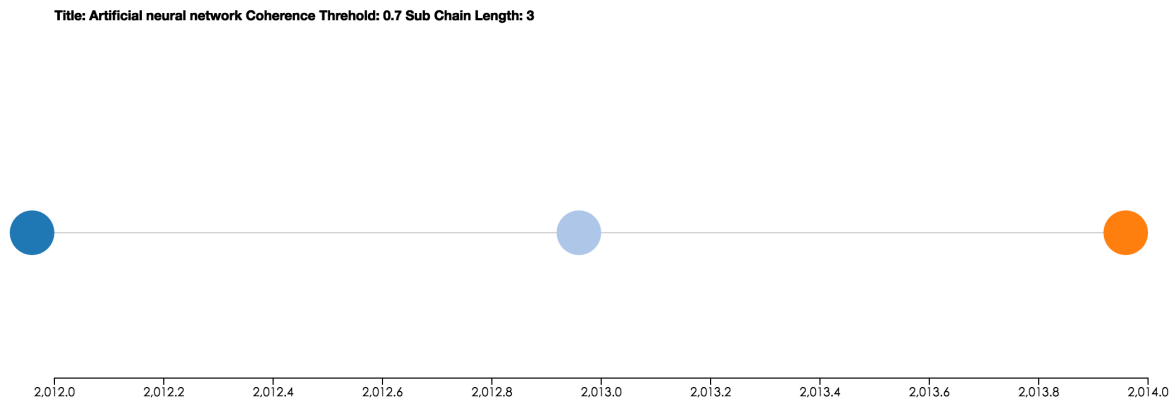


Figure 5.9: The coherence graph of search query "Artificial Neural Network" with Coherence Threshold 0.7

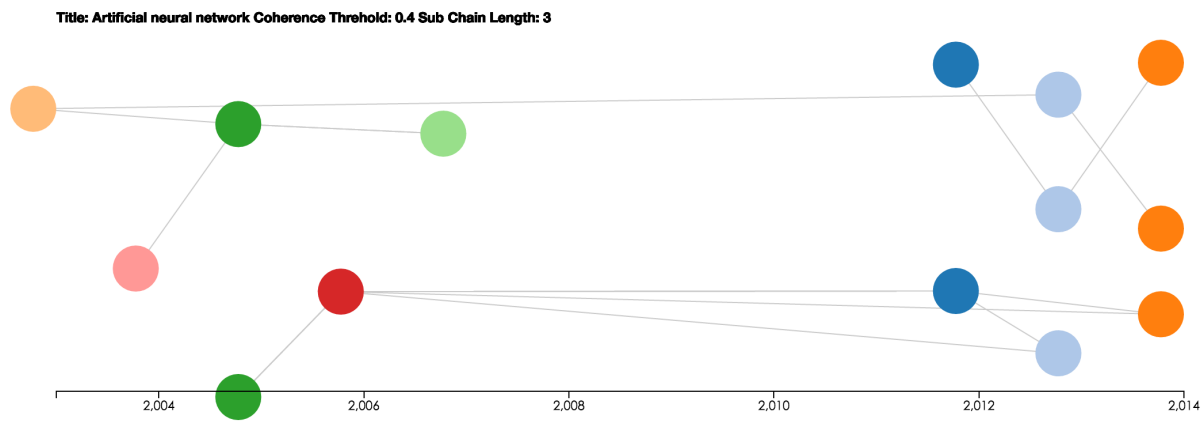


Figure 5.10: The coherence graph of search query "Artificial Neural Network" with with Coherence Threshold 0.4

Sub-Chain Length

Storyline in this project is generated by divide-and conquer algorithm. The coherence graph is composed of sub chains with the same length. After enlarging this parameter, some chains will be removed if they do not meet the length requirement. This means the storyline contains less papers will be eliminated easily. The figure 5.11 shows the coherence graph with default sub-chain length and the sub-chain length parameters is increased to 4 in figure 5.12 and 5 in figure 5.13. Both of them are simpler than figure default setting, then users can find the longer evolution storyline.

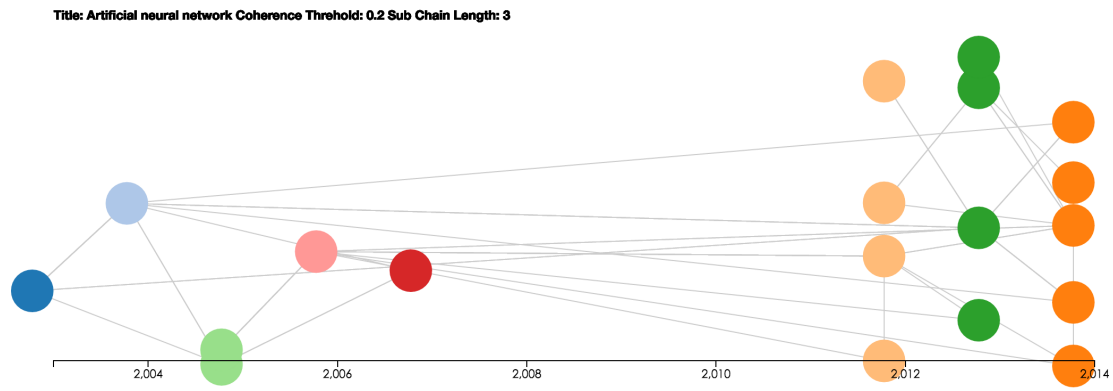


Figure 5.11: The coherence graph of search query "Artificial Neural Network" with coherence threshold 0.2 and sub-chain length 3

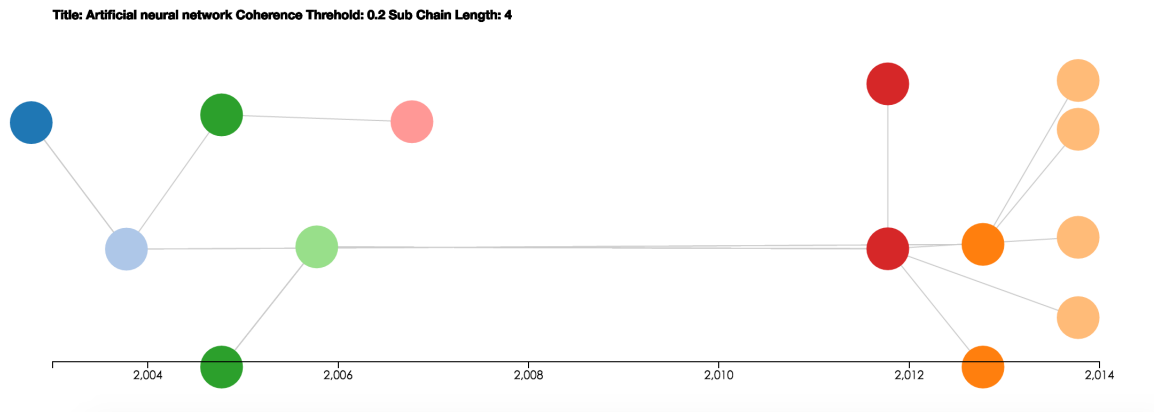


Figure 5.12: The coherence graph of search query "Artificial Neural Network" with coherence threshold 0.2 and sub-chain length 4

