RESTFUL API FOR INTEGRATIVE LITERATURE INFORMATION AND KNOWLEDGE SERVICE

by

Xu Zhu

A thesis submitted to the Faculty of the University of Delaware in partial fulfillment of the requirements for the degree of Master of Science in Bioinformatics and Computational Biology

Summer 2017

RESTFUL API FOR INTEGRATIVE LITERATURE INFORMATION AND KNOWLEDGE SERVICE

by

Xu Zhu

Approved: ____
Cathy H. Wu, Ph.D.
Professor in charge of thesis on behalf of the Advisory Committee

Approved: ____
Kathleen F. McCoy, Ph.D.
Chair of the Department of Computer and Information Sciences

Approved: ____
Babatunde Ogunnaike, Ph.D.
Dean of the College of Engineering

Approved: ____
Ann L. Ardis, Ph.D.
Senior Vice Provost for Graduate and Professional Education

ACKNOWLEDGMENTS

I wish to thank my adviser, Dr. Cathy H Wu; and my committee members Dr. Chuming Chen, and Dr. Li Liao for their continuous advice, guidance, and academic support during the past two years. I must also thank my professional friends and colleagues, Jia Ren, Gang Li, and M. Joseph Tomlinson IV, who have supported and helped me throughout my graduate education.

This manuscript is dedicated to my parents, Fanzhu Zhu, Fang Sun and Yu Zhu for their unconditional love; also to my friend Xi Jiao for her motivating and inspiring.

TABLE OF CONTENTS

Appendix

LIST OF FIGURES

LIST OF TABLES

ABSTRACT

Biocuration needs efficient and high-quality data sources as well as automatic literature curation as the result of dramatically increasing the volume of literature in life science and biology fields. API is a stable and scalable mechanism to deliver programmable data, which can become the data source for biocuration and data presentation. RESTful API for Integrative Literature Information and Knowledge Service (abbreviated to iLINKS) is designed to expose the text-mining data gained from analyzing PubMed articles to support potential application development. The iLINKS system integrated three text-mining tools, RLIMS-P, miRTex and PubTator to capture phosphorylation relation, gene-miRNA relation and biological relation in general. With the data generated by those text-mining tools, three different types of APIs are built to deliver PubMed article full information, relation-specific information, and across-article entity information. Following the best practice of API design principle as well as the business rules in iLINKS system, the APIs are used to deliver text-mining results along with PubMed article abstract/full text. API documentation website is set to help developers to develop applications using the APIs. An example PubMed ID based search web page can help biocurators quickly searching for biological entities and relations for a given PubMed article. The

RESTful APIs will be widely used for developing other iLINKS applications, which include Cytoscape knowledge network visualization, and BRAT literature annotation.

Chapter 1

INTRODUCTION

Biocuration is a data collection process including identification, interpretation, and integration of biological information, and reformat data into biological databases. [1] Computational biology is an analytical technique built on biological data from experiments and literature curation, whose availability and quality will influence the downstream analysis results and conclusion [2]. The information curated from literature can also be set as the reference and standard for related computational analysis and high-throughput data assembly [3]. Since the middle 1990s, the dramatic increase in the volume of life science literatures has published the urgent need for intelligent tools to extract information and convert it to consistent format [2]. Providing accessible and programmable data is another highly demanded function to help researchers gain insight of the certain topic rapidly, so data integration and presentation are also crucial in biocuration progresses [3].

API (Application Programming Interface) is a practical and scalable solution for efficient biocuration data accessing methods. [4] Following the transformation of Web 2.0, REST (Representational state transfer) API became the main trend of building web APIs. [5] [6] There are three reasons why RESTful API service is suitable to handle the current difficulties we are facing in biocuration. First, the driver of API is exposing capabilities, which matches the purposes of curators in terms of delivering data content and functions. Second, API is a great support for the developers to integrate data or functions into their applications, which helps to push biocuration into a higher engineering level comparing to manual curation. Third, from the data providers' point of view, providing API access can boost their business by having a wider group of customers and users. So building RESTful APIs for Integrative Literature Information and Knowledge Service (abbreviated to iLINKS) is chosen as my Master degree thesis project. (iLINKS Text-mining System, the Center for Bioinformatics & Computational Biology (CBCB), University of Delaware.)

The thesis is organized as follows: Chapter 1 introduces the motivation of the thesis and describes the high-level structure of the thesis. Chapter 2 presents the basic background knowledge for building APIs and related work. Chapter 3 describes the important role API plays in the iLINKS text-mining system project and presents the API design principle and API endpoints descriptions. Chapter 4 focuses on API implementation and documentation. Chapter 5 introduces example use case as searching and returning data tables populated by the APIs. Chapter 6 discusses the potential development directions for iLINKS API version 2.0; Chapter 7 concludes the thesis and highlights the contribution of this thesis project.

Chapter 2

BACKGROUND

As a way to expose the data integrated in iLINKS text-mining database, RESTful API is chosen as the media to deliver the data to all external users. Different from web user interfaces (WUI) that has been widely used and viewed by users with a web browser, API, application program interface, is a programmable data or function delivery service. The consumers of APIs would mainly be developers and engineers. [7] Programming and developing skills and expertise are required for API endpoints users, but all those steps are taken to ultimately benefit a larger group of users. Because this thesis project is exclusively designing APIs to deliver PubMed article text content and text-mining results as the data objects, in this chapter, RESTful APIs and their applications in biology/bioinformatics field are firstly demonstrated, then followed by some similar API designs that are also aiming for exposing PubMed article and text-mining function.

2.1 RESTful API

iLINKS text-mining system processes PubMed articles and stores text-mining results into a relational database. To present data in a programmable format, and make the text-mining results available for programmatic access, data in the relational database are processed and published as APIs. API is a media that allows application/software to directly interact with the backend database. Some applications can use multiple APIs from different resources, and good APIs can organize data into building blocks that are easier to interact. REST (also called RESTful web service) is applied by HTTP request and mainly doing textual representations access and manipulation, such as GET, PUT, POST, DELETE and so on. [8] Those textual representations of web resources could be documents or URL-identified files, which can be read through by a browser. Many large-scale websites are using RESTful APIs, such as, Google, Amazon, Twitter and LinkedIn.

Most of the biological databases and sources provide APIs as the external accesses for part of their data or data processing functions. For example, NCBI (National Center for Biotechnology Information) has entity-centric databases like Gene, Protein and Probe, article-centric databases like PubMed and PubMed Central, also with functions like Blast, and those data and functions are also accessible through NCBI E-Utilities API system. [9] Other biology-related API sources are like KEGG, tools like PubTator, Cytoscape, and some APIs provided by research groups like PSAMM API from Computational Molecular Ecology Lab at the University of Rhode Island. [10] With more and more APIs offered in biotechnology and bioinformatics field, researchers and developers who want to build their own web applications or mobile applications can directly use the information or functions provided by those online biological sources. And with other general APIs in computer and software engineering field, they can also combine different APIs from different sources to increase app's functionality. Ultimately, as part of the research group, we want to design and build our own APIs for both internal and external use.

2.2 Related work

Among the two hundred open source science APIs, there are more than one hundred API sources are related to biology field, including bioinformatics (58 out of 195), genetics (44 out of 195) and biology (35 out of 195), according to the data collected by programmableweb.com. [11] NCBI Entrez API (E-Utilities) is the most popular one with different functions and databases. One NCBI API access (named BioC API for PubMed) that provides PubMed article data is directly related to this project. In addition, PubTator also developed REST API as one of their new features. [12] The three APIs (BioC API for PubMed, PubTator REST API, and iLINKS API in this project) all use PubMed articles as data source. The main differences among the three APIs are, BioC API for PubMed contains only article information, PubTator REST API and iLINKS API contain text-mining data. The difference between PubTator REST API and iLINKS API are, in data content aspect, iLINKS API contains the data generated from PubTator and two other text-mining tools (RLIMS-P and miRTex), but PubTator REST API only has PubTator text-mining as the data source; in API structure aspect, iLINKS API has three different API structures but PubTator REST API only has one.

BioC API for PubMed

PubMed has several article presentation views, including common web view, XML, MEDLINE and plain text. The access URL is defined in this format: https://www.ncbi.nlm.nih.gov/pubmed/?term={PMID}&report={format}&format=text

Two parameters can be modified by users, one is {PMID}, which should be replaced with PubMed ID (for example, 19330006), the other is {format} which can be Summary, Summary (text), Abstract – the PubMed basic page, Abstract (text), MEDLINE, XML and PMID List.

BioC API for PubMed is a RESTful API developed by Zhiyong Lu team, as part of the NCBI text-mining service. [13] [14] The data source for this API is PubMed database, and the API only displays article text and citation information without integrating any text-mining or natural language processing (NLP) functions. The endpoint for this API is defined as https://www.ncbi.nlm.nih.gov/bionlp/RESTful/pubmed.cgi/BioC_{format}/{PMID}/{encoding}

Three parameters can be customized by users, one is {format} with available value 'json' and 'xml', another one is {PMID} for PubMed ID, and the last one is {encoding} with available value 'unicode' or 'ascii'. It is quite straightforward to know how this endpoint works, even though there are no testing cases or any other detailed documentation (with interactive try out).

For sample code (with parameters set as format = json, PMID = 19330006, encoding = unicode), the request and response are shown below.

Request:

Method:

GET

Endpoint:

https://www.ncbi.nlm.nih.gov/research/bionlp/RESTful/pubmed.cgi/BioC_json/19330006/unicode

Respond:

```
{
  "date": "20170414",
  "source": "PubMed",
  "infons": {},
  "documents": [
    {
      "infons": {},
      "passages": [
        {
          "text": "A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX
in neural stem cell fate determination.",
          "offset": 0,
          "relations": [],
          "infons": {
            "type": "title"
          },
          "sentences": [],
          "annotations": []
        },
        {
          "text": "MicroRNAs have been implicated as having important roles in stem cell
biology. MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and
may be involved in neural stem cell self-renewal and differentiation. We showed previously
that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal.
Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell
proliferation and accelerate neural differentiation. Introducing a TLX expression vector
that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and
inhibited precocious differentiation. In utero electroporation of miR-9 in embryonic brains
led to premature differentiation and outward migration of the transfected neural stem cells.
Moreover, TLX represses expression of the miR-9 pri-miRNA. By forming a negative regulatory
loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell
proliferation and differentiation.",
          "offset": 113,
          "relations": [],
          "infons": {
            "type": "abstract"
          },
          "sentences": [],
          "annotations": []
        }
      ],
      "id": "19330006",
      "relations": []
    }
  ],
  "key": "collection.key"
}
```

Figure 2.1: Example BioC API for PubMed (Endpoint: https://www.ncbi.nlm.nih.gov/research/bionlp/RESTful/pubmed.cgi/BioC_json/19330006/unicode)

The API contains basic document information (no citation information), title text, and abstract text. Comparing to PubTator API and iLINKS API in this project, this article text API stores article title and abstract separately, which on one hand is convenient for displaying the article text (title and abstract structure) on the web page, but on the other hand requires extra steps if users/developers want to process title text and abstract text together.

PubTator REST API

PubTator, as introduced in the previous chapter, is an online text-mining tool with search and annotation functions based on NCBI PubMed database. The text-mining/annotation data, along with the original article is open for FTP download (ftp://ftp.ncbi.nlm.nih.gov/pub/lu/PubTator/), and PubTator developer team also added REST API access recently (with brief documentation, without interactive try out). [14] The data source for this API is still PubMed database, and the biological concepts that can be recognized by PubTator integrated text-mining tools are gene, disease, specie, mutation, and chemical component. The endpoint for this API is defined as https://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/RESTful/tmTool.cgi/{bio-concept}/{PMID}/{format}/

There are three parameters, one is {bio-concept} with available values 'Gene', 'Disease', 'Chemical', 'Species', 'Mutation' and 'BioConcept' (for all included); {PMID} is simple PubMed ID, and {format} can be 'PubTator', 'BioC', and 'PubAnnotation' (or 'JSON').

The sample code (with parameters set as bio-concept = BioConcept, PMID = 19330006, format = PubAnnotation), the request and response are shown below.

Request:

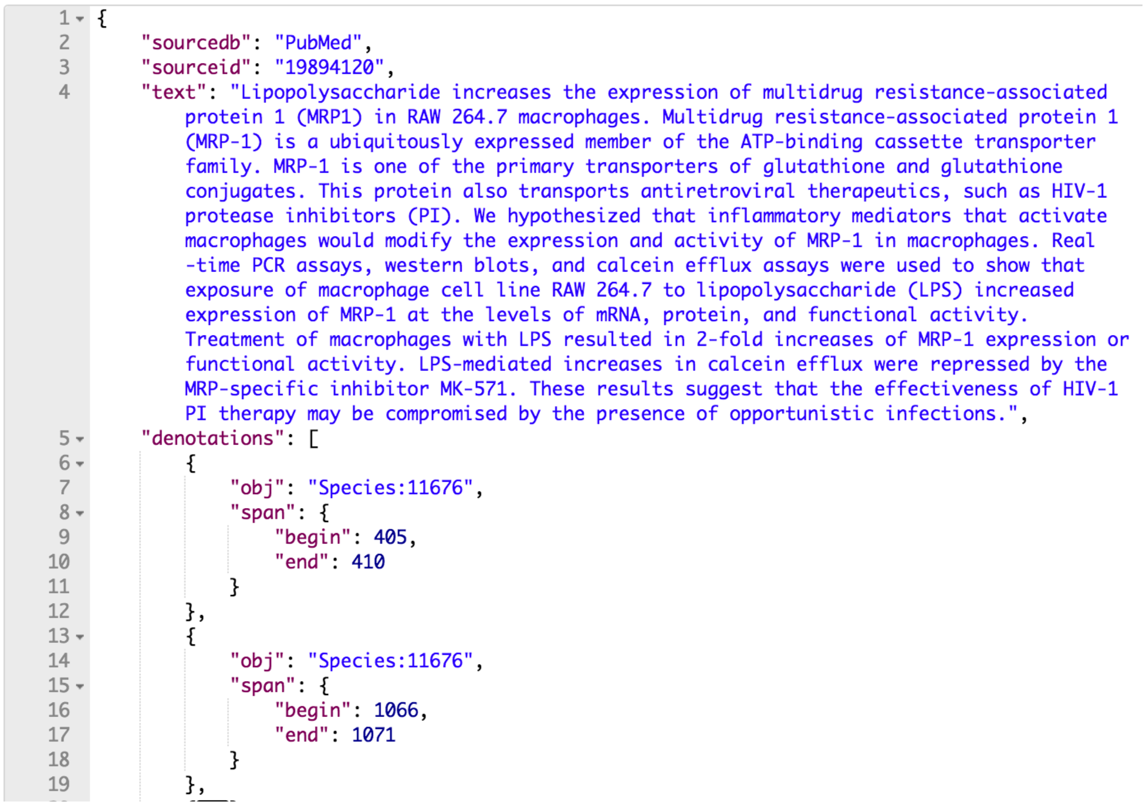
Method:

GET

Endpoint:

https://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/RESTful/tmTool.cgi/BioConcept/19894120/PubAnnotation/

Response: (not full data)



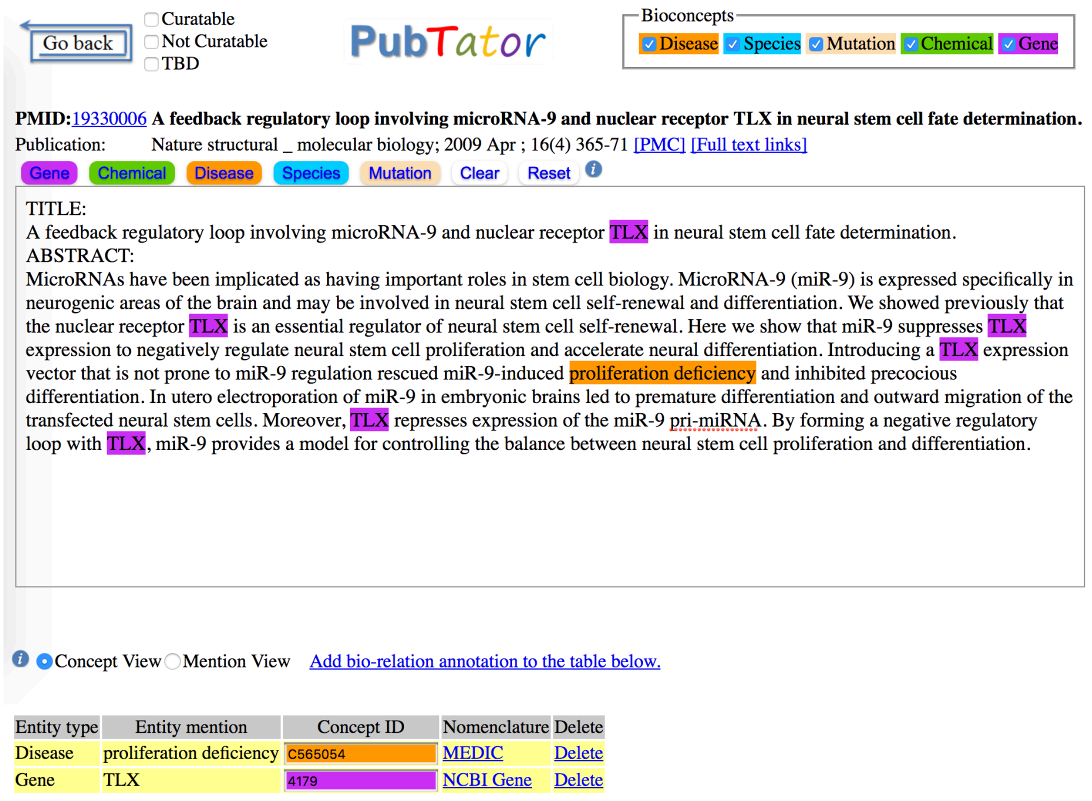
```
{
    "sourcedb": "PubMed",
    "sourceid": "19894120",
    "text": "Lipopolysaccharide increases the expression of multidrug resistance-associated
        protein 1 (MRP1) in RAW 264.7 macrophages. Multidrug resistance-associated protein 1
        (MRP-1) is a ubiquitously expressed member of the ATP-binding cassette transporter
        family. MRP-1 is one of the primary transporters of glutathione and glutathione
        conjugates. This protein also transports antiretroviral therapeutics, such as HIV-1
        protease inhibitors (PI). We hypothesized that inflammatory mediators that activate
        macrophages would modify the expression and activity of MRP-1 in macrophages. Real
        -time PCR assays, western blots, and calcein efflux assays were used to show that
        exposure of macrophage cell line RAW 264.7 to lipopolysaccharide (LPS) increased
        expression of MRP-1 at the levels of mRNA, protein, and functional activity.
        Treatment of macrophages with LPS resulted in 2-fold increases of MRP-1 expression or
        functional activity. LPS-mediated increases in calcein efflux were repressed by the
        MRP-specific inhibitor MK-571. These results suggest that the effectiveness of HIV-1
        PI therapy may be compromised by the presence of opportunistic infections.",
    "denotations": [
        {
            "obj": "Species:11676",
            "span": {
                "begin": 405,
                "end": 410
            }
        },
        {
            "obj": "Species:11676",
            "span": {
                "begin": 1066,
                "end": 1071
            }
        },
```

Figure 2.2: Example PubTator REST API (Endpoint:

https://www.ncbi.nlm.nih.gov/CBBresearch/Lu/Demo/RESTful/tmTool.cgi/BioConcept/

19894120/PubAnnotation/)

The iLINKS full-information API has similar structure as PubTator REST API, which consists of article information (source and ID), article text (title and abstract together), and annotation information (with start and end character positions within the article text). This structure is simple and convenient to work with when each biological entity is marked by a unique ID. The API can be directly used by annotation tools such as Brat Rapid Annotation Tool (http://brat.nlplab.org/).


Go back
☐ Curatable
☐ Not Curatable
☐ TBD
PubTator
Bioconcepts
✓ Disease ✓ Species ✓ Mutation ✓ Chemical ✓ Gene

PMID:19330006 **A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination.**
Publication: Nature structural _ molecular biology; 2009 Apr ; 16(4) 365-71 [PMC] [Full text links]

Gene Chemical Disease Species Mutation Clear Reset

TITLE:
A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination.
ABSTRACT:
MicroRNAs have been implicated as having important roles in stem cell biology. MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation. We showed previously that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal. Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation. Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation. In utero electroporation of miR-9 in embryonic brains led to premature differentiation and outward migration of the transfected neural stem cells. Moreover, TLX represses expression of the miR-9 pri-miRNA. By forming a negative regulatory loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and differentiation.

[● Concept View] ○ Mention View Add bio-relation annotation to the table below.

Entity type	Entity mention	Concept ID	Nomenclature	Delete
Disease	proliferation deficiency	C565054	MEDIC	Delete
Gene	TLX	4179	NCBI Gene	Delete

Figure 2.3: Example PubTator annotation webpage view (19330006 as PubMed ID)

From the aspect of article data display and annotation, PubTator has its built-in article annotation view, which can be the reference development direction for iLINKS API usage. This potential usage also influenced the design of iLINKS APIs. [15]

Regarding three APIs (BioC API for PubMed, PubTator REST API and the RESTful API for iLINKS text-mining system as this thesis project), they are all PubMed article centric data access APIs. When comparing BioC API for PubMed and RESTful API for iLINKS text-mining system, BioC API for PubMed only contains PubMed article text and article related information and there is no text-miming data included; RESTful API for iLINKS text-mining system, on the other hand, contains PubMed article text, article information, and text-miming data. When comparing PubTator REST API and the RESTful API for iLINKS text-mining system, RESTful API for iLINKS text-mining system has two special target-specific text-mining tools (RLIMS-P and miRTex) that PubTator system does not have. And one unique feature the RESTful APIs for iLINKS text-mining system have is that there is one subtype of APIs that accesses the information across multiple PubMed articles as they have been integrated into the underlying relational database. In other words, BioC API for PubMed and PubTator REST API only have APIs for single PubMed article, but RESTful APIs for iLINKS text-mining system have APIs that can deliver related information from multiple PubMed articles.

