# ir

*by* Snigdha Report

**Submission date:** 03-May-2018 02:23PM (UTC+0530)
**Submission ID:** 958164441
**File name:** snigdha_ir-report.pdf (1.49M)
**Word count:** 4400
**Character count:** 24021

Information Retrieval Project Report

On

# A natural language interface to a graph-based bibliographic information retrieval system

## Submitted by

15IT149 SNIGDHA TADURU
15IT210 RISHITA GOLLA
15IT244 SHREYA SHETTY

Under the Guidance of

**Dr. Sowmya Kamath**

Assistant Professor

**Date of Submission: 30th April, 2018**


**Department of Information Technology**
**National Institute of Technology Karnataka, Surathkal.**
**2017-2018**

# Certificate

This is to certify that this project report entitled A natural language interface to a graph-based bibliographic information retrieval system submitted to Department of Information Technology,NITK, is a bona fide record of work done by the team members under my supervision from January 2018 to March 2018.

Guide: Dr.Sowmya Kamath

Signature of guide:

Date:

# Declaration

We hereby declare that the project work done here and submitted to the IT Department, is a record of an original work done by us under the guidance of Dr. Sowmya Kamath, Department of Information Technology, National Institute of Technology, Karnataka - Surathkal. This project work is submitted in the partial fulfillment of the requirements for the award of the degree of Bachelor of Technology in Information Technology as a Mini-Project for Information Retrieval. The results embodied in this project report have not been submitted to any other University or Institute for the award of any degree or diploma.

Snigdha Taduru 15IT149

Rishita Golla 15IT210

Shreya Shetty 15IT244

## Summary

A bibliographic database is an organized digital collection of references to published journals, reports, papers, patents, conference proceedings, etc.
With the rapid increase in bibliographic data there is a need for a interface to successfully maintain, organize and store the vast growing data. Several systems have been developed to organize the data, these systems include Google Scholar, Web of Science and Scopus. These interfaces have their respective drawbacks. Web of Science uses a form-based interface where individuals have to first select their fields and then respective values in those fields. This is troublesome as users have to take a burden in understanding what tags and fields they should select. Google Scholar uses a keyword based interface. This provides a drawback of understanding the queries and relations between the keywords. Natural language interfaces are considered to be more effective in comparison to form and keyword interfaces as users are allowed to express their need more easily in a quicker manner.
Our work involves making a natural language based bibliographic interface to easily query and retrieve bibliographic records. Here we convert a natural query to a graph query and then query a graph database. Converting a natural query to a graph query involves a series of steps. These steps include recognizing bibliographic entities, tokenizing the natural query, finding dependency relations among the tokens, generating graph relations and converting the graph relations to a graph query. After obtaining the graph query we can query the graph database, for implementation neo4j graph database has been used.

# Abstract

It is common knowledge that, there are over millions of research papers that have been published over the years. If not for a very capable system it would be very difficult to retrieve required results for a particular query. This project aims to build a natural language interface to a bibliographic graph database that is the data set of bibliographic data is converted to a graph database for easy retrieval. The natural language query is converted into a graph query to retrieve from the graph database. This structure uses a series of text and linguistic-based techniques such as tokenization, named-entity recognition and syntactic analysis. A NLI-GIBIR interface which is used effectively represents and addresses complex bibliographic needs.

# Contents

# List of Figures

# 1 Introduction

With the rapid increase in the number of research papers being published every year there is an ever-increasing importance of a system for bibliographic retrieval of records. Some systems which have been developed include Web of Science, Scopus and Google scholar. Web o Science and Scope use a form based interface where the user has to select fields and values. Although this interface is effective for simple queries, they prove to be inadequate for complex queries as they burden the users with the task of understanding and selecting fields or tags. Google Scholar uses a keyword based interface. It much more convenient in comparison to form based interfaces but it provides users limited options for representing complex bibliographic queries. Hence we have implemented a natural language based interface which queries from a graph based interface.
The following documentation explains the literature survey, methodology, implementation and conclusions.

# 2 Literature Survey

## 2.1 Background

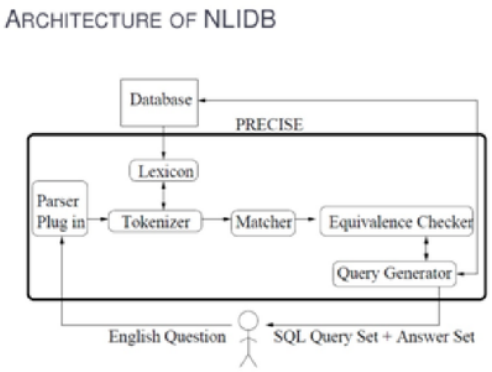
We review three areas of research that directly connect to natural language-enabled bibliographic information retrieval systems: natural language interfaces, named entity recognition, and syntactic analysis.