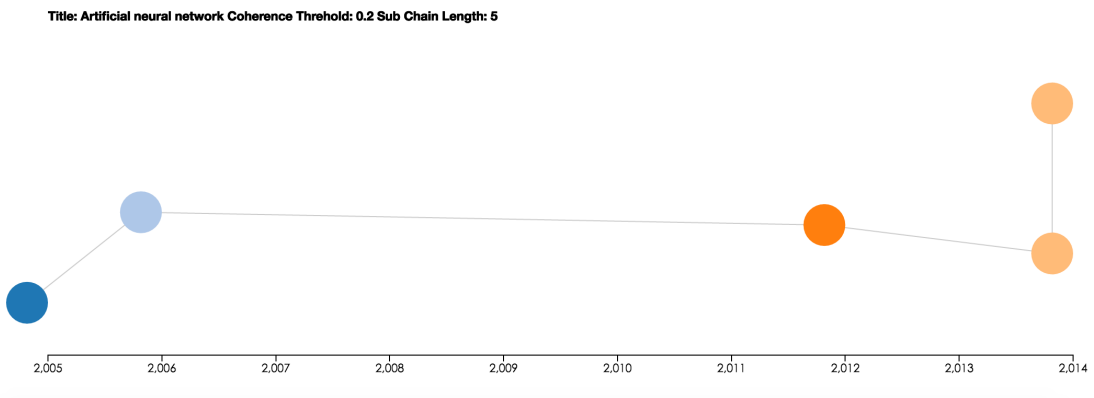


Figure 5.13: The coherence graph of search query "Artificial Neural Network" with coherence threshold 0.2 and sub-chain length 5

Testing and Summary

Some tests are conducted to check the complexity of the graph under different parameters. The number of chains is calculated to represent the complexity as table 5.2 shows. The number of chains will decrease and tend to stable when the numbers of Monte Carlo Simulation increasing. Small numbers of simulation may cause some edges with low weight appear in final graph, which cause the result inaccurate. However, when the times of simulation in-

creased to 100, the number of chain tends to stable, which mean the times of simulation is enough. The number of chains will increased by decreasing the coherence threshold, increasing the number of nodes and rising the length of sub-chain. To be conclude, there was a significant correlation between parameters and the complexity of the graph.

Number of chains

Nodes:100,sub-chains: 3,coherence threshold: 0.5		Nodes:100,sub-chains:3, Monte Carlo num:100		Monte Carlo num:100, Sub-chains:3,Coherence threshold: 0.5		Nodes:100,Monte Carlo num: 100, Coherence threshold: 0.5	
Number of Monte Carlo Simulation	number of chains	Coherence threshold	number of chains	number of nodes	number of chains	sub-chain length	number of chains
20	182	0.1	32	50	0	2	32
50	44	0.2	25	100	12	3	17
100	17	0.3	25	150	17	4	12
200	18	0.5	16	200	23	5	7
300	18	0.8	7	300	35	6	0

Table 5.2: The number of chains of search query “Artificial neural network”

5.4.2 Evaluation of an Example

The case experiment of search query “Artificial Neural Network” is used in this section, with 100 as node number, 3 as sub-chain length and 0.4 as coherence threshold. In this experiment, 8 chains are generated and 4 of them are showed in table 5.3 to be evaluated.

The first chain contains 3 articles and all of them are relevant to neural network. Other than the similarity in title, contents analysed of these paper also shows that they all related to subtopic of language processing. The paper 52 is cited by paper 79 while 79 and 62 have same ancestral. As mentioned in implementation chapter, 2 papers connected directly will lead to one of them become the same ancestor. This is showed in paper 52 and 79 which have both cite and ancestral label edges. There is no other connections of paper 62 in this table (still has connection to papers which are not contained in these 8 chains). Thus, the storyline is ended with length of three.

The chain 4 can be extended by chain 6 and 7 to different directions. Author concludes them into two storylines. The paper 46 and 54 have same author while paper 54 cited by 58. These three papers are talking about language processing and translation under neural network. The paper 58 and 59 have same ancestral, which lead the storyline to statistical machine translation area. The paper 58 and 72 have same ancestral and keep the storyline on language processing. Both of these two storylines is ended with length of four.

5.5 Speed of Program

This application spends a lot of time on processing the Java program which generates the coherence graph. The speed is influenced by number of Monte Carlo Simulation and number of nodes added. Concurrency strategy is used and will be evaluate in this section.

Search query: Artificial neural network , nodes: 100, Coherence Threshold: 0.4, Sub-chain length: 3

chain 0:	52	79	62	chain 6:	54	58	59
Title	Deep Neural Network Language Models	Adaptation Data Selection using Neural Language Models: \nExperiments in Machine Translation	Minimum Translation Modeling with Recurrent Neural Networks	Title	Continuous Space Language Models for Statistical Machine Translation	Continuous Space Translation Models for Phrase-Based Statistical Machine Translation	Additive Neural Networks for Statistical Machine Translation
Relation	79--1.0--cite	52--1.0--ancestral		Relation	58--0.933649--cite	59--0.4986903739688794--ancestral	58--0.4986903739688794--ancestral
Relation	79--1.0--ancestral	62--0.7--ancestral		Relation			
chain 4:	46	54	58	chain 7:	54	58	72
Title	Proceedings of Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing	Continuous Space Language Models for Statistical Machine Translation	Continuous Space Translation Models for Phrase-Based Statistical Machine Translation	Title	Continuous Space Language Models for Statistical Machine Translation	Continuous Space Translation Models for Phrase-Based Statistical Machine Translation	Transduction Recursive Auto-Associative Memory: Learning Bilingual Compositional Distributed Vector Representations of Inversion Transduction Grammars
Relation	54--0.5136025263300433--same_author	58--0.933649--cite	59--0.4986903739688794--ancestral	Relation	58--0.933649--cite	72--0.5238342321242233--ancestral	
Relation	54--0.5136025263300433--ancestral			Relation			

Table 5.3: Chains picked from storyline of Artificial neural network

5.5.1 Time of Each Part

The search query "Artificial Neural Network" with 100 nodes and 100 times of Monte Carlo Simulation is tested as figure 5.14 shows. Program spend most time in adding ancestral labels. The second part is adding citation and same author labels. After applying concurrency, the speed of these two parts are a little increased and they will be discussed in next section. The time spend on synchronous and the time spend on asynchronous is closely because the workload distributed to each thread is almost same. The threads of synchronous concurrency do not cost too much time on waiting other threads to be completed.

Search query: Artificial neural network , nodes: 100, Monte Carlo number: 100

Time(ms)	Single thread	Synchronous concurrency (4 threads)	Asynchronous concurrency (4 threads)
Construct graph	279	116	112
Add citation and same author label	2226	1603	1522
Normalize citation graph	9.0	7.0	7
Add ancestral label	4231	3892	3621
Construct coherence graph	18	22	19

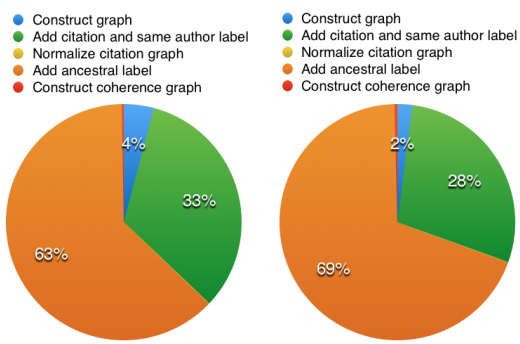


Figure 5.14: Time spends on single thread, synchronous concurrency and asynchronous concurrency

5.5.2 Time Spends on Adding Ancestral Labels

The test is conducted in a 4 core computer[1]. As figure 5.15 shows, the time spend on multiple threads is longer than single thread when the number of simulation is small. In theory, the works divided into 4 threads should be quicker than only using one thread. However, the multiple threads may cost more time on input(IO). The figure 5.15 also shows the time spend on multiple threads increase slower than the single thread. This may be caused by the percentage of time spend on IO decreased.