Chapter 3

API DESIGN

API design in general follows some universal design principles and patterns, and there always is a trade-off between flexibility and usability. Flexibility indicates that API should contain more than one function and is able to support different use cases; usability indicates that if the API can serve its basic purpose and be consumed by users properly. The increase in flexibility will add complexity to API, which may have a negative impact on the user experience and make the API hard to use. [16] [17] For this project, we are targeting the specific field - biocuration, so instead of trying to design a mashup API with multiple functionalities, we want to create simple data delivery APIs. [18] Since data is the main component of the APIs, in this chapter, the data source is introduced first, followed by API design principles and API service description.

3.1 iLINKS text-mining system

Before diving into the API design, we want to present the high-level architecture of iLINKS text-mining system. The data source of the APIs is also a big component of the whole pipeline. RESTful API plays an important role in overall data flow. It supports data manipulation to backend database to keep the system stable using self-checking method; it also supports potential client application development.

3.1.1 Architecture

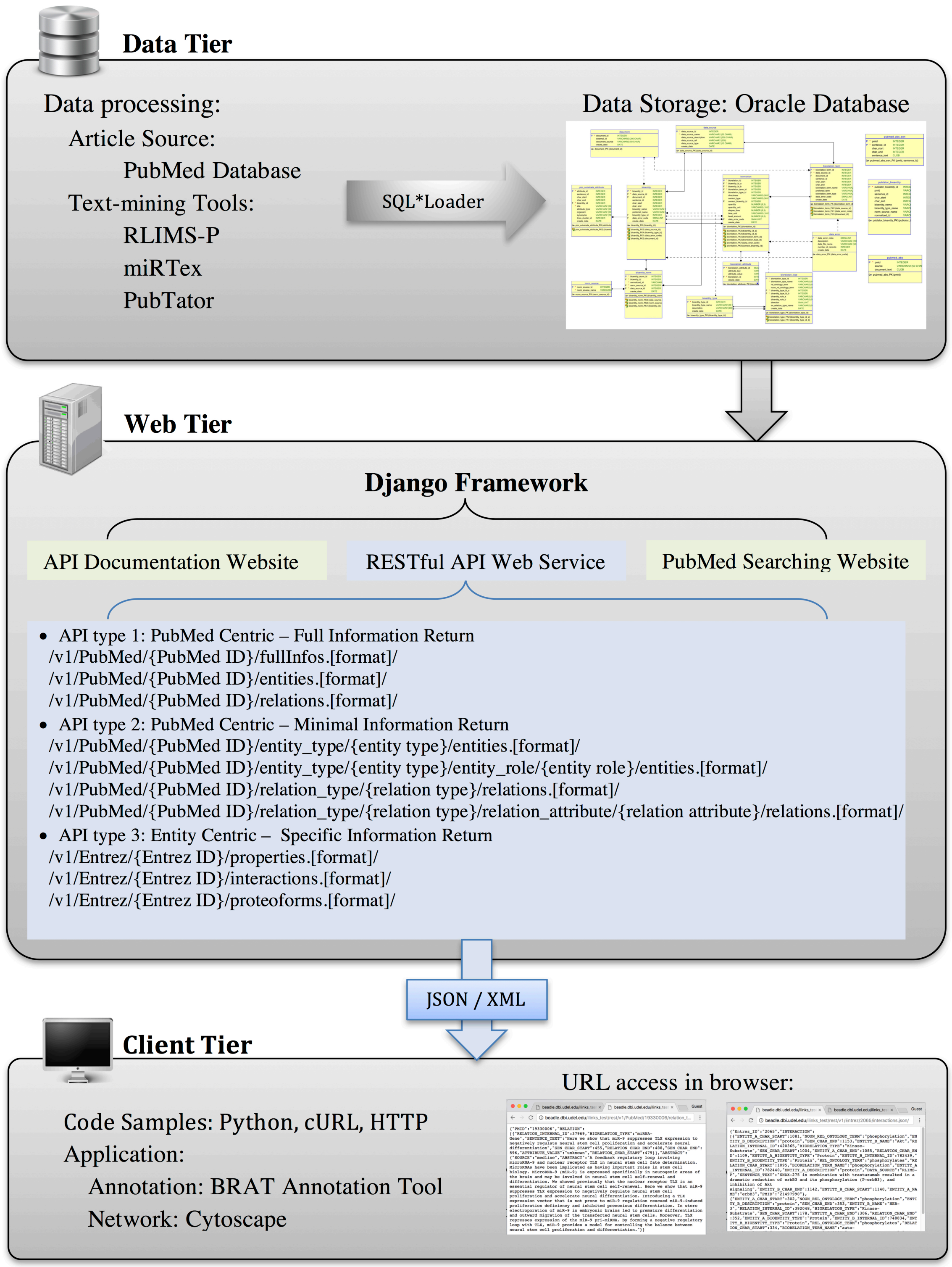

Data Tier
Data processing:
Article Source:
PubMed Database
Text-mining Tools:
RLIMS-P
miRTex
PubTator
SQL*Loader
Data Storage: Oracle Database
Web Tier
Django Framework
API Documentation Website
RESTful API Web Service
PubMed Searching Website
API type 1: PubMed Centric – Full Information Return
/v1/PubMed/{PubMed ID}/fullInfos.[format]/
/v1/PubMed/{PubMed ID}/entities.[format]/
/v1/PubMed/{PubMed ID}/relations.[format]/
API type 2: PubMed Centric – Minimal Information Return
/v1/PubMed/{PubMed ID}/entity_type/{entity type}/entities.[format]/
/v1/PubMed/{PubMed ID}/entity_type/{entity type}/entity_role/{entity role}/entities.[format]/
/v1/PubMed/{PubMed ID}/relation_type/{relation type}/relations.[format]/
/v1/PubMed/{PubMed ID}/relation_type/{relation type}/relation_attribute/{relation attribute}/relations.[format]/
API type 3: Entity Centric – Specific Information Return
/v1/Entrez/{Entrez ID}/properties.[format]/
/v1/Entrez/{Entrez ID}/interactions.[format]/
/v1/Entrez/{Entrez ID}/proteoforms.[format]/
JSON / XML
Client Tier
URL access in browser:
Code Samples: Python, cURL, HTTP
Application:
Annotation: BRAT Annotation Tool
Network: Cytoscape

Figure 3.1: iLINKS Text-Mining System architecture [19]

Figure 3.1.1 shows an overall architecture of iLINKS text-mining system. Because this project concentrates on PubMed article text-mining for biocurators to do annotation and further development, the original data content is retrieved from PubMed Database (MEDLINE), which covers article information (PubMed ID), and article text (title and abstract). Then three text-mining tools (RLIMS-P, miRTex, and PubTator) are applied to the article data to capture biological entity data and biological relation data (entity A and entity B interact to form a relation). The text-mining output data are integrated into Oracle relational database.

The text-mining output data are re-organized in the relational database, stored procedures are used to generate the initial data content for each iLINKS API, and those stored procedures in the database are called from Web tier directly using Python cursor function. Next step is to define the URLs/endpoints for APIs and follow up with Django views to handle the input parameters from URL and pass them to the stored procedure calling class (Python class). So it forms a complete data flow from API request to API response.

A Django website is created to document APIs (includes introductions and one case study utilizing three APIs). In order to serve the purpose of benefiting both internal and external users and developers to do integrity checking, data monitoring or article searching, a search web interface is also created that takes one PubMed ID as input and return article text (title and abstract) and its biological entities and relations. This interface is also built upon the same Django framework to keep the consistency and independence. For other on-going application development supported by APIs, Cytoscape (knowledge network) and BRAT (article annotation) are chosen as two typical visualization applications, which are under development by other team members.

3.1.2 Data sources

Because all the APIs' data content is generated in Oracle database by stored procedures, the data source and data pre-processing need to be described. As shown in the Figure 3.1.2, the main biological article source is MEDLINE/PubMed, there are three text-mining tools processing the articles, and the text-mining output data are reformatted before being loaded into a database.

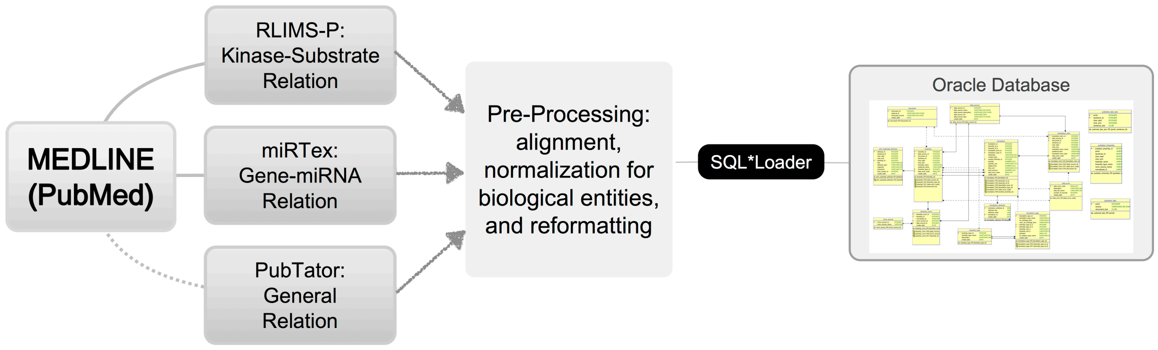

MEDLINE (PubMed)
RLIMS-P: Kinase-Substrate Relation
miRTex: Gene-miRNA Relation
PubTator: General Relation
Pre-Processing: alignment, normalization for biological entities, and reformatting
SQL*Loader
Oracle Database

Figure 3.2 API data source and processing steps (Article source – MEDLINE, text-mining tools – RLIMS-P, miRTex, PubTator; the dashed line represents no-physical connection.)

In the figure above, the dashed line from MEDLINE to PubTator represents that even though MEDLINE is the article source for this tool but we do not need to collect article data from MEDLINE manually, instead, PubTator has an inner connection with MEDLINE and the text-mining result can be downloaded directly through its FTP.

In terms of article resource chosen for this project, MEDLINE (Medical Literature Analysis and Retrieval System Online) is originated in 1964 and contains more than 23 million life science journal citations and abstracts. [20] It covers biology, biochemistry, medicine, pharmacy, and health care, with the recommendation and selection of the Literature Selection Technical Review Committee (LSTRC). MEDLINE is the major component of PubMed database, which is managed by NLM National Center for Biotechnology Information (NCBI). As the most widely used open bioinformatics literature resource, The MEDLINE/PubMed Baseline Repository is also chosen as the first and primary literature source for this thesis project.

In terms of the article text processing method, text-mining tool is one component of the data processing workflow, which utilizes the natural language processing systems to capture biological entities and relations in the literatures. In this thesis project, there are three text-mining tools being used to processing the PubMed abstracts, which are PubTator developed by NCBI Computational Biology Branch (CBB), RLIMS-P and miRTex developed by the Center for Bioinformatics & Computational Biology (CBCB), University of Delaware and the Department of Computer and Information Sciences, University of Delaware.

RLIMS-P (abbreviation of Rule-based Literature Mining System for Phosphorylation), is an enhanced text-mining tool, specifically tagging protein phosphorylation relation by identifying kinase, substrate, and phosphorylation site as biological entities and also distinguishing the phosphorylation trigger words as biological relations. It ingrates the natural language processing and information-extracting modules to process articles and recognize patterns for phosphorylation relations, to which the data resource is all the articles from MEDLINE/PubMed with PubMed ID, title, and abstract data. RLIMS-P 2.0 is capable of processing full-length articles from PubMed Central (PMC OA database), and extract relation information beyond regular expression patterns and across sentences. The accuracy and overall performances of RLIMS-P 2.0 were checked by development team using high expression variety articles (abstracts and full-length articles) data and achieved 0.9 and higher F-scores for capturing kinase, substrate, and phosphorylation site. Additional evaluation was done using 2013 BioNLP-ST GE task content and achieved F-score 0.87 on phosphorylation core task. [21] RLIMS-P output is in JSON format, containing PubMed ID, abstract text, phosphorylation relation type tagging, biological entity types (kinase, substrate and phosphorylation site) tagging.

miRTex is another text-mining tools used in this project, and similar to RLIMS-P, miRTex is designed to identify one type of biological relation – the relation between miRNA and the gene it regulates. Different from RLIMS-P, miRTex can specify the relation by adding direction and directness features; if miRNA directly regulates the gene, it is called ‘miRNA-target’ relation; if miRNA indirectly regulates the gene or the directness is unknown, it is called ‘miRNA-gene’ relation; and if it is gene that regulates the miRNA, it is called ‘gene-miRNA’ relation. Aside from the capability of capturing the ‘gene-miRNA’ relation, another design that makes miRTex unique is its precision. [22] Previous text-mining systems usually use a co-occurrence-based approach or use machine-learning, but their overall accuracy is not quite satisfying even though they may achieve a good result in recall. [23] The output data format is similar to RLIMS-P which makes it convenient for backend developers to do integrity checking and data reformatting before importing into the database.

PubTator, which is an online open-source texting-mining and annotation tool for biocuration, is used as the third text-mining tool (also text-mining data resource) in this project. Different from RLIMS-P and miRTex, PubTator is a web-based system, and its major task is to provide article annotations dynamically, with searching and selecting functions. Another difference is that PubTator targets on multiple biological relations and entity types at the same time, including gene, disease, specie, mutation, and chemical component. And to be specific, PubTator integrates multiple biological entity tracking and normalizing tools, which are GeneTUKit [24] for gene identification (with GenNorm [25] for normalization), DNorm [26] for recognizing diseases, tmVar [27] for tracking mutations. and SR4GN [28] for tagging species, and a dictionary-based lookup approach [29] for chemicals. [12] PubTator contains all the content in PubMed (NCBI) database, which matches the data source for RLIMS-P and miRTex, and it is available through FTP access. In this project, the combined package for all 5 concepts (Gene, Disease, Species, Mutation, and Chemical) was downloaded and put into use (named bioconcepts2pubtator_offsets). The data structure is for this data file is as below:

```
<PMID>|t|<TITLE>
<PMID>|a|<ABSTRACT>
<PMID> <START OFFSET 1> <LAST OFFSET 1> <MENTION 1> <TYPE 1> <IDENTIFIER 1>
<PMID> <START OFFSET 2> <LAST OFFSET 2> <MENTION 2> <TYPE 2> <IDENTIFIER 2>
```

This format is similar to the data output from RLIMS-P and miRTex, and because of the similar data set structure, a relational database was built to store all the data with the potential for adding more data from different text-mining tools, which includes pGenN (for Proteoform), DiMex (for Variant), and eFIP for Phosphorylation PPI (protein-protein interaction).

An Oracle relational database is used to store text-mining results. The text-mining output data are integrated through pre-processing, including alignment, normalization for biological entities, and reformatting to fit the staging table in the Oracle database for direct importing. SQL*Loader, a powerful data parsing engine that can load data from external files to Oracle database under Linux system, is used to import all the text-mining data into staging tables in the Oracle database. Later the data in staging tables are mapped into the productive schema for this relational database, which is mainly composed of entity, entity-feature, relation, relation-feature and document, document-feature table structure. In the productive database schema, short strings (for describing words) are stored in data type VARCHAR, and long strings (like title and abstracts) are stored in data type CLOB. A separate MongoDB is also in use to store the article full texts and sentences.

3.2 API design principles

A design principle is crucial for every kind of design and engineering, not just limited to API design. Simplicity and functionality are, for example, universal principles. When designing something, the first request would be if it can work, then it would come to if it is easy to use. For API design, functionality is still the primary requirement; an API should be able to deliver enough functions or data that can fulfill the needs from API consumers and all other users (through the entire API usage cycle). For example, in this project, the API usage cycle would be data import, article text display or processing, and text-mining data analyzing. So each API should contain those three components so that when users execute one API, they can get all the data they want at once, instead of jumping around several APIs. As mentioned at the beginning of this chapter, there is a simplicity-functionality trade-off in API design. The simplicity can ensure users a good user experience and also lower the learning curve for API usage and consumption. One possible solution would be to build granulated APIs. Granulated APIs mean the functions or data developer wanted to expose through APIs are divided into small reusable sections. This concept does not violate the requirement of containing enough programmable capabilities, because, even one API is a small function block, it still needs to be able to stand alone and support the application actions.

Industrial standard is a great guide to follow, it is also a principle for API design; by adapting to the same standard, both providers and consumers of APIs can have a familiar circumstance to work in and also they can have more support from other groups and resources. [29] So for the API design in this thesis project, RESTful API is chosen other than SOAP API, and the reason is RESTful is a simple and popular standard API type for web application, which has become the most widely used API type. JSON and XML are set as the two formats for API data format, because they are the most commonly used API data formats, and they are also the standard API format across all types of APIs provided by different organizations. Meet the industry standard does not mean to ignore the special requirement of individual project. For example, in biology/bioinformatics field, there is a large portion of researchers and scientist working with the BioC data type. So for the future development of the API version 2.0 (Current API is in version 1.0), BioC will serve as the third API format. As to API endpoint design, strict rules need to be followed. The rules include 1) use a noun instead of a verb; 2) use plurals instead of single nouns; 3) versioning should be part of API endpoints; 4) API format should be specified at the end of each endpoint; 5) API depth should be controlled into an appropriate range. [30] Those rules may vary partially from one API project to another, but the ideal status remains the same.

API documentation and dogfooding (also known as self-checking) may not be required from all the API providers, but they are highly recommended and demanded from users and direct consumers. [31] A good API documentation should at least contain API description, endpoints, method, parameters, request, response, and code samples for try out. [32] A website dedicated to documenting APIs is built as part of this thesis project and will be described in detail in the next chapter. On the other hand, as mentioned in the previous chapter, two related work for building APIs, neither of them has a complete API documentation. That also makes the APIs here more user-friendly and competitive. Dogfooding is not always applicable because sometimes the agencies that published APIs only came up with some use cases but they did not build a real application using the APIs they provided. However, in this project, a PubMed search web page is created using three API endpoints. The data in APIs are processed to generate a search response, which consists of PubMed article text, biological entity information table, and biological relation information table. This search page will also be described in more detail in Chapter 5.

3.3 API endpoints and description

Following the design principles described previously, the versioning is included in the API root URL, which is 'http://beadle.dbi.udel.edu/ilinks_test/rest/v1/'. For current development progress in the iLINKS text-mining system, this API project is running on development branch with the embedded URL as '/ilinks_test/'; all current APIs are in version 1.0, which is also indicated in root URL as '/v1/'. This root URL stays consistent through all the APIs designed in this thesis project. In terms of URL structure and depth, there are general patterns for all APIs. (Not all the APIs have the same depth.) /grandparent/{grandparent_ID}/parent/{parent_ID}/siblingA/{siblingA_ID}/siblingB/{siblingB_ID}/endfile.format

There are four layers of depth, which are grandparent, parent, sibling, and end file. There are three main types of APIs, with different concentrations and data coverage. Figure 3.3.1 stands for the general hieratical structure of how the APIs distinguish from each other and interconnect with each other.

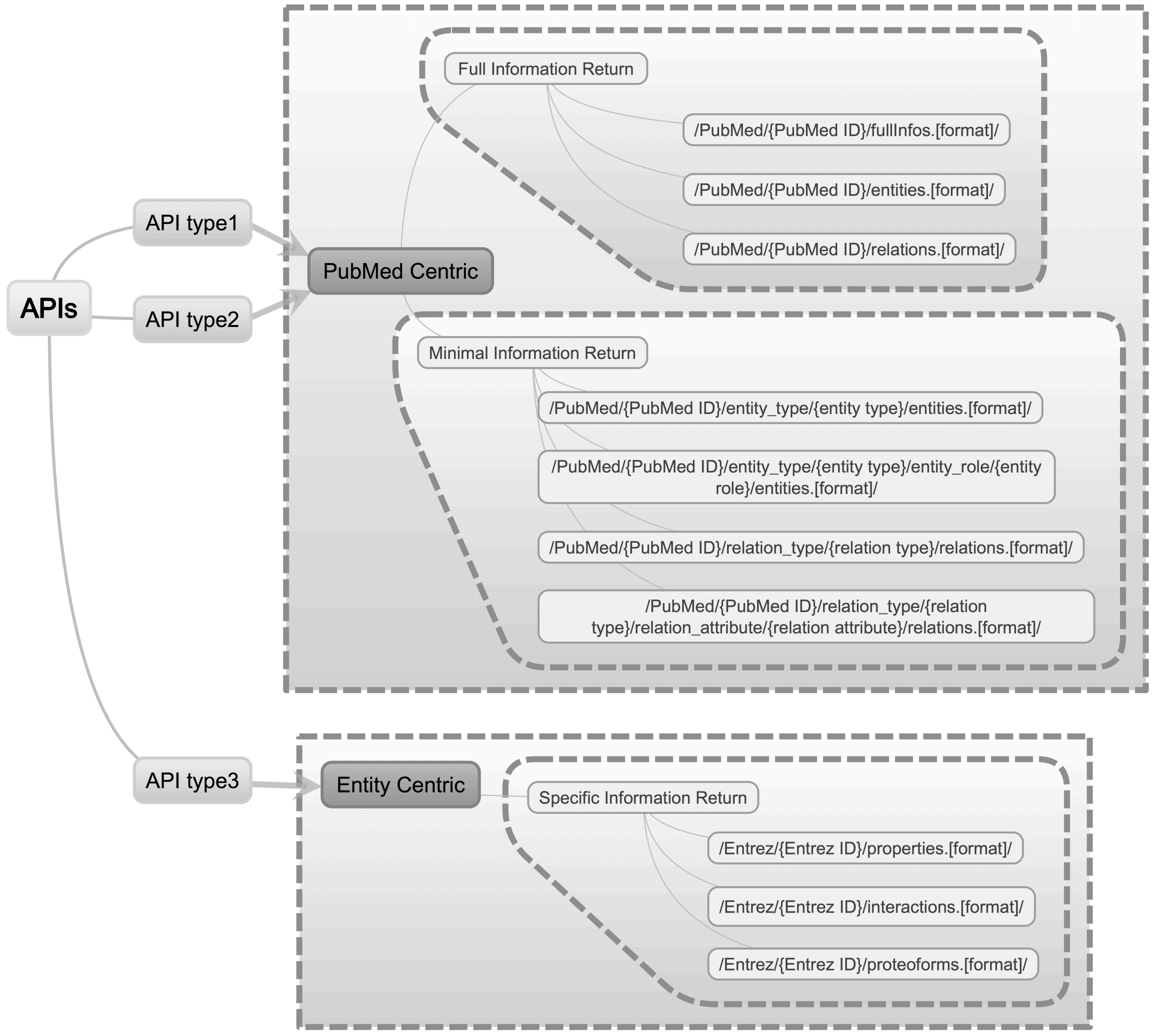

APIs
API type1
API type2
API type3
PubMed Centric
Full Information Return
/PubMed/{PubMed ID}/fullInfos.[format]/
/PubMed/{PubMed ID}/entities.[format]/
/PubMed/{PubMed ID}/relations.[format]/
Minimal Information Return
/PubMed/{PubMed ID}/entity_type/{entity type}/entities.[format]/
/PubMed/{PubMed ID}/entity_type/{entity type}/entity_role/{entity role}/entities.[format]/
/PubMed/{PubMed ID}/relation_type/{relation type}/relations.[format]/
/PubMed/{PubMed ID}/relation_type/{relation type}/relation_attribute/{relation attribute}/relations.[format]/
Entity Centric
Specific Information Return
/Entrez/{Entrez ID}/properties.[format]/
/Entrez/{Entrez ID}/interactions.[format]/
/Entrez/{Entrez ID}/proteoforms.[format]/

Figure 3.3: Hieratical API interconnection structure.

