**Assignment 4 – Report**

**Step1:**

We have performed data cleansing for the **species.txt** file in the first mapper file. Following cleansing expressions were applied in the text file:

1. Separated the web page and the its outlinks as key and value.
2. Cleaned the unwanted data in the outlinks using and considering the value before the comma.





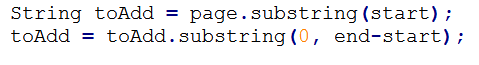
1. Removed all the species name in the foreign languages and taken English names into consideration.



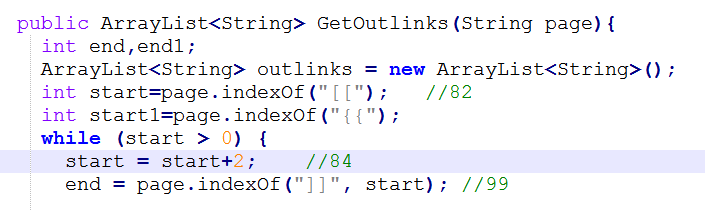
1. Extracted the species name in both the “{{ }}” and “[[ ]]” brackets using indexes.



1. Using substring and finding index, we will extract the data inside the brackets.



1. Calculate the outlinks using the below function.



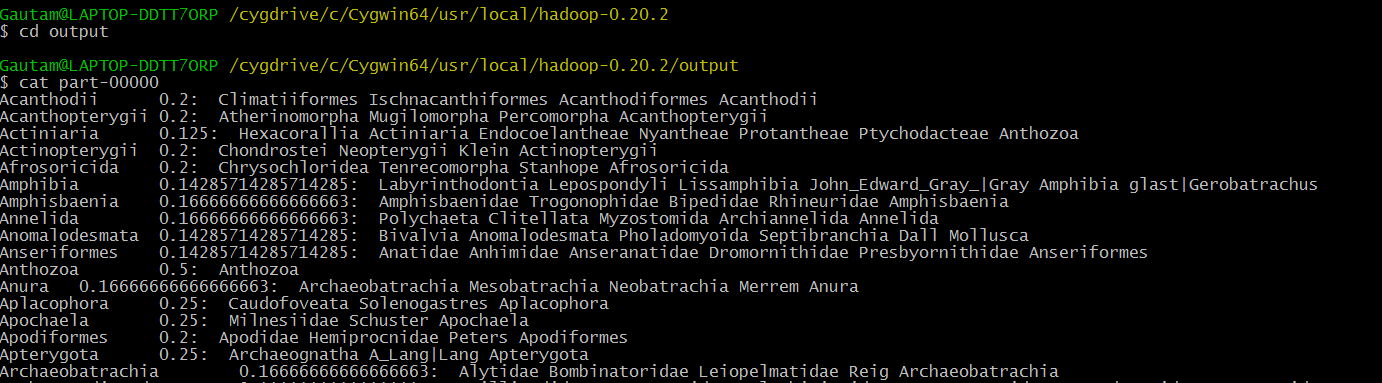
**Step2:**

In the SpeciesGraphBuilderReducer class, counted the page rank of the key(title) and sent it to the next mapper, where count is the number of web pages and each’s web page rank is assumed as 1/no. of web pages.



After running SpeciesGraphBuilder class we will get the output that contains:

* Key(Title)
* Page Rank of Title
* Outlinks from the page.



**Step3:**

* We will split the Title, Page Rank and the number of outlinks in the second mapper.
* In the next step, we will put the outlinks in the key and the key(title) in the value.
* We have taken dampening factor as 0.85.