Three areas of research are reviewed which are paramount to connect a natural language-enabled bibliographic information retrieval

Natural language interface

Natural language user interface (NLI) is a type of computer human interface where linguistic phenomena such as verbs, phrases and clauses act as user interface controls for creating, selecting and modifying data in software applications. NLI can be classified broadly into two types: one is natural language interface to databases (NLIDB), and the other natural language interface to knowledge bases (NLIKB).

NLIDB In NLIDB a relational database is used to store structured information[1]. Asking questions to databases in natural language is a very convenient and easy method of data access from database system especially for normal users who do not understand complex database query languages such as SQL. This enables a user to simply enter queries in English to the Natural Language Database Interface.




NLIKB NLIKB is a natural language interface that use an ontology to manage information[2][3]. The vision is that of users being able to tell a computer what they would like to find, using any number of sentences and as many details as requested. NLIKB began with the semantic web hence has a concise record. NLIKB usually performs better[3, c]ecause it makes use of rich ontologies represented in

the resource description framework[5]

The NLI used in this project is a NLIDB as they have prominent scalability making them better for real-world applications even though their expressive power is debatable. Also they have been used extensively in information retrieval systems. Given the graph like characteristics used in bibliographic data[6], a NLI to graph database-based bibliographic information retrieval systems provides a practical way of accessing and retrieving bibliographic data.

### 2.1.1 Named Entity Recognition

Named entity recognition(NER) is a sub task of information extraction that seeks to locate and classify named entities in text to predefined categories such as the names of persons, organizations, locations, expressions of times, quantities, monetary value, percentage etc. So basically it refers to a data extraction task that is responsible for finding, sorting and storing textual content into default categories.

A dictionary-based NER system utilizes pre-dened dictionaries and performs a look-up in texts[7, 8]. The method is widely used in domains such as biomedicine, in which named entities are well-recorded and managed: for instance, in protein recognition [9] and drug recognition [10]. Another popular category of NER is statistical NER (e.g., [13]). Widely used statistical NER methods include maximum entropy (ME)- [8], hidden Markov models (HMM)- [6], and conditional random elds (CRF)-based NER systems [32]. Some NER systems use more than one type of NER: for example, Stanford NER [20] provides both dictionary- and statistical-based NER through a gazette feature.

The bibliographic data used in this project is from ARNETMinor. A dictionary based approach is used to retrieve bibliographic named entities (i.e., authors, papers, organizations, terms, and sources) from a natural language query. The relations between these entities and answer query is extracted along with the entities.

### 2.1.2 Syntactic analysis (Parsing)

Parsing is usually done fro a string based on a structure of sentence formation for phrases defined before. (i.e context free grammar)[1]. After data representation such as Penn Treebank [2] which is annotated data was introduced a number of statistical parsers were proposed and became popular. Two popular ways of representing syntactic structures are constituency and dependency. For constituency, words in a sentence are organized into nested constituents; for dependency, dependent relations between words are shown. This project uses a dependency structure to identify grammatical relations among bibliographic named entities present in NL queries. NaLIR proposed by Li and Jagadish [3]defines a NLI for relational

databases that can be applied to bibliographic database retrieval. However in this project the system is designed to answer bibliographic query of relations(e.g., authors of papers that were cited by papers that were written by John). , not logics as in the paper by Li and Jagadish(e.g., return the author who has the most publications in database area.

## 2.2 Identified Gaps

There are a few limitations identified in the implemented project. First, this project can be applied effectively only on bibliographic kind of data-sets so as to get relations between named entities. Because of some rules applied on the interface the performance of the framework worked out so well. It may not apply to other domains. Second, the retrieval results are highly dependent on the query and the formulation of the natural language query. It may have to be resolved using query expansion techniques which has not been implemented in this project. Third, the Stanford NER parser results were scraped from the website instead of using the library in python as the results were not obtained in the required form. Finally, the ARNETMinor data set used in this project does not include a cited and citing column that is it is not possible to retrieve satisfactory results for a query such as Papers that are written by John and cited by Happy University .

## 2.3 Problem Statement

Implementation of a natural language interface for easy retrieval of bibliographic records using a graph database- Neo4j.