We want to distinguish PubMed-centric APIs from Entity-centric APIs because of their two major data coverage directions. For PubMed centric APIs, users can choose to return full information or minimal information as two different data content types. For full information content, there are still three different options 'fullinfos' (entities and relations combined), 'entities' and 'relations'. For minimal information content, the user can specify the entity type/role or relation type/attribute to narrow the data to a specific biological topic they are interested in. For the third main type of APIs, they are Entity-centric, meaning the data are collected according to one specific biological entity (for example, one protein with Entrez ID 2065). Furthermore, there are three data content options for users, which are entity property, entity interaction, and proteoform.

API type 1

Using PubMed ID to get all the text-mining data extracted from its title and abstract. This type of APIs has full coverage on all the information stored in the database related to single PubMed article. The API consumers and data users can execute this API to get the article text along with its text-mining result. This design is applicable for applications like basic data table presentation, document annotation, interaction network, statistics/metadata generation and so on. Because it contains the full information, it can be called once and feed many applications at the same time. With this one-stop data delivery feature, this API can be coordinated in PubMed search functions to generate article text display and table views. (This example will also be described in detail in Chapter 5.)

There are three sub-types for this API design, which varies in the data content it contains, entity data only, relation data only, or full data (with both entity and relation data). All of them contain basic PubMed information including PubMed ID, source, and article text (title and abstract together). So the API method and request (extending the API root URL mention before) will be like below. Here only GET method is allowed and all data are read-only. '/v1/' is still part of API root URL, showed here to emphasize the versioning. The section marked as '{PubMed ID}' is expected to be replaced entirely by the real PubMed ID that the API consumers are interested in. For version 1.0, JSON and XML are available as format options, which are marked as [format] in the URL.

- GET full data:

  /v1/PubMed/{PubMed ID}/fullInfos.[format]/

- GET entity data:

  /v1/PubMed/{PubMed ID}/entities.[format]/

- GET relation data:

  /v1/PubMed/{PubMed ID}/relations.[format]/

To be more specific, for example, an API consumer has one PubMed ID as 19330006, and he wants to return data in JSON format. The three sub-type of APIs should be called by the URL as shown below. For XML format data, the consumers can simply replace '.json' with '.xml' at the end of each URL.

- GET full data:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/

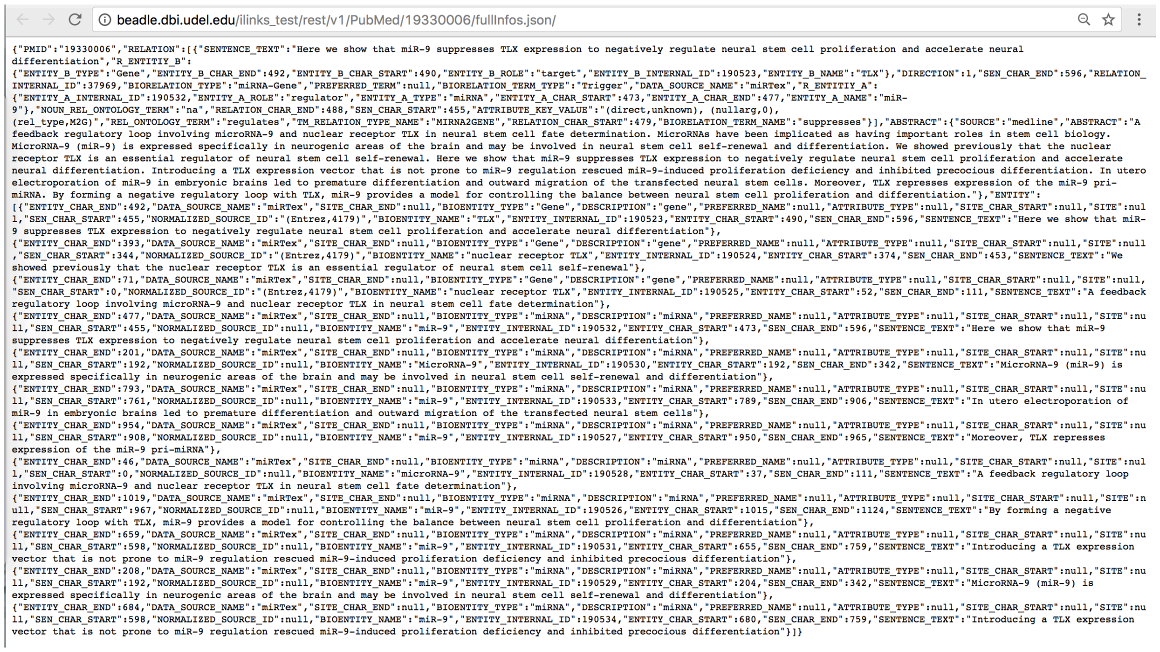
- GET entity data:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/entities.json/

- GET relation data:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/relations.json/

The response viewed in browser is shown in Figure 3.3.2 below.



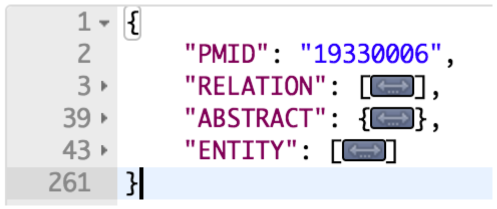
```
{"PMID":"19330006","RELATION":[{"SENTENCE_TEXT":"Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural
differentiation","R_ENTITIY_B":
{"ENTITY_B_TYPE":"Gene","ENTITY_B_CHAR_END":492,"ENTITY_B_CHAR_START":490,"ENTITY_B_ROLE":"target","ENTITY_B_INTERNAL_ID":190523,"ENTITY_B_NAME":"TLX"},"DIRECTION":1,"SEN_CHAR_END":596,"RELATION_
INTERNAL_ID":37969,"BIORELATION_TYPE":"miRNA-Gene","PREFERRED_TERM":null,"BIORELATION_TERM_TYPE":"Trigger","DATA_SOURCE_NAME":"miRTex","R_ENTITIY_A":
{"ENTITY_A_INTERNAL_ID":190532,"ENTITY_A_ROLE":"regulator","ENTITY_A_TYPE":"miRNA","ENTITY_A_CHAR_START":473,"ENTITY_A_CHAR_END":477,"ENTITY_A_NAME":"miR-
9"},"NOUN_REL_ONTOLOGY_TERM":"na","RELATION_CHAR_END":488,"SEN_CHAR_START":455,"ATTRIBUTE_KEY_VALUE":"(direct,unknown), (nullarg,0),
(rel_type,M2G)","REL_ONTOLOGY_TERM":"regulates","TM_RELATION_TYPE_NAME":"MIRNA2GENE","RELATION_CHAR_START":479,"BIORELATION_TERM_NAME":"suppresses"}],"ABSTRACT":{"SOURCE":"medline","ABSTRACT":"A
feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination. MicroRNAs have been implicated as having important roles in stem cell biology.
MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation. We showed previously that the nuclear
receptor TLX is an essential regulator of neural stem cell self-renewal. Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate
neural differentiation. Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation. In utero
electroporation of miR-9 in embryonic brains led to premature differentiation and outward migration of the transfected neural stem cells. Moreover, TLX represses expression of the miR-9 pri-
miRNA. By forming a negative regulatory loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and differentiation."},"ENTITY":
[{"ENTITY_CHAR_END":492,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"Gene","DESCRIPTION":"gene","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nul
l,"SEN_CHAR_START":455,"NORMALIZED_SOURCE_ID":"(Entrez,4179)","BIOENTITY_NAME":"TLX","ENTITY_INTERNAL_ID":190523,"ENTITY_CHAR_START":490,"SEN_CHAR_END":596,"SENTENCE_TEXT":"Here we show that miR-
9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation"},
{"ENTITY_CHAR_END":393,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"Gene","DESCRIPTION":"gene","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":null
,"SEN_CHAR_START":344,"NORMALIZED_SOURCE_ID":"(Entrez,4179)","BIOENTITY_NAME":"nuclear receptor TLX","ENTITY_INTERNAL_ID":190524,"ENTITY_CHAR_START":374,"SEN_CHAR_END":453,"SENTENCE_TEXT":"We
showed previously that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal"},
{"ENTITY_CHAR_END":71,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"Gene","DESCRIPTION":"gene","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":null,
"SEN_CHAR_START":0,"NORMALIZED_SOURCE_ID":"(Entrez,4179)","BIOENTITY_NAME":"nuclear receptor TLX","ENTITY_INTERNAL_ID":190525,"ENTITY_CHAR_START":52,"SEN_CHAR_END":111,"SENTENCE_TEXT":"A feedback
regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination"},
{"ENTITY_CHAR_END":477,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":455,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190532,"ENTITY_CHAR_START":473,"SEN_CHAR_END":596,"SENTENCE_TEXT":"Here we show that miR-9
suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation"},
{"ENTITY_CHAR_END":201,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":192,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"MicroRNA-9","ENTITY_INTERNAL_ID":190530,"ENTITY_CHAR_START":192,"SEN_CHAR_END":342,"SENTENCE_TEXT":"MicroRNA-9 (miR-9) is
expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation"},
{"ENTITY_CHAR_END":793,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":761,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190533,"ENTITY_CHAR_START":789,"SEN_CHAR_END":906,"SENTENCE_TEXT":"In utero electroporation of
miR-9 in embryonic brains led to premature differentiation and outward migration of the transfected neural stem cells"},
{"ENTITY_CHAR_END":954,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":908,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190527,"ENTITY_CHAR_START":950,"SEN_CHAR_END":965,"SENTENCE_TEXT":"Moreover, TLX represses
expression of the miR-9 pri-miRNA"},
{"ENTITY_CHAR_END":46,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nul
l,"SEN_CHAR_START":0,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"microRNA-9","ENTITY_INTERNAL_ID":190528,"ENTITY_CHAR_START":37,"SEN_CHAR_END":111,"SENTENCE_TEXT":"A feedback regulatory loop
involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination"},
{"ENTITY_CHAR_END":1019,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":n
ull,"SEN_CHAR_START":967,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190526,"ENTITY_CHAR_START":1015,"SEN_CHAR_END":1124,"SENTENCE_TEXT":"By forming a negative
regulatory loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and differentiation"},
{"ENTITY_CHAR_END":659,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":598,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190531,"ENTITY_CHAR_START":655,"SEN_CHAR_END":759,"SENTENCE_TEXT":"Introducing a TLX expression
vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation"},
{"ENTITY_CHAR_END":208,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":192,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190529,"ENTITY_CHAR_START":204,"SEN_CHAR_END":342,"SENTENCE_TEXT":"MicroRNA-9 (miR-9) is
expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation"},
{"ENTITY_CHAR_END":684,"DATA_SOURCE_NAME":"miRTex","SITE_CHAR_END":null,"BIOENTITY_TYPE":"miRNA","DESCRIPTION":"miRNA","PREFERRED_NAME":null,"ATTRIBUTE_TYPE":null,"SITE_CHAR_START":null,"SITE":nu
ll,"SEN_CHAR_START":598,"NORMALIZED_SOURCE_ID":null,"BIOENTITY_NAME":"miR-9","ENTITY_INTERNAL_ID":190534,"ENTITY_CHAR_START":680,"SEN_CHAR_END":759,"SENTENCE_TEXT":"Introducing a TLX expression
vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation"}]}
```

Figure 3.4: API response for request

http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/

The data contained in those APIs as multiple arrays/objects, and the definitions is consistent through all the API design since all the APIs are generated from the same data source.

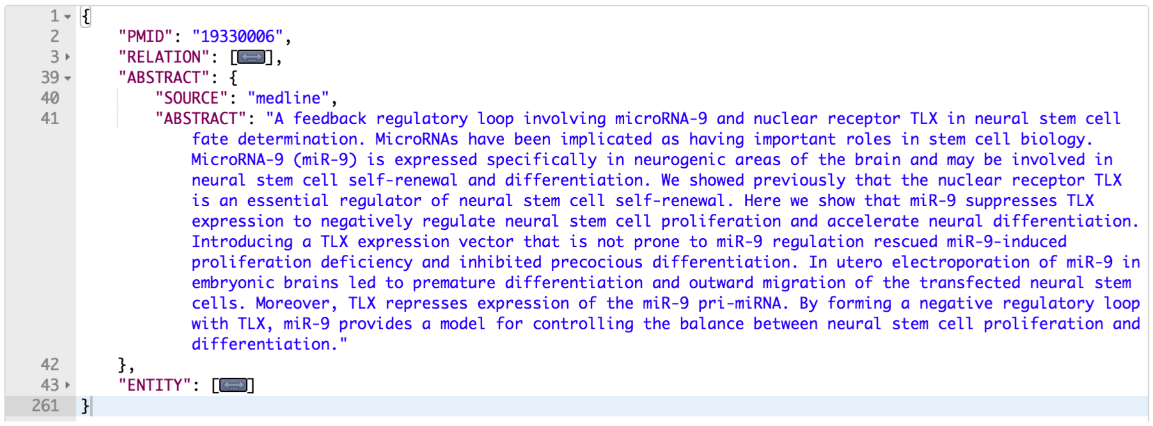
For API with full information returned (in JSON format), there are four main objects:



```
{
    "PMID": "19330006",
    "RELATION": [↔],
    "ABSTRACT": {↔},
    "ENTITY": [↔]
}
```

Figure 3.5: Main objects in PubMed centric full information API (JSON format).

In the ‘ABSTRACT’ object, there are two key-value pairs storing the article source (‘SOURCE’) and article text (‘ABSTRACT’).



```
{
    "PMID": "19330006",
    "RELATION": [⟺],
    "ABSTRACT": {
        "SOURCE": "medline",
        "ABSTRACT": "A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell
            fate determination. MicroRNAs have been implicated as having important roles in stem cell biology.
            MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in
            neural stem cell self-renewal and differentiation. We showed previously that the nuclear receptor TLX
            is an essential regulator of neural stem cell self-renewal. Here we show that miR-9 suppresses TLX
            expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation.
            Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced
            proliferation deficiency and inhibited precocious differentiation. In utero electroporation of miR-9 in
            embryonic brains led to premature differentiation and outward migration of the transfected neural stem
            cells. Moreover, TLX represses expression of the miR-9 pri-miRNA. By forming a negative regulatory loop
            with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and
            differentiation."
    },
    "ENTITY": [⟺]
}
```

Figure 3.6: ABSTRACT object in PubMed centric full information API (JSON format).

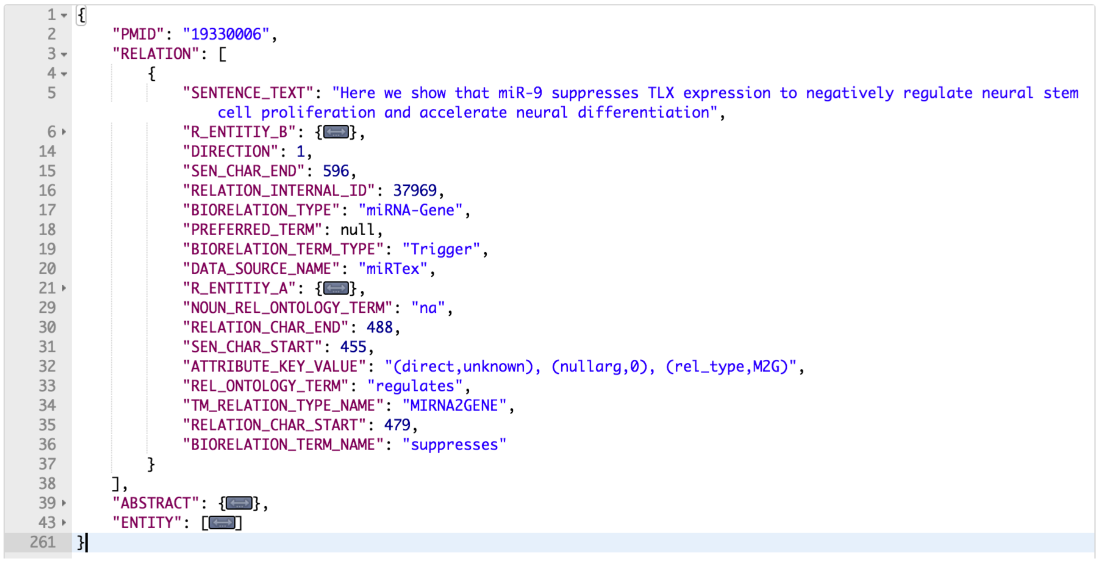
In the ‘RELATION’ array, relation records are appended to a list/array in the JSON API; within each relation object, there are 16 key-value pairs storing biological relation information and 2 objects storing the information for two biological entities that interact with each other. Similarly, the ‘ENTITY’ array contains the biological entity information. Because the initial goal for building the APIs is to feed data to annotation (BRAT annotation tool), network (Cytoscape) and data tables (jQuery DataTable) applications. The structural requirements from those applications do have an impact on how data are organized in those APIs. For example, the jQuery DataTable prefers all data stored as objects in one array and all have exactly the same keys with each object. And in order to only provide the necessary data, consumers can decide which objects they want. The three combinations are:

- GET full data: PMID + ARTICLE + ENTITY + RELATION

- GET entity data: PMID + ARTICLE + ENTITY

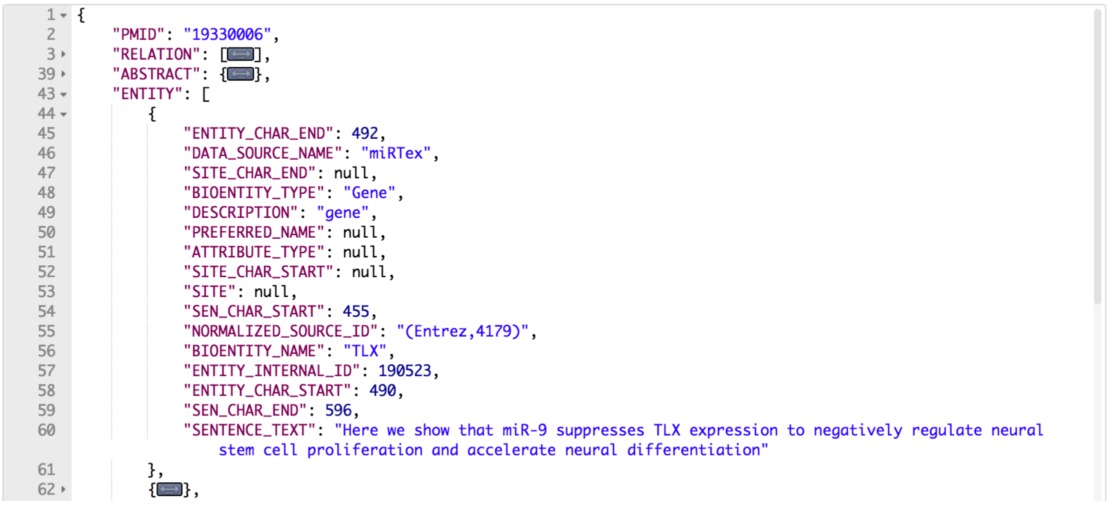
- GET relation data: PMID + ARTICLE + RELATION

The pretty formatted API data are shown as below:



```
{
    "PMID": "19330006",
    "RELATION": [
        {
            "SENTENCE_TEXT": "Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem
                cell proliferation and accelerate neural differentiation",
            "R_ENTITIY_B": {↔},
            "DIRECTION": 1,
            "SEN_CHAR_END": 596,
            "RELATION_INTERNAL_ID": 37969,
            "BIORELATION_TYPE": "miRNA-Gene",
            "PREFERRED_TERM": null,
            "BIORELATION_TERM_TYPE": "Trigger",
            "DATA_SOURCE_NAME": "miRTex",
            "R_ENTITIY_A": {↔},
            "NOUN_REL_ONTOLOGY_TERM": "na",
            "RELATION_CHAR_END": 488,
            "SEN_CHAR_START": 455,
            "ATTRIBUTE_KEY_VALUE": "(direct,unknown), (nullarg,0), (rel_type,M2G)",
            "REL_ONTOLOGY_TERM": "regulates",
            "TM_RELATION_TYPE_NAME": "MIRNA2GENE",
            "RELATION_CHAR_START": 479,
            "BIORELATION_TERM_NAME": "suppresses"
        }
    ],
    "ABSTRACT": {↔},
    "ENTITY": [↔]
}
```

Figure 3.7: RELATION object in PubMed centric full information API (JSON format).



```
{
    "PMID": "19330006",
    "RELATION": [↔],
    "ABSTRACT": {↔},
    "ENTITY": [
        {
            "ENTITY_CHAR_END": 492,
            "DATA_SOURCE_NAME": "miRTex",
            "SITE_CHAR_END": null,
            "BIOENTITY_TYPE": "Gene",
            "DESCRIPTION": "gene",
            "PREFERRED_NAME": null,
            "ATTRIBUTE_TYPE": null,
            "SITE_CHAR_START": null,
            "SITE": null,
            "SEN_CHAR_START": 455,
            "NORMALIZED_SOURCE_ID": "(Entrez,4179)",
            "BIOENTITY_NAME": "TLX",
            "ENTITY_INTERNAL_ID": 190523,
            "ENTITY_CHAR_START": 490,
            "SEN_CHAR_END": 596,
            "SENTENCE_TEXT": "Here we show that miR-9 suppresses TLX expression to negatively regulate neural
                stem cell proliferation and accelerate neural differentiation"
        },
        {↔},
```

Figure 3.8: ENTITY object in PubMed centric full information API (JSON format).

