CHAPTER FOUR

METHODOLOGY

4.1 EUTILS

Entrez system’s core aspects are generally accessed by eUtils in searching and retrieval aspects. Most of the datasets exist outside the Entrez system. Availability of desired data is to be checked before starting a project with the eUtils.

4.1.1 The Seven Entrez Eutils

- **EInfo**: information related to record count indexed in each field in addition to upadtion dates and other links available.
- **EGQuery**: matching records in Entrez database is provided with.
- **ESearch**: Responds with list of UIDs with respect to quire applied, along with the term translations of the query.
- **ESummary**: specifies UIDs in correspondence with document summary.
- **Post**: Stores History Server for each UID, respond with the corresponding query key and Web environment.
- **EFetch**: UIDs with the corresponding data records
- **ELink**: List of UIDs in a given database with IDs list.
Figure 4.1: PubMed parsing data

Excision of the lambda prophage from the chromosome of its Escherichia coli host requires the products of the two viral genes int and xis. This paper reports a purification of the lambda xis gene product using a complementation assay in which functional Xis must be added to purified Int and an E. coli-derived host factor extract. Excision recombination between a left (attL) and right (attR) prophage attachment site cloned on the same plasmid DNA substrate occurred efficiently under these conditions. Purified Int and Xis together could not carry out excision in vitro unless an extract derived from the E. coli host was added; purified integration host factor satisfied this requirement. Xis appears to have a molecular weight of 8500 as determined by polyacrylamide gel electrophoresis in the presence of sodium dodecyl sulfate. It possesses no detectable endonuclease or topoisomerase activities, does not appear to bind DNA to filters, and does not increase the ability of Int to bind DNA. The addition of Xis not only stimulated excisive recombination in vitro but also inhibited integrative recombination. Xis protected int protein from heat inactivation, suggesting a possible interaction between the two proteins. In light of these observations, possible roles for Xis in recombination are discussed.
4.1.2 Parser Design

Parsers are designed to work on the basis of events. Parser includes Scanner and Consumer objects.

4.1.2.1 Scanners

Scanners acquire the input from an information source and analyses it completely with care. And whenever it finds any useful information, it sends an event. Suppose if the scanner comes across a line having an organism name, it creates an “organism_name” event.

4.1.2.2 Consumers

The events that are created by scanners are then received by objects called Consumers. As per the above given strategy, consumer attains the information related to “organism_name” advent, and trails the same based on the requirement.

4.1.3 Events

In general Events are categorized into two forms, i.e., info events and section events. Info-events emphasizes on the position of information residing in a data drifts wherein the section events embarks on emphasizing sections within that stream. Info events relates to the pointing of data in particular location wherein section-events relates to onset and offset of particular process. In general section event names are specified as start_task and end_task where task is the event (task) name.
4.1.4 Parsing PubMed with ElementTree

Continuing with the theme of using stock Python to parse XML from NCBI, a sample parser code may look like

```python
def parse_myArticle(article):
    D = dict()
    D['pmid'] = article.findtext('MedlineCitation/PMID')
    journalinfo = article.find('MedlineCitation/MedlineJournalInfo')
    D['journal'] = journalinfo.findtext('MedlineTA')
```

Continuing with the theme of using stock Python to parse XML from NCBI, a sample parser code may look like

```python
def parse_myArticle(article):
    D = dict()
    D['pmid'] = article.findtext('MedlineCitation/PMID')
    journalinfo = article.find('MedlineCitation/MedlineJournalInfo')
    D['journal'] = journalinfo.findtext('MedlineTA')
```
a = article.find('MedlineCitation/Article')

D['volume'] = a.findtext('Journal/JournalIssue/Volume')

The above code may parse the XML output of PubMed and finds for ‘PMID’, ‘Journal information’ and ‘Volume’, and stores them as Dictionary object for better output.

### 4.2 Software Architecture

#### 4.2.1 System Requirements

- **Operating System**: Windows XP / VISTA
- **RAM**: 256 mb or higher
- **Software**: Python 2.5.4 with additional modules
  - NumPy, Biopython

#### 4.2.2 Additional Packages / Modules

Sometimes in spite of Python’s extensive standard library provides most of the programming needs, we may require to incorporate some new functionality to the Python installation as third party modules. In such case, it must be written in Python.
4.3 Experimental Set Up

4.3.1 Software Configuration

Software testing provides an objective, independent view of the software to allow the business to appreciate and understand the risks of software implementation. Test techniques include, but are not limited to the process of executing a program or application with the intent of finding software bugs.