## 2.4 Objective

1)Fast retrieval for the ever-growing bibliographic data.
2)Easy and efficient storage of the data in the form of a graph database.
3)Enables users to input their query in a natural way with minimum number of restrictions.

# 3 Methodology and Framework

The work flow diagram comprises of the following steps.

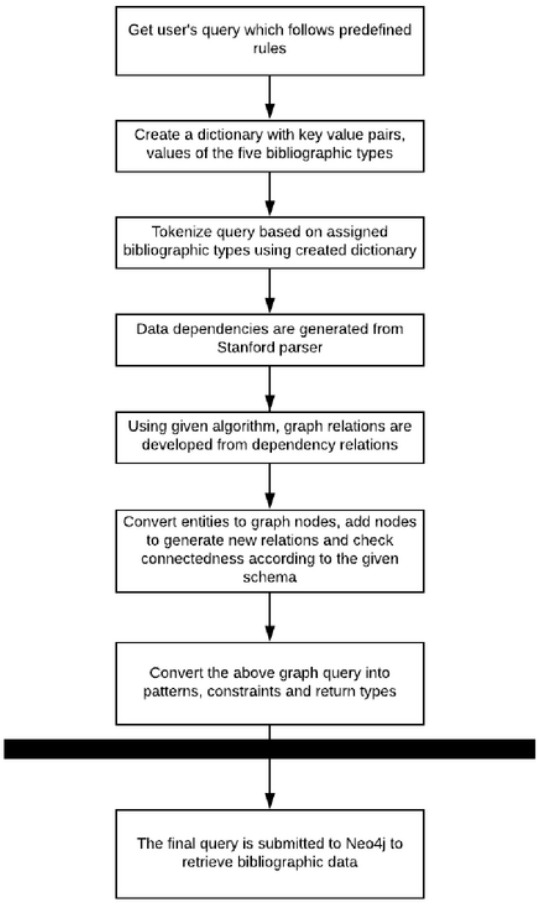



Figure 1: Work flow diagram

The system is designed to take a natural language query as input and return corresponding bibliographic data as output. The steps involve
1) Converting a natural query to a graph query
2) Querying a graph database

## 3.1 Converting Natural Query to a Graph Query

Converting a natural query to a graph query involves a series of steps including
1) Formulating a natural query
2) Recognizing and extracting named entities
3) Tokenizing natural query based on named entities
4) Extraction of dependency relations
5) Generating graph relations
6) Formulation of a graph query

### 3.1.1 Formulating a natural query

The user can formulate his/her query to retrieve information regarding bibliographic records. More specifically, the system can return information regarding the paper, author, term, source and organization of the bibliographic records. The interface can also return the citation information.
However there are certain rules to the queries that can be given to the system.
First the query should be noun phrases instead of complete sentences with interrogative pronouns i.e. instead of "*What are the papers written by Adam?*" the query should be "*papers that were written by Adam*"
Second, the query should not have grammatical errors.
Third the query should not omit relative pronouns like "*that*" and "*which*" i.e. "*papers that were written by Adam*" is preferred to "*papers written by Adam*"

### 3.1.2 Recognizing and extracting named entities

There are five types of bibliographic named entities in this system, them being paper, author, term, source and organization. A dictionary is constructed by preprocessing the dataset. The keys of the dictionary are the entities extracted from the dataset and their corresponding values are either one of the five (paper, author, term, source, organization).
Five other entities are added to the dictionary. The added entities include (paper,class_paper), (author,class_author), (term,class_term), (source,class_source) and (organization,class_organization).
Example for an entry in the dictionary is (*Information retrieval* , *paper*).

### 3.1.3 Tokenizing natural query based on named entities

The query specified by the user is tokenized using the above created dictionary. Tokenizing with NER is important as "*Information Retrieval*" is treated as one token instead of two tokens "*Information*" and "*Retrieval*".

### 3.1.4 Extraction of dependency relations

Stanford parser is used to parse the queries. It generates Stanford dependencies. Stanford dependencies use 56 grammatical relations to represent grammatical relations among tokens. Parsing is used to find grammatical relations among bibliographic named entities represented by tokens.
Queries which have citations are divided into two parts, a citing part and a cited part. For example a query "*papers about information retrieval that were cited by papers written by John*" is divided into two parts "*papers about information retrieval*" and "*papers that were written by John*". We do this in order to reduce the complexity of parsing a large query.