API type 2

Because we have the enhanced text-mining tools dedicated for phosphorylation relation and miRNA-gene relation (recognizing kinase, substrate, miRNA and target genes), topic-specific APIs using PubMed ID to get specific data by limiting the entity type/role or relation type/attribute are designed. This type of APIs has very similar structural design comparing to the API type 1. However, this API only contains the minimal information for the biological entities and relations. The purpose of this special design is to quickly return clean data with specific topics that the users are interested in. So instead of returning all the potential information captured by text-mining tools, users are allowed to put filters on the data content, in other words, users can specify the biological entity type/role, or biological relation type/attribute and request a subset of full information data.

The relation between Biological entity type and entity role in the current version 1.0 with three text-mining tools (RLIMS-P, miRTex, and PubTator) are shown as below. Based on the current database status (supporting API version 1.0), there are six biological entity types available (Protein, Proteoform, Gene, miRNA, Variant, and Unknown - the yellow rectangles in the middle line). There are four biological entity roles (Kinase, Interactant, Regulator and Unknown) available for BIOENTITY A and four (Substrate, Interactant, Target, and Unknown) for BIOENTITY B in the BIENTITY A - BIORELATION - BIENTITY B relation structure. In the figure below, the BIOENTITY A is on the left side in green and BIOENTITY B is on the right side in blue; the lines connecting two rectangles indicates they have been combined to describe one biological entity.

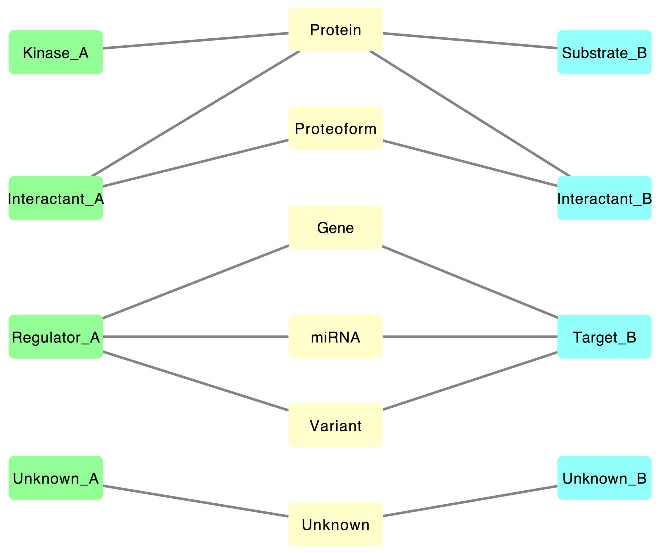


Figure 3.9: Version 1.0 API available biological entity type and role (left-side green entities and right side blue entities represent of entity roles of Biological Entity A and B; middle part yellow entities are the types of Biological Entity; the lines indicate they are signed to the same biological entity)

The URLs designed for this type of APIs are listed below.

- GET entity data with type filter:

/v1/PubMed/{PubMed ID}/entity_type/{entity type}/entities.[format]/

- GET entity data with type and role filter:

  /v1/PubMed/{PubMed ID}/entity_type/{entity type}/entity_role/{entity role}/entities.[format]/

- GET relation data with type filter:

  /v1/PubMed/{PubMed ID}/relation_type/{relation type}/relations.[format]/

- GET relation data with type and attribute filter:

  /v1/PubMed/{PubMed ID}/relation_type/{relation type}/relation_attribute/{relation attribute}/relations.[format]/

For example, an API consumer still has one PubMed ID as 19330006, and he wants to return data in JSON format. He also chooses 'miRNA' as entity type, 'regulator' as entity role, 'miRNA-Gene' as relation type, and 'unknown' as relation attribute. Then the APIs should be called by the URL as below.

- GET entity data with type filter:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/entity_type/miRNA/entities.json/

- GET entity data with type and role filter:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/entity_type/miRNA/entity_role/regulator/entities.json/

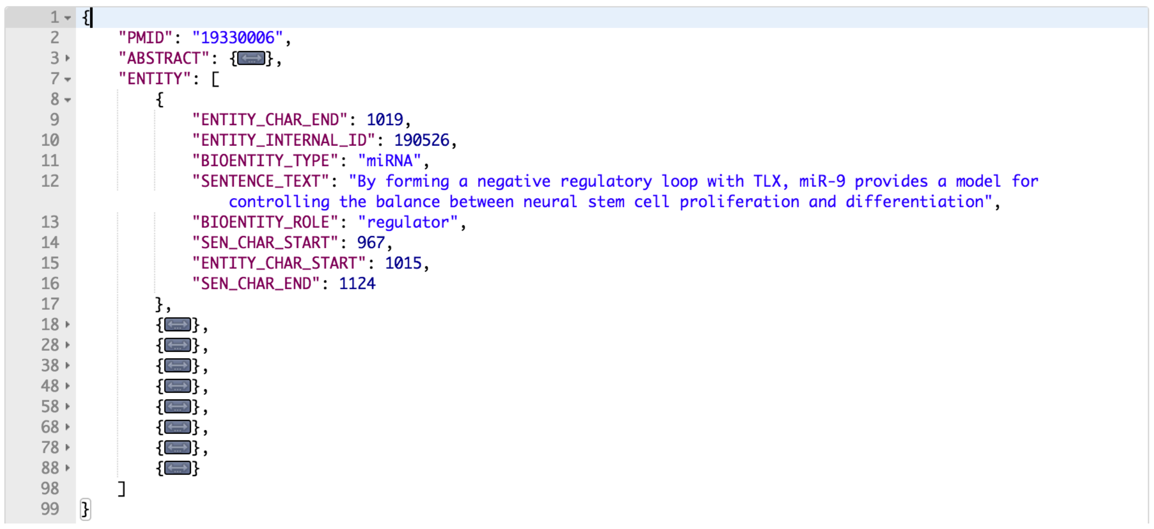
- GET relation data with type filter:

http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/relation_type/miRNA-Gene/relations.json/

- GET relation data with type and attribute filter:

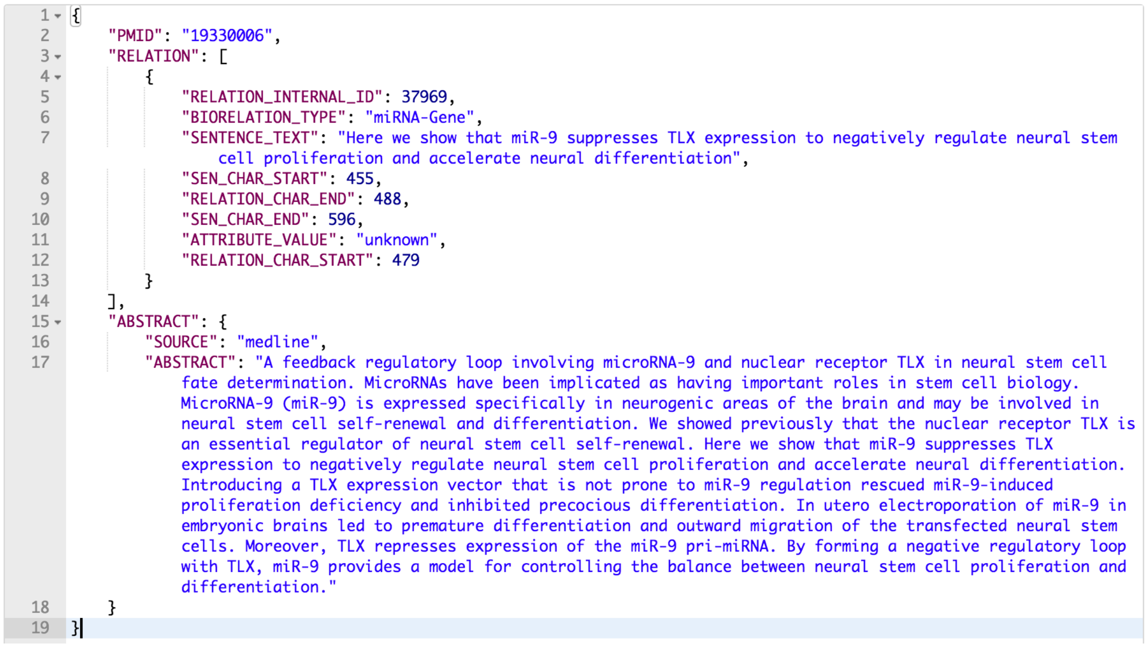
  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/relation_type/miRNA-Gene/relation_attribute/unknown/relations.json/

In terms of the minimal data subset, for each biological entity or relation object, the sentence (mentioning this entity/relation), entity/relation filter values, offsets, and one unique entity/relation tracking ID are included. Aside from this, the PMID and ABSTRACT objects remind the same. So the sample response according to the requests:



```
{
    "PMID": "19330006",
    "ABSTRACT": {⟺},
    "ENTITY": [
        {
            "ENTITY_CHAR_END": 1019,
            "ENTITY_INTERNAL_ID": 190526,
            "BIOENTITY_TYPE": "miRNA",
            "SENTENCE_TEXT": "By forming a negative regulatory loop with TLX, miR-9 provides a model for
                controlling the balance between neural stem cell proliferation and differentiation",
            "BIOENTITY_ROLE": "regulator",
            "SEN_CHAR_START": 967,
            "ENTITY_CHAR_START": 1015,
            "SEN_CHAR_END": 1124
        },
        {⟺},
        {⟺},
        {⟺},
        {⟺},
        {⟺},
        {⟺},
        {⟺},
        {⟺}
    ]
}
```

Figure 3.10: ENTITY object in PubMed centric minimal information API (JSON format).



```
{
    "PMID": "19330006",
    "RELATION": [
        {
            "RELATION_INTERNAL_ID": 37969,
            "BIORELATION_TYPE": "miRNA-Gene",
            "SENTENCE_TEXT": "Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem
                cell proliferation and accelerate neural differentiation",
            "SEN_CHAR_START": 455,
            "RELATION_CHAR_END": 488,
            "SEN_CHAR_END": 596,
            "ATTRIBUTE_VALUE": "unknown",
            "RELATION_CHAR_START": 479
        }
    ],
    "ABSTRACT": {
        "SOURCE": "medline",
        "ABSTRACT": "A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell
            fate determination. MicroRNAs have been implicated as having important roles in stem cell biology.
            MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in
            neural stem cell self-renewal and differentiation. We showed previously that the nuclear receptor TLX is
            an essential regulator of neural stem cell self-renewal. Here we show that miR-9 suppresses TLX
            expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation.
            Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced
            proliferation deficiency and inhibited precocious differentiation. In utero electroporation of miR-9 in
            embryonic brains led to premature differentiation and outward migration of the transfected neural stem
            cells. Moreover, TLX represses expression of the miR-9 pri-miRNA. By forming a negative regulatory loop
            with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and
            differentiation."
    }
}
```

Figure 3.11: PubMed centric minimal information API (JSON format)

API type 3

Using normalized ID (Entrez ID) to get full data across articles. This is a special design utilizing the data in the database. Instead of focusing on each single PubMed article, this type of APIs collects related data from all the articles from PubMed. This design can provide users the text-mining data from all different articles, which will have a higher coverage on the knowledge and may discover the hidden relations. With the similar purpose, eGIFT (http://biotm.cis.udel.edu/eGIFT/) offers a gene-searching base to extract information from all kinds of text sources. Similar to eGIFT, the APIs in this section contains sentences text and some basic feature descriptions for biological entity/relation; the difference would be the approach to gain the knowledge. If the user wants direct visualized result, with clickable interface, eGIFT would be a great option, however, eGIFT does not provide API services, so if the user wants a programmable data access, this type 3 API will be a better choice. [33] There are three options for API users depending on how they are looking for protein/gene properties, interactions or proteoforms information.

The URLs designed for this type of APIs are listed below.

- GET full property data related to one entity:

  /v1/Entrez/{Entrez ID}/properties.[format]/
- GET full interaction data related to one entity:

  /v1/Entrez/{Entrez ID}/interactions.[format]/
- GET full proteoform data related to one entity:

  /v1/Entrez/{Entrez ID}/proteoforms.[format]/

For example, if one user wants to get the data for the gene with Entrez ID 2065, the three URLs would be:

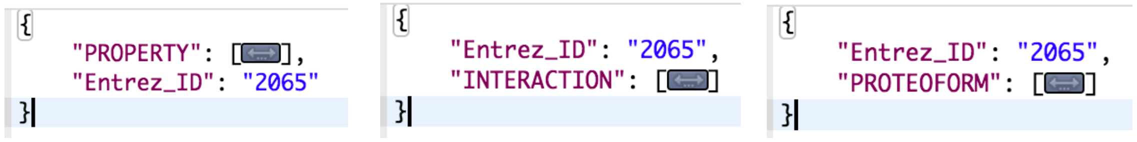
- GET full property data related to one entity:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/properties.json/
- GET full interaction data related to one entity:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/interactions.json/
- GET full proteoform data related to one entity:

  http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/proteoforms.json/

Because this type of APIs is entity centric, the main data objects in each APIs are only two, one is Entrez ID, the other could be one of 'PROPERTY', 'INTERACTION' and 'PROTEOFORM'.



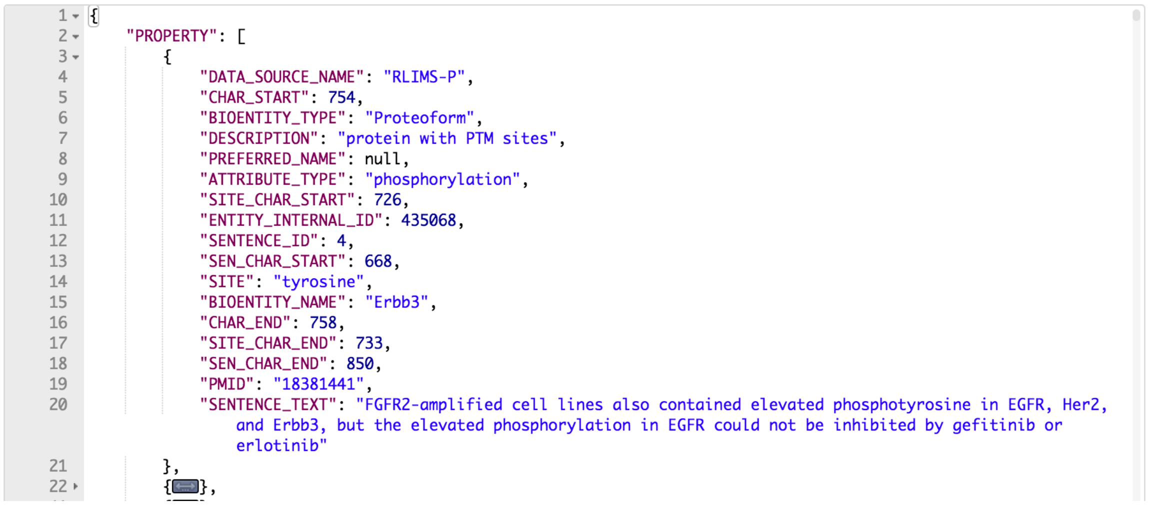
```
{
    "PROPERTY": [⟺],
    "Entrez_ID": "2065"
}
```

```
{
    "Entrez_ID": "2065",
    "INTERACTION": [⟺]
}
```

```
{
    "Entrez_ID": "2065",
    "PROTEOFORM": [⟺]
}
```

Figure 3.12: Main objects in entity centric APIs (JSON format).

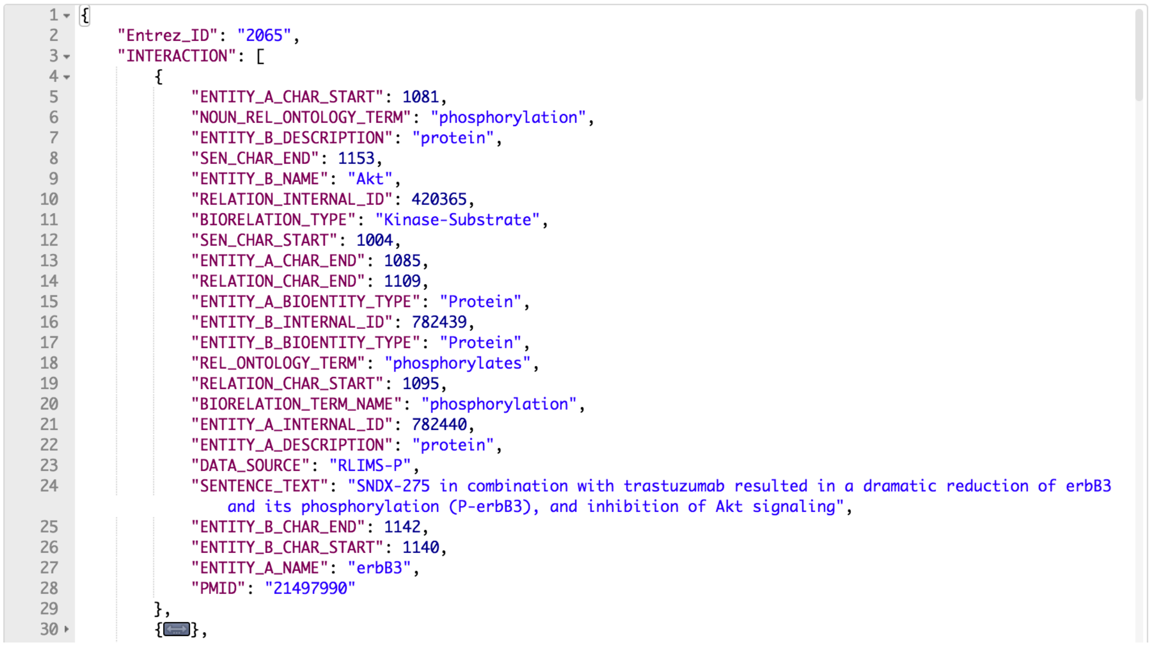
According to the feature of each subtype API, key-value pairs are selected and organized into one data object for one mention of the target gene/protein. Supporting description for all the keys in three major types of APIs are appended at the end of the thesis. Here are the sample responses with parameter Entrez ID set as 2065.



```
{
    "PROPERTY": [
        {
            "DATA_SOURCE_NAME": "RLIMS-P",
            "CHAR_START": 754,
            "BIOENTITY_TYPE": "Proteoform",
            "DESCRIPTION": "protein with PTM sites",
            "PREFERRED_NAME": null,
            "ATTRIBUTE_TYPE": "phosphorylation",
            "SITE_CHAR_START": 726,
            "ENTITY_INTERNAL_ID": 435068,
            "SENTENCE_ID": 4,
            "SEN_CHAR_START": 668,
            "SITE": "tyrosine",
            "BIOENTITY_NAME": "Erbb3",
            "CHAR_END": 758,
            "SITE_CHAR_END": 733,
            "SEN_CHAR_END": 850,
            "PMID": "18381441",
            "SENTENCE_TEXT": "FGFR2-amplified cell lines also contained elevated phosphotyrosine in EGFR, Her2,
                and Erbb3, but the elevated phosphorylation in EGFR could not be inhibited by gefitinib or
                erlotinib"
        },
        {⟺},
```

Figure 3.13: Entity centric entity property API (JSON format), in response to

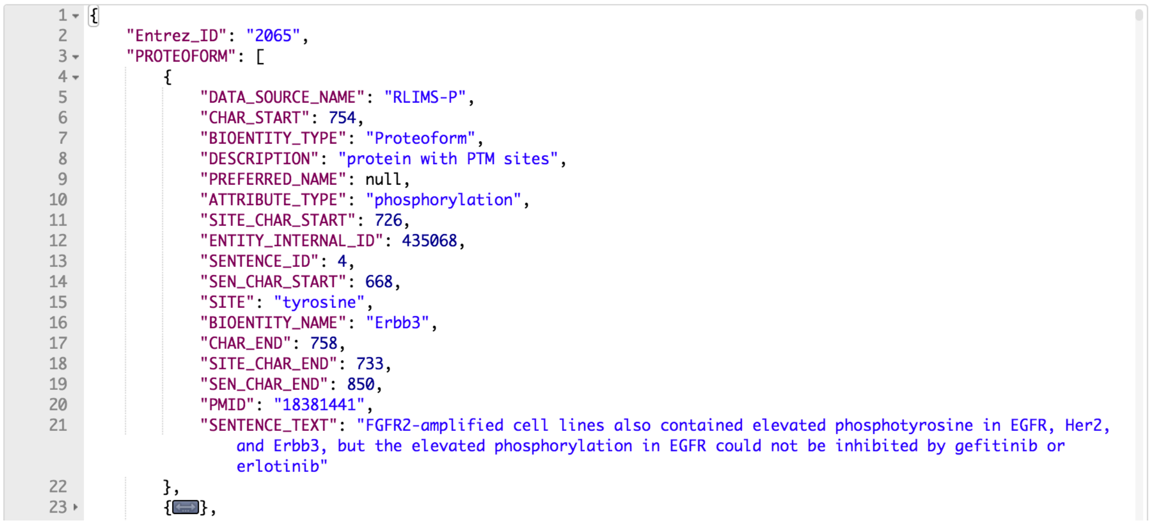
http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/properties.json/



```
{
    "Entrez_ID": "2065",
    "INTERACTION": [
        {
            "ENTITY_A_CHAR_START": 1081,
            "NOUN_REL_ONTOLOGY_TERM": "phosphorylation",
            "ENTITY_B_DESCRIPTION": "protein",
            "SEN_CHAR_END": 1153,
            "ENTITY_B_NAME": "Akt",
            "RELATION_INTERNAL_ID": 420365,
            "BIORELATION_TYPE": "Kinase-Substrate",
            "SEN_CHAR_START": 1004,
            "ENTITY_A_CHAR_END": 1085,
            "RELATION_CHAR_END": 1109,
            "ENTITY_A_BIOENTITY_TYPE": "Protein",
            "ENTITY_B_INTERNAL_ID": 782439,
            "ENTITY_B_BIOENTITY_TYPE": "Protein",
            "REL_ONTOLOGY_TERM": "phosphorylates",
            "RELATION_CHAR_START": 1095,
            "BIORELATION_TERM_NAME": "phosphorylation",
            "ENTITY_A_INTERNAL_ID": 782440,
            "ENTITY_A_DESCRIPTION": "protein",
            "DATA_SOURCE": "RLIMS-P",
            "SENTENCE_TEXT": "SNDX-275 in combination with trastuzumab resulted in a dramatic reduction of erbB3
                and its phosphorylation (P-erbB3), and inhibition of Akt signaling",
            "ENTITY_B_CHAR_END": 1142,
            "ENTITY_B_CHAR_START": 1140,
            "ENTITY_A_NAME": "erbB3",
            "PMID": "21497990"
        },
        {↔},
```

Figure 3.14: Entity centric entity interaction API (JSON format), in response to

http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/interactions.json/



```
{
    "Entrez_ID": "2065",
    "PROTEOFORM": [
        {
            "DATA_SOURCE_NAME": "RLIMS-P",
            "CHAR_START": 754,
            "BIOENTITY_TYPE": "Proteoform",
            "DESCRIPTION": "protein with PTM sites",
            "PREFERRED_NAME": null,
            "ATTRIBUTE_TYPE": "phosphorylation",
            "SITE_CHAR_START": 726,
            "ENTITY_INTERNAL_ID": 435068,
            "SENTENCE_ID": 4,
            "SEN_CHAR_START": 668,
            "SITE": "tyrosine",
            "BIOENTITY_NAME": "Erbb3",
            "CHAR_END": 758,
            "SITE_CHAR_END": 733,
            "SEN_CHAR_END": 850,
            "PMID": "18381441",
            "SENTENCE_TEXT": "FGFR2-amplified cell lines also contained elevated phosphotyrosine in EGFR, Her2,
                and Erbb3, but the elevated phosphorylation in EGFR could not be inhibited by gefitinib or
                erlotinib"
        },
        {↔},
```

Figure 3.15: Entity centric entity proteoform API (JSON format), in response to

http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/proteoforms.json/

Chapter 4

API IMPLEMENTATION AND DOCUMENTATION

API building is supposed to be a back-end engineering task, however, without a good front end interface for API documentation, the API design is not complete. And as one of the API design principles, dogfooding and self-checking is usually required for each API provider. So in this chapter, all the API building techniques will be described in detail in implementation section followed by example use case and API documentation website.

