Read Me for Index Project

Name Search:

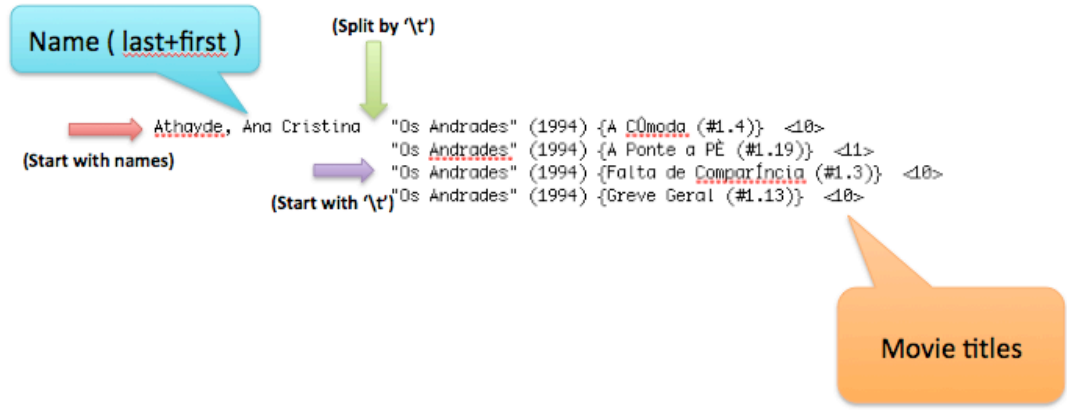
Index Structure:

I choose **associative array** (dictionary) in Python to be my structure of this index project, because the associative array has key and value function, once I got the key value and I can find the value related to this particular key.

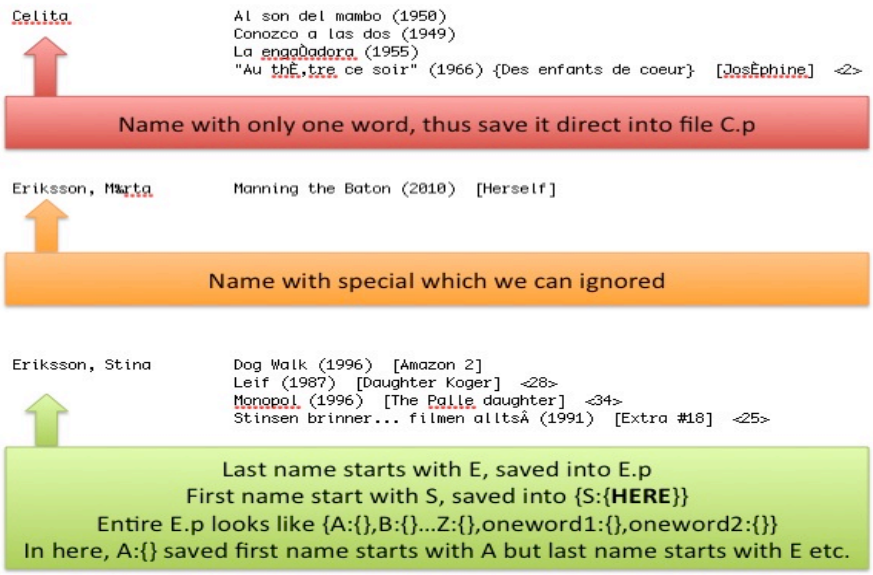
In this project, we have actors and actress list in two separate files; my job is to integrate these two files into one index structure by using **dictionary** in Python. My design is to separate this gigantic file by the first letter from A to Z for both first name and last name, therefore, there are two dictionaries actually, it is dictionary inside dictionary, one is categorized by first name, the other one is categorized by last name. Because the entire actor and actress text file were created in a solid method, hence there are few rule that we can create to use for all of the each data in these gigantic files. For example, first name and last name are divided by comma etc.

Index creation step by step:

