# Report of third programming problems

Introduction to Big Data Management

Menoli Riccardo, riccardo.menoli@helsinki.fi

I used the python streaming in order to complete the exercise. I followed this guide <https://www.glennklockwood.com/data-intensive/hadoop/streaming.html>, suggested by the Professor Jiaheng Lu.

## Contents

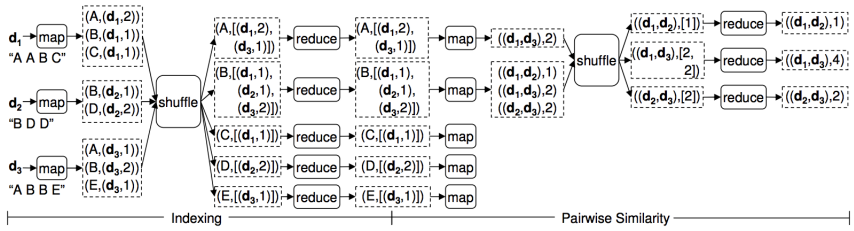



Figure 1:

# 1 Assigment

Implement Hadoop MapReduce programs to perform the similarity string join with n-gram Jaccard similarity. Download datasets: Gram2018.zip, where you will find two Wikipedia samples. Each of them contains 10k lines of Wikipedia categories like this. These datasets are now inconsistent due to misspellings. Can you find similar pairs of records, one from each dataset, which have Jaccard similarity no less than 0.15?

1. Fill the following table:

| Gramsize | Result size | Single machine | Multiple machines |
|---|---|---|---|
| 2 (bi-gram) | | | |
| 3 (tri-gram) | | | |

2. Describe your algorithm for this problem, and upload your source codes and analyze the performance of your codes.

# 2 Result

I would like to talk first about the second request: *Describe your algorithm for this problem, and upload your source codes and analyze the performance of your codes.*

## 2.1 Describe your algorithm

My code follows the idea proposed by Elsayed, Lin and Oard 2008 [2]. The main idea of the paper is described in Figure 1. The computation is performed in two MapReduce steps:

1. **Indexing**: build a standard inverted index, where each n-gram is associated with a list of sentences that contain it and the number of occurrences of the n-gram.

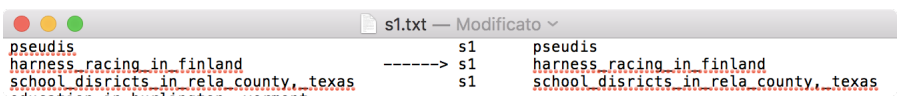
2. **Pairwise Similarity:** create pairs of sentences that share the same n-grams, find how many n-grams each pair shares and compute the similarity.

Elsayed, Lin and Oard 2008 assumes that all sentences are in the same file. They consider th following similarity measure: $sim(d_i, d_j) = \sum_{t \in V} w_{t,i} \cdot w_{t,j}$ where $sim(d_i, d_j)$ is the similarity between sentences $d_i$ and $d_j$, $V$ is the vocabulary set and $w_t$ is the weight which indicate the importance of each term $t$. If a term is not in the sentences the weight of the term for that document is 0.

In order to complete the task of the assignment (i.e. finding the similar pairs between the two documents and use as similarity distance the Jaccard similarity), I change a little the idea of Elsayed, Lin and Oard 2008. My solution performs two MapReduce job (indexing and pairwise similarity) plus a *preprocessing* step.

### 2.1.1 Preprocessing

*s1.txt* and *s2.txt* contain a sentence per row. In order to perform my solution, for each row they should have the following structure: the name of the file, tab, sentence.




To perform this task you have to use the script *name.py*.

The preprocessing give the possibility to write only one map function that is able to distinguish if a sentence belongs to file *s1.txt* or to *s2.txt*.

### 2.1.2 Indexing

The indixing part is composed by a map function written in *inv-index-mapper.py* and a reduce function written in *inv-index-reducer.py*.

**Map function:** the map function takes as input the preprocessed file. For every sentence met, it divide it in n-grams, create for each n-gram a key-value pair where the key is a tuple *(n-gram; name of the file)* and the value is the sentence.

```
aa;s1   leinster_gaa_club_stubs 1
aa;s2   companies_based_in_shaanxi      1
ab;s1   1490s_establishments_in_spain   1
ab;s1   1780s_establishments_in_hungary 1
ab;s1   1789_establishments_in_korea    1
ab;s1   1815_establishments_in_portugal 1
ab;s1   1824_establishments_in_tghe_kingdom_of_sardinia 1
ab;s1   1844_disestablishments_in_noth_amercia  1
ab;s1   1855_disestablishments_in_england       1
ab;s1   1860s_establishments_in_germany 1
ab;s1   1868_establishments_in_chile    1
ab;s1   1871_establishments_in_uruguay  1
ab;s1   1890s_disestablishments_in_spain        1
ab;s1   1898_establishments_in_switzerland      1
```