### 3.1.5 Generation of graph relations

An algorithm which has been specified by the paper has been used to generate graph relations from graph dependencies. The algorithm suggests that a relation should be selected if both the subject and object of the relation are named entities and the dependency relation is not a conjunction.
Candidate relations are also created. If the subject or object of the dependency relation is a named entity and the dependency relation is either "acl:relcl", "nmod" or "dobj" then add the relation to a candidate relation list. For every candidate relation initialize an empty tuple and add the subject of "acl:relcl" and the object of "nmod" or "dobj" relation to the tuple. Then add the tuple to the graph relation.

### 3.1.6 Formulation of graph queries

Here the bibliographic named entities are converted to graph nodes and graph nodes are checked for connectedness and direction. The cited and citing parts are also integrated in this step.
At the end we have to convert a graph query to a graph language that can be given to the graph database. Neo4j graph database and Cypher graph language has been used. The graph relations generated are used to derive patterns, a constraint is derived from a constraint node and the return type is a answer node, with these the query can be made.

## 3.2 Querying a Graph Database

After obtaining the query the query is submitted to Neo4j graph database. The querying language used is Cypher.

# 4 Implementation

## 4.1 Work Done

A natural language interface to a graph based bibliographic system was implemented.
The conversion of a natural query to a query language involves various steps like formulation of the natural language queries, recognition and extraction of bibliographic named entities, tokenization of natural language queries, extraction of grammatical errors, generation of graph relations from graph dependencies, formulation of graph graph queries, converting of bibliographic named entities to graph nodes, checking connectedness and direction of graph relations, integrating the cited and citing parts, translation of a graph query into a graph query language.
After these steps are done we use the query to query a graph database.

### 4.1.1 Creation of a dictionary

A dictionary is created by preprocessing the [1] dataset. The papers, authors, term, sources and organizations from the dataset were identified and kept as keys and their corresponding named entities i.e. paper, author, term, source and organization are kept as values. Example ('*Information Retrieval*','*paper*') where (*key*,*value*). Five more entries are added to the dictionary, the entries being ('*paper*','*class_paper*'), ('*author*','*class_author*'), ('*term*','*class_term*'), ('*source*','*class_source*') and ('*organization*','*class_organization*').
Using this dictionary named entities are obtained. The keys of the dictionary are the named entities.
The named entities derived are further used for tokenization.

### 4.1.2 Tokenization of the Query

The user query is tokenized by looking-up the previously created dictionary. First bigram words are detected by dividing the query into two-two words are searching if that pair is present in the dictionary further the sentence is divided into one word each omitting the words already classified as named entities i.e. tokens. These unigram words are searched in the dictionary to see if it is a named entity. After this procedure tokens which are named entities from the user query are obtained.

### 4.1.3 Deriving dependency relations

The dependency relations are a set of 56 grammatical relations which are used to derive grammatical relations among tokens. These dependencies are directly generated from the Stanford Parser.

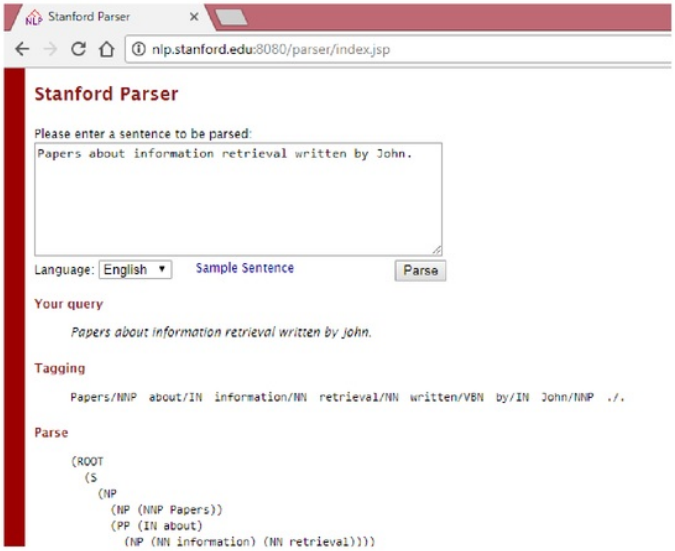



Figure 2: Stanford Parser

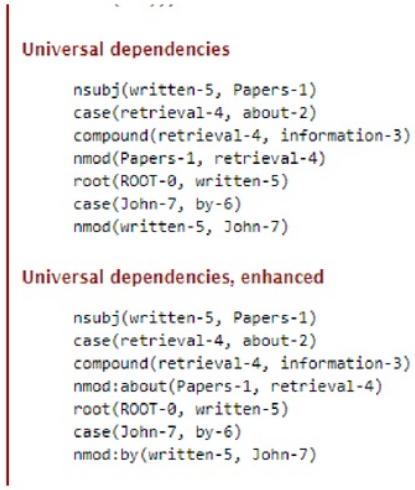



Figure 3: Universal Dependencies

We have our query to the Stanford Parser and the enhanced dependencies were

scraped via using BeautifulSoup.
From the scraped text, dependencies were made. These dependencies will be used further to derive graph relations.