Monte Carlo Concurrency , Nodes: 100

Number of Monte Carlo Simulation	Single thread	Synchronous concurrency (4 threads)	Asynchronous concurrency (4 threads)
1	139	2761	2819
4	248	3099	2985
20	769	3600	3515
200	7310	9734	9925
500	18985	18177	18763
1000	36875	32208	33047
2000	83987	59954	60458
4000	154432	99534	95231
8000	287152	174712	165384

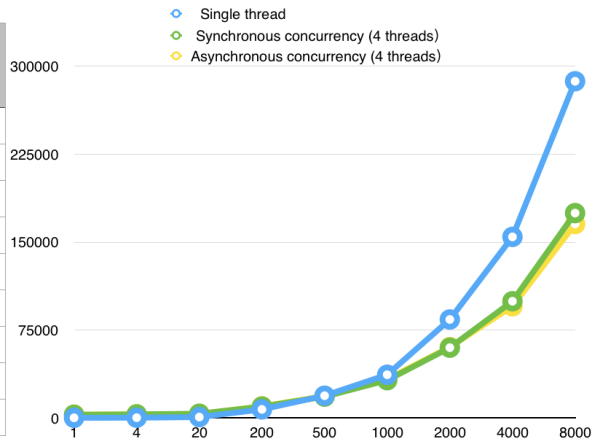


Figure 5.15: Time spend according to numbers of Monte Carlo Simulation

5.5.3 Time Spends on Adding Citation and Same Author Labels

The time spend on this section increases exponentially with the number of nodes added as figure 5.16 shows. The effective of time saving by concurrency strategy is more obvious when the number of nodes is larger.

Citation Graph Concurrency, Monte Carlo num: 100

Number of nodes	Single thread	Synchronous concurrency (4 threads)	Asynchronous concurrency (4 threads)
50	1005	1003	1087
100	2226	1794	1814
500	38465	16566	11856
1000	140883	62293	59541

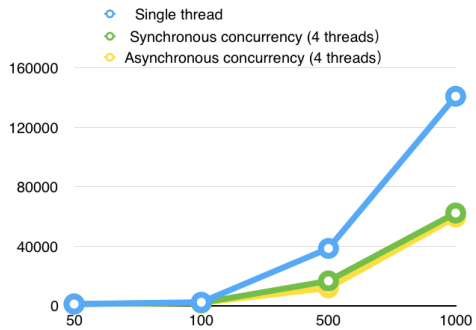


Figure 5.16: Time spends according to numbers of Nodes

[1]MacBook Pro (Retina, 13-inch, Early 2015)

Chapter 6

Conclusion and Future work

6.1 Limitation

Takes Too Long Time to Index

All of data are indexed to Solr using POST command. However, this command do not accept too much arguments when more than 200 papers are updated. The FIND command is used to post the papers separately, which needs consuming longer time. The indexing process of whole data costs several hours. The programmer needs to update the whole data when one field of schema is added. This make the design and debug more difficult.

Speed of Java Program is Slow

The Java program which constructs coherence graph needs to spend much time to get the result. Even concurrency strategy is applied to improve the speed of the program, the time consumed will be increased to dozens of minutes when lots of nodes are added to the graph. This limits the development of the graph to include more information. The web application is designed to feedback immediately, which needs the program be finished under seconds. However, when multiple users are accessed to the system, the waiting time for users will be increased apparently.

To Much Parameters of Graph Adjusted by Users

Users need adjust the number of nodes, coherence threshold and sub-chain length of the coherence graph to get right result. In some situation, the adjusting procedure is complicated and time-consuming, which will degrade the user experience.

Without Customized Settings

The papers used in coherence graph do not support customized settings. Users can not obtain paper in range of years or certain authors. The content type such as article, book, or conference publications also can not be chosen.

The Score of Paper only Influenced by Semantic Similarity

The score of paper only influenced by semantic similarity is not accurate. The paper which mentions the topic many times will get high mark in this algorithm. However, this paper should be less important if it does not have any citation relationship to other papers in this

topic. The influence of citation relation, which is essential, does not included in the procedure of calculation score.

Third-party Search Platform

This project uses Solr as search platform which is low cost, quick using and has developed open source community. However, it limited the development of this project. All operations need to be based on the API of Solr and its' customized settings. The intermediate process and result of information retrieval can not be accessed.

6.2 Future work

Use Distributed System to Speed Up

Apache Hadoop is one software framework for distributed storage and distributed processing, which was consisted by Hadoop Distributed File System (HDFS) and MapReduce processing model. It can scale up from single servers to thousands of machines. The mapReduce procedures shown in figure 6.1 can explain the processing of datasets in a distributed fashion over several machines. It distributes the works to mappers and maps it to ¡key,value¿ pairs. The reducers will process each group of output data and combine the ¡key,value¿ pairs. Hadoop supports our Java program to implement the "map" and "reduce" parts. [28] The Solr also can be implemented with the Apache Hadoop framework [29]. Thus, it can afford large number of users and provide quick search speed.

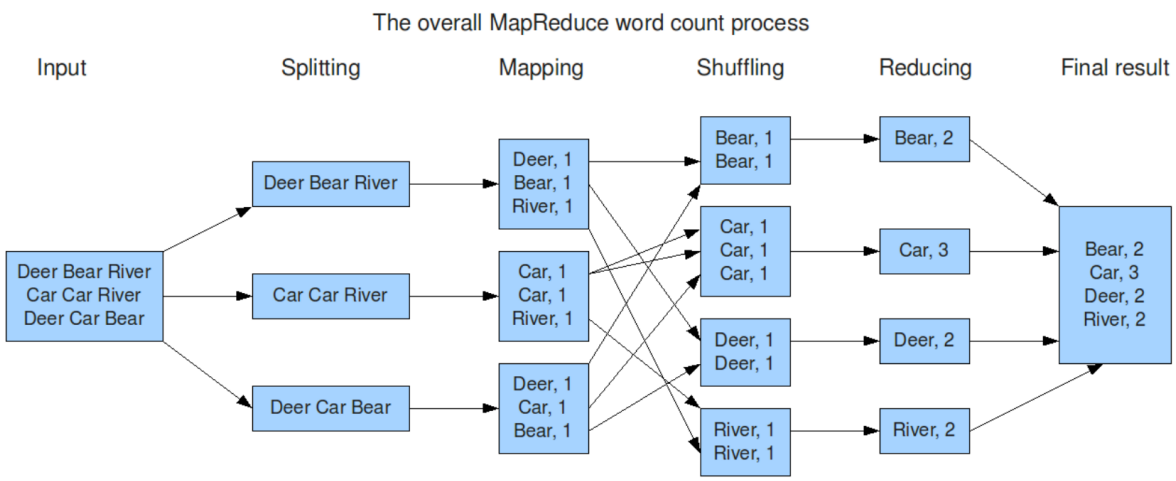


Figure 6.1: MapReduce of word count process

The Single Node Setup of Hadoop is already applied to the virtual machine of Apachecloudstack of Imperial College London. The word count function based on mapReduce framework is achieved and tested. After configure the Cluster Setup of multi-node Hadoop and apply our whole program to mapReduce framework, the distributed system can be built in the future work.

Adjust Parameters of Graph Automatically

The complexity of graph depends on the parameters entered by users. However, the result got by this procedure is unforeseeable. An algorithm to calculate the complexity of graph

can be designed which may according to the number of nodes and edges. When speed of generating graph increased, the program can generate graphs of different parameters and choose the most appropriate one.

Add Customized Settings

Some customized settings of search can be added in future. Firstly, author wants select valid papers of search in range of date. Then, the papers also can be filtered by authors given by users. After that, the given words appear in title and abstract can increase the score of paper which is also controlled by users. Finally, users can select the required papers from the table of our website to generate the coherence graph. The components of selecting is already completed. The procedure of transmit data of this function can be achieved in future.