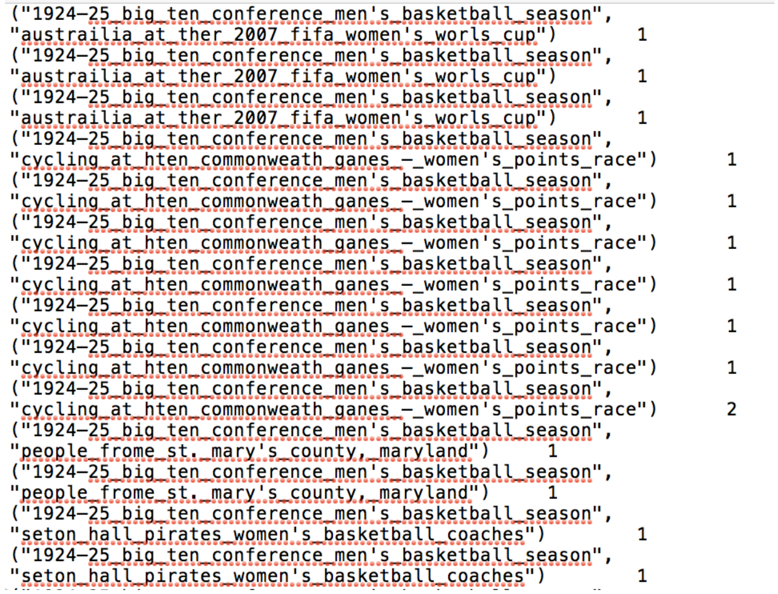
**Reduce function:** the reduce function per each key puts together all the sentences found in a python dictionary style. The output is the following:


```
aa;s1   {"leinster_gaa_club_stubs": 1}
aa;s2   {"companies_based_in_shaanxi": 1}
ab;s1   {"1490s_establishments_in_spain": 1,"1780s_establishments_in_hungary": 1,"1789_establishments_in_korea":
1,"1815_establishments_in_portugal": 1,"1824_establishments_in_tghe_kingdom_of_sardinia":
1,"1844_disestablishments_in_noth_amercia": 1,"1855_disestablishments_in_england":
1,"1860s_establishments_in_germany": 1,"1868_establishments_in_chile": 1,"1871_establishments_in_uruguay":
1,"1890s_disestablishments_in_spain": 1,"1898_establishments_in_switzerland": 1,"1901_establishments_in_sweden":
1,"1906_establishments_in_austria-hungary": 1,"1908_establishments_in_bohemia":
1,"1910s_establishments_in_cuba": 1,"1916_establishments_in_new_brunswick": 1,"1940s_establishments_in_beligum":
1,"1952_establishments_in_kenya": 1,"1965_establishments_in_indonesia": 1,"1970s_establishments_in_armenia":
1,"1997_establishments_in_new_jersey": 1,"1998_disestablishments_in_estonia":
1,"1999_establishments_in_greenland": 1,"2005_establishments_in_serbia": 1,"2012_establishments_in_nepal":
1,"21st-century_disestablishments_in_hten_palistinian_terriories": 1,"abell_richness_clas_4":
1,"consituencies_established_in_1801": 1,"establishments_in_sourth_dakota_by_eyar":
1,"history_books_abbout_ther_latter_day_saint_movment": 1,"labor_disputes_in_japan": 1,"libraries_in_alabama":
1,"local_govenrment_in_saudi_arabia": 1,"magazines_disestablished_in_1876":
1,"museums_in_cabell_county,_west_virginia": 1,"philippines-united_arab_emirates_realtions":
1,"populated_places_established_in_hten_9th_centuty_bc": 1,"publishing_companies_established_in_teh_1710s":
1,"reaccurring_sporting_events_established_in_1996": 1,"schools_in_labette_county,_kansas":
1,"sports_venues_in_ashgabat": 1,"states_and_terriories_established_in_1557":
1,"sustainable_building_by_ocuntry": 1,"technnology_companies_established_in_1919":
1,"wikipedia_requested_photographs_in_greene_county,_alabama": 1,"zimbabwean_emigrants_to_ther_united_states":
1}
ab;s2   {"1572_establishments": 1,"1690s_establishments_in_europe": 1,"1760_disestablishments_in_france":
1,"1776_disestablishments_in_south_america": 1,"1793_establishments_in_ireland":
1,"1814_establishments_in_the_united_states": 1,"1847_establishments_in_england":
1,"1868_establishments_in_chile": 1,"1877_establishments": 1,"1881_establishments_in_japan":
1,"1882_establishments_in_the_philippines": 1,"1906_establishments_in_austria-hungary":
1,"1909_establishments_in_asia": 1,"1945_establishments_in_cyprus": 1,"1946_establishments_in_peru":
1,"1958_establishments_in_chad": 1,"1960s_disestablishments_in_mexico": 1,"1986_establishments_in_india":
```