4.1 Implementation

The relational database used in iLINKS system is Oracle Database 12c Enterprise Edition Release 12.1.0.2.0 - 64bit Production. The APIs are implemented using Django REST framework (version 3). The test web site is implemented in Django Framework (version 1.10.5, released on 01/04 2017). The figure below indicates the data flow from user request to the response returned by the server. Request (with input parameters) is passed to server to invoke a specific call of stored procedures in Oracle database. The data table generated by the stored procedure is rendered into data object (in JSON or XML format) and sent back to the user as API response. The typical request and response cycle takes less than 700 milliseconds.

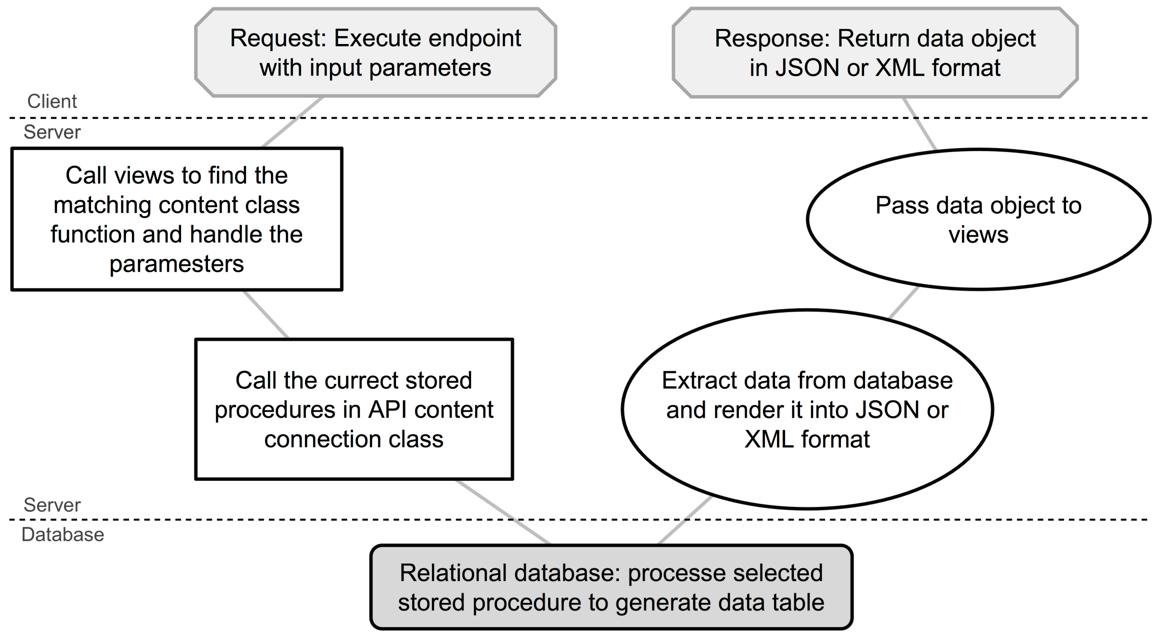

Request: Execute endpoint with input parameters
Response: Return data object in JSON or XML format
Client
Server
Call views to find the matching content class function and handle the paramesters
Pass data object to views
Call the currect stored procedures in API content connection class
Extract data from database and render it into JSON or XML format
Server
Database
Relational database: processe selected stored procedure to generate data table

Figure 4.1: API request and response cycle

Starting from the database (the bottom section in the figure), instead of using Django REST framework default database (sqlite3), Oracle database is chosen because of its scalability (handling a large quantity of data), security, and availability. In fact, most of the projects in the Center for Bioinformatics & Computational Biology (CBCB), University of Delaware, that require databases, are using Oracle database. As described in Chapter 3.1, one Oracle relational database is built to store the text-mining data and PubMed article text. [Dr. Sheng-Chih Chen, NCI Text-mining Integration System project.] In general, the database logic structure is shown in figure below.

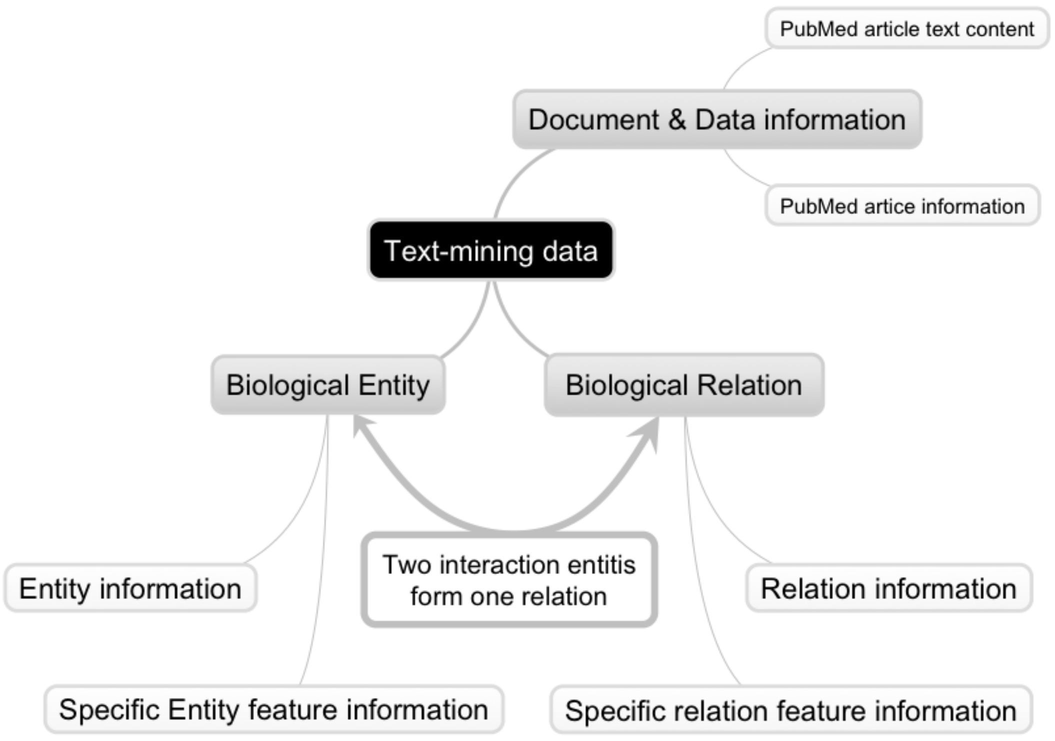

PubMed article text content
Document & Data information
PubMed artice information
Text-mining data
Biological Entity
Biological Relation
Entity information
Two interaction entitis form one relation
Relation information
Specific Entity feature information
Specific relation feature information

Figure 4.2: iLINKS relational database general logic structure

The database consists of three major sections, biological entity (BIOENTITY), biological relation (BIORELATION), and document and data information (DOCUMENT, PUBMENT_ABS); each section contains several tables for basic information and features (or text content). 10 stored procedures are used to generate the initial data tables, in other words, the data for the APIs. Stored procedures are chosen in this project for two reasons. One, stored procedure itself is stable and scalable in terms of handling different kinds of data attribute and the procedures that return tables providing a stable data output for downstream development and analysis, which can ensure the modifications on the database being handled within the database and will not affect the frontend applications. The other reason is that Oracle database stored procedure has been tested out for the applicability in connecting to the webserver and passing data (on iPTMnet data integrity checking project), which is especially crucial in this project.

The stored procedures are called from Python using cx_Oracle module, which should be imported in advance, followed by 'cx_Oracle.connect' function (to connect to the relational database) and 'cursor.callproc' function to get the data from stored procedures. [36] Within the same Python class, data retrieved from database are firstly stored as Python dictionary, then rendered into JSON and XML (format) data objects. To render data objects into JSON format, we use Django REST framework JSON render decorator '@renderer_classes((JSONRenderer,))'; to render data objects into XML format, we use Python package 'dicttoxml' (version 1.7.4), which requires installation in advance. [36] [37] There are 10 Python functions in this data generation class, which match the 10 APIs described in the previous chapter (three types of APIs). This functionality does not belong to Django framework, instead, it is created manually as a method to connect data tier and web tier. In other words, if considering this API design as a three-tier architecture, (data tier, logic tier and presentation tier), using stored procedures can help API developers to put most of the business logic inside the database. [38] [39] This method not only meets the requirements in this project, but also provides one option or reference for other developers who want to build database-oriented applications. [40]

In Django Framework, there are two files (urls.py and views.py) set to design URL patterns and imply views to handle request and response. [41] [42] Those two files are also utilized in this project for API endpoint building and parameters passing. For example, for PubMed centric full information API with URL:

http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/{PubMed ID}/fullInfos.[format]/

The URL created in urls.py file would be:

```
urlpatterns = format_suffix_patterns([
        url(r'^v1/PubMed/(?P<pmid>[0-9]+)/fullInfos.(?P<api_format>[-\w]+)/',fullInfo,
        name='fullInfo'),
])
```

Here '(?P<pmid>[0-9]+)' allows integer/number input and stores the URL input value in the parameter named 'pmid'; '(?P<api_format>[-\w]+)' allows letters input and stores string value in the parameter named 'api_format'. When a HTTP request is made using the pattern of URL, Python function 'fullInfo' is invoked to process the request by passing the parameters to database connection class, which will call certain stored procedures to generate data object. For error checking, if the input for 'pmid' is not integer there would be a 404 error (page not found) indicating the input is not valid; 404 error will also happen when other parts of the URL are not correct. If the input value for 'api_format' is not valid, a customized error message will be returned to users. For example, use puts 'api_format' as 'abc', the error message will be:

```
{
  "ERROR_MESSAGE":"Input format abc is not correct or available",
  "ERROR_TYPE":"Wrong format input",
  "HTTP Status Code":"404 Not Found"
```

}

In API version 1.0, JSON and XML format is available, so the input for 'api_format' could be 'json' or 'xml'; in other words, the user can have two options in this API with URLs like:

- http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/{PubMed ID}/fullInfos.json/
- http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/{PubMed ID}/fullInfos.xml/

4.2 Documentation

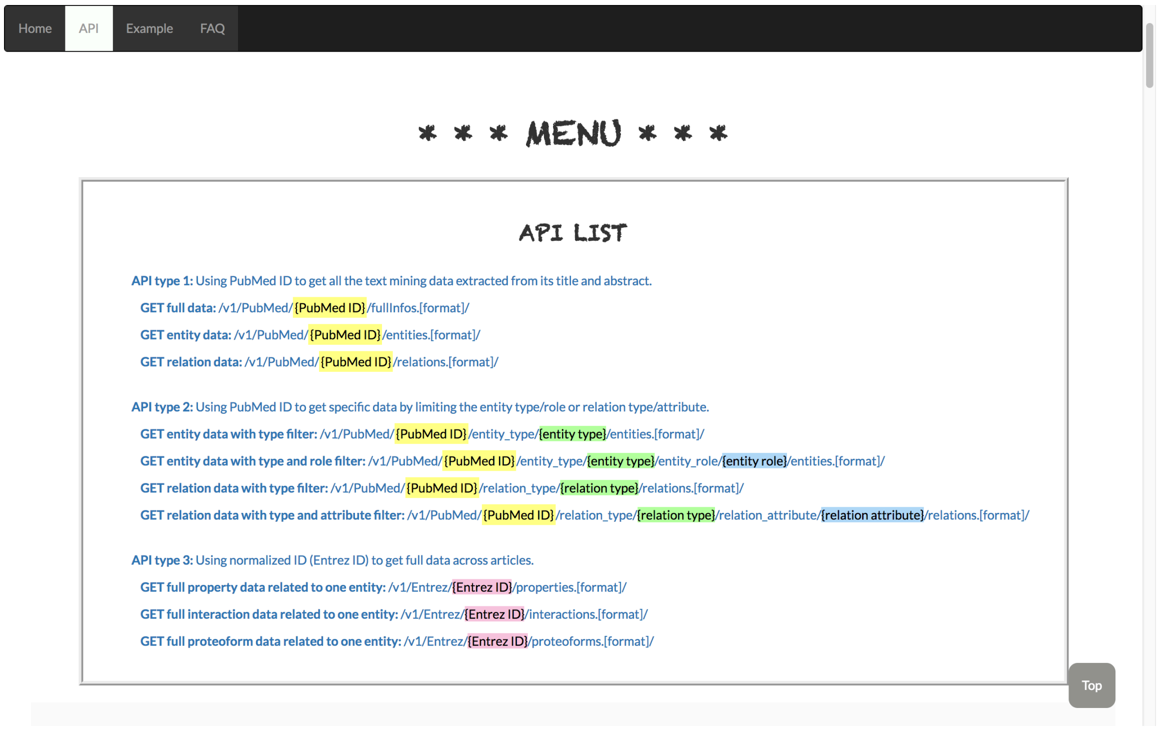
According to the survey hosted by programmableweb.com (an open source for publishing API, current has over 17,000 APIs and thousands of application, available through https://www.programmableweb.com/) in 2013, an accurate and complete documentation has been selected as the most important factor for API development. [43] [44] [45] So for the APIs in this thesis project, we built the documentation website for iLINKS RESTful APIs. The website consists of four sections, 'Home', 'API', 'Example' and 'FAQ'. For API version 2.0, the 'What is new' section will be include in home page tracking the differences between two versions.

The website is available at http://beadle.dbi.udel.edu/ilinks_test/rest/.

API page

The 'API' page is main page for API documentation, which has clickable API list as a guideline and each type of APIs has its introduction section followed with tables for attribute definition and detailed API description.

In the API list, parameters are marked in different colors, which are consistent through the entire page; users can use the color mark as a reference of where to put the input value for each parameter and how many parameters are there in each API. Also this API list has all the endpoints/URLs for APIs, users can simply click on each API or type description to jump to the section they want.



API LIST

API type 1: Using PubMed ID to get all the text mining data extracted from its title and abstract.

GET full data: /v1/PubMed/{PubMed ID}/fullInfos.[format]/

GET entity data: /v1/PubMed/{PubMed ID}/entities.[format]/

GET relation data: /v1/PubMed/{PubMed ID}/relations.[format]/

API type 2: Using PubMed ID to get specific data by limiting the entity type/role or relation type/attribute.

GET entity data with type filter: /v1/PubMed/{PubMed ID}/entity_type/{entity type}/entities.[format]/

GET entity data with type and role filter: /v1/PubMed/{PubMed ID}/entity_type/{entity type}/entity_role/{entity role}/entities.[format]/

GET relation data with type filter: /v1/PubMed/{PubMed ID}/relation_type/{relation type}/relations.[format]/

GET relation data with type and attribute filter: /v1/PubMed/{PubMed ID}/relation_type/{relation type}/relation_attribute/{relation attribute}/relations.[format]/

API type 3: Using normalized ID (Entrez ID) to get full data across articles.

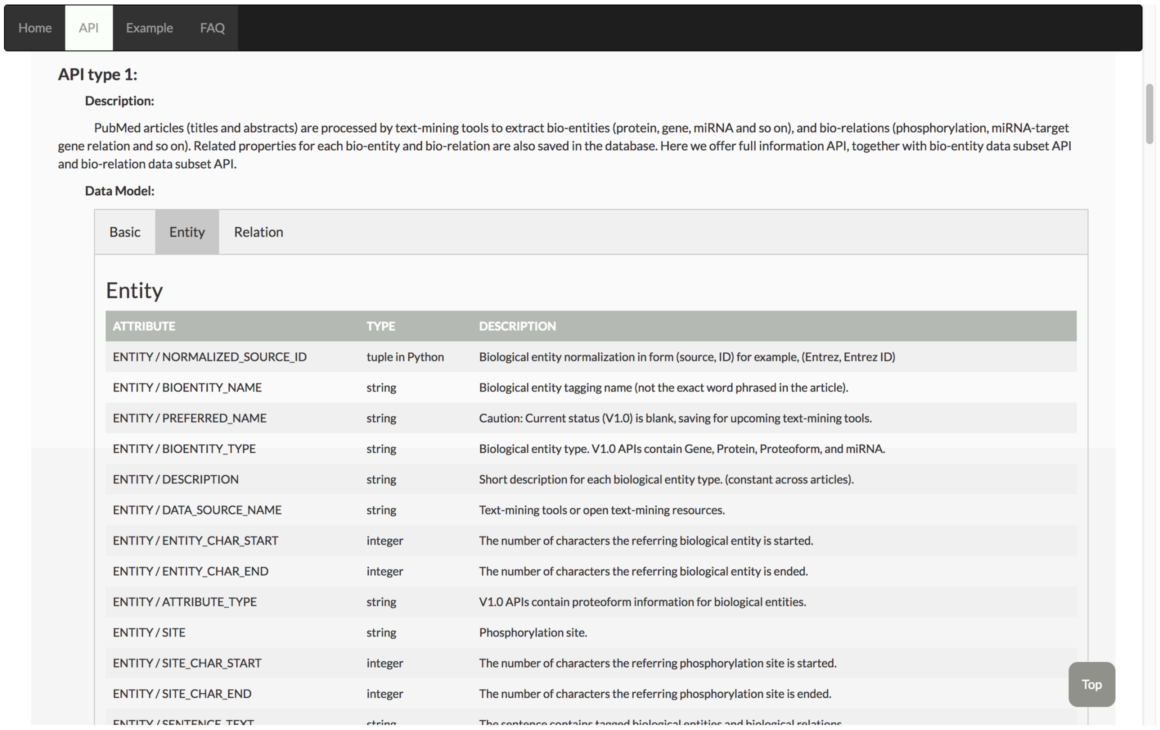
GET full property data related to one entity: /v1/Entrez/{Entrez ID}/properties.[format]/

GET full interaction data related to one entity: /v1/Entrez/{Entrez ID}/interactions.[format]/

GET full proteoform data related to one entity: /v1/Entrez/{Entrez ID}/proteoforms.[format]/

Figure 4.3: API list in API page (http://beadle.dbi.udel.edu/ilinks_test/rest/API/)

For each type of APIs, there is a brief introduction of its design and data content. Data Model part is used to show all the attributes (key-value pairs) mentioned in the APIs. For example, type 1 API would contain ‘ABSTRACT’, ‘ENTITY’ and ‘RELATION’, three objects (each has multiple key-value pairs), so the abstract, entity and relation data are described in three tables in this part. There are duplications in three types of APIs, because all the API data is generated from the same database, but with different structure. The duplications are kept to ensure users' convenience. This page will be updated according to the modifications in API version 2.0, but the general structure will remain the same.



API type 1:

Description:

PubMed articles (titles and abstracts) are processed by text-mining tools to extract bio-entities (protein, gene, miRNA and so on), and bio-relations (phosphorylation, miRNA-target gene relation and so on). Related properties for each bio-entity and bio-relation are also saved in the database. Here we offer full information API, together with bio-entity data subset API and bio-relation data subset API.