#### 4.1.4 Generation of graph relations

Graph relations were generated using the below algorithm.

INPUT: DR (dependency relations): A list of triples including the subject, relation, and object of a relation
OUTPUT: GR (graph relations): A list of tuples including the subject and object of a relation

INITIALIZE CR (candidate relations): An ordered list of dependency relations
FOR each ***dependency relation*** in DR
  IF ***subject*** and ***object*** of ***dependency relation*** are named entities THEN
    IF ***dependency relation*** is NOT "conj" THEN
      initialize an empty tuple and add ***subject*** and ***object*** to the tuple
      add the tuple to GR
    ENDIF
  ENDIF
  IF ***subject*** or ***object*** of ***dependency relation*** is a named entity THEN
    IF ***dependency relation*** is "acl:relcl" THEN
      add ***dependency relation*** to CR
    ENDIF
    IF ***dependency relation*** is "nmod" or "dobj" THEN
      add ***dependency relation*** to CR
    ENDIF
  ENDIF
ENDFOR

FOR every two **candidate relations** in CR
  initialize an empty tuple
  add the ***subject*** of "acl:relcl" relation and the ***object*** of "nmod" or "dobj" relation to the tuple
  add the tuple to GR
ENDFOR

Figure 4: Algorithm for generation of graph relations

#### 4.1.5 Converting bibliographic named entities to graph nodes

First, we identify the bibliographic named entity that the query is asking for and is labeled as answer node. The answer node is the root node obtained from the set of dependency relations. Second, each bibliographic named entity is assigned a unique instance that will be used when generating a graph query language. Third,

the constraint nodes are identified. If the type of the named entity does not contain the string "*class_*" the named entity is a constraint node.

INPUT: NE (named entities), DR (dependency relations)
OUTPUT: GN (graph nodes)

FOR each ***named entity*** in NE
  IF the ***dependency relation*** of ***named entity*** in DR is "root" THEN
    SET node type of ***named entity*** as "Answer Node"
  ENDIF
  IF the type of ***named entity*** NOT includes the string "class_" THEN
    SET node type of ***named entity*** as "Constraint Node"
  ENDIF
  SET a unique instance name to ***named entity***
  add ***named entity*** to GN
ENDFOR

Figure 5: Algorithm for converting named entities to graph nodes

#### 4.1.6 Checking connectedness and direction

The named entity should be connected and confer to the database schema. The database schema is shown below.

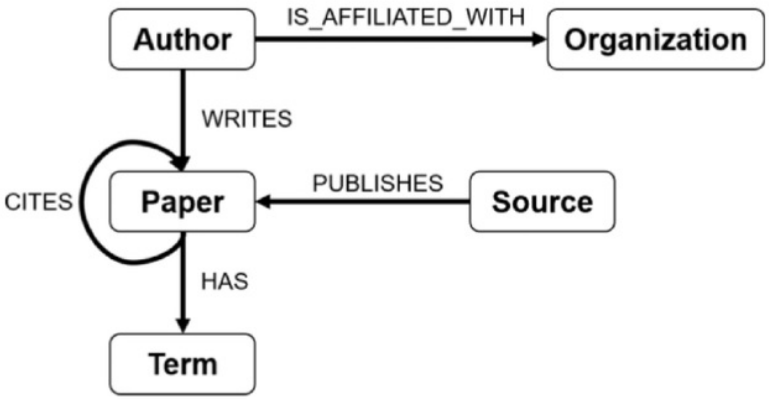



Figure 6: Algorithm for generation of graph relations

Connectedness determines whether the entities are connected in the database schema. They should be connected in such a way that they confer to the schema. The direction is checked and is reversed according to the schema. If two entities don't have a direct connection in accordance to the schema extra nodes are added. These nodes are one of the five named entities, connections are now made to suit the database schema.

### 4.1.7 Integration of citing and cited parts

The cited and citing parts are integrated together using the below algorithm.

```
INPUT: CN1 (cited nodes), CR1 (cited relations), CN2 (citing nodes), CR2 (citing relations)
OUTPUT: GN (graph nodes), GR (graph relations)

Add CN1 and CN2 to GN
IF CN1 and CN2 include nodes with the type of Paper THEN
   initialize an empty graph relation
   SET the subject of the graph relation Paper node in CN2
   SET the object of the graph relation Paper node in CN1
   add the graph relation to GR
ENDIF
IF CN1 or CN2 includes a node with the type of Paper THEN
   initialize an empty graph relation
   IF CN1 includes a node with the type of Paper THEN
      SET the object of the graph relation Paper node in CN1
      create a graph node with the type of Paper, add the graph node to GN, set it as the subject of the graph relation
      add the graph relation to GR
   ENDIF
   ELSE
      SET the subject of the graph relation Paper node in CN2
      create a graph node with the type of Paper, add the graph node to GN, set it as the object of the graph relation
      add the graph relation to GR
ENDIF
IF CN1 and CN2 NOT include nodes with the type of Paper THEN
   initialize an empty graph relation
   create a graph node with the type of Paper, add the graph node to GN, set it as the subject of the graph relation
   create a graph node with the type of Paper, add the graph node to GN, set it as the object of the graph relation
   add the graph relation to GR
ENDIF
```



Figure 7: Algorithm for integrating cited and citing parts

#### 4.1.8 Converting graph query into a graph query language

The graph query is changed into a graph query language using patterns, answer node and constraint nodes. The graph used is Cypher. Cypher language is used to query Neo4j database.

INPUT: GN (graph nodes), GR (graph relations)
OUTPUT: GQL (graph query language)

FOR each ***graph node*** in GN
  IF node type of ***graph node*** is "Answer Node" THEN
    add ***graph node*** to RETURN clause of GQL
  ENDIF
  IF node type of ***graph node*** is "Constraint Node" THEN
    add ***graph node*** to WHERE clause of GQL
  ENDIF
ENDFOR
FOR each ***graph relation*** in GR
  extract instance names of subject and object of ***graph relation***
  attach types of subject and object of ***graph relation*** to instances names
  connect two instance names and add to GQL
ENDFOR

Figure 8: Algorithm for generating graph query language

#### 4.1.9 Creating a graph database

The dataset is converted to a csv file and the csv file is feed into Neo4j. Relations are created using Cypher according to the provided database schema.

#### 4.1.10 Querying from Neo4j

The obtained query is given to Neo4j database and results are obtained.

Query- Papers that were written by John

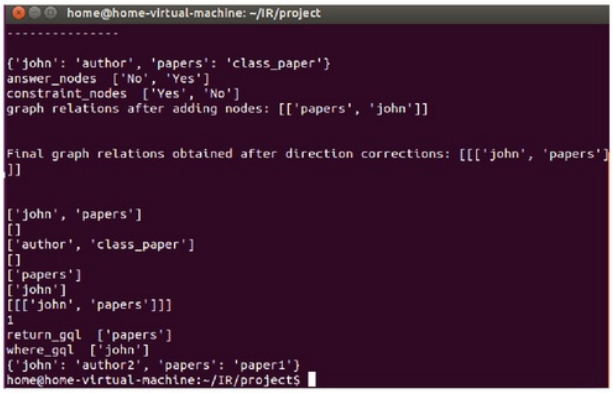



Figure 9: Cypher Query Generation 1

Query- Papers that were published in ACM.

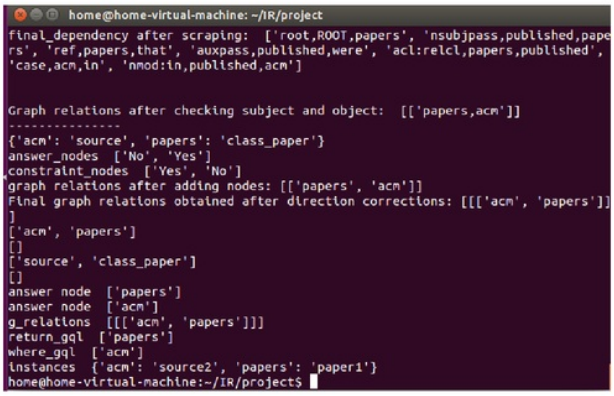



Figure 10: Cypher Query Generation 2

## 4.2 Results and Analysis

The final graph database is shown below.