### 2.1.3 Pairwise similarity

The Pairwise similarity part is composed by a map function written in *pair-mapper.py* and a reduce function written in *pair-reducer.py.*

**Map function:** the map function put together the sentences in *s1.txt* and *s2.txt* that have at least one n-gram in common. The key will be a pair of sentences that belong to two different files and that have at least one n-gram in common, the value will be how many time that specific n-gram have the sentence in common. So, it has to make the cross product of all the sentence that appears in the value of *(n-gram;s1)* and *(n-gram,s2)* where n-gram is the same. The value is computed as the minimum between the values of the previous reduce step. The output is the following:



```
("1924–25_big_ten_conference_men's_basketball_season",
"austrailia_at_ther_2007_fifa_women's_worls_cup")       1
("1924–25_big_ten_conference_men's_basketball_season",
"austrailia_at_ther_2007_fifa_women's_worls_cup")       1
("1924–25_big_ten_conference_men's_basketball_season",
"austrailia_at_ther_2007_fifa_women's_worls_cup")       1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      1
("1924–25_big_ten_conference_men's_basketball_season",
"cycling_at_hten_commonweath_ganes_–_women's_points_race")      2
("1924–25_big_ten_conference_men's_basketball_season",
"people_frome_st._mary's_county,_maryland")     1
("1924–25_big_ten_conference_men's_basketball_season",
"people_frome_st._mary's_county,_maryland")     1
("1924–25_big_ten_conference_men's_basketball_season",
"seton_hall_pirates_women's_basketball_coaches")        1
("1924–25_big_ten_conference_men's_basketball_season",
"seton_hall_pirates_women's_basketball_coaches")        1
```

It is the bottleneck of the entire computational flow. Infect, due to the fact each sentence contains a lot of n-grams, the size of that file dramatically increases with the number (and the length) of the words in the two file. Whit the documents *s1.txt* and *s2.txt* this file is more than 20 GB. Clearly create this file in a VM is impossible (I believe that in a cluster it would be possible). For this reason, I decided to take the first 714 sentences of *s1.txt* and *s2.txt.* I try to modify a little bit the idea of Elsayed, Lin and Oard 2008 [2] by writing only the number of lines and the name of the file for each sentence. However, the resulting size still remains more than 6 GB. I would point out that with this algorithm we write ***only*** the pairs of sentences that have at least one n-grams in common. If a pair of sentences does not have n-grams in common it will be not written. So all the things written in this file are necessary to compute the exact Jaccard similarity between the sentences. I tried to cut in some way the size of this file by doing approximation based on the result of the output but the result were not good. In the end, I would underline that Elsayed, Lin and Oard 2008 [2] declare that *"Empirically, we find that running time increases linearly with collection size, which is an extremely desirable property".* I don't know if functions *job1.setOutputFormatClass(SequenceFileOutputFormat.class)* and j*ob2.setInputFormatClass(SequenceFileInputFormat.class)* suggested in the hint of the question sheet would solve this size problem. I didn't found their correspondence in python.

**Reduce function:** the reduce function of the pairwise step computes the Jaccard distance, according to Wikipedia it is:

$$J_\delta(A,B) = 1 - J(A,B).$$

Where $J(A,B) = |A \cap B|/|A \cup B|$ is the Jaccard index. I followed the idea of

Bank and Cole 2008 [1] to compute the Jaccard index:

*To compute the Jaccard similarity coefficient for a pair of X elements, calculation of two quantities is needed. The first is the number of Y elements that occur with both X elements, and the second is the number of Y elements that occur with one or both. Calculating the first quantity was relatively straightforward, but for the second quantity, one must first calculate the number of Y elements that occur with the first X element plus the number of Y elements that occur with the second X element and then subtract the number of Y elements that occur with both. Finally, the first quantity is divided by the second quantity to arrive at the proper coefficient.*

To compute the first quantity the reducer should simply sum the values that shares the same key. To compute the second quantity the reducer should do: $l_1+l_2-2-fistquantity$ for bi-grams and $l_1+l_2-4-fistquantity$ for tri-grams. Where $l_1$ and $l_2$ are the length of the first and second sentences of the key. This calculation is provided by the fact that in a word of length $n$ there are $n-1$ bi-grams and $n-2$ tri-grams. To summarize the reducer perform the following operation:

$$J_\delta(A,B) = 1 - \frac{sum(values\ with\ same\ key)}{l_1 + l_2 - n - sum(values\ with\ same\ key)}.$$

Where $n$ is equal to 2 if we use bi-grams, $n$is equal to 4 if we use tri-grams.

## 2.2 Result table

The results are riported in the table above. I would underline that it is the results reported are related only to the first 714 sentences of each file. The reson of this limitation are described in the subsession 2.1.3.

| Gramsize | Result size | