Data Wrangling with MongoDB

Lesson4:ProblemSet

Preparing Data:

processing.py

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
In this problem set you work with another type of infobox data, audit it, clean it,
come up with a data model, insert it into a MongoDB and then run some queries against your database.
The set contains data about Arachnid class.
Your task in this exercise is to parse the file, process only the fields that are listed in the
FIELDS dictionary as keys, and return a list of dictionaries of cleaned values.

The following things should be done:
- keys of the dictionary changed according to the mapping in FIELDS dictionary
- trim out redundant description in parenthesis from the 'rdf-schema#label' field, like "(spider)"
- if 'name' is "NULL" or contains non-alphanumeric characters, set it to the same value as 'label'.
- if a value of a field is "NULL", convert it to None
- if there is a value in 'synonym', it should be converted to an array (list)
  by stripping the "{}" characters and splitting the string on "|". Rest of the cleanup is up to you,
  eg removing "*" prefixes etc. If there is a singular synonym, the value should still be formatted
  in a list.
- strip leading and ending whitespace from all fields, if there is any
- the output structure should be as follows:
{ 'label': 'Argiope',
  'uri': 'http://dbpedia.org/resource/Argiope_(spider)',
  'description': 'The genus Argiope includes rather large and spectacular spiders that often ...',
  'name': 'Argiope',
  'synonym': ["One", "Two"],
  'classification': {
                  'family': 'Orb-weaver spider',
                  'class': 'Arachnid',
                  'phylum': 'Arthropod',
                  'order': 'Spider',
                  'kingdom': 'Animal',
                  'genus': None
                  }
}
  * Note that the value associated with the classification key is a dictionary with
    taxonomic labels.
"""
```

```
import codecs
import csv
import json
import pprint
import re

DATAFILE = 'arachnid.csv'
FIELDS ={'rdf-schema#label': 'label',
        'URI': 'uri',
        'rdf-schema#comment': 'description',
        'synonym': 'synonym',
        'name': 'name',
        'family_label': 'family',
        'class_label': 'class',
        'phylum_label': 'phylum',
        'order_label': 'order',
        'kingdom_label': 'kingdom',
        'genus_label': 'genus'}

def process_file(filename, fields):

    process_fields = fields.keys()
    data = []
    with open(filename, "r") as f:
        reader = csv.DictReader(f)
        for i in range(3):
            l = reader.next()

        for line in reader:
            # YOUR CODE HERE
            for key,value in FIELDS.items():
                line[value] = line.pop(key)
                line[value] = line[value].strip(' ')

            # discard unwanted pairs
            for key in line.keys():
                if key not in FIELDS.values():
                    del line[key]

            # strip junk from label and name
            line['label'] = re.sub(r' (.*)', '', line['label'])
            if line['name'] == 'NULL' or re.match(r'\W|\w*\s\W', line['name']):
                line['name'] = line['label']

            # replace 'NULL' with None
            for key,value in line.items():
                if value == 'NULL':
                    line[key] = None
            # turn synonym string into a list
            if line['synonym'] != None:
                if line['synonym'][0] != '{':
                    line['synonym'] = [line['synonym']]
                elif line['synonym'][0] == '{':
                    line['synonym'] = line['synonym'].replace('{', '').replace('}', '').replace('*', '').split('|')
                    line['synonym'] = [syn.strip(' ') for syn in line['synonym']]

            # combine certain pairs into classification
            line['classification'] = {}
            for key in ['family','class','phylum','order','kingdom','genus']:
                line['classification'][key] = line[key]
                del line[key]

            data.append(line)
            pass
    return data

def parse_array(v):
    if (v[0] == "{") and (v[-1] == "}"):
        v = v.lstrip("{")
        v = v.rstrip("}")
        v_array = v.split("|")
        v_array = [i.strip() for i in v_array]
        return v_array
    return [v]

def test():
    data = process_file(DATAFILE, FIELDS)
    print "Your first entry:"
    pprint.pprint(data[0])
    first_entry = {
        "synonym": None,
        "name": "Argiope",
        "classification": {
            "kingdom": "Animal",
            "family": "Orb-weaver spider",
            "order": "Spider",
            "phylum": "Arthropod",
            "genus": None,
            "class": "Arachnid"
        },
        "uri": "http://dbpedia.org/resource/Argiope_(spider)",
        "label": "Argiope",
        "description": "The genus Argiope includes rather large and spectacular spiders that often have a
strikingly coloured abdomen. These spiders are distributed throughout the world. Most countries in
tropical or temperate climates host one or more species that are similar in appearance. The etymology
of the name is from a Greek name meaning silver-faced."
    }

    assert len(data) == 76
    assert data[0] == first_entry
    assert data[17]["name"] == "Ogdenia"
    assert data[48]["label"] == "Hydrachnidiae"
    assert data[14]["synonym"] == ["Cyrene Peckham & Peckham"]

if __name__ == "__main__":
    test()
```

processing.py output:

```
Your first entry:

{'classification': {'class': 'Arachnid',

                    'family': 'Orb-weaver spider',

                    'genus': None,

                    'kingdom': 'Animal',

                    'order': 'Spider',

                    'phylum': 'Arthropod'},

 'description': 'The genus Argiope includes rather large and spectacular spiders that often have
a strikingly coloured abdomen. These spiders are distributed throughout the world. Most count
ries in tropical or temperate climates host one or more species that are similar in appearance. T
he etymology of the name is from a Greek name meaning silver-faced.',

 'label': 'Argiope',

 'name': 'Argiope',

 'synonym': None,

 'uri': 'http://dbpedia.org/resource/Argiope_(spider)'}
```

<u>Inserting into DB:</u>

<u>dbinsert.py</u>