Contribute the Score of Paper by Citation Relationship

The papers should influenced by citation relationship of the given topic. As the page rank algorithm used for ranking website, the importance of paper A should be improved if it is cited by an important paper B. This can be achieved in our citation graph. Some other algorithms of page rank also could be applied.[30]

6.3 Conclusion

The original aims of the project can be assessed to check the extent of completion. The application achieved functions that enable users both to generate storyline graph under identical search input and to see the visualized graph on the designed website.The search system is built which can use search query to obtain valid papers ordered by score. The Java graph construction system and its API is also completed. And programmer success in finishing web application and the visualization of coherence graph. All objectives are basically achieved in this project.

How well does it work? The search system based on Solr return correct and fast results under given search query, but intermediate outcome is invisible and can not be accessed. The Java graph construction system can return valuable results contains the data structure of the coherence graph for visualization. However, in certain complex situations, the time spent on the whole procedure would be longer and possibly reached to tens of minutes, even in the case applied concurrency strategy. In order to obtain a coherence graph with acceptable complexity, three parameters, including number of nodes, sub-chain length and coherence threshold, could be adjusted by user manually. Fortunately, in most cases, the coherence graph generated in default setting is within the range of a suitable complexity. The website created by React is indeed user friendly with clear instructions and is easy to operate. Nonetheless, the appearance of the visualized coherence graph displayed on the website could be improved in the future.

To conclude, this project achieved an application to generate and visualize coherence graph, aiming at providing a more clear understanding for users in identical academic field. To further optimise this project, potential interested developers could start with improving the speed in search and graph construction stages and raising the accuracy rate of algorithm in generate storyline graph.

Bibliography

[1] T. Chamorro-Premuzic. (2014,) How the web distorts reality and impairs our judgement skills. [Online]. Available: https://www.theguardian.com/media-network/media-network-blog/2014/may/13/internet-confirmation-bias

[2] S. Grimes. (2008,) Unstructured data and the 80 percent rule. [Online]. Available: https://breakthroughanalysis.com/2008/08/01/unstructured-data-and-the-80-percent-rule/

[3] IEEE. (2016) Ieee xplore digital library. [Online]. Available: http://ieeexplore.ieee.org/Xplore/home.jsp

[4] A. Anthology. (2016) A digital archive of research papers in computational linguistics. [Online]. Available: http://www.aclweb.org/anthology/

[5] L. Lu, "Application on unstructured data for academic literature," 2015, unpublished thesis.

[6] E. D. Liddy. (2005,) Document retrieval, automatic. [Online]. Available: http://surface.syr.edu/cgi/viewcontent.cgi?article=1055&context=istpub

[7] R. I. Cristian Moral, Angélica de Antonio and J. Ramírez. (2014) A survey of stemming algorithms in information retrieval. [Online]. Available: http://www.informationr.net/ir/19-1/paper605.html#.V8cAr5MrLVo

[8] M. Jursic. (2010) Lemmagen: Multilingual lemmatisation with induced ripple-down rules. [Online]. Available: http://kt.ijs.si/nada_lavrac/Publications/JUCS-2010-LemmaGen.pdf

[9] H. C. WU and R. W. P. LUK. (2008) Interpreting tf-idf term weights as making relevance decisions. [Online]. Available: http://delivery.acm.org/10.1145/1370000/1361686/a13-wu.pdf

[10] Y. Z. Shize Xu, Shanshan Wang. (2013) Summarizing complex events: a cross-modal solution of storylines extraction and reconstruction. [Online]. Available: http://www.aclweb.org/anthology/D13-1127

[11] R. G. James Allan and V. Khandelwal. (2001) Temporal summaries of news topics. [Online]. Available: http://maroo.cs.umass.edu/getpdf.php?id=2267

[12] D. Shahaf, C. Guestrin, and E. Horvitz, "Metro maps of science," 2012,. [Online]. Available: http://www.cs.cmu.edu/~dshahaf/kdd2012-shahaf-guestrin-horvitz.pdf

[13] D. Shahaf and C. Guestrin, "Connecting the dots between news articles," 2010,. [Online]. Available: http://www.cs.cmu.edu/~dshahaf/kdd2010-shahaf-guestrin.pdf

[14] D. Shahaf, C. Guestrin, and E. Horvitz, "Trains of thought: Generating information maps," 2012,. [Online]. Available: http://i.stanford.edu/~dshahaf/fp0590-shahaf.pdf

[15] Solr. (2016) Apache solr™ 6.2.0. [Online]. Available: http://lucene.apache.org/solr/

[16] Facebook. (2016) React a javascript library for building user interfaces. [Online]. Available: https://facebook.github.io/react/

[17] ant design. (2016) Ant-design: An enterprise-class ui design language and react-based implementation. [Online]. Available: https://github.com/ant-design/ant-design

[18] Spark. (2016) Spark - a micro framework for creating web applications in java 8 with minimal effort. [Online]. Available: http://sparkjava.com/

[19] D3.js. (2016) Data-driven documents. [Online]. Available: https://d3js.org/

[20] D. Skau. (2013) Why d3.js is so great for data visualization. [Online]. Available: http://www.scribblelive.com/blog/2013/01/29/why-d3-js-is-so-great-for-data-visualization/

[21] A. Ranganathan. (2009) cross-site xmlhttprequest with cors. [Online]. Available: https://hacks.mozilla.org/2009/07/cross-site-xmlhttprequest-with-cors/

[22] C. Bradford. (2015) Going cross-origin with solr. [Online]. Available: http://opensourceconnections.com/blog/2015/03/26/going-cross-origin-with-solr/

[23] K. El-Arini and C. Guestrin. (2011) Beyond keyword search: Discovering relevant scientific literature. [Online]. Available: http://www.cs.cmu.edu/~kbe/bks-elarini-guestrin-tr.pdf

[24] D. Åse. (2015) Setting up spark with maven. [Online]. Available: https://sparktutorials.github.io/2015/04/02/setting-up-a-spark-project-with-maven.html

[25] F. M. G. III. (2016) Implement cors in spark. [Online]. Available: https://sparktutorials.github.io/2016/05/01/cors.html

[26] J. K. Hart and K. Martinez. (2006) Environmental sensor networks: A revolution in the earth system science sensor data: a preliminary study. [Online]. Available: http://eprints.soton.ac.uk/263093/1/esn.pdf

[27] P. K. H. V. h. Lode Vuegen, Bert Van Den Broeck. (2013) Automatic monitoring of activities of daily living based on real-life acoustic sensor data: a preliminary study. [Online]. Available: http://www.aclweb.org/anthology/W/W13/W13-3918.pdf

[28] C. Lam, *Hadoop in Action*. Manning Publications, 2011.

[29] T. P. Tery Grainger, *Manning - Solr in Action*. Manning Publications, 2014.

[30] D. Sullivan. (2007) What is google pagerank? a guide for searchers webmasters. [Online]. Available: http://searchengineland.com/what-is-google-pagerank-a-guide-for-searchers-webmasters-11068

Appendices

Appendix A

Github

The whole achievements of this paper are upload to github as two projects.

Imperial_final_project:

- Search System : Return valid paper data according to search query
- Java Graph Construction System: Using data of valid papers to construct coherence graph which shows the storylines of the search query
- Java Restful API server: Call Java program and return the coherence graph to web application

This part is avaiable at `https://github.com/bingweichen/Imperial_final_project`

Search_Web:

- Website : communicate with users and get required parameter.
- Visualization : using D3.js to show the coherence graph

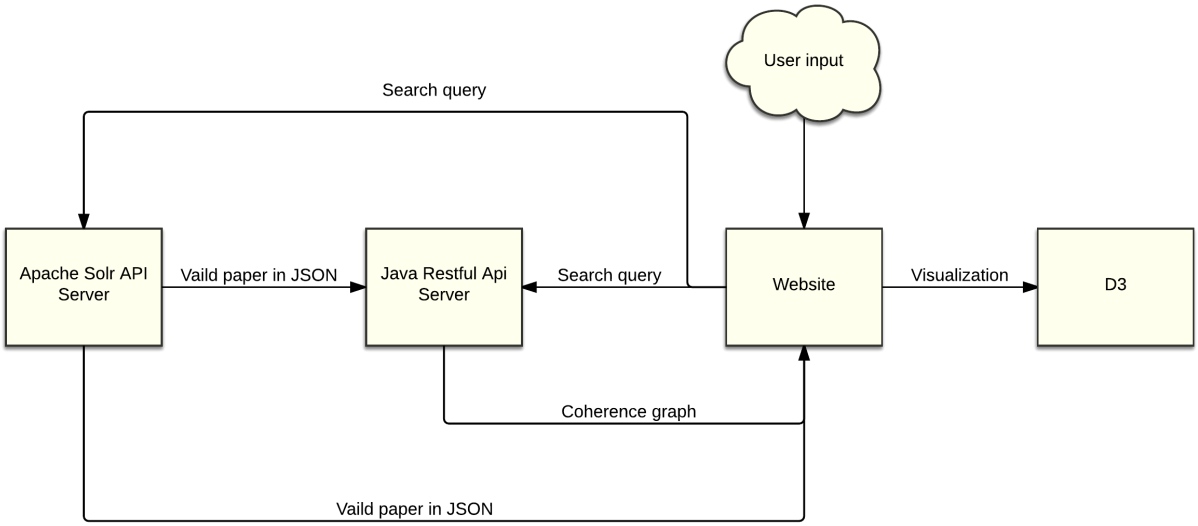
This part is avaiable at `https://github.com/bingweichen/Search_Web`

porject title: Scientific survey creation

author: bingwei chen

keywords: storyline,coherence graph, serach engine, web application, visualization

This is a project which can show stroyline graph of search keywords on web application.



this is the overview of this project.

process:

1. get the query of user input from website
2. sent the query to solr server return vaild papers sorted by score in JSON
3. the web application show the information of these papers to users
4. sent the query to Java Restful API server. after get the papers' information from Solr, the Java progrom will process the data and constract coherence graph which will be return to website.
5. after receive the coherence graph, the information will be visualized by D3

it will be implemented by fellow 5 steps:

1. Pre-process
2. Solr
3. Coherence graph java
4. Java api server
5. Web

Pre-process

In last year, the data of ACL was crawled and processed to XML by Lingyu Lu.
In this step, all data of paper in XML will be convert to JSON and the information of date and url of each paper will be added.

XML&&JSON Download url:
https://www.dropbox.com/sh/aqy4k19az24b0in/AADeJucNMZE2UcgY2nME6F0-a?dl=0

XML: SectLabel,PharseHed,ParsCit

JSON: id,title,author,url,date,content,abstract, references

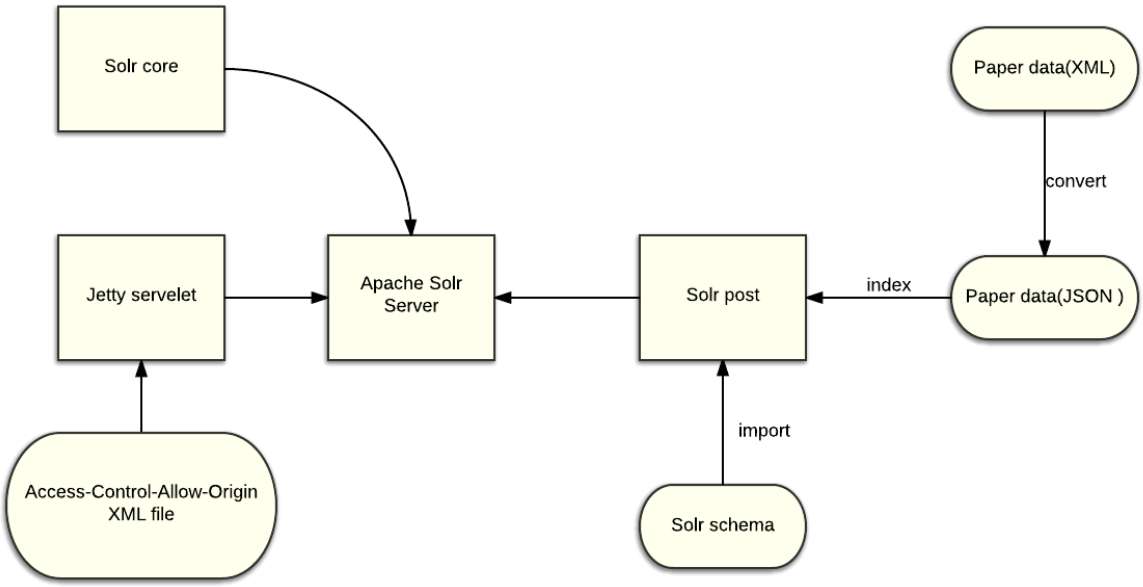
download:

json-simple-1.1.1.jar

mysql-connector-java-5.1.39-bin.jar

Solr

All commands and API in README.md of Solr file.



1. Download Solr from: http://www.apache.org/dyn/closer.lua/lucene/solr/6.2.0
2. using commands and API
    1. construct Solr
    2. upload schema
    3. post all papers of JSON to Solr
3. change the webdefault.xml of jetty to solve cross domain problem

After finish this, the solr api can be used to get required paper in JSON.

Coherence graph java

1. get vaild paper data in JSON from Solr
2. construct graph by adding each paper to vertex

3. adding citation label edges
4. adding same author label edges
5. normalize
6. adding ancestral label edges
7. construct coherence graph using divide-and-conquer algorithm

download:

apache-httpcomponents-httpclient.jar

apache-httpcomponents-httpcore.jar

httpclient-4.3-beta1.jar

java-json.jar

json-simple-1.1.1.jar

detail:

1. get vaild paper data in JSON from Solr
    1. sent http request to Solr using the search query
    2. store the returned JSON object which contains all vaild paper
2. construct graph by adding each paper to vertex
    1. create MyGraph object which contains Vertex and Node
    2. Vertex used for store the object of paper and all Nodes of this Vertex
    3. Nodes used for edge between two Vertex
3. adding citation label edges
    1. compare the title and reference title
    2. using word segmentation and compare alogrithm to return a score
    3. if score exceed threshold value, add edge with citation label from cited paper to citing paper
    4. the weight of the edge depend on the score of paper cited
4. adding same author label edges
    1. compare the authors of each paper
    2. add edge with same author label from earlier paper to latest if they have at least one same author
    3. the weight of the edge depend on the score of paper published earlier.
5. normalize
    1. for each Vertex, add all weight of citation label point to this Vertex to Z(normalization constant)
    2. for each Vertex, calculate the number of edges l point to this Vertex with same author label.
    3. add the weight of same author label divided by l to Z
    4. for each edge of this Vertex,
6. adding ancestral label edges
    1. monteCarloSimulation

    2. generateRandomsubgraph
7. construct coherence graph using divide-and-conquer algorithm
    1. get all pairs which coherence exceed the threshold value
    2. using chain object to store these pair of paper and put the chain into priorty queue
    3. collect the chain with highest coherence and extend it if it is short than length of sub-chain
    4. connect chains in same storyline

Java API Server

1. set up Spark

   https://sparktutorials.github.io/2015/04/02/setting-up-a-spark-project-with-maven.html
2. implement CORS in spark

   https://sparktutorials.github.io/2016/05/01/cors.html

Web

React

1. download ant-design

   https://github.com/ant-design/ant-design
2. install npm
3. install node.js
4. `npm install antd`
5. `npm start`

D3.js

1. download from

   https://github.com/d3/d3.git
2. using force layout

debug

1. change same author label function

   edge from earlier to latest
2. change monte carlo function

   same ancestor x->c1 and x->c2

Commands

Start
`bin/solr start -e cloud -noprompt`

Delete All
`bin/post -c gettingstarted -d "<delete><query> * : *</query></delete>"`

Delete
`bin/post -c gettingstarted -d "<delete><id>40</id></delete>"`

Stop All
`bin/solr stop -all`

Post
`bin/post -c gettingstarted /Users/chen/Documents/individulal_project/data/testjson/*.json`

API

Search

```
http://54.171.189.58:8983/solr/gettingstarted/select?indent=on&q=*:*&wt=json
```

Update

```
curl -X POST -H 'Content-type:application/json' --data-binary '[
{
  "id": "bsdsds"
  }
]' http://54.171.189.58:8983/solr/gettingstarted/update
```

Delete ALL

```
   curl -X POST -H 'Content-type:text/xml' --data-binary "<delete><query>*:*
</query></delete>"  http://54.171.189.58:8983/solr/gettingstarted/update
```

Delete

```
curl -X POST -H 'Content-type:text/xml' --data-binary "<delete><query>id:5555
6666</query></delete>"  http://54.171.189.58:8983/solr/gettingstarted/update
```

Normal mode

`$ bin/solr start`

`$ bin/solr start -p 8984`

```
$ bin/solr create -c <name>
```

Solution for Commands Argument too long

```
find /Users/chen/Documents/individulal_project/data/json -name '*.json' -exec
/Users/chen/project_run/solr/solr-6.1.0_2/bin/post -c  gettingstarted {} \;
```

Using for upload all JSON file

Appendix B

User Manual

- enter search query and press search button
- press citation graph navigator to check the citation graph
- press coherence graph navigator coherence graph construction page
- change default parameters if needed
- press Generate Graph button, the graph will showed when the load is finished
- move mouse on the circle to see the title of the paper
- the titles of all papers can be showed by clicking one of circles
- the detail information can be showed by double clicking the circle
- the circle can be dragged vertically by mouse down and move

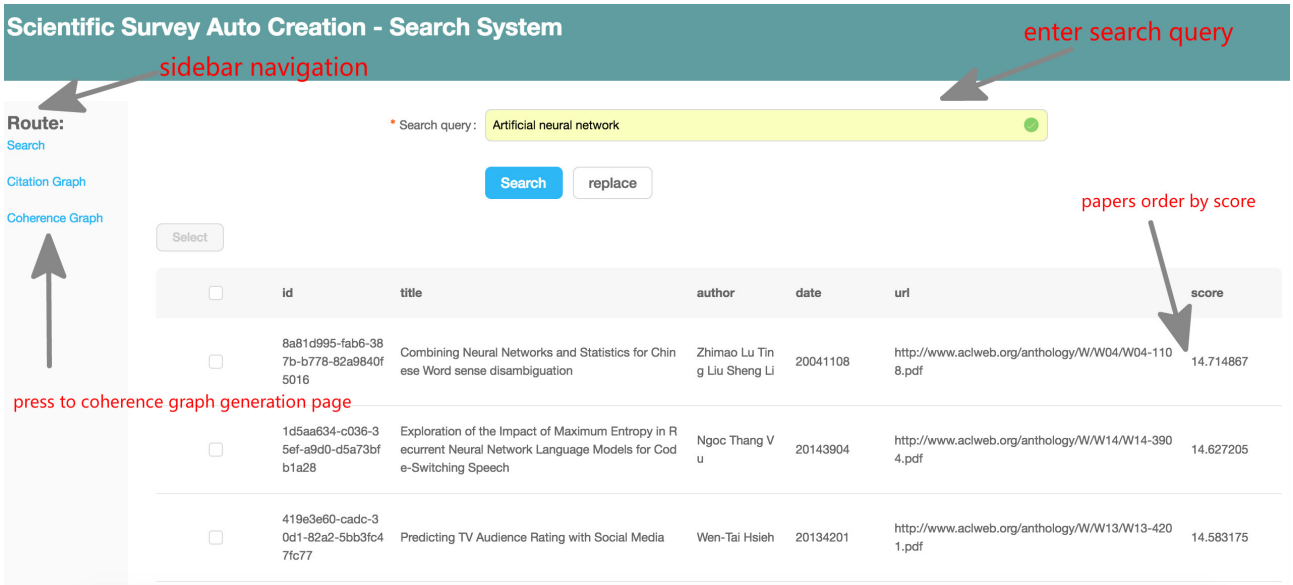


Figure B.1: Search page

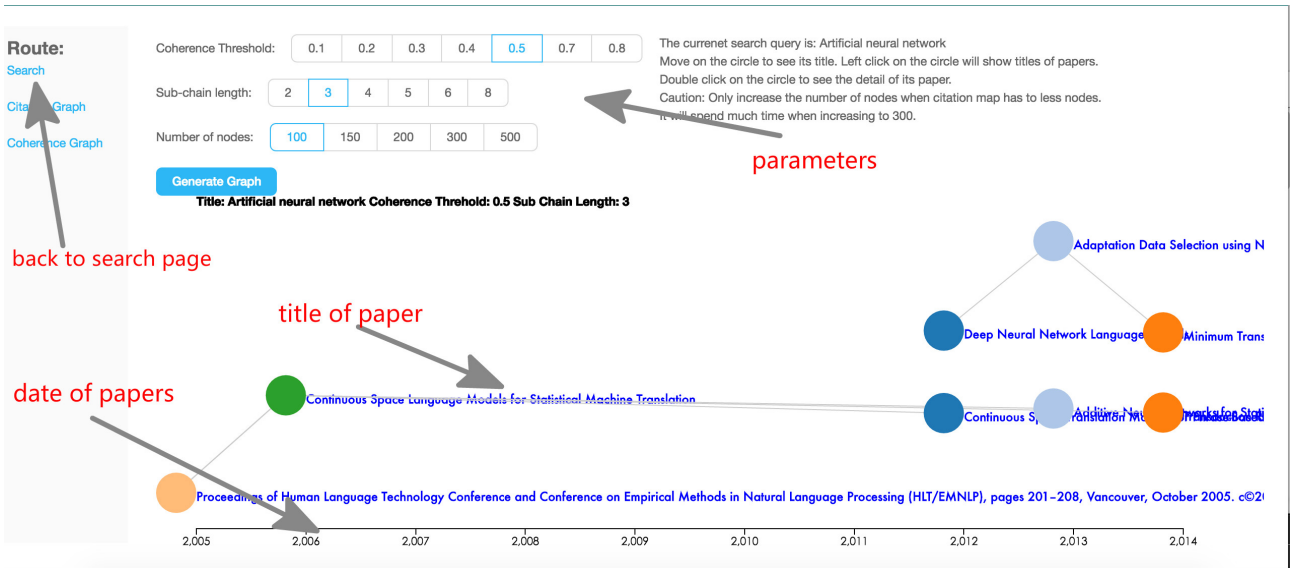


Figure B.2: Search page

Appendix C

User Survey

User A

I searched the keys words ”primary market and secondary market”, ”effective market theory” and ”option pricing” in the system.
The results show clearly the development and evolution of financial market hypothesis, for example, the coherence graph for ”primary market and secondary market” demonstrates the time line and relationships of financial market from forming to maturing. I believe this system could help user to see the inside relation of certain area.