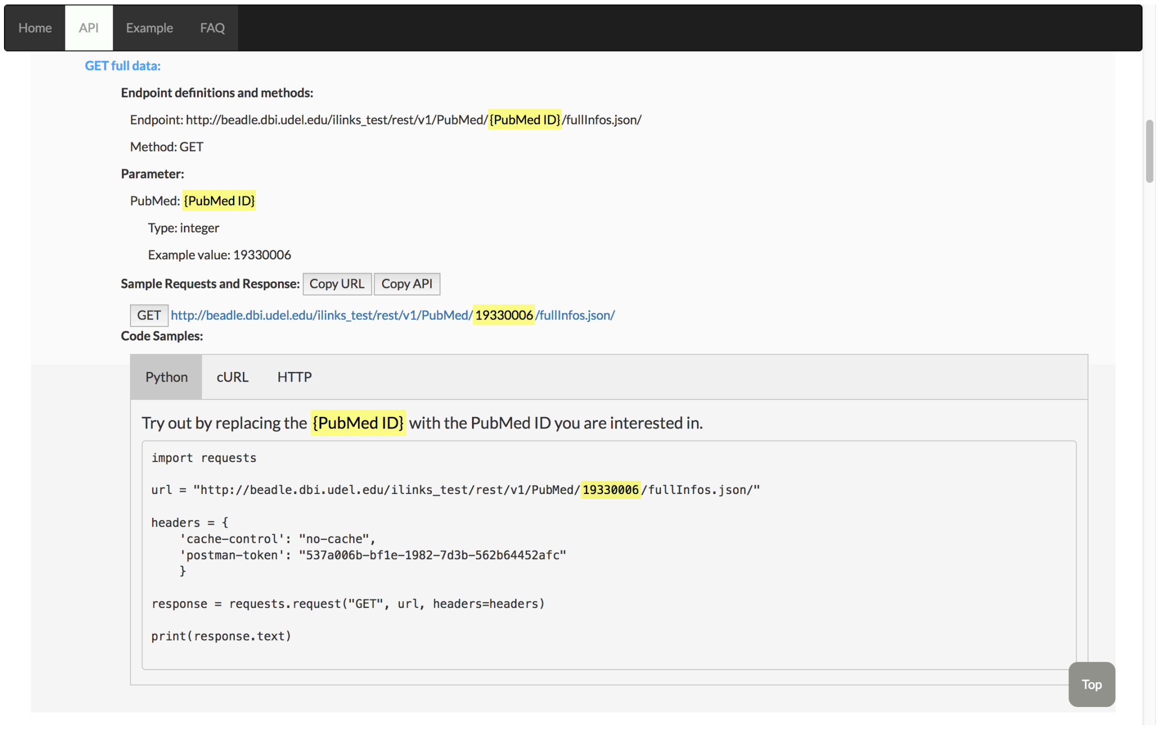
Data Model:

Entity

ATTRIBUTE	TYPE	DESCRIPTION
ENTITY / NORMALIZED_SOURCE_ID	tuple in Python	Biological entity normalization in form (source, ID) for example, (Entrez, Entrez ID)
ENTITY / BIOENTITY_NAME	string	Biological entity tagging name (not the exact word phrased in the article).
ENTITY / PREFERRED_NAME	string	Caution: Current status (V1.0) is blank, saving for upcoming text-mining tools.
ENTITY / BIOENTITY_TYPE	string	Biological entity type. V1.0 APIs contain Gene, Protein, Proteoform, and miRNA.
ENTITY / DESCRIPTION	string	Short description for each biological entity type. (constant across articles).
ENTITY / DATA_SOURCE_NAME	string	Text-mining tools or open text-mining resources.
ENTITY / ENTITY_CHAR_START	integer	The number of characters the referring biological entity is started.
ENTITY / ENTITY_CHAR_END	integer	The number of characters the referring biological entity is ended.
ENTITY / ATTRIBUTE_TYPE	string	V1.0 APIs contain proteoform information for biological entities.
ENTITY / SITE	string	Phosphorylation site.
ENTITY / SITE_CHAR_START	integer	The number of characters the referring phosphorylation site is started.
ENTITY / SITE_CHAR_END	integer	The number of characters the referring phosphorylation site is ended.
ENTITY / SENTENCE_TEXT	string	The sentence contains tagged biological entities and biological relations.

Figure 4.4: API type introduction in API page (http://beadle.dbi.udel.edu/ilinks_test/rest/API/#A10)

For each API, endpoint, method, parameters, request, response and sample code are provided. Because the APIs in this project are designed as read-only, GET is the only access method available. User can replace the parameter with the value they are interested in or use the example value to try out the API call. API can be checked through browser by typing in the URL (user can also click on the sample request URL). Aside from that, users can also try Python function, cURL (Linux command), or HTTP request to access the API. In order to give users a clear view of how the API looks like, pretty printed (with line break and indentation) API data are also provided. User can click on the GET button to see the API in JSON and XML format.



GET full data:

Endpoint definitions and methods:

Endpoint: http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/{PubMed ID}/fullInfos.json/

Method: GET

Parameter:

PubMed: {PubMed ID}

Type: integer

Example value: 19330006

Sample Requests and Response: Copy URL Copy API

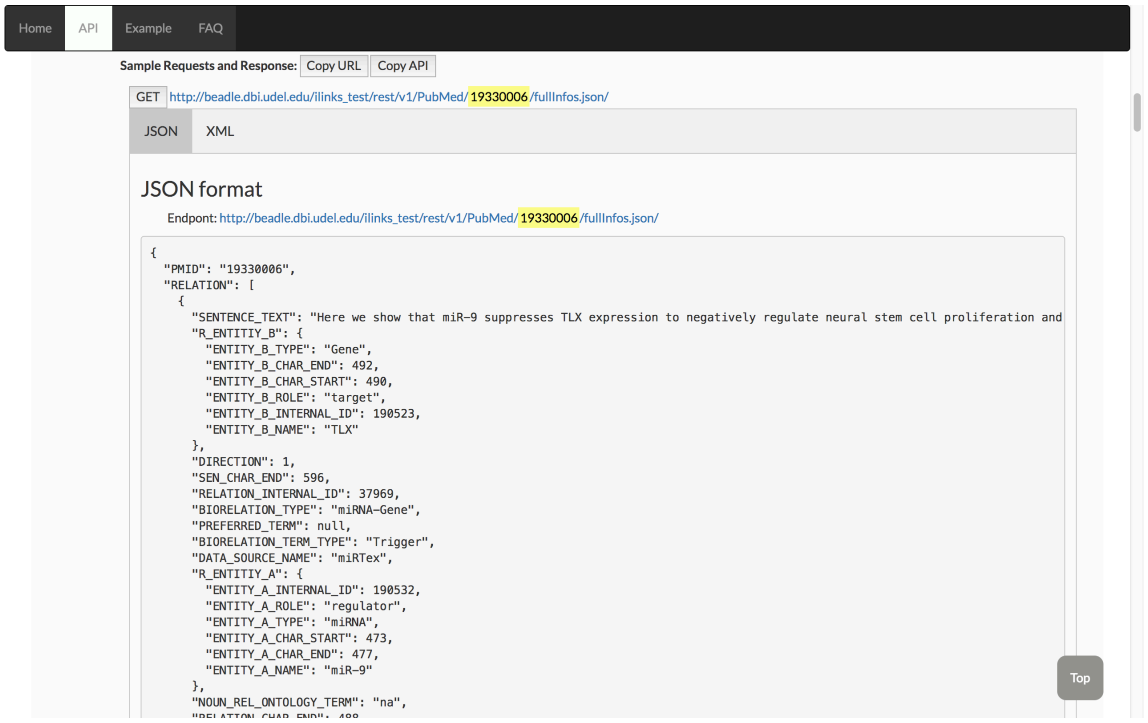
GET http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/

Code Samples:

Try out by replacing the {PubMed ID} with the PubMed ID you are interested in.

```
import requests

url = "http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/"

headers = {
    'cache-control': "no-cache",
    'postman-token': "537a006b-bf1e-1982-7d3b-562b64452afc"
    }

response = requests.request("GET", url, headers=headers)

print(response.text)
```

Figure 4.5: API description (sample code) in API page (http://beadle.dbi.udel.edu/ilinks_test/rest/API/#A11)



http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/

JSON format

Endpont: http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/

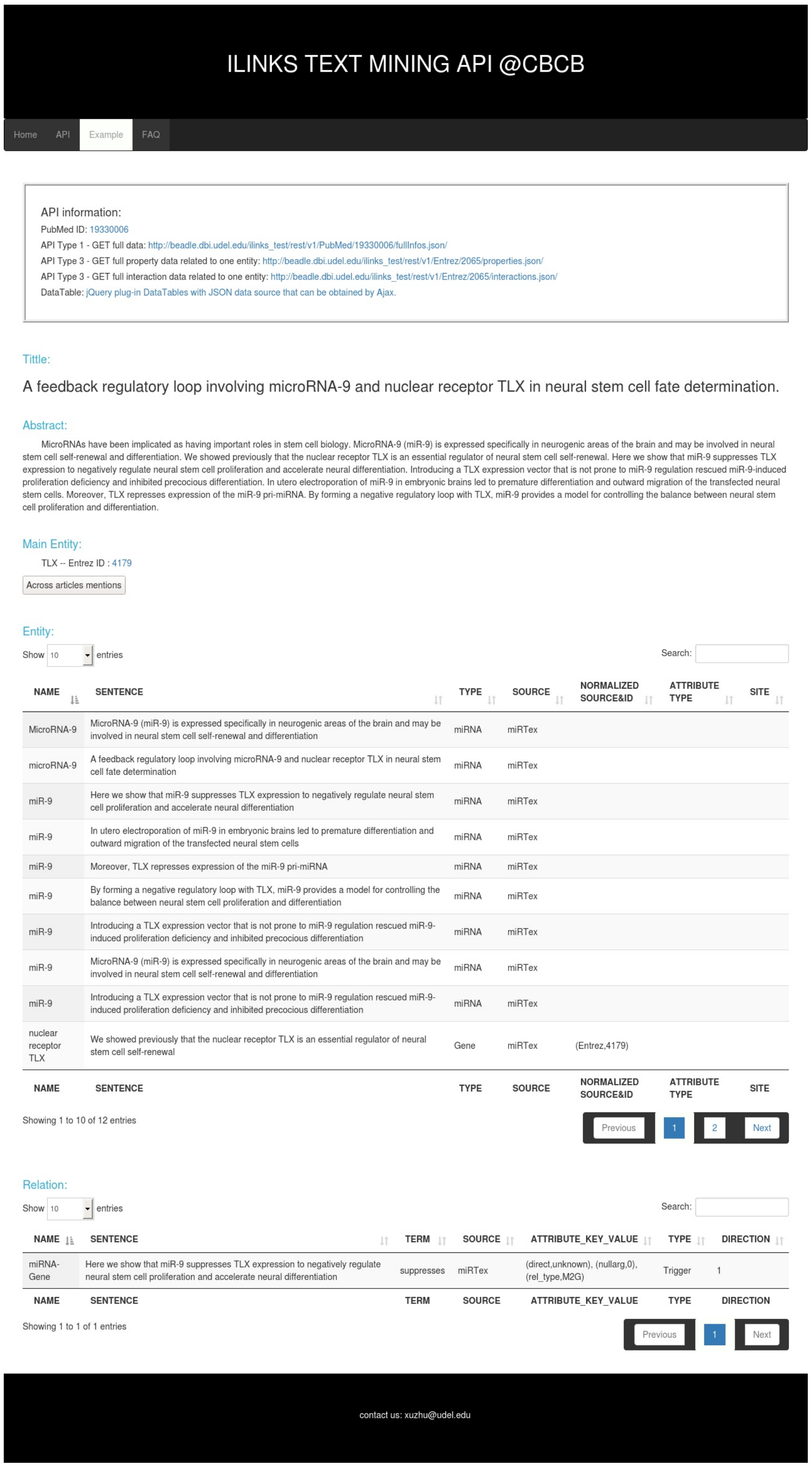
```
{
  "PMID": "19330006",
  "RELATION": [
    {
      "SENTENCE_TEXT": "Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and
      "R_ENTITIY_B": {
        "ENTITY_B_TYPE": "Gene",
        "ENTITY_B_CHAR_END": 492,
        "ENTITY_B_CHAR_START": 490,
        "ENTITY_B_ROLE": "target",
        "ENTITY_B_INTERNAL_ID": 190523,
        "ENTITY_B_NAME": "TLX"
      },
      "DIRECTION": 1,
      "SEN_CHAR_END": 596,
      "RELATION_INTERNAL_ID": 37969,
      "BIORELATION_TYPE": "miRNA-Gene",
      "PREFERRED_TERM": null,
      "BIORELATION_TERM_TYPE": "Trigger",
      "DATA_SOURCE_NAME": "miRTex",
      "R_ENTITIY_A": {
        "ENTITY_A_INTERNAL_ID": 190532,
        "ENTITY_A_ROLE": "regulator",
        "ENTITY_A_TYPE": "miRNA",
        "ENTITY_A_CHAR_START": 473,
        "ENTITY_A_CHAR_END": 477,
        "ENTITY_A_NAME": "miR-9"
      },
      "NOUN_REL_ONTOLOGY_TERM": "na",
      "RELATION_CHAR_END": 488
```

figure 4.6: API response (JSON format result) in API page

(http://beadle.dbi.udel.edu/ilinks_test/rest/API/#A11)

Example page

Flowing the API design principle, dogfooding is implemented on this page. At the top of this page, related resources and APIs are listed with clickable links. We use three APIs (from two types) to generate this example page. With these three APIs, user can have PubMed article text, text-mining result in Entity table and Relation table, and one special section to highlight the main entity in this article (across article mentions are listed in entity property table and entity interaction table). User can click on the 'Across article mention' button to expand or hide tables. The overlay of this page is also the guide for building the iLINKS web site search page (presented in next chapter).



ILINKS TEXT MINING API @CBCB

API information:

PubMed ID: 19330006

API Type 1 - GET full data: http://beadle.dbi.udel.edu/ilinks_test/rest/v1/PubMed/19330006/fullInfos.json/

API Type 3 - GET full property data related to one entity: http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/properties.json/

API Type 3 - GET full interaction data related to one entity: http://beadle.dbi.udel.edu/ilinks_test/rest/v1/Entrez/2065/interactions.json/

DataTable: jQuery plug-in DataTables with JSON data source that can be obtained by Ajax.

Tittle:

A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination.

Abstract:

MicroRNAs have been implicated as having important roles in stem cell biology. MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation. We showed previously that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal. Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation. Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation. In utero electroporation of miR-9 in embryonic brains led to premature differentiation and outward migration of the transfected neural stem cells. Moreover, TLX represses expression of the miR-9 pri-miRNA. By forming a negative regulatory loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and differentiation.

Main Entity:

TLX -- Entrez ID : 4179

Entity:

Show 10 entries Search:

NAME	SENTENCE	TYPE	SOURCE	NORMALIZED SOURCE&ID	ATTRIBUTE TYPE	SITE
MicroRNA-9	MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation	miRNA	miRTex			
microRNA-9	A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination	miRNA	miRTex			
miR-9	Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation	miRNA	miRTex			
miR-9	In utero electroporation of miR-9 in embryonic brains led to premature differentiation and outward migration of the transfected neural stem cells	miRNA	miRTex			
miR-9	Moreover, TLX represses expression of the miR-9 pri-miRNA	miRNA	miRTex			
miR-9	By forming a negative regulatory loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and differentiation	miRNA	miRTex			
miR-9	Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation	miRNA	miRTex			
miR-9	MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation	miRNA	miRTex			
miR-9	Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation	miRNA	miRTex			
nuclear receptor TLX	We showed previously that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal	Gene	miRTex	(Entrez,4179)		
NAME	SENTENCE	TYPE	SOURCE	NORMALIZED SOURCE&ID	ATTRIBUTE TYPE	SITE

Showing 1 to 10 of 12 entries

Relation:

Show 10 entries Search:

NAME	SENTENCE	TERM	SOURCE	ATTRIBUTE_KEY_VALUE	TYPE	DIRECTION
miRNA-Gene	Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation	suppresses	miRTex	(direct,unknown), (nullarg,0), (rel_type,M2G)	Trigger	1
NAME	SENTENCE	TERM	SOURCE	ATTRIBUTE_KEY_VALUE	TYPE	DIRECTION

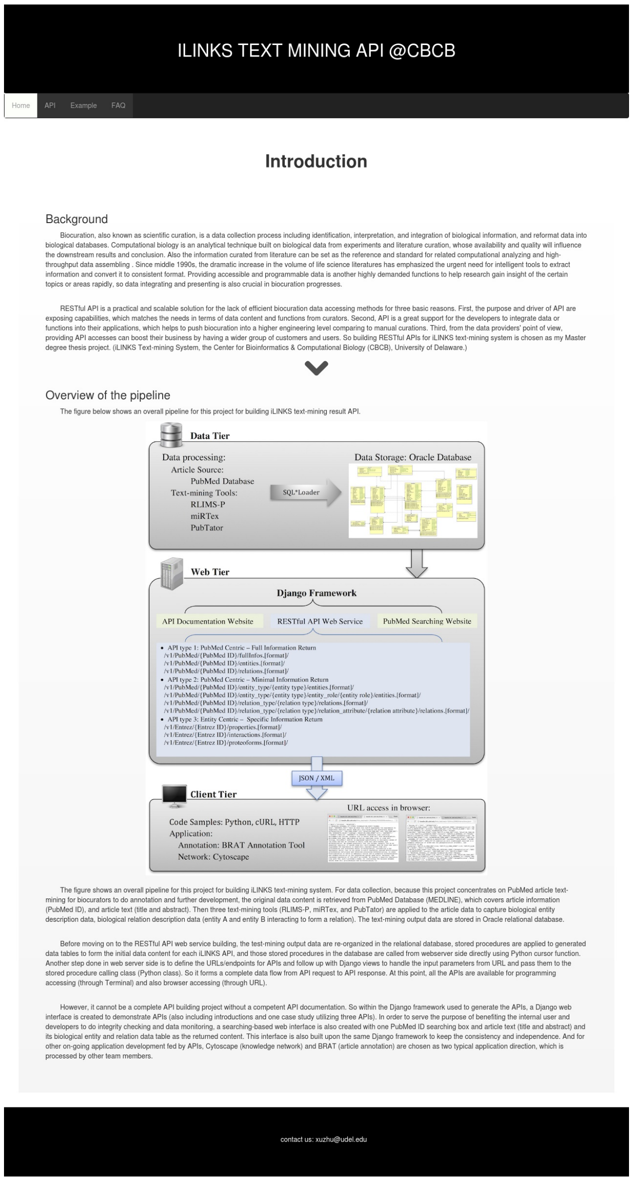
Showing 1 to 1 of 1 entries

contact us: xuzhu@udel.edu

Figure 4.7: Example page (http://beadle.dbi.udel.edu/ilinks_test/rest/example/)

Home page and FAQ page

Both 'Home' and 'FAQ' pages are served as informational page, 'Home' page contains the background and overall pipeline of iLINKS text-mining system; 'FAQ' page is about frequently asked questions about iLINKS RESTful APIs.



Introduction

Background

Biocuration, also known as scientific curation, is a data collection process including identification, interpretation, and integration of biological information, and reformat data into biological databases. Computational biology is an analytical technique built on biological data from experiments and literature curation, whose availability and quality will influence the downstream results and conclusion. Also the information curated from literature can be set as the reference and standard for related computational analyzing and high-throughput data assembling . Since middle 1990s, the dramatic increase in the volume of life science literatures has emphasized the urgent need for intelligent tools to extract information and convert it to consistent format. Providing accessible and programmable data is another highly demanded functions to help research gain insight of the certain topics or areas rapidly, so data integrating and presenting is also crucial in biocuration progresses.

RESTful API is a practical and scalable solution for the lack of efficient biocuration data accessing methods for three basic reasons. First, the purpose and driver of API are exposing capabilities, which matches the needs in terms of data content and functions from curators. Second, API is a great support for the developers to integrate data or functions into their applications, which helps to push biocuration into a higher engineering level comparing to manual curations. Third, from the data providers' point of view, providing API accesses can boost their business by having a wider group of customers and users. So building RESTful APIs for iLINKS text-mining system is chosen as my Master degree thesis project. (iLINKS Text-mining System, the Center for Bioinformatics & Computational Biology (CBCB), University of Delaware.)

Overview of the pipeline

The figure below shows an overall pipeline for this project for building iLINKS text-mining result API.

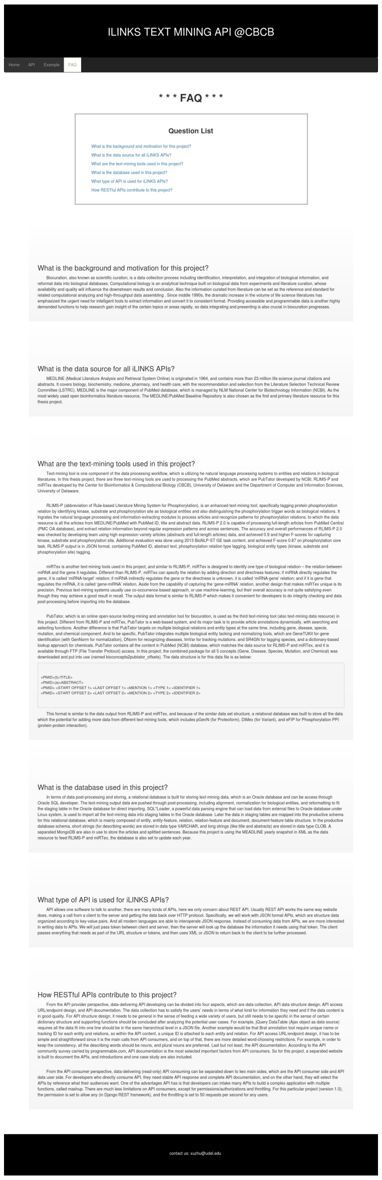
- API type 1: PubMed Centric – Full Information Return
  /v1/PubMed/{PubMed ID}/fullInfos.[format]/
  /v1/PubMed/{PubMed ID}/entities.[format]/
  /v1/PubMed/{PubMed ID}/relations.[format]/
- API type 2: PubMed Centric – Minimal Information Return
  /v1/PubMed/{PubMed ID}/entity_type/{entity type}/entities.[format]/
  /v1/PubMed/{PubMed ID}/entity_type/{entity type}/entity_role/{entity role}/entities.[format]/
  /v1/PubMed/{PubMed ID}/relation_type/{relation type}/relations.[format]/
  /v1/PubMed/{PubMed ID}/relation_type/{relation type}/relation_attribute/{relation attribute}/relations.[format]/
- API type 3: Entity Centric – Specific Information Return
  /v1/Entrez/{Entrez ID}/properties.[format]/
  /v1/Entrez/{Entrez ID}/interactions.[format]/
  /v1/Entrez/{Entrez ID}/proteoforms.[format]/

The figure shows an overall pipeline for this project for building iLINKS text-mining system. For data collection, because this project concentrates on PubMed article text-mining for biocurators to do annotation and further development, the original data content is retrieved from PubMed Database (MEDLINE), which covers article information (PubMed ID), and article text (title and abstract). Then three text-mining tools (RLIMS-P, miRTex, and PubTator) are applied to the article data to capture biological entity description data, biological relation description data (entity A and entity B interacting to form a relation). The text-mining output data are stored in Oracle relational database.

