Farzaneh Motahari, uni: fm2475
Ting-Chu Lin, uni: tl2583

****

Files:
python Files:
1.run.py
2.actor_attr_extraction.py
3.author_attr_extraction.py
4.businessperson_attr_extraction.py
5.league_attr_extraction.py
6.person_attr_extraction.py
7.sportsTeam_attr_extraction.py
8.output.py
9.questions.py

Extra Files:
We have two query files: one for questions, which include all test cases asked by the TA as well as some extra questions for more precise checking of our code against yours. The same applies to info box checking, there are 20 queries, the first queries are the test cases and the rest are some extra checking. our results are saved in files called test_infobox.result and test_question.result.

README
MAKEFILE

****

Usage:
1. python run.py -key <Freebase API key> -q <query> -t <infobox | question>
2. python run.py -key <Freebase API key> -f <file of queries> -t <infobox | question>

where:
* <Freebase API key > is the Google account key for Freebase.
* -q <query> is the query. For infobox, it is a list of words in double quotes (e.x., “Bill Gates”). For question, it is a string that matches this patten “Who created [X]?”, where [X] is the target item. “W” and “w” is both accepted to our implementation, and “?” can be skipped.
* -f <file of queries> is a text file with one query per line.
* -t <infobox | question> indicates the query type.

****

Person Attributes		
Freebase Property	Property of interest	Comments

Person Attributes		
Name	/type/object/name	The first level '/type/object/name' propert includes the person's name. Since everyone has one full name, the value is a list of one object and we used value for key 'text'
Birthday	/people/person/ date_of_birth	Since everyone has one birthdate, we used the first entry from the array of values which in fact includes only one entry. from there used the value for key 'text'
Place of Birth	/people/person/ place_of_birth	Since everyone has one place of birth, we used the first entry from the array of values which in fact includes only one entry. from there used the value for key 'text'
Death(Place,Date,Cause)	/people/ deceased_person/ date_of_death','/people/ deceased_person/ place_of_death','/ people/ deceased_person/ cause_of_death'	if the person is dead and because everyone has only one deceased date as well as place , we used the first entry from the array of values which in fact includes only one entry. from there used the value for key 'text'. A person might have multiple causes of death so we traverse the array of cause_of_death to retrieve those causes and we combined all these attributes in a format like : <2009-06-25> <at Holmby Hills> <, cause: (Cardiac arrest, Homicide)> so that is place of death is missing then it will just show 'date , cause: ..'
Siblings	/people/person/ sibling_s, /people/ sibling_relationship/ sibling	For siblings if the person has the first proper ie '/ people/person/sibling_s' then we will retrieve all values and for each value we look for the second property which is '/people/sibling_relationship/ sibling' and from there we extract the 'text'
Spouses	/people/person/ spouse_s, '/people/marriage/ spouse'	Since a person might have multiple marriages, we traverse the array of values for the given property and then for each value entry we look for '/people/ marriage/spouse' property and for that we used the first value entry and extract the text
Description	/common/topic/ description	We use the first order property and from the first entry in values array we extract the 'value' because it includes the comprehensive description

Author Attributes		
Freebase Property	property of interest	comments
Books	/book/author/ works_written'	For this property, we traversed the arrays of values and retrieve the ‘text’ to get the books by author
Books About	/book/book_subject/ works'	For this property, we traversed the arrays of values and retrieve the ‘text’ to get the books about author
Influenced	/influence/ influence_node/ influenced'	For this property, we traversed the arrays of values and retrieve the ‘text’ to get the who the author influenced
Influenced By	/influence/ influence_node/ influenced_by'	For this property, we traversed the arrays of values and retrieve the ‘text’ to get the who the author was influenced by

Actor Attributes		
Freebase propert	property of interest	comments
Film Name	/film/performance/ character	for any film that the actor has played in we extract film name as well as character in the film from the array of values for this property
Character	/film/performance/ character	using the array of values from above we extract for each film, the character of the actor in that film

Business Person Attributes		
Freebase Property	property of interest	comments
Founded	/organization/ organization_founder/ organizations_founded	we extract ‘text’ from array of values for this property