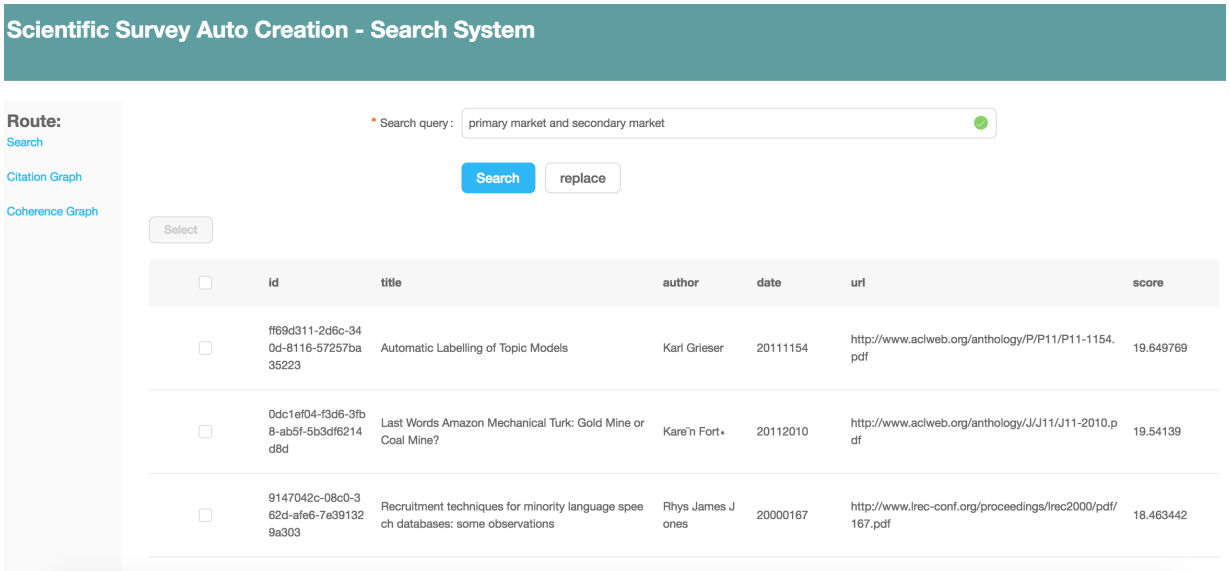


Figure C.1: primary market and secondary market

User B

I tested the search system with keywords ”Inductive Logic Programming” and ”Inverse Entailment”. The system would list all related theses and also provides a coherence graph for each keyword.

It is very helpful to use such coherence graph to study the development history of logic programming, which provides a whole picture with underlying relations of this subject. I will choose and support this search system.

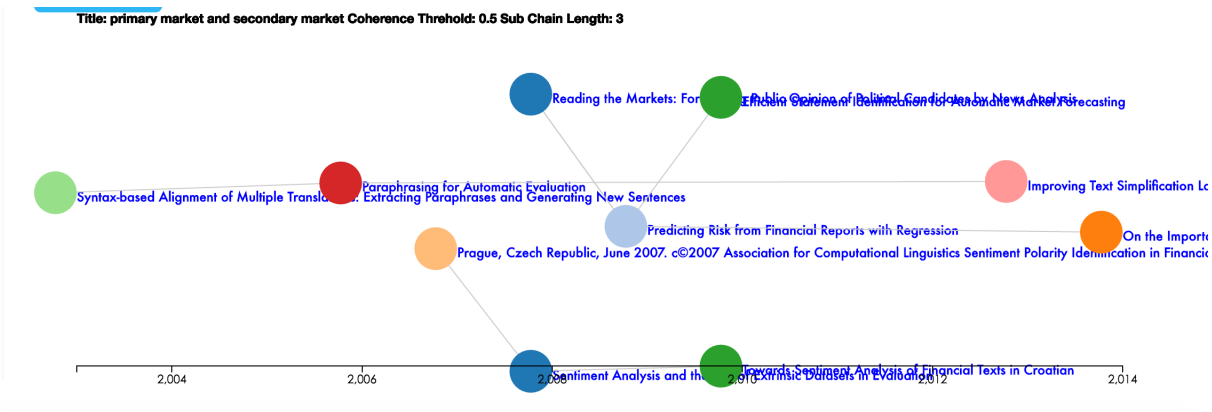


Figure C.2: primary market and secondary market

58114aa6-eb45-3a35-9331-6aa59ec826a5	DESIGN OF A KNOWLEDGE-BASED REPORT GENERATOR	Karen Kukich	19831022	http://www.aclweb.org/anthology/P/P83/P83-1022.pdf	9.57251
9019b86c-c1cb-3017-9643-f760e97f2883	A Corpus-based, Comparative Study of Mandarin Chinese, English and Spanish	Siaw-Fong Chung	20031014	http://www.aclweb.org/anthology/Y/Y03/Y03-1014.pdf	9.143704
2a39c3f3-b481-3294-bee2-3405abc8aeef	Summarization of Multimodal Information	Saif Ahmad	20040502	http://www.lrec-conf.org/proceedings/lrec2004/pdf/502.pdf	8.924136
02919e1b-bc24-32db-be29-f50fa697d450	SYSTEM DEMONSTRATION	INTERACTIVE GENERATION	19981435	http://www.aclweb.org/anthology/W/W98/W98-1435.pdf	8.736851
df830e4a-0a16-32fb-b50a-01c73a8c7a9c	Prolog for Natural Language Processing	Annie Gal	19924012	http://www.aclweb.org/anthology/J/J92/J92-4012.pdf	8.736851
e2c3f150-d49a-3888-8d8b-1ba9f8709228	Social Event Radar: A Bilingual Context Mining and Sentiment Analysis Summarization System	Wen-Tai Hsieh Chen-Ming Wu	20123028	http://www.aclweb.org/anthology/P/P12/P12-3028.pdf	8.711426
14666245-1fc9-397b-b692-566e369b0510	A Sentiment Detection Engine for Internet Stock Message Boards	Maria Milosavljevic	20091012	http://www.aclweb.org/anthology/U/U09/U09-1012.pdf	8.701132

Figure C.3: effective market theory

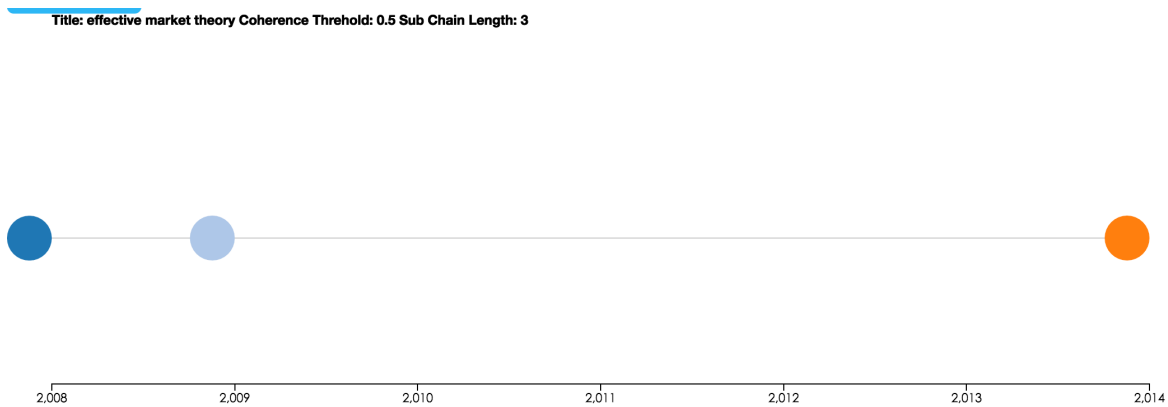


Figure C.4: effective market theory

User C

I typed ”Principal Component Analysis” into the search system. The searching results displayed by the system give both detailed information and an overview structure.

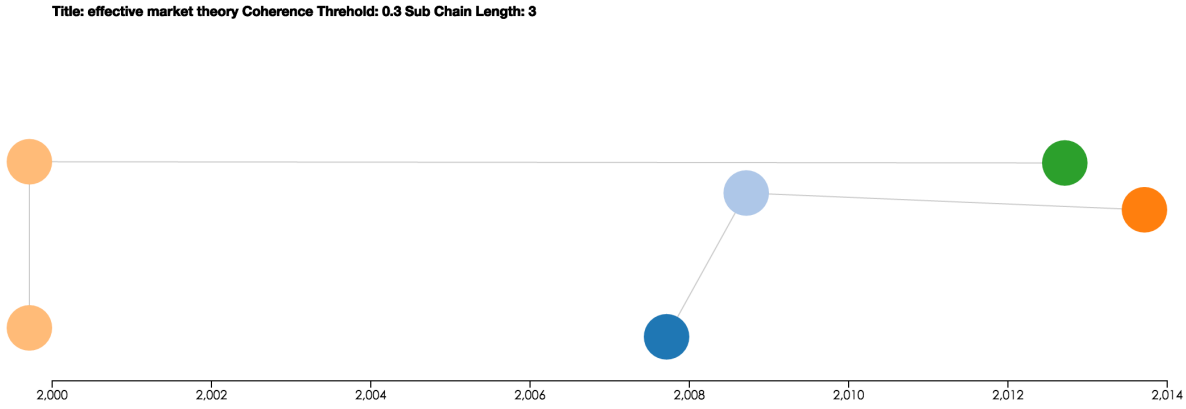


Figure C.5: effective market theory

	7c857b23-77eb-3396-81f1-cc9434ca8896	Automatic inference of indexing rules for MEDLINE	Aure´lie Ne´ve´ol	20080612	http://www.aclweb.org/anthology/W/W08/W08-0612.pdf	15.437466
	49de51ca-0cbf-304e-8390-c9a9d4c94f74	Sydney, July 2006. c©2006 Association for Computational Linguistics A Hybrid Relational Approach for WSD – First Results	Lucia Specia	20063010	http://www.aclweb.org/anthology/P/P06/P06-3010.pdf	15.343369
	fb6e4b19-2872-398e-9fb3-19ad6dfbebfe	An Entity-Mention Model for Coreference Resolution with Inductive Logic Programming	Xiaofeng Yang Jian Su Jun Lang Chew Lim Tan Ting Liu Sheng Li	20081096	http://www.aclweb.org/anthology/P/P08/P08-1096.pdf	15.143748
	93411f07-5807-3092-a938-77583bd86325	Learning Expressive Models for Word Sense Disambiguation	Lucia Specia	20071006	http://www.aclweb.org/anthology/P/P07/P07-1006.pdf	14.916866
	9addeb92-502d-3a43-95f5-fb89b9a3cf41	Inferring syntactic rules for word alignment through Inductive Logic Programming	Sylwia Ozdowska	20100878	http://www.lrec-conf.org/proceedings/lrec2010/pdf/878_Paper.pdf	14.81502
	5d9ff714-7940-37bc-b42d-c7da567a18bd	Learning Semantic-Level Information Extraction Rules by Type-Oriented ILP	Yutaka Sasaki	20002101	http://www.aclweb.org/anthology/C/C00/C00-2101.pdf	14.478109

Figure C.6: Inductive Logic Programming

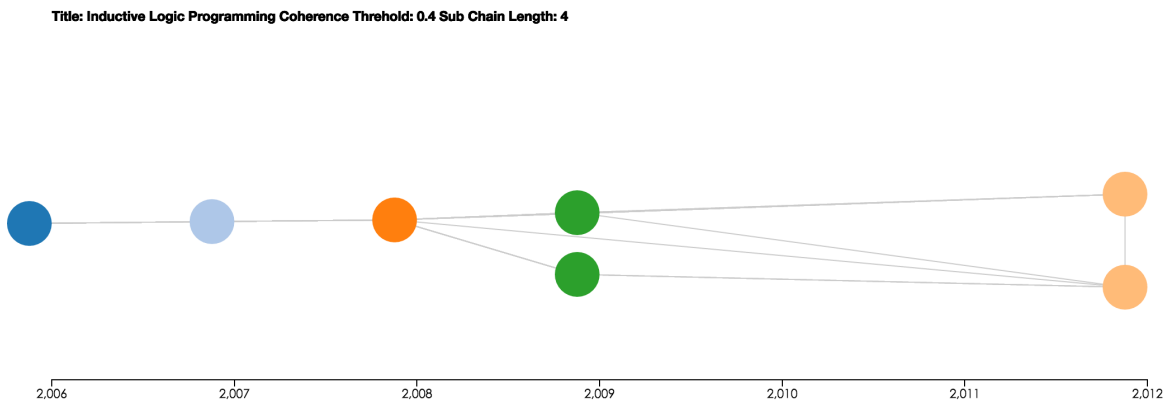


Figure C.7: Inductive Logic Programming

I suppose that by following the connection shows in coherence graph the user could have a comprehensive understanding of the significance of Principal Component Analysis in statistic. From a user sight of view, the system provides a powerful function.

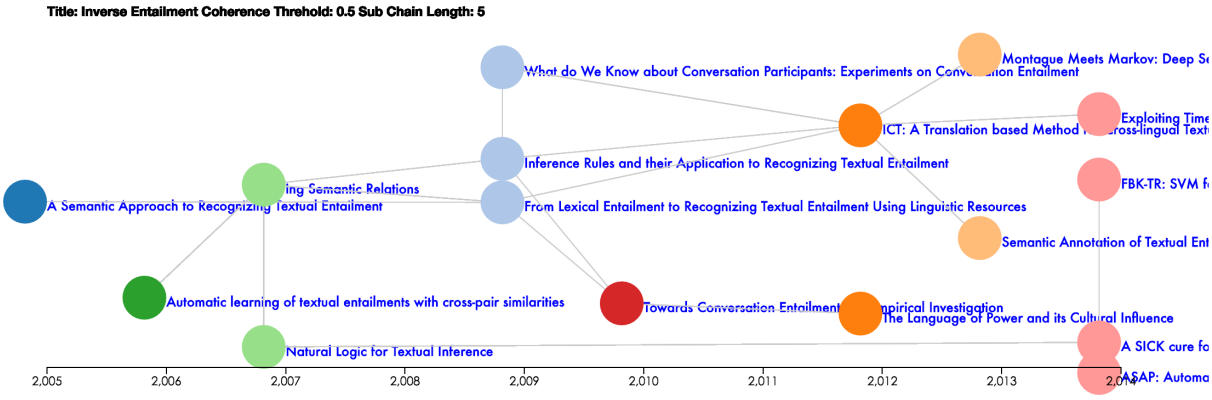


Figure C.8: Inverse Entailment

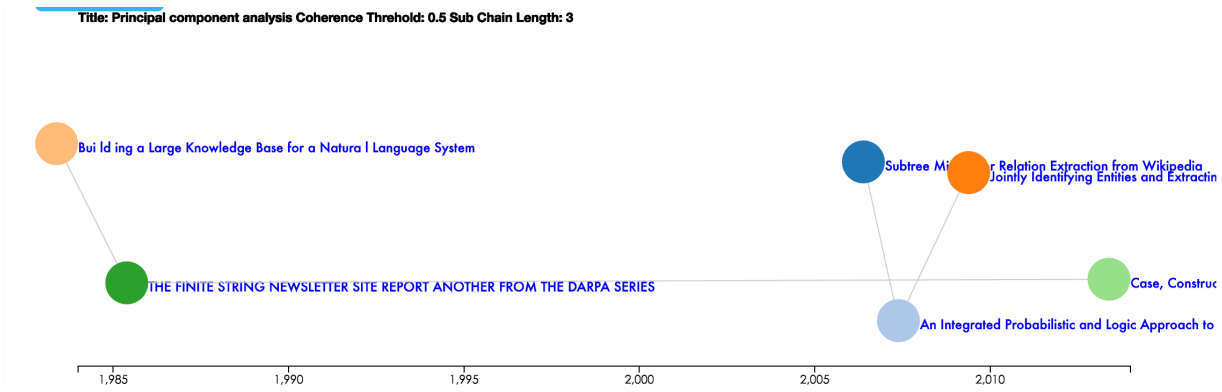


Figure C.9: Inverse Entailment