Before moving on to the RESTful API web service building, the test-mining output data are re-organized in the relational database, stored procedures are applied to generated data tables to form the initial data content for each iLINKS API, and those stored procedures in the database are called from webserver side directly using Python cursor function. Another step done in web server side is to define the URLs/endpoints for APIs and follow up with Django views to handle the input parameters from URL and pass them to the stored procedure calling class (Python class). So it forms a complete data flow from API request to API response. At this point, all the APIs are available for programming accessing (through Terminal) and also browser accessing (through URL).

However, it cannot be a complete API building project without a competent API documentation. So within the Django framework used to generate the APIs, a Django web interface is created to demonstrate APIs (also including introductions and one case study utilizing three APIs). In order to serve the purpose of benefiting the internal user and developers to do integrity checking and data monitoring, a searching-based web interface is also created with one PubMed ID searching box and article text (title and abstract) and its biological entity and relation data table as the returned content. This interface is also built upon the same Django framework to keep the consistency and independence. And for other on-going application development fed by APIs, Cytoscape (knowledge network) and BRAT (article annotation) are chosen as two typical application direction, which is processed by other team members.

contact us: xuzhu@udel.edu



* * * FAQ * * *

Question List

- What is the background and motivation for this project?
- What is the data source for all iLINKS APIs?
- What are the text-mining tools used in this project?
- What is the database used in this project?
- What type of API is used for iLINKS APIs?
- How RESTful APIs contribute to this project?

What is the background and motivation for this project?

Biocuration, also known as scientific curation, is a data collection process including identification, interpretation, and integration of biological information, and reformat data into biological databases. Computational biology is an analytical technique built on biological data from experiments and literature curation, whose availability and quality will influence the downstream results and conclusion. Also the information curated from literature can be set as the reference and standard for related computational analyzing and high-throughput data assembling . Since middle 1990s, the dramatic increase in the volume of life science literatures has emphasized the urgent need for intelligent tools to extract information and convert it to consistent format. Providing accessible and programmable data is another highly demanded functions to help research gain insight of the certain topics or areas rapidly, so data integrating and presenting is also crucial in biocuration progresses.

What is the data source for all iLINKS APIs?

MEDLINE (Medical Literature Analysis and Retrieval System Online) is originated in 1964, and contains more than 23 million life science journal citations and abstracts. It covers biology, biochemistry, medicine, pharmacy, and health care, with the recommendation and selection from the Literature Selection Technical Review Committee (LSTRC). MEDLINE is the major component of PubMed database, which is managed by NLM National Center for Biotechnology Information (NCBI). As the most widely used open bioinformatics literature resource, The MEDLINE/PubMed Baseline Repository is also chosen as the first and primary literature resource for this thesis project.

What are the text-mining tools used in this project?

Text-mining tool is one component of the data processing workflow, which is utilizing he natural language processing systems to entities and relations in biological literatures. In this thesis project, there are three text-mining tools are used to processing the PubMed abstracts, which are PubTator developed by NCBI, RLIMS-P and miRTex developed by the Center for Bioinformatics & Computational Biology (CBCB), University of Delaware and the Department of Computer and Information Sciences, University of Delaware.

RLIMS-P (abbreviation of Rule-based Literature Mining System for Phosphorylation), is an enhanced text-mining tool, specifically tagging protein phosphorylation relation by identifying kinase, substrate and phosphorylation site as biological entities and also distinguishing the phosphorylation trigger words as biological relations. It ingrates the natural language processing and information-extracting modules to process articles and recognize patterns for phosphorylation relations, to which the data resource is all the articles from MEDLINE/PubMed with PubMed ID, title and abstract data. RLIMS-P 2.0 is capable of processing full-length articles from PubMed Central (PMC OA database), and extract relation information beyond regular expression patterns and across sentences. The accuracy and overall performances of RLIMS-P 2.0 was checked by developing team using high expression variety articles (abstracts and full-length articles) data, and achieved 0.9 and higher F-scores for capturing kinase, substrate and phosphorylation site. Additional evaluation was done using 2013 BioNLP-ST GE task content, and achieved F-score 0.87 on phosphorylation core task. RLIMS-P output is in JSON format, containing PubMed ID, abstract text, phosphorylation relation type tagging, biological entity types (kinase, substrate and phosphorylation site) tagging.

miRTex is another text-mining tools used in this project, and similar to RLIMS-P, miRTex is designed to identify one type of biological relation – the relation between miRNA and the gene it regulates. Different than RLIMS-P, miRTex can specify the relation by adding direction and directness features; if miRNA directly regulates the gene, it is called 'miRNA-target' relation; if miRNA indirectly regulates the gene or the directness is unknown, it is called 'miRNA-gene' relation; and if it is gene that regulates the miRNA, it is called 'gene-miRNA' relation. Aside from the capability of capturing the 'gene-miRNA' relation, another design that makes miRTex unique is its precision. Previous text-mining systems usually use co-occurrence-based approach, or use machine-learning, but their overall accuracy is not quite satisfying even though they may achieve a good result in recall. The output data format is similar to RLIMS-P which makes it convenient for developers to do integrity checking and data post-processing before importing into the database.

PubTator, which is an online open-source texting-mining and annotation tool for biocuration, is used as the third text-mining tool (also text-mining data resource) in this project. Different from RLIMS-P and miRTex, PubTator is a web-based system, and its major task is to provide article annotations dynamically, with searching and selecting functions. Another difference is that PubTator targets on multiple biological relations and entity types at the same time, including gene, disease, specie, mutation, and chemical component. And to be specific, PubTator integrates multiple biological entity tacking and normalizing tools, which are GeneTUKit for gene identification (with GenNorm for normalization), DNorm for recognizing diseases, tmVar for tracking mutations. and SR4GN for tagging species, and a dictionary-based lookup approach for chemicals. PubTator contains all the content in PubMed (NCBI) database, which matches the data source for RLIMS-P and miRTex, and it is available through FTP (File Transfer Protocol) access. In this project, the combined package for all 5 concepts (Gene, Disease, Species, Mutation, and Chemical) was downloaded and put into use (named bioconcepts2pubtator_offsets). The data structure is for this data file is as below:

```
<PMID>|t|<TITLE>
<PMID>|a|<ABSTRACT>
<PMID> <START OFFSET 1> <LAST OFFSET 1> <MENTION 1> <TYPE 1> <IDENTIFIER 1>
<PMID> <START OFFSET 2> <LAST OFFSET 2> <MENTION 2> <TYPE 2> <IDENTIFIER 2>
```

This format is similar to the data output from RLIMS-P and miRTex, and because of the similar data set structure, a relational database was built to store all the data which the potential for adding more data from different text-mining tools, which includes pGenN (for Proteoform), DiMex (for Variant), and eFIP for Phosphorylation PPI (protein-protein interaction).

What is the database used in this project?

In terms of data post-processing and storing, a relational database is built for storing text-mining data, which is an Oracle database and can be access through Oracle SQL developer. The text-mining output data are pushed through post-processing, including alignment, normalization for biological entities, and reformatting to fit the staging table in the Oracle database for direct importing. SQL*Loader, a powerful data parsing engine that can load data from external files to Oracle database under Linux system, is used to import all the text-mining data into staging tables in the Oracle database. Later the data in staging tables are mapped into the productive schema for this relational database, which is mainly composed of entity, entity-feature, relation, relation-feature and document, document-feature table structure. In the productive database schema, short strings (for describing words) are stored in data type VARCHAR, and long strings (like title and abstracts) are stored in data type CLOB. A separated MongoDB are also in use to store the articles and splitted sentences. Because this project is using the MEADLINE yearly snapshot in XML as the data resource to feed RLIMS-P and miRTex, the database is also set to update each year.

What type of API is used for iLINKS APIs?

API allows one software to talk to another, there are many kinds of APIs, here we only concern about REST API. Usually REST API works the same way website does, making a call from a client to the server and getting the data back over HTTP protocol. Specifically, we will work with JSON format APIs, which are structure data organized according to key-value pairs. And all modern languages are able to interoperate JSON response. Instead of consuming data from APIs, we are more interested in writing data to APIs. We will just pass token between client and server, then the server will look up the database the information it needs using that token. The client passes everything that needs as part of the URL structure or tokens, and then uses XML or JSON to return back to the client to be further processed.

How RESTful APIs contribute to this project?

From the API provider perspective, data-delivering API developing can be divided into four aspects, which are data collection, API data structure design, API access URL/endpoint design, and API documentation. The data collection has to satisfy the users' needs in terms of what kind for information they need and if the data content is in good quality. For API structure design, it needs to be general in the sense of feeding a wide variety of users, but still needs to be specific in the sense of certain dictionary structure and supporting functions should be concluded after analyzing the potential user cases. For example, jQuery DataTable (Ajax object as data source) requires all the data fit into one line should be in the same hierarchical level in a JSON file. Another example would be that Brat annotation tool require unique name or tracking ID for each entity and relations, so within the API content, a unique ID is attached to each entity and relation. For API access URL/endpoint design, it has to be simple and straightforward since it is the main calls from API consumers, and on top of that, there are more detailed word-choosing restrictions. For example, in order to keep the consistency, all the describing words should be nouns, and plural nouns are preferred. Last but not least, the API documentation. According to the API community survey carried by programmable.com, API documentation is the most selected important factors from API consumers. So for this project, a separated website is built to document the APIs, and introductions and one case study are also included.

From the API consumer perspective, data-delivering (read-only) API consuming can be separated down to two main sides, which are the API consumer side and API data user side. For developers who directly consume API, they need stable API response and complete API documentation, and on the other hand, they will select the APIs by reference what their audiences want. One of the advantages API has is that developers can intake many APIs to build a complex application with multiple functions, called mashup. There are much less limitations on API consumers, except for permissions/authorizations and throttling. For this particular project (version 1.0), the permission is set to allow any (in Django REST framework), and the throttling is set to 50 requests per second for any users.

contact us: xuzhu@udel.edu

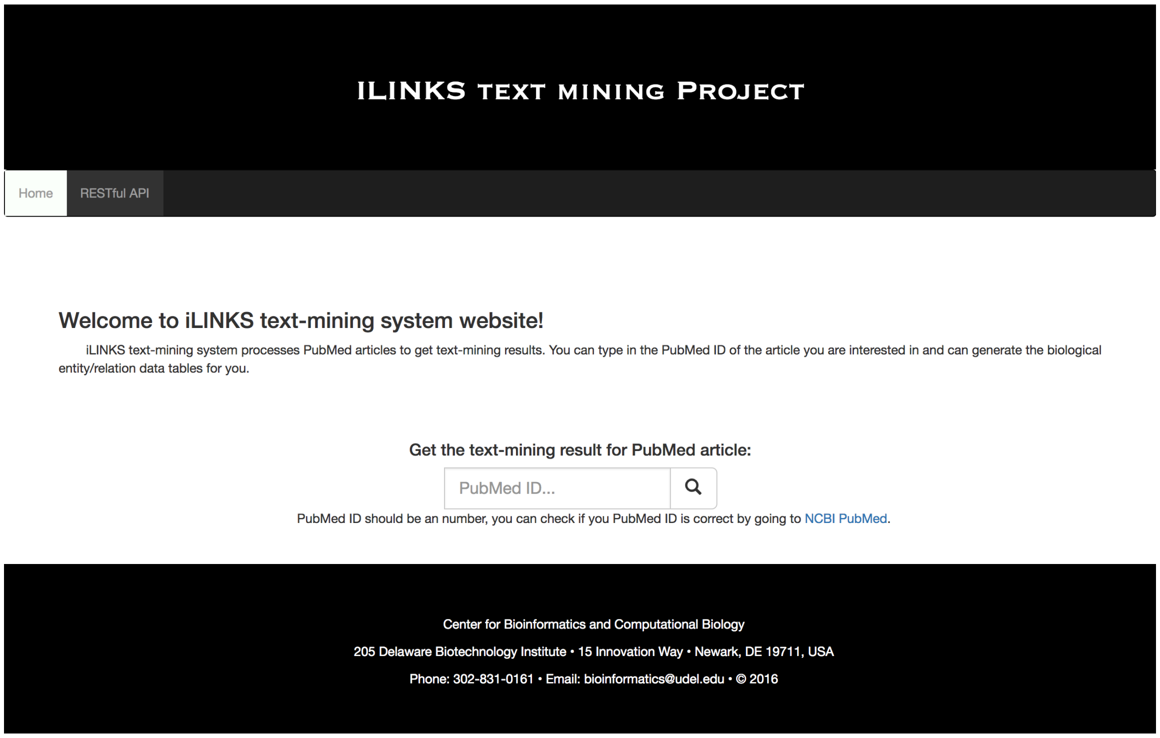
Figure 4.8: API Home page (http://beadle.dbi.udel.edu/ilinks_test/rest/) and FAQ page (http://beadle.dbi.udel.edu/ilinks_test/rest/faq/)

Chapter 5

EXAMPLE USE CASE

As part of iLINKS text-mining system website, a PubMed ID search page is built for external users to get the text-mining data in table format and for internal users to do integrity checking. Because this page is populated by the data returned from the APIs, this web page also serves as one self-checking mechanism for the API design and implementation. API developer can be sure that the APIs worked as expected. For external users and API consumers, the design and implementation of this page can also be used as an example and reference for them to develop their applications. The API testing website, together with API documentation web site, all developed using Django framework, which is open for plugging in new applications populated by the APIs, and also leaves potential API upgrade. Other than that, this all-in-one framework makes it easy to move to Docker or AWS (Amazon Web Services).

The PubMed ID search page contains two sections, one is search input box, the other is search result, which consists of PubMed source link, PubMed article text, text-mining data (biological entity table, biological relation table). The search input is first validated before passing to other functions. The input data type is limited to integer numbers because PubMed ID is an integer number. [46] If the input is not a valid PubMed ID (not found in the database), an alert box will pop out indicating 'Invalid PubMed ID!', there will be no search result shown in the result section. We also check if there are text-mining data for input PubMed article available in the database, in other words, if the 'ENTITY' and 'RELATION' are not detected by our text-mining tools, 'No text-mining data available!' alert message will pop up.


iLINKS TEXT MINING PROJECT

Home | RESTful API

Welcome to iLINKS text-mining system website!

iLINKS text-mining system processes PubMed articles to get text-mining results. You can type in the PubMed ID of the article you are interested in and can generate the biological entity/relation data tables for you.

Get the text-mining result for PubMed article:

PubMed ID...

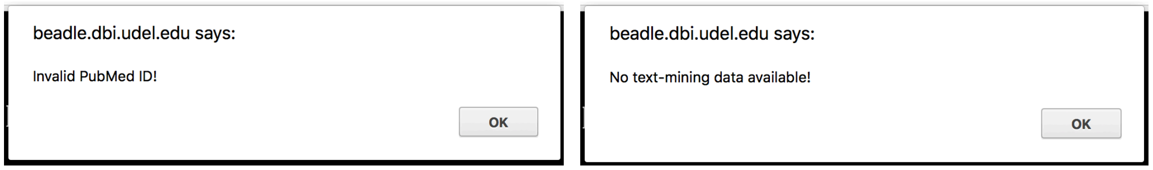
PubMed ID should be an number, you can check if you PubMed ID is correct by going to NCBI PubMed.

Center for Bioinformatics and Computational Biology

205 Delaware Biotechnology Institute • 15 Innovation Way • Newark, DE 19711, USA

Phone: 302-831-0161 • Email: bioinformatics@udel.edu • © 2016

Figure 5.1: Search page in iLINKS text-mining system website (http://beadle.dbi.udel.edu/ilinks_test/)


beadle.dbi.udel.edu says:

Invalid PubMed ID!

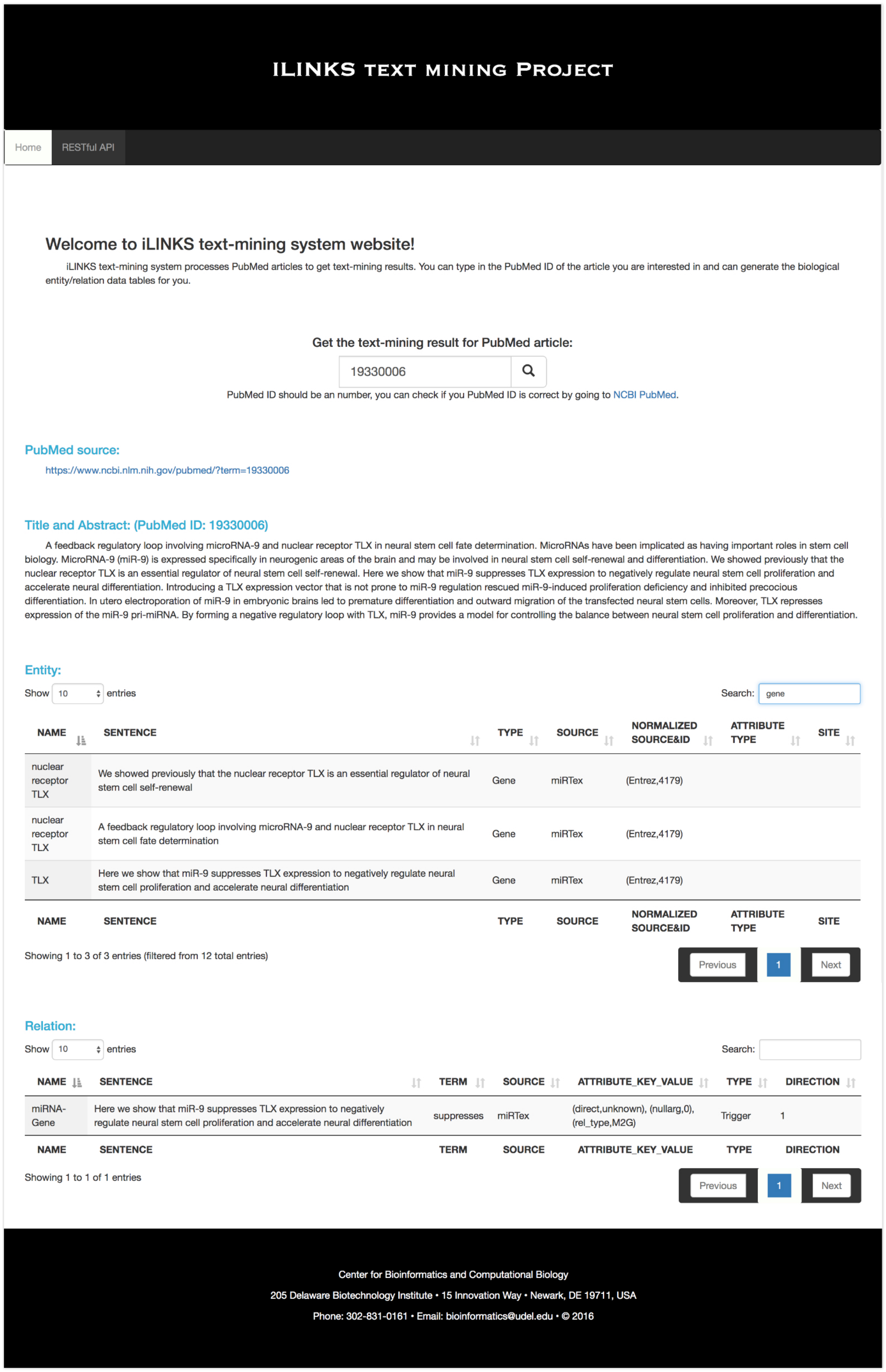
OK

beadle.dbi.udel.edu says:

No text-mining data available!

OK

Figure 5.2: ‘Invalid PubMed ID’ alert (with input ‘1234567890’) and ‘No text-mining data available’ alert (with input ‘1’) on search page



iLINKS text mining Project

Home RESTful API

Welcome to iLINKS text-mining system website!

iLINKS text-mining system processes PubMed articles to get text-mining results. You can type in the PubMed ID of the article you are interested in and can generate the biological entity/relation data tables for you.

Get the text-mining result for PubMed article:

19330006

PubMed ID should be an number, you can check if you PubMed ID is correct by going to NCBI PubMed.

PubMed source:

https://www.ncbi.nlm.nih.gov/pubmed/?term=19330006

Title and Abstract: (PubMed ID: 19330006)

A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination. MicroRNAs have been implicated as having important roles in stem cell biology. MicroRNA-9 (miR-9) is expressed specifically in neurogenic areas of the brain and may be involved in neural stem cell self-renewal and differentiation. We showed previously that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal. Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation. Introducing a TLX expression vector that is not prone to miR-9 regulation rescued miR-9-induced proliferation deficiency and inhibited precocious differentiation. In utero electroporation of miR-9 in embryonic brains led to premature differentiation and outward migration of the transfected neural stem cells. Moreover, TLX represses expression of the miR-9 pri-miRNA. By forming a negative regulatory loop with TLX, miR-9 provides a model for controlling the balance between neural stem cell proliferation and differentiation.

Entity:

Show 10 entries Search: gene

NAME	SENTENCE	TYPE	SOURCE	NORMALIZED SOURCE&ID	ATTRIBUTE TYPE	SITE
nuclear receptor TLX	We showed previously that the nuclear receptor TLX is an essential regulator of neural stem cell self-renewal	Gene	miRTex	(Entrez,4179)		
nuclear receptor TLX	A feedback regulatory loop involving microRNA-9 and nuclear receptor TLX in neural stem cell fate determination	Gene	miRTex	(Entrez,4179)		
TLX	Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation	Gene	miRTex	(Entrez,4179)		
NAME	SENTENCE	TYPE	SOURCE	NORMALIZED SOURCE&ID	ATTRIBUTE TYPE	SITE

Showing 1 to 3 of 3 entries (filtered from 12 total entries)

Previous 1 Next

Relation:

Show 10 entries Search:

NAME	SENTENCE	TERM	SOURCE	ATTRIBUTE_KEY_VALUE	TYPE	DIRECTION
miRNA-Gene	Here we show that miR-9 suppresses TLX expression to negatively regulate neural stem cell proliferation and accelerate neural differentiation	suppresses	miRTex	(direct,unknown), (nullarg,0), (rel_type,M2G)	Trigger	1
NAME	SENTENCE	TERM	SOURCE	ATTRIBUTE_KEY_VALUE	TYPE	DIRECTION

Showing 1 to 1 of 1 entries

Previous 1 Next

Center for Bioinformatics and Computational Biology

205 Delaware Biotechnology Institute • 15 Innovation Way • Newark, DE 19711, USA

Phone: 302-831-0161 • Email: bioinformatics@udel.edu • © 2016

Figure 5.3: Search result display (PubMed-19330006), with ‘gene’ filter on entity table

The PubMed source link is linking back to the original page in PubMed website for the article, which is added here for users to double check if their PubMed IDs are correct. Under the search box, the link for PubMed home page is also provided for users who are not quite sure about the PubMed ID they want to search. The article text (titles and abstract combined) is displayed after the source link, in which article title is the first sentence of the paragraph. The two tables are shown above (entity table and relation table) are both jQuery DataTable, which has its basic functions including page length (number of rows per page), previous/next page, sorting (alphabet or number order), and a search function within the table for any text content. These plugins provide users an easy-to-use interface to check the data and find the information they want. With jQuery DataTable as a base, there are more modifications and style changes can be done later on to fit different development purposes.