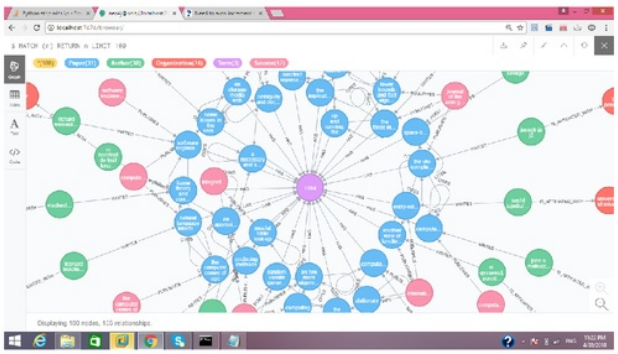


Figure 11: Graph Database

When a user enters queries into Neo4j in the form of Cypher language. The below results are displayed. The input has returned proper results when its given to the graph database or most of the queries.

Query- Papers that were written by John
Results- 25 retrieved all relevant


Figure 12: Query Result 1

Query- Authors of Information Retrieval
Results- 3 retrieved all relevant


Figure 13: Query Result 2

Query- Organization of David
Results - 18 retrieved all relevant


Figure 14: Query Result 3

Query- Papers that were published in ACM.
Results- 49 retrieved all relevant


Figure 15: Query Result 4

Query- Papers about operating systems that were published in IEEE.
Results- 1 retrieved which is relevant

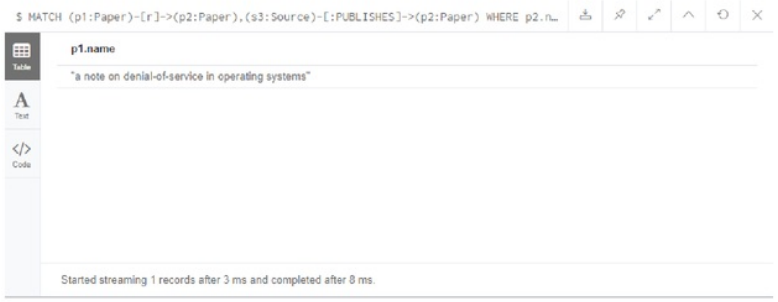



Figure 16: Query Result 5

## 4.3 Innovative Work

The interface doesn't return any results for queries like 'Information retrieval'. Hence we have added a new node 'paper' to mapped class of query. The relation is again checked for connectedness and directions. After the final query is given to the graph database, it results all papers on information retrieval.

## 4.4 Individual Contribution

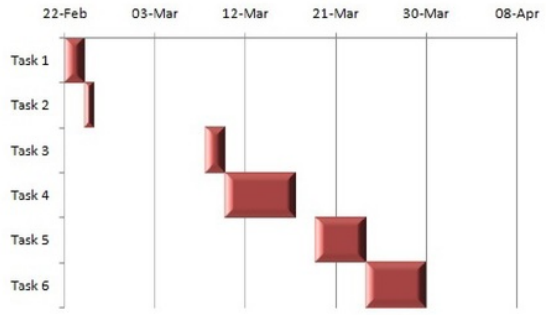



Figure 17: Ghantt Chart

**Extraction of dependency relations** - Shreya Shetty
**Generation of graph relations** - Rishita Golla
**Formulation of graph queries** - Snigdha Taduru

# 5 Conclusion and Future Work

In this project, we implemented a natural language interface for searching bibliographic data from the ARNETMinor dataset. A NLI-GIBIR interface is a natural language interface to a graph based bibliographic data. One of the vital steps in interpreting the natural language queries is to recognize the named entities of the bibliographic data in the natural language query. This structure not only achieves this but also finds dependency among those recognized entities. It is presumed that the natural language interface to graph databases will improve appreciably as the underlying technologies like of natural language processing advance. Such interfaces will help users overcome the difficulty in productively searching and navigating within the humongous and ever-growing collection of scientific publications.

On the basis of few of the limitations identifies in this project the following can be the future work on this project. First , query expansion techniques can be used to incorporate any format and vocabulary of natural language query. Second, Stanford NER Parser library can be incorporated in the python code instead of scraping from the web. Third, a better data set can be used where double queries can be executed like the example given above and more importance can be given to user testing. In this project the testing was done by only the project group members. Finally, the query given in natural language is converted to a graph query and the instances for the graph query is displayed however the query is not given in the format required for the Neo4j database. The conversion to the format of Neo4j query can be considered as future work.