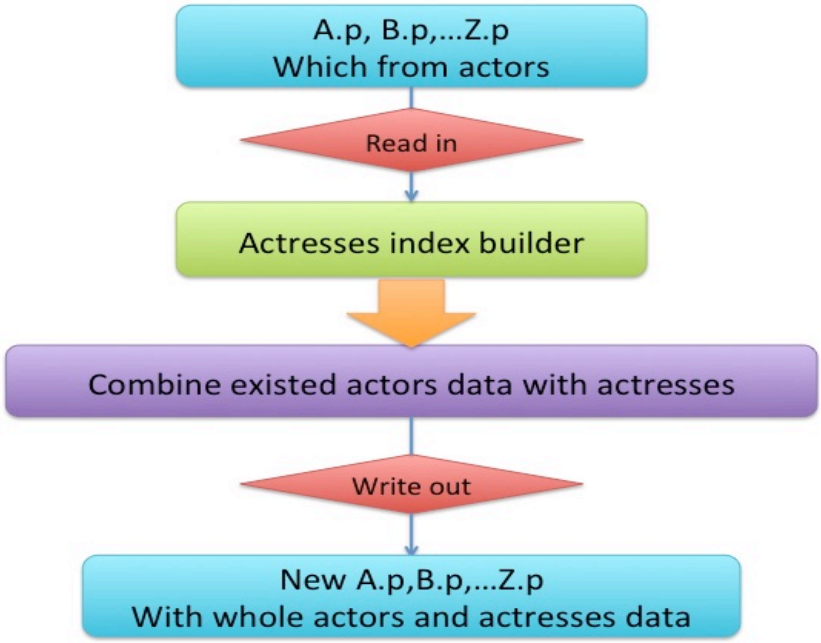
1. To read entire line by line
2. Analysis each line, if it starts with is '\t' and it is not empty line ('\n'), it means this line only contains the movies, not having actor or actress. Otherwise, it does not start with '\t', it means this line contains name of actors or actress. Thus, I save the name by split the line with '\t', the front part is name and the back part is the movie name.



3. Now we got the name of the actors and actresses, first of all, use filter class I create to eliminate some name that contain some special character that we did not want to search such as German or French characters. Secondly, separate first name and last name by splitting with comma, because last name is placed before first name, I categorized first by last name, if its first name starts with A then it is save in the 'A.p' file etc., ".p" is the pickle file created by Python. Thirdly, if the last name all start with same letter, I will categorize one more time by separation the first letter of first name.



4. Because the actors and actresses file already sorted the name of actors and actresses by letter, thus, all I have to do is to detect the first letter of last name change or not, if it changed, it means the all actors or actresses whose name start with this letter are all saved into dictionary, now I just need to save this whole dictionary into file for usage later.
5. Executed program, file from A.p to Z.p will generated after program terminated, then to executed another similar program for actresses. There are some differences between integration of actors index and actresses index, when I want to generated actress index, the actors index is already existed, thus, all I need to do is to read from A.p to Z.p in by each file, and combine actresses dictionary with actors dictionary and then resave from A.p to Z.p again. After that, the entire actors and actresses index were categorized well by dictionary structure in Python.



Search:

To build function of name search, I have a gigantic index categorized by letter already, thus, all I need to do is read this gigantic index into my search program, combine them together into one dictionary for searching by alphabets, such as {A:{data saved in A.p}, B:{data saved in B.p}……Z:{data saved in Z.p}}. Second step, to read query file in and then input one query name once a time until all queries are searched.

How to search? Because I have first name and last name categorized already, hence all I need to do is detect this query name has one word or two word, if it has two word then it must be first name and last name, to look the first letter of these two word, this is the key value, then I can find value related to this key, that is what I want to print out.

Implementation:

Building Index:

1. To place the **actors.list** and **actresses.list** with **actorIndex.py** and **actressIndex.py** in a same directory
2. To execute **actorsIndex.py** and after all A to Z pickle got generated, executing **actressIndex.py**

Search:

1. To place the index files which is A to Z pickle files and query file into same directory with search file. Executing **nameSearch.py** it will load all pickle in then we can type our query files.
2. To save query name line by line, last name first and then first name, separate by comma and there is one space right after comma.
3. To execute program after all A to Z pickle are read, then we can enter the query file, to quit the program, please type "**exit**".

Movie Search:

Index Structure:

I tried to used **prefix tree** in this search method, but I did not quite create this index with prefix tree successful, thus I give up this structure and just try to read entire file in an then search the keyword line by line, if the line contains the keyword, then print it. It means there is no particular data structure in this search part.

How to build index? Once the program executed, if the line is not empty ("\n"), read line by line and stored each line as an element into a list. I have to store all file because there is no anything I can ignore.

Search:

To execute searching function, pop out each line stored in the list before, compare whether the query keyword contain in this line or not, if yes, print the line, if not, skip this line. I used partial match, it means the output will contain exactly match and partial match, for example, if query keyword is **king**, then **kingdom** will appear in the output result, too.

Implementation:

1. To place query file, **movie.py** and **movies.list** into same directory.
2. After executing program, it will build index first by loading the entire movie list, and then execute search by entering query file. We can enter the query file now, to shut down the program by typing "**exit**".