4.3.1.1 Objectives

The process of testing has the following objectives:

1. Checking for an error while execution of a program.
2. probability of finding a undiscovered error, and
3. uncovering an undiscovered error
4.3.1.2 Installation of Python 2.5.4

If one selects the Python (command line) menu item, one will see the Python (command line) window. This will contain:

C:\Python25>python

Python 2.5.4 (r254:67916, Dec 23 2008, 15:10:54) [MSC v.1310 32 bit (Intel)] on

win32

On IDLE:

![Python Command line window](image)

**Figure 4.2** : Python Command line window

4.3.1.3 Installation of BioPython

A quick test must be made to ensure the correct installation of Biopython. Interestingly, this can be done by Python itself. Biopython could be established into higher level Bio and BioSQL catalogue, making sure that these
specified set of directories do exist in $PYTHONPATH ecological attribute. Care must be taken if the process is not done by default installing methodology, if so the PYTHONPATH must be set as follows:

The installation can be tested with the following code:

$ python

Python 2.5 (r25:51908, Nov 23 2009, 18:40:28)
[GCC 4.1.1 20061011 (Red Hat 4.1.1-30)] on linux2
Type "help", "copyright", "credits" or "license" for more information.

>>> from Bio.Seq import Seq
>>> from Bio.Alphabet.IUPAC import unambiguous_dna
>>> new_seq = Seq('GATCAGAAG', unambiguous_dna)
>>> new_seq[0:2]
Seq('GA', IUPACUnambiguousDNA())
>>> new_seq.translate()
Seq('DQK', HasStopCodon(IUPACProtein(), '*'))

4.3.1.4 Installation of Numpy

NumPy is the primary set of programming constraints for scientific approach with python consisting:

- an N-array object
- skeptical (broadcasting) methods.
- Methodologies for combining different levels of code like C/C++ and Fortran.
- Working aspects related to linear algebra, Fourier transform, and random number capabilities.

NumPy, in addition to its scientific aspects could also be used as a prominent n-array pool of data which is generally of generic type using which one could easily define contemporary data-types. NumPy is generally used in integration purpose for combination of different databases more fastly.

To make sure everything went right during the install, the python interpreter has to be fired at and the import of NumPy without any errors also has to be ensured.

After the installation the unit test suits must be run in Python:

These should run without errors.

We can also verify that NumPy is using optimized BLAS and LAPACK libraries by checking:

```python
>>> import numpy
>>> numpy.show_config()
```
**Guidelines for Software Installation**

For Windows, click-and-run installers have to be provided specific to the version of python, but first prerequisite software, in particular, NumPy must be installed.

An optional Biopython package is included in most Linux distributions that handles any prerequisite software.

One ought to install Apple's XCode in addition to 10.4 SDK which could be taken as mandatory.

Otherwise, one usually install it through source by uncompressing the downloaded archive, typically as we do in respect to MAC OS X and then by running those commands.

**Required Software**

- Python above 2.4 version is recommended.
- For C compiler (if compiling from source) support of distutuls is recommended. As I general gcc is said to have prominent role on UNIX-like platforms, the same care need not be taken in the case of working with Windows. The installation of Apple’s XCode is recommended on Mac OS.
- NumPy (Numerical Python) earlier to BioPython 1.49, [the older Numeric library] was used by Biopython.

**Elective Software**

For compiling Biopython:

- flex, The Fast Lexical Analyzer is used as a builder for Bio.PDB.mmCIF.MMCIFlex, used in the mechanism of parsing macromolecular Crystallographic Information Files (mmCIF)

Basic articulations of Biopython includes the other python libraries like:

- ReportLab, which includes methodologies for generating pdf graphics cipher.
- MySQLdb, acting as bridge between BioSQL and MySQL database

### 4.4 Text Mining From NCBI PubMed Databases

NCBI PubMed literature database was selected for this study. Initially, a generalized search without any limits was employed to retrieve articles related to *bioinformatics* and *computational biology*. As search results indicated the presence of keyword anywhere in the article (title, abstract, address, keywords and text), a more stringent search criterion was employed to identify the
number of articles that appeared when a search is performed either by individual or in combinations of keywords by limiting the search to Title and Abstract.