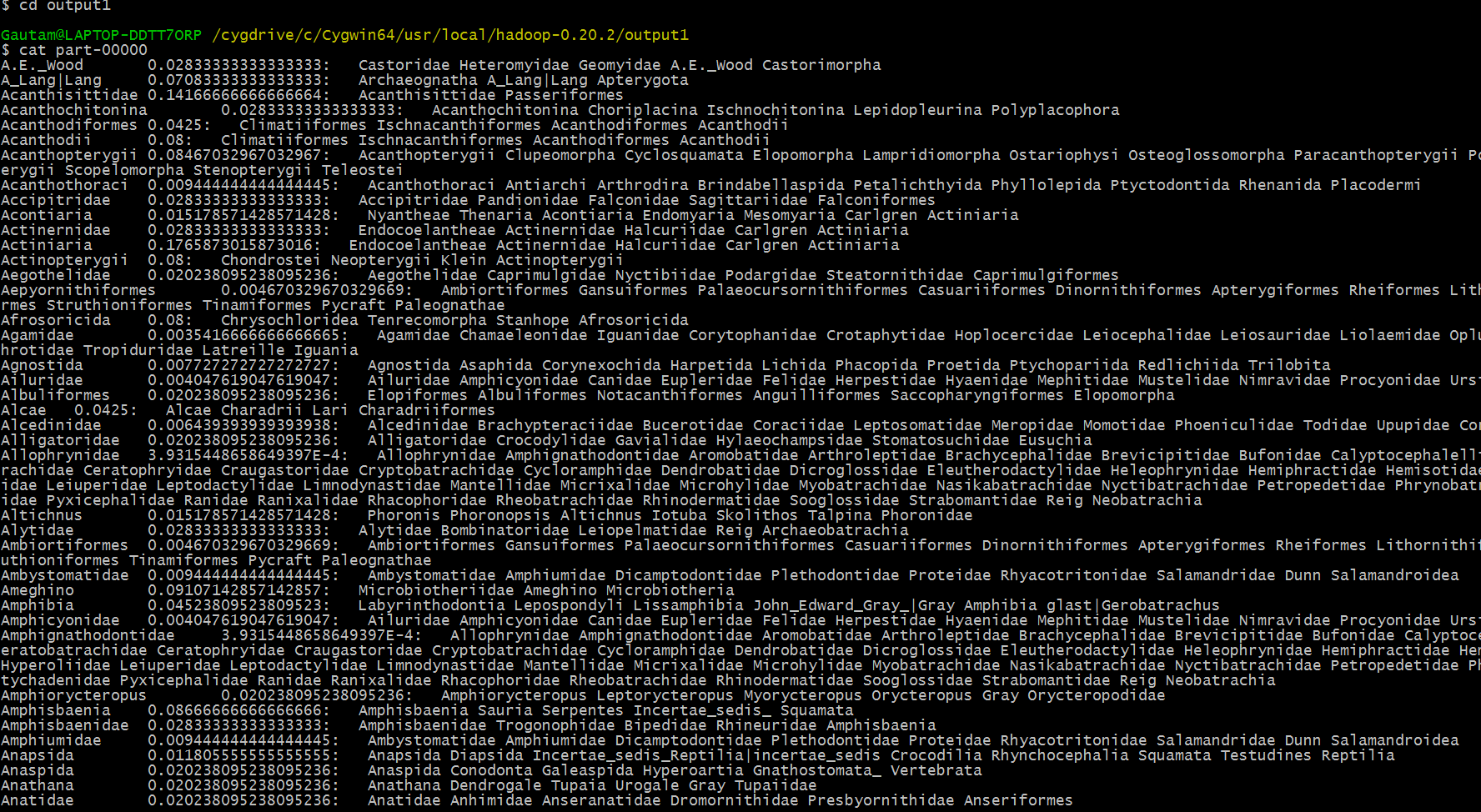
Where currscore is the page rank of the inlinks and numoutlinks is the no of outlinks from the inlink.

* To handle the case when user directly accesses the URL or web page we have taken the 0.15 probability.



Suppose a key is appeared more than once then in the reducer, aggregation will take place where all the corresponding inlinks page ranks will be added according to the formula.

* We will send the above key value pair to the reducer
* Run the Driver file for the above and we will get the below output.
* Key-Web page(Inlinks from the previous mapper)
* Page Rank of the web page.
* Outlinks of the web page.



