7.5 Python Packages of Interest

7.5.1 JSON

JavaScript Object Notation (JSON) is an easy to read and write data-interchange format. JSON is used as an alternative to XML and is is easy for machines to parse and generate. JSON is built on two structures - a collection of name-value pairs (e.g. a Python dictionary) and ordered lists of values (e.g.. a Python list).

JSON format is often used for serializing and transmitting structured data over a network connection, for example, transmitting data between a server and web application. Box 7.43 shows an example of a Twitter tweet object encoded as JSON.

Exchange of information encoded as JSON involves encoding and decoding steps. The Python JSON package [88] provides functions for encoding and decoding JSON.

Box 7.44 shows an example of JSON encoding and decoding.

■ Box 7.44: Encoding & Decoding JSON in Python

```
>>>import json

>>>message = {
    "created": "Wed Jun 31 2013",
    "id":"001",
    "text":"This is a test message.",
}

>>>json.dumps(message)
'{"text": "This is a test message.", "id": "001", "created": "Wed Jun 31 2013"}'
```

```
>>>decodedMsg = json.loads( '{"text": "This is a test message.", "id": "001", "created":
"Wed Jun 31 2013"}')

>>>decodedMsg['created']
u'Wed Jun 31 2013'
>>>decodedMsg['text']
u'This is a test message.'
```

7.5.2 XML

XML (Extensible Markup Language) is a data format for structured document interchange. Box 7.45 shows an example of an XML file. In this section you will learn how to parse, read and write XML with Python. The Python *minidom* library provides a minimal implementation of the Document Object Model interface and has an API similar to that in other languages. Box 7.46 shows a Python program for parsing an XML file. Box 7.47 shows a Python program for creating an XML file.

Box 7.41 shows the inverted index reducer program. The key-value pairs emitted by the map phase are shuffled to the reducers and grouped by the key. The reducer reads the key-value pairs grouped by the same key from the standard input (stdin) and creates a list of document-IDs in which the word occurs. The output of reducer contains key value pairs where key is a unique word and value is the list of document-IDs in which the word occurs.

Box 7.42 shows the commands to run the inverted index MapReduce program. First, we copy the directory containing the input to Hadoop filesystem. The input contains an aggregated file in which each line contains a document-ID and the contents of the document separated by a tab. A Hadoop streaming job is then created by specifying the input mapper and reducer programs and the locations of the input and output. When the streaming job completes, the output directory will have a file containing the inverted index.

■ Box 7.40: Inverted Index Mapper in Python

```
#!/usr/bin/env python
import sys

for line in sys.stdin:
    doc_id, content = line.split('')

    words = content.split()
    for word in words:
        print '%s%s' % (word, doc_id)
```

■ Box 7.41: Inverted Index Reducer in Python

```
#!/usr/bin/env python
import sys

current_word = None
```

```
current_docids = []
word = None

for line in sys.stdin:
    # remove leading and trailing whitespace
    line = line.strip()

    # parse the input we got from mapper.py
    word, doc_id = line.split('')

    if current_word == word:
        current_docids.append(doc_id)
    else:
        if current_word:
            print '%s%s' % (current_word, current_docids)
        current_docids = []
        current_docids.append(doc_id)
        current_word = word
```

■ Box 7.42: Running Inverted Index MapReduce on Hadoop Cluster

```
$HADOOP_HOME/bin/hadoop fs -copyFromLocal /documents input

$ HADOOP_HOME/bin/hadoop jar contrib/streaming/hadoop-*streaming*.jar
-mapper mapper.py -reducer reducer.py
-file mapper.py
-file reducer.py
-input input/* -output output
```

■ Box 7.47: Creating an XML file with Python

```
#Python example to create the following XML:
#' <?xml version="1.0" ?> <Class> <Student>
#<Name>Alex</Name> <Major>ECE</Major> </Student > </Class>

from xml.dom.minidom import Document
doc = Document()

# create base element
base = doc.createElement('Class')
doc.appendChild(base)

# create an entry element
entry = doc.createElement('Student')
base.appendChild(entry)

# create an element and append to entry element
name = doc.createElement('Name')
nameContent = doc.createTextNode('Alex')
name.appendChild(nameContent)
entry.appendChild(name)

# create an element and append to entry element
major = doc.createElement('Major')
majorContent = doc.createTextNode('ECE')
major.appendChild(majorContent)
entry.appendChild(major)

fp = open('foo.xml','w')
doc.writexml()
fp.close()
```