Only the Title and abstract search is considered in this study because the Title field in some articles refers to the most important keywords relative to the subject. Therefore, there is a validated disparity in information retrieved through text mining limited only to Titles and Abstract terms. Articles belonging to bioinformatics and computational biology are explicitly reported in journals. Some may have the term in Title/Abstract while some are representative of the field without keywords. Therefore, though a myriad of pertinent articles are located, preference is given to the two search techniques, Title and Abstract.

Title/Abstract is selected as limits to search the database in order to overcome false hits and to identify true positives. Therefore an article is considered true by positive only if the keyword is explicitly identified in Title/Abstract. Records without abstracts are considered true positives only if the title contains the keywords [88]. Finally, year wise growth in number of articles in each field was carried to find out the enormous amount of data deposited in PubMed.
4.5 Python Program for Mining Textual Information From PubMed Databases

urllib module in Python standard library, helps in requesting Entrez server directly which emphasis on general interactions. NCBI generally advices to take eUtils into account. Certain request takes time for computation (for > 100 requests) and situates that there will be maximum of one request for couple of seconds. Big jobs can be easily accomplished for prompting the user to submit certain prerequisite details.

```python
>>> import urllib

>>> Entrez.email = "123@123.com"

Raw-input statement can be used to provide search patterns, offset and onset time and size of record dynamically and "input" statement for identifiers. ‘\n’ or a print statement can be used for formatting appearance patterns.

```python
>>> Term = raw_input("Enter Search Term to find No. of Records: ")

>>> startYear = input("Enter the start Year: ")

>>> endYear = input("Enter the end Year: ")

>>> recordSize = input("No. of Records to be Retrieved: ")
```
Newline:

>>> print "\n Connecting to PubMed database /*/*/*/*/*.

Entrez esearch stems search patterns and repository to connect and inclination to elude onset and offset date, patterns of such are defined beforehand.

>>> handle = Entrez.esearch(db="pubmed", term= Term , mindate= startYear, maxdate=endYear, retmax=recordSize)

The ElementTree wrapper enumerates code to charge XML as basics of Element objects, and re-emancipate them. ElementTree instance can be directly loaded fastly by the aid of parse method.

Scripted interaction with NCBI is supposed to use eutils. Many of the eutils (like ESearch) return only XML, while others like EFetch respond to rettype=fasta or rettype=text. This is the executive summary. Using WebEnv, one can gain more control over the output, but that is another post. NCBI also makes extensive use of another format called ASN.

>>> handle = Entrez.esearch(db="pubmed", term= Term , mindate= startYear, maxdate=endYear, retmax=recordSize)
Entrez efetch affidavit retrieves data from repository depending on search patterns defined.

Provides parser to dissect records redeemed basing on Entrez.efetch method. Medline dissects such patterns.

Predefined PubMed exploration field confession and tags can be labeled for retrieval of output by Entrez.efetch statement.

**PubMed ID [PMID]:**

```python
>>> print "PUBMED ID:" , record.get("PMID")
```

**Title [TI]:**

```python
>>> print "TITLE:" , record.get("TI", "Error (or) No Title")
```

**Authors [AU]:**

```python
>>> print "AUTHORS:" , record.get("AU", "Error (or) No Authors")
```

**Abstract [AB]:**

```python
>>> print "ABSTRACT:" , record.get("AB", "Error (or) No Abstract")
```

**Source [SO]:**
4.6 A Retrieval Performance Study of PubMed, Citeseer, ACM

and IEEE Databases

NCBI PubMed literature database was selected for this study as already mentioned. Initially a generalized search without any limits was employed to retrieve articles related to bioinformatics. An advanced search criteria is more robust and can handle several parsing keywords enabling Boolean operators in PubMed. NCBI eutils package has a very advanced search option where the user can integrate the database into the program.

In addition, IEEE database was also searched for articles that were published on bioinformatics as the key word. However, an advanced search to filter hits based on date and year or author specific is missing. A generalized key word search program was adapted from google in IEEE, where a search is performed on the complete web site and the results are more general, if not, narrow. Hence, the database was not further used for comparison. Instead, ACM database was selected for comparison because ACM database has implemented its own custom search code to retrieve hits.

The fourth database employed in the work was citeseer database, which is an automatic citation indexing system representing a digital library for
scientific and academic papers. A generalized option as well as an advanced search option is available at citeseer. Only text fields and date range criteria were implemented to search metadata.