Business Person Attributes		
BoardMember(From, To, Organization, Role, Title),	‘/business/ board_member/ organization_board_memberships’,’/ organization/ organization_board_membership/organization’,’/ organization/ organization_board_membership/title’,’/ organization/ organization_board_membership/from’,’/ organization/ organization_board_membership/to’,’/ organization/ organization_board_membership/role'	using the first property we first check wether the business person is a board member of a company or not, then we extract the properties of interest iteratively for each value entry for this property. To get the properties of interest we extract all the properties in each value entry of ‘/business/ board_member/organization_board_memberships’ and store it in an object called company. Then for the inner layer properties of title, role, … we check whether if they exist in company object, if they do then we extracted the ‘text from the first value entry within them, otherwise we keep them as empty strings to be able to fill in our infobox. So doing so we are able to save all the organizations the person is a boradmember of in a list and each company is of the form {'Organization':'', 'Title':'', 'Role':'', ‘From-To':''} in that list of coardmembership
Leadership((From, To, Organization, Role, Title)	’/organization/ leadership/ organization’,’/ organization/leadership/ title’,’/organization/ leadership/role’,’/ organization/leadership/ from’,’/organization/ organization_board_membership/to'	using the first property we first check wether the business person is a board member of a company or not, then we extract the properties of interest iteratively for each value entry for this property. To get the properties of interest we extract all the properties in each value entry of ‘/business/ board_member/organization_board_memberships’ and store it in an object called leader_of. Then for the inner layer properties of title, role, … we check whether if they exist in leader_of object, if they do then we extracted the ‘text from the first value entry within them, otherwise we keep them as empty strings to be able to fill in our infobox. So doing so we are able to save all the organizations the person is a leader of in a list and each company is of the form {'Organization':'', 'Title':'', 'Role':'', ‘From-To':''} in that list of leadership

League Attributes		
Freebase Properties	properties of interest	comments
Name	/type/object/name	From the first order /type/object/name property we extract the name of the league from ‘text’
Championship	/sports/sports_league/ championship	From the value for this property we extract the ‘text’ to get championship for this league

League Attributes		
Sport	/sports/sports_league/sport	to get the sport of the league we used this property and for its value array we used the first entry and got the text for that
Slogan	/organization/organization/slogan	to get the slogan of the league we used this property and for its value array we used the first entry and got the text for that
OfficialWebsite	/common/topic/official_website	to get the OfficialWebsite of the league we used this property and for its value array we used the first entry and got the text for that
Description	/common/topic/description	to get the Description of the league we used this property and for its value array we used the first entry and used 'value' to get the full description
Teams	/sports/sports_league/teams	we iterated through the value array to get teams and saved them in a list

Sports Team Attributes		
Freebase Properties	property of interest	comments
Name	/type/object/name	from the outer property '/type/object/name' we extract the name from the first value entry and from 'text'
Sport	/sports/sports_team/sport	for this property since each team is playing one sport, from the first value entry we extracted 'text'
Arena	/sports/sports_team/arena_stadium	for this property since each team has one Arena, from the first value entry we extracted 'text'
Championships	/sports/sports_team/championships	we iterated the value array of this property and for each value we extracted 'text'
Founded	/sports/sports_team/founded	since each team has one founded date, we extracted the 'text' for the first value entry
Leagues	/sports/sports_team/league,'/sports/sports_league_participation/league'	since each team is playing in one league at a time we extracted the first value entry we extracted the second property and for that property's value we got 'text'
Locations	/sports/sports_team/location	from the value array we extracted the 'text'

Sports Team Attributes		
Coaches(name, title, from/to)	/sports/sports_team/coaches','/sports/sports_team_coach_tenure/coach','/sports/sports_team_coach_tenure/position',/sports/sports_team_coach_tenure/from','/sports/sports_team_coach_tenure/to	Coaches in a list of coaches with each coach represented in a dictionary, with keys Name, Position, From/To. so for each coach we iterated through the array values of the first property and we used the other properties to retrive name, title, from/to
PlayerRosters	/sports/sports_team/roster', '/sports/sports_team_roster/player','/sports/sports_team_roster/position','/sports/sports_team_roster/number','/sports/sports_team_roster/from','/sports/sports_team_roster/to'	PlayersRoster is a list, each player represented in a dictionary, position in each dictionary is a list of positions. So for the array of values for the first property, we traverse that array and extract name, positions, number, from/to. Position might be multiple so for that property we also traverse the array of values to get each position. For all of them we used 'text'
Description	/common/topic/description	We used the first value entry and used 'value' to get the thorough description

Part 2 Properties	
/organization/organization_founder	/organization/organization_founder/organizations_founded
/book/author	/book/author/works_written

Description of the code:
In this package all the queries submitted to APIs are url-escaped by using url lib.quote(query)
In run.py, in main method, we first get the input from terminal and check whether the format of the command matches the two given formats or not, and if not it will exit and will give 'wrong input format.
Otherwise if it's file then we extract the queries and save them in a list of queries, if a single query then we don't do anything.
Then we check wether the query is of type info box or question. in either case the related method in the same file is called. If the query type is neither question nor info box it will give query type error and will exit.