The 'RESTful API' tab in the navigation bar will redirect the users to the API documentation website introduced in the previous chapter.

Chapter 6

DISCUSSION AND FUTURE WORK

These APIs designed and implemented in this thesis project fulfilled the purpose of transferring data and providing a stable, scalable working environment/base for downstream application development. During the progress of designing and generating APIs, we did integrity checking constantly. As the result, the APIs not only provide the necessary functionalities for rest of the team project, they also helped other team members to improve their data processing and storing mechanisms. This API implementation also becomes a guideline and reference for other API building projects. Actually, as the iLINKS text-mining system project continues, intermediate step APIs (for internal checking) and the APIs for potential applications (Cytoscape and BRAT) will be built in the future.

The RESTful APIs for iLINKS text-mining system has served a crucial role in delivering data and supporting other application development. As iLINKS text-mining system project progresses, the APIs should move on to the next version with a wider data source, better-organized data content and a higher coverage on biological concepts. For version 1.0, the data source is PubMed article, but in version 2.0, full-length article will be processed and integrated into APIs. More text-mining tools will be introduced into the project to process articles and capture different biological concepts. Because of the larger amount of text content and a more complex text-mining data output, an upgrade in API structure design is needed in order to ensure the data quality of the APIs and the accuracy of their downstream applications. Data normalization and simplification would be two of the approaches to organizing new data in the database. Web tier API generation also need to have the ability to handle more parameters and potential errors by adding more parameter checking and exceptions reporting.

In API implementation, Flask can be used to as another option for implementing APIs. Flask is very similar to Django framework, which also has a Python based model–view–controller (MVC) architecture. [47] However, Django framework is an opinionated framework, which has some assumptions and pre-built structures, Flask, on the other hand, is a micro web-development framework. [48] [49] This feature makes Flask quicker than Django, but the performance will also depend on other factors. [50] Also because Flask has a much simple structure, the learning curve will be much lower than learning Django framework; this becomes important when the project is hosted at school and constantly needs new students to get involved. Flask-RESTful is an extension from Flask that can help to build RESTful APIs quickly and effortlessly. [51]

For the client-side code, React (a JavaScript library for creating interfaces) can be a good choice for the iLINKS text-mining system project to support client-side application development. [52] As to store the text-mining data and article text, Mango Database could also be an option. [53]

In API version 2.0, the documentation should be updated accordingly. We can choose to either update the documentation built for version 1.0 APIs or redo the documentation using Swagger UI to have a more standardized interface. [54]

Chapter 7

CONCLUSION

In conclusion, the RESTful APIs played an important role in not only the iLINKS text-mining system but also the biocuration field for three reasons. First, the APIs achieved the goal of helping the biocuration for PubMed articles with programmable data access methods. The APIs built in this project can be consumed by developers to develop different kinds of applications for biocuration; the search website supported by the APIs can be a go-to tool for all biocurators who want to read the PubMed article and get the extracted biological entity and relation information. Second, with the functionality supported by the APIs, iLINKS text-mining system is capable of providing biological entity and relationship by PubMed ID based search, Cytoscape network visualization, and BRAT annotation. Other potential application development will also benefit from these APIs. Third, with the support of these RESTful APIs, the entire data processing pipeline can be broken down into functional building blocks and reassembled.

This thesis describes the design and implementation of the RESTful APIs in detail along with application examples. For the wide variety of readers, the technical background and related works are also included. The thesis aims to deliver a clear description of the API building process and also serves as a reference in APIs development in the biocuration field. The API documentation website is at http://beadle.dbi.udel.edu/ilinks_test/rest/.

RESERENCES

1. Burge, Sarah et al. “Biocurators and Biocuration: Surveying the 21st Century Challenges.” Database: The Journal of Biological Databases and Curation2012 (2012): bar059. PMC. Web. 13 May 2017.

2. Bourne, Philip E, and Johanna McEntyre. “Biocurators: Contributors to the World of Science.” *PLoS Computational Biology* 2.10 (2006): e142. *PMC*. Web. 14 May 2017.

3. Howe, Authorship Doug et al. “Big Data: The Future of Biocuration.” Nature455.7209 (2008): 47–50. PMC. Web. 13 May 2017.

4. Christensson, Per. "API Definition." TechTerms. Sharpened Productions, 20 June 2016. Web. 06 July 2017. <https://techterms.com/definition/api>.

5. Graham, Paul (November 2005). "Web 2.0". Retrieved 2006-08-02. I first heard the phrase 'Web 2.0' in the name of the Web 2.0 conference in 2004.

6. "Web Services Architecture". World Wide Web Consortium. 11 February 2004. 3.1.3 Relationship to the World Wide Web and REST Architectures. Retrieved 29 September 2016.

7. Nations, Daniel. "Understanding the Purpose of Web Applications." Lifewire. Lifewire, 17 Oct. 2016. Web. 08 July 2017. <https://www.lifewire.com/what-is-a-web-application-3486637>.

8. Berners-Lee, Tim; Fielding, Roy T.; Nielsen, Henrik Frystyk. "Method Definitions". Hypertext Transfer Protocol -- HTTP/1.0. IETF. pp. 30-32. sec. 8. RFC 1945.

9. Sayers E. A General Introduction to the E-utilities. In: Entrez Programming Utilities Help [Internet]. Bethesda (MD): National Center for Biotechnology Information (US); 2010-. Available from: https://www.ncbi.nlm.nih.gov/books/NBK25497/

10. Steffensen JL, Dufault-Thompson K, Zhang Y. PSAMM: A Portable System for the Analysis of Metabolic Models. PLOS Comput Biol. Public Library of Science; 2016;12: e1004732. doi:10.1371/journal.pcbi.1004732.

11. Santos, Wendell. "195 Science APIs: Springer, EPA and NCBI." ProgrammableWeb. ProgrammableWeb, 28 Mar. 2012. Web. 28 June 2017. <https://www.programmableweb.com/news/195-science-apis-springer-epa-and-ncbi/2012/03/28>.

12. Wei, Chih-Hsuan, Hung-Yu Kao, and Zhiyong Lu. “PubTator: A Web-Based Text Mining Tool for Assisting Biocuration.” Nucleic Acids Research 41.Web Server issue (2013): W518–W522. PMC. Web. 12 June 2017.

13. Chih-Hsuan Wei, Robert Leaman, Zhiyong Lu; Beyond accuracy: creating interoperable and scalable text-mining web services. *Bioinformatics* 2016; 32 (12): 1907-1910. doi: 10.1093/bioinformatics/btv760

14. Wei, Chih-Hsuan. "NCBI Text Mining Web APIs." National Center for Biotechnology Information. U.S. National Library of Medicine, 16 Feb. 2016. Web. 28 June 2017. <https://www.ncbi.nlm.nih.gov/research/bionlp/APIs/>.

15. Wei CH et. al., PubTator: A PubMed-like interactive curation system for document triage and literature curation, in Proceedings of BioCreative 2012 workshop, Washington DC, USA, 145-150, 2012

16. Lidwell, William, and Kritina Holden, Jill Butler (2010). Universal Principles of Design, Revised and Updated: 125 Ways to Enhance Usability, Influence Perception, Increase Appeal, Make Better Design Decisions, and Teach through Design. London: Rockport Publishers. pp. 102–103. ISBN 1592535879

17. Mitra, Ronnie. "5 Universal API Design Principles." API Design and Strategy. API Academy, 12 Dec. 2015. Web. 28 June 2017. <http://www.apiacademy.co/resources/5-universal-api-design-principles/>.

18. Benslimane, Djamal, Schahram Dustdar, and Amit Sheth. "Services mashups: The new generation of web applications." IEEE Internet Computing 12.5 (2008).

19. Petersen, Jeremy. "Benefits of using the n-tiered approach for web applications." URL: http://www. adobe. com/devnet/coldfusion/articles/ntier. html (2001).

20. NIH. "MEDLINE Fact Sheet." U.S. National Library of Medicine. National Institutes of Health, 5 Dec. 2013. Web. 28 June 2017. <https://www.nlm.nih.gov/pubs/factsheets/medline.html#>.

21. Torii, Manabu et al. “RLIMS-P 2.0: A Generalizable Rule-Based Information Extraction System for Literature Mining of Protein Phosphorylation Information.” IEEE/ACM transactions on computational biology and bioinformatics / IEEE, ACM 12.1 (2015): 17–29. PMC. Web. 12 June 2017.

22. Li, Gang et al. “miRTex: A Text Mining System for miRNA-Gene Relation Extraction.” Ed. Andrey Rzhetsky. PLoS Computational Biology 11.9 (2015): e1004391. PMC. Web. 12 June 2017.

23. Bagewadi, Shweta et al. “Detecting miRNA Mentions and Relations in Biomedical Literature.” F1000Research 3 (2014): 205. PMC. Web. 12 June 2017.

24. Huang, Minlie, Jingchen Liu, and Xiaoyan Zhu. “GeneTUKit: A Software for Document-Level Gene Normalization.” Bioinformatics 27.7 (2011): 1032–1033. PMC. Web. 12 June 2017.

25. Wei, Chih-Hsuan, and Hung-Yu Kao. “Cross-Species Gene Normalization by Species Inference.” BMC Bioinformatics 12.Suppl 8 (2011): S5. PMC. Web. 12 June 2017.

26. Leaman, Robert, Rezarta Islamaj Doğan, and Zhiyong Lu. “DNorm: Disease Name Normalization with Pairwise Learning to Rank.” Bioinformatics 29.22 (2013): 2909–2917. PMC. Web. 12 June 2017.

27. Wei, Chih-Hsuan et al. “tmVar: A Text Mining Approach for Extracting Sequence Variants in Biomedical Literature.” Bioinformatics 29.11 (2013): 1433–1439. PMC. Web. 12 June 2017.

28. Wei, Chih-Hsuan, Hung-Yu Kao, and Zhiyong Lu. “SR4GN: A Species Recognition Software Tool for Gene Normalization.” Ed. Jan Aerts. PLoS ONE 7.6 (2012): e38460. PMC. Web. 12 June 2017

29. "Ubiquitous standards" Principles — API Design Guide 0.1 documentation. AusDTO, 02 June 2017. Web. 08 July 2017. <http://apiguide.readthedocs.io/en/latest/principles/standards.html>.

30. "Use RESTful service URLs." Building and using APIs — API Design Guide 0.1 documentation. AusDTO, 02 June 2017. Web. 08 July 2017. <http://apiguide.readthedocs.io/en/latest/build_and_publish/use_RESTful_urls.html>.

31. "Dogfooding" Principles — API Design Guide 0.1 documentation. AusDTO, 02 June 2017. Web. 08 July 2017. <http://apiguide.readthedocs.io/en/latest/principles/dogfood.html>.

32. Johnson, Tom. "Documenting APIs: A guide for technical writers." Documenting APIs: A guide for technical writers | Document REST APIs. I'd Rather Be Writing, 27 Mar. 2016. Web. 08 July 2017. <http://idratherbewriting.com/learnapidoc/>.

33. Tudor, Catalina O, Carl J Schmidt, and K Vijay-Shanker. "eGIFT: Mining Gene Information from the Literature." BMC Bioinformatics 11 (2010): 418. PMC. Web. 5 July 2017.

34. Christie, Tom. "Django REST Framework." Home - Django REST framework. N.p., 18 Aug. 2015. Web. 08 July 2017. <http://www.django-rest-framework.org/>.

35. "Database Cloud Service | Database as a Service (DbaaS) | Oracle Cloud." Cloud Service | Oracle Cloud. N.p., n.d. Web. 08 July 2017. <https://cloud.oracle.com/database>.

36. Christie, Tom. "Renderers." Renderers - Django REST framework. N.p., 18 Aug. 2015. Web. 08 July 2017. <http://www.django-rest-framework.org/api-guide/renderers/>.

37. "Dicttoxml 1.7.4." Dicttoxml 1.7.4: Python Package Index. Python Software Foundation, 08 July 2016. Web. 08 July 2017. <https://pypi.python.org/pypi/dicttoxml>.

38. Piotrowski, Przemyslaw. "Calling database stored procedures and other interesting aspects of advanced Python programming." Mastering Oracle Python, Part 5: Stored Procedures, Programming Python. Oracle, Mar. 2010. Web. 08 July 2017. <http://www.oracle.com/technetwork/articles/prez-stored-proc-084100.html>.

39. Eckerson, Wayne. "Three tier client/server architecture: Achieving scalability, performance and efficiency in client server applications." Open Information Systems 10.1 (1995).

40. Linthicum, David S. Next generation application integration: from simple information to Web services. Addison-Wesley Longman Publishing Co., Inc., 2003.

41. "Documentation." URL dispatcher | Django documentation | Django. Django Software Foundation, 1 Sept. 2016. Web. 08 July 2017. <https://docs.djangoproject.com/en/1.10/topics/http/urls/>.

42. "Documentation." Writing views | Django documentation | Django. Django Software Foundation, 1 Aug. 2016. Web. 08 July 2017. <https://docs.djangoproject.com/en/1.10/topics/http/views/>.

43. Johnson, Tom. "Most important factor in APIs is complete and accurate documentation." I'd Rather Be Writing - Tom Johnson. I'd Rather Be Writing, 15 Jan. 2015. Web. 08 July 2017. <http://idratherbewriting.com/2015/01/15/most-important-factor-in-apis-is-complete-and-accurate-documentation/>.

44. DuVander, Adam. "API Consumers Want Reliability, Documentation and Community." ProgrammableWeb. N.p., 07 Jan. 2013. Web. 08 July 2017. <https://www.programmableweb.com/news/api-consumers-want-reliability-documentation-and-community/2013/01/07>.

45. "Search the Largest API Directory on the Web." ProgrammableWeb. N.p., n.d. Web. 08 July 2017. <https://www.programmableweb.com/category/all/apis>.

46. "PMCID - PMID - Manuscript ID - DOI Converter." National Center for Biotechnology Information. U.S. National Library of Medicine, n.d. Web. 09 July 2017. <https://www.ncbi.nlm.nih.gov/pmc/pmctopmid/#converter>.

47. Ronacher, Armin. "Welcome to Flask." Welcome to Flask — Flask Documentation (0.12). Flask, n.d. Web. 09 July 2017. <http://flask.pocoo.org/docs/0.12/>.

48. Bitra, Tejaswi. "What does it mean by a 'opinionated' framework or library?" Quora. Quora, 11 Jan. 2017. Web. 9 July 2017. <https://www.quora.com/What-does-it-mean-by-a-opinionated-framework-or-library>.

49. "Web Frameworks for Python." WebFrameworks - Python Wiki. Python Software Foundation, 07 July 2017. Web. 09 July 2017. <https://wiki.python.org/moin/WebFrameworks>.

50. Klenov, Kirill. "Python frameworks' benchmarks." N.p., 09 June 2016. Web. 09 July 2017. <http://klen.github.io/py-frameworks-bench/>.

51. Burke, Kevin. "Flask-RESTful." Flask-RESTful — Flask-RESTful 0.2.1 documentation. Flask, 18 Mar. 2013. Web. 09 July 2017. <https://flask-restful.readthedocs.io/en/0.3.5/>.

52. "React - A JavaScript library for building user interfaces." Facebook.github.io. Facebook Open Source, 14 June 2017. Web. 09 July 2017. <https://facebook.github.io/react/>.

53. "The MongoDB 3.4 Manual." The MongoDB 3.4 Manual — MongoDB Manual 3.4. MongoDB, 29 Nov. 2016. Web. 09 July 2017. <https://docs.mongodb.com/manual/>.

54. "SWAGGER UI." Swagger. SmartBear Software, n.d. Web. 09 July 2017. <https://swagger.io/swagger-ui/>.

APPENDIX

A API CONTENT DATA OBJECT KEYS DEFINITION

ATTRIBUTE	TYPE	DESCRIPTION
PMID	integer	PubMed ID - as parameter.
ABSTRACT / SOURCE	string	Article/paper source - MEDLINE/PubMed for iLINKS API version 1.0.
ABSTRACT / ABSTRACT	string	Article/paper abstract (including title - the first sentence).
ENTITY / NORMALIZED_SOURCE_ID	tuple in Python	Biological entity normalization in form (source, ID) for example, (Entrez, Entrez ID)
ENTITY / BIOENTITY_NAME	string	Biological entity tagging name (not the exact word phrased in the article).
ENTITY / PREFERRED_NAME	string	Caution: Current status (V1.0) is blank, saving for upcoming text-mining tools.
ENTITY / BIOENTITY_TYPE	string	Biological entity type. V1.0 APIs contain Gene, Protein, Proteoform, and miRNA.
ENTITY / DESCRIPTION	string	Short description for each biological entity type. (constant across articles).
ENTITY / DATA_SOURCE_NAME	string	Text-mining tools or open text-mining resources.
ENTITY / ENTITY_CHAR_START	integer	The number of characters the referring biological entity is started.
ENTITY / ENTITY_CHAR_END	integer	The number of characters the referring biological entity is ended.
ENTITY / ATTRIBUTE_TYPE	string	V1.0 APIs contain proteoform information for biological entities.
ENTITY / SITE	string	Phosphorylation site.
ENTITY / SITE_CHAR_START	integer	The number of characters the referring phosphorylation site is started.
ENTITY / SITE_CHAR_END	integer	The number of characters the referring phosphorylation site is ended.
ENTITY / SENTENCE_TEXT	string	The sentence contains tagged biological entities and biological relations.

ENTITY / SEN_CHAR_START	integer	The number of characters the referring sentence is started.
ENTITY / SEN_CHAR_END	integer	The number of characters the referring sentence is ended.
ENTITY / ENTITY_INTERNAL_ID	integer	Biological entity tracking ID, unique for each entity, identical to BIOENTITY_ID for internal use.
RELATION / BIORELATION_TERM_NAME	string	Biological relation tagging term (not the exact word phrased in the article).
RELATION / PREFERRED_TERM	string	Caution: Current status (V1.0) is blank, saving for upcoming text-mining tools.
RELATION / BIORELATION_TERM_TYPE	string	Complimentary information for biological relation term.
RELATION / RELATION_CHAR_START	integer	The number of characters the referring biological relation is started.
RELATION / RELATION_CHAR_END	integer	The number of characters the referring biological relation is ended.
RELATION / BIORELATION_TYPE	string	Biological relation tagging type (not the exact word phrased in the article).
RELATION / REL_ONTOLOGY_TERM	string	Complementary phrasing for BIORELATION_TERM.
RELATION / NOUN_REL_ONTOLOGY_TERM	string	Complementary phrasing for BIORELATION_TERM.
RELATION / DIRECTION	integer	Biological relation direction, 1 for BIOENTITY_A to BIOENTITY_B, 2 for BIOENTITY_B to BIOENTITY_A, 0 for unknown.
RELATION / TM_RELATION_TYPE_NAME	string	Text-mining tools in using biological relation type tags.
RELATION / ATTRIBUTE_KEY_VALUE	string	Supporting descriptions for biological relations.
RELATION / DATA_SOURCE_NAME	string	Text-mining tools or open text-mining resources.
RELATION / SENTENCE_TEXT	string	The sentence contains tagged biological entities and biological relations.
RELATION / SEN_CHAR_START	integer	The number of characters the referring sentence is started.
RELATION / SEN_CHAR_END	integer	The number of characters the referring sentence is ended.

RELATION / RELATION_INTERNAL_ID	integer	Biological relation tracking ID, unique for each entity, identical to BIORELATION_ID for internal use.
RELATION / R_ENTITIY_A / ENTITY_A_INTERNAL_ID	integer	Biological entity A tracking ID, unique for each entity, identical to BIOENTITY_ID for internal use.
RELATION / R_ENTITIY_A / ENTITY_A_NAME	string	Biological entity A tagging name (not the exact word phrased in the article).
RELATION / R_ENTITIY_A / ENTITY_A_TYPE	string	Biological entity A type. V1.0 APIs contain Gene, Protein, Proteoform, and miRNA.
RELATION / R_ENTITIY_A / ENTITY_A_ROLE	string	Biological entity A role. V1.0 APIs contain Kinase, Interactant, Regulator, and Unknown.
RELATION / R_ENTITIY_A / ENTITY_A_CHAR_START	integer	The number of characters the referring biological entity A is started.
RELATION / R_ENTITIY_A / ENTITY_A_CHAR_END	integer	The number of characters the referring biological entity A is ended.
RELATION / R_ENTITIY_B / ENTITY_B_INTERNAL_ID	integer	Biological entity B tracking ID, unique for each entity, identical to BIOENTITY_ID for internal use.
RELATION / R_ENTITIY_B / ENTITY_B_NAME	string	Biological entity B tagging name (not the exact word phrased in the article).
RELATION / R_ENTITIY_B / ENTITY_B_TYPE	string	Biological entity B type. V1.0 APIs contain Gene, Protein, Proteoform, and miRNA.
RELATION / R_ENTITIY_B / ENTITY_B_ROLE	string	Biological entity B role. V1.0 APIs contain Substrate, Interactant, Target, and Unknown.
RELATION / R_ENTITIY_B / ENTITY_B_CHAR_START	integer	The number of characters the referring biological entity B is started.
RELATION / R_ENTITIY_B / ENTITY_B_CHAR_END	integer	The number of characters the referring biological entity B is ended.

Table A.1: API content data object keys definition.