## References


[1] J. Tang, J. Zhang, L. Yao, J. Li, L. Zhang, & Z. Su, Arnetminer: extraction and mining of academic social networks, in: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, ACM, 2008, pp. 990998.

[2] F. Li, H.V. Jagadish, Constructing an interactive natural language interface for relational databases, Proceed. VLDB Endow. 8 (1) (2014) 7384.

[3] A.B. Abacha, P. Zweigenbaum, MEANS: a medical question-answering system combining NLP techniques and semantic web technologies, Inform. Process. Manag. 51 (5) (2015) 570594. http://dx.doi.org/10.1016/j.ipm.2015.04.006.

[4] E. Kaufmann, A. Bernstein, Evaluating the usability of natural language query languages and interfaces to Semantic Web knowledge bases, Web Semant.: Sci., Serv. Agents World Wide Web 8 (4) (2010) 377393.

[5] E. Miller, An introduction to the resource description framework, Bull. Am. Soc. Inform. Sci. Technol. 25 (1) (1998) 1519.

[6] Y. Zhu, E. Yan, I.-Y. Song, The use of a graph-based system to improve bibliographic information retrieval: system design, implementation, and evaluation, J. Assoc. Inform. Sci. Technol. 68 (2) (2016) 480490. http://dx.doi.org/10.1002/asi.23677

[7] P. Ryu, M. Jang, H. Kim, Open domain question answering using wikipedia-based knowledge model, Inform. Process. Manag. 50 (5) (2014) 683692. http://dx.doi.org/10.1016/j.ipm.2014.04.007

[8] X. Mu, K. Lu, H. Ryu, Explicitly integrating MeSH thesaurus help into health information retrieval systems: an empirical user study, Inform. Process. Manag. 50 (1) (2014) 2440. http://dx.doi.org/10.1016/j.ipm.2013.03.005.

[9] Y. Tsuruoka, J.I. Tsujii, Boosting precision and recall of dictionary-based protein name recognition, in: Proceedings of the ACL 2003 Workshop on Natural Language Processing in Biomedicine, Association for Computational Linguistics, Volume 13, 2003, pp. 4148.

[10] T.C. Rindesch, L. Tanabe, J.N. Weinstein, L. Hunter, EDGAR: extraction of drugs, genes and relations from the biomedical literature. in: Proceedings of Pacic Symposium on Biocomputing, Vol. 5, 2000, pp. 514525.

# ir

ORIGINALITY REPORT

| 27% | 9% | 23% | % |
|---|---|---|---|
| SIMILARITY INDEX | INTERNET SOURCES | PUBLICATIONS | STUDENT PAPERS |

PRIMARY SOURCES

| | | |
|---|---|---|
| 1 | Yongjun Zhu, Erjia Yan, Il-Yeol Song. "A natural language interface to a graph-based bibliographic information retrieval system", Data & Knowledge Engineering, 2017<br>Publication | 18% |
| 2 | maslopo.blogspot.com<br>Internet Source | 1% |
| 3 | www.ripublication.com<br>Internet Source | 1% |
| 4 | upcommons.upc.edu<br>Internet Source | 1% |
| 5 | www.absoluteastronomy.com<br>Internet Source | 1% |
| 6 | documents.mx<br>Internet Source | 1% |
| 7 | nlp.kiv.zcu.cz<br>Internet Source | 1% |
| 8 | www.scribd.com<br>Internet Source | 1% |

| | | |
|---|---|---|
| 9 | dspace.thapar.edu:8080<br>Internet Source | 1% |
| 10 | zenodo.org<br>Internet Source | <1% |
| 11 | www.ww.htwm.de<br>Internet Source | <1% |
| 12 | www.aou.edu.lb<br>Internet Source | <1% |
| 13 | dblp.dagstuhl.de<br>Internet Source | <1% |
| 14 | aclweb.org<br>Internet Source | <1% |
| 15 | kmi.open.ac.uk<br>Internet Source | <1% |
| 16 | ijarcsse.com<br>Internet Source | <1% |
| 17 | Parssian, Amir, William Yeoh, and Mong Shan Ee. "Quality-Based SQL: Specifying Information Quality in Relational Database Queries", Computer, 2015.<br>Publication | <1% |
| 18 | etd.lib.metu.edu.tr<br>Internet Source | <1% |
| | www.slideshare.net | |

19 Internet Source

<1%

| | | | |
|---|---|---|---|
| Exclude quotes | On | Exclude matches | < 3 words |
| Exclude bibliography | On | | |