**Step4:**

In the next step above data will be sent to the Species Viewer Mapper class. Here we will perform following operations:

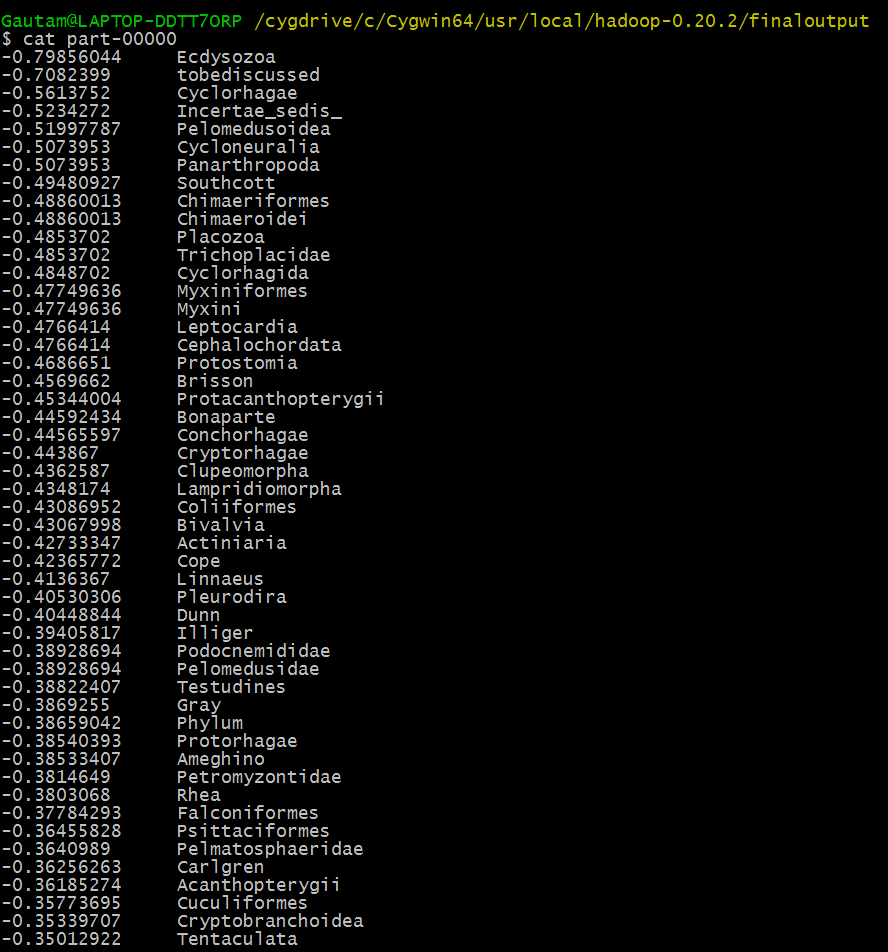
1) Get the page and split it into title and page rank.



2) Collect the page rank and send it to the next Viewer driver class with page rank as key and species name as value.

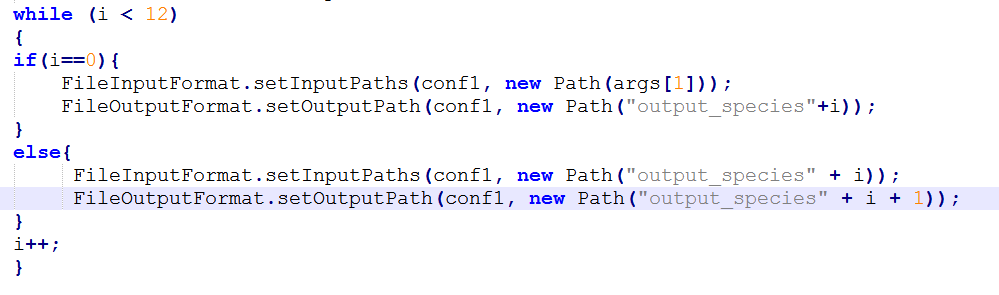


3) Run the Viewer Driver class to get the finalize page rank of the species web links in the descending order.



**Converging with Iterations:**

To Achieve a Converge page rank, We have iterated the jobs multiple number of times(12) times. After that we will get a constant page rank for a web page. Following snapshots will describe on how to perform the iteration:



* If value of i is not 0 , then the output of the iterator will be the input for the SpecieViewer Mapper class.
* Else if we are left with iterations to complete, each iterator’s output will be fed to the iterator mapper class only as described in the above figure.

After 1st iteration, value of below was observed:



After 11, it was



Finally, after 12 iterations we got the same value as the 1st one i.e. the converge value of page rank was generated.