```
import json

def insert_data(data, db):

    # Your code here. Insert the data into a collection 'arachnid'
    db.arachnid.insert(data)

    pass

if __name__ == "__main__":

    from pymongo import MongoClient
    client = MongoClient("mongodb://localhost:27017")
    db = client.examples

    with open('arachnid.json') as f:
        data = json.loads(f.read())
        insert_data(data, db)
        print db.arachnid.find_one()
```

<u>dbinsert.py</u> output:

```
{u'synonym': None, u'description': u'The genus Argiope includes rather large and spectacular sp
iders that often have a strikingly coloured abdomen. These spiders are distributed throughout t
he world. Most countries in tropical or temperate climates host one or more species that are si
milar in appearance. The etymology of the name is from a Greek name meaning silver-faced.', u
'classification': {u'kingdom': u'Animal', u'family': u'Orb-weaver spider', u'order': u'Spider', u'phyl
um': u'Arthropod', u'genus': None, u'class': u'Arachnid'}, u'uri': u'http://dbpedia.org/resource/A
rgiope_(spider)', u'label': u'Argiope', u'_id': ObjectId('5560ef06afbf3024e065be78'), u'name': u'
Argiope'}

Correct! You successfully inserted the data into the 'arachnid' collection!
```

<u>Updating Scema:</u>
<u>update.py</u>

```
#!/usr/bin/env python
# -*- coding: utf-8 -*-
"""
In this problem set you work with another type of infobox data, audit it, clean it,
come up with a data model, insert it into a MongoDB and then run some queries against your database.
The set contains data about Arachnid class.
For this exercise, the arachnid data is already in the database. You have been given the task of
including 'binomialAuthority' information in the records. You will do this by processing the arachnid.csv
to extract binomial authority data and then using this data to update the corresponding data base
records.
The following things should be done in the function add_field:
- process the csv file and extract 2 fields - 'rdf-schema#label' and 'binomialAuthority_label'
- clean up the 'rdf-schema#label' the same way as in the first exercise - removing redundant "(spider)"
suffixes
- return a dictionary with the cleaned 'rdf-schema#label' field values as keys,
  and 'binomialAuthority_label' field values as values
- if 'binomialAuthority_label' is "NULL" for a row in the csv, skip the item

The following should be done in the function update_db:
- query the 'label' field in the database using rdf-schema#label keys from the data dictionary
- update the documents by adding a new item under 'classification' with the key 'binomialAuthority' and
the binomialAuthority_label value from the data dictionary as the value

For item {'Argiope': 'Jill Ward'} in the data dictionary, the resulting document structure
should look like this:

{ 'label': 'Argiope',
  'uri': 'http://dbpedia.org/resource/Argiope_(spider)',
  'description': 'The genus Argiope includes rather large and spectacular spiders that often ...',
  'name': 'Argiope',
  'synonym': ["One", "Two"],
  'classification': {
                'binomialAuthority' : 'Jill Ward'
                'family': 'Orb-weaver spider',
                'class': 'Arachnid',
                'phylum': 'Arthropod',
                'order': 'Spider',
                'kingdom': 'Animal',
                'genus': None
                }
}

Note that the value in the 'binomialAuthority' field is a placeholder; this is only to
```

```
demonstrate the output structure form, for the entries that require updating.
"""

import codecs
import csv
import json
import pprint
import re
DATAFILE = 'arachnid.csv'
FIELDS ={'rdf-schema#label': 'label',
        'binomialAuthority_label': 'binomialAuthority'}

def add_field(filename, fields):

    process_fields = fields.keys()
    data = {}
    with open(filename, "r") as f:
        reader = csv.DictReader(f)
        for i in range(3):
            l = reader.next()
        # YOUR CODE HERE
        for line in reader:

            # swap keys, strip whitespace
            for key,value in FIELDS.items():
                line[value] = line.pop(key)
                line[value] = line[value].strip(' ')

            # discard unwanted pairs
            for key in line.keys():
                if key not in FIELDS.values():
                    del line[key]

            # strip junk from label and name
            line['label'] = re.sub(r' (.*)', '', line['label'])

            # turn binomialAuthority string into a list
            if line['binomialAuthority'] != None:
                if line['binomialAuthority'][0] != '{':
                    line['binomialAuthority'] = [line['binomialAuthority']]
                elif line['binomialAuthority'][0] == '{':
                    line['binomialAuthority'] = line['binomialAuthority'].replace('{', ' ').replace('}', ' ').replace('*', '
').split('|')
                    line['binomialAuthority'] = [syn.strip(' ') for syn in line['binomialAuthority']]

            # no NULLs
            if line['binomialAuthority'][0] != 'NULL':
                data[line['label']] = line['binomialAuthority']
    return data

def update_db(data, db):
    # YOUR CODE HERE
    for key,value in data.items():
        db.arachnid.update({"label" : key},
                        {"$set" : {"classification" : {"binomialAuthority" : value[0]}}})
    pass

def test():
    # Please change only the add_field and update_db functions!
    # Changes done to this function will not be taken into account
    # when doing a Test Run or Submit, they are just for your own reference
    # and as an example for running this code locally!

    data = add_field(DATAFILE, FIELDS)
    from pymongo import MongoClient
    client = MongoClient("mongodb://localhost:27017")
    db = client.examples

    update_db(data, db)

    updated = db.arachnid.find_one({'label': 'Opisthoncana'})
    assert updated['classification']['binomialAuthority'] == 'Embrik Strand'
    pprint.pprint(data)

if __name__ == "__main__":
    test()
```

update.py Output:

```
The update was successful!

Correct! Your program worked correctly!
```