In question method we first check to see whether the question if of the correct format ie, who create … if not warning is given and will exit. it will discard all white spaces and handles all the errors.
In case everything is fine, we extract all the answers related to the question from /book/author/works_written and /organization/organization_founder/organizations_founded. Response from API is saved and then submitted to questions.py file to extract the desired results, i.e. name of author and the books with the query in the title or name of business person with the companies he has which have the query in their company name.
The results for both author and company are combined and returned in terminal in alphabetical order.

MQL queries:

```
[{
  "/book/author/works_written": [{
    "a:name": null,
    "name~=": "Mona Lisa"
  }],
  "id": null,
  "name": null,
  "type": "/book/author"
}]
```

```
[{
  " /organization/organization_founder/organizations_founded": [{
    "a:name": null,
    "name~=": "Mona Lisa"
  }],
  "id": null,
  "name": null,
  "type": " /organization/organization_founded"
}]
```

In infobox, we first get the json results of the first API for the query which gives us the mid values. if there is nothing returned then appropriate message is shown to the user, otherwise, we extract mid values and from the highest score to the last one we check wether any of the 6 entity types of interest are in the API for each mid value, the first mid which has any one of the entities is the one we use for information extraction for the query.
If none of the mid values has any of the 6 entities, then we give appropriate message. To check the existence of any of these 6 entity types we use the method type_match for the josh result of each mid value.
For each entity matched, we then use the associated file i.e. for entity person we use person_attr_extraction.py to extract the requested properties, ie name, date of birth and etc.
We run associated file for each existing entity in our entity list.
we group the results in two categories, person/actor/author/businessPerson or league/sporteTeam. More info is available below in additional information.

We then had our own drawing of table in output.py. Each result type ie, Name, Spouses or Films each need their own way of printing. ‘Name', 'Birthday', 'Place_of_Birth', 'Death', ‘Description'

are considered as paragraphs, they can take one or more continuous line. 'Spouses', 'Siblings', 'Books', 'Books_About', 'Influenced','Influenced_By', ‘Founded' take multiple lines per entry, and finally 'Films', 'Board_Member', ‘Leadership' are printed in tables. Each of these printing modes have their own specific function with some mathematical computations on when to replace the rest of the string with ‘…’ and so on and so forth.

****

Freebase API KEY: AIzaSyA5c1nxOEDPual-jfeLVt-i571ngRFost4
requests per second per user: 100,000 requests/day and 20.0 requests/second/user

****

Additional Information:

1. For the case that if a query is matched as both a person and a sport team/league, we will consider it as a sport team/league, because the match to a sport team/league is more likely to be unique. For example, we prefer to display “NFL” as a sport league rather than as a person whose name related to the string “NFL”.
2. We assume that if a query is matched as a league, it should not be a sport team, but if this case dose happen, we display it as a league.
3. In the second part we might have the same author with different id types, like when you query ‘Who Created Ebay’, then you would have Michael Miller in the returned json file, as two objects with different ids:


```
{
"type": "/book/author",
"name": "Michael Miller",
"/book/author/works_written": [
{
"a:name": "Easy eBay"
},
{
"a:name": "Tricks of the eBay(R) Business Masters"
},
{
"a:name": "Guide to Ebay"
},
{
"a:name": "eBay Auction Templates Starter Kit"
}
],
"id": "/authority/us/gov/loc/na/n86056832"
},
```


and:


```
{
"type": "/book/author",
"name": "Michael Miller",
```

"/book/author/works_written": [
{
"a:name": "Tricks of the eBay business masters"
},
{
"a:name": "Making a living from your eBay business"
}
],
"id": "/user/hangy/viaf/18369810"
},

So we will have two lines of result in our output shown to the user. We might have different authors with the same name.

4. In the reference implementation, spouse has date of birth and place of marriage, however I asked Fotis and he said we don't need to show those information as it is not asked in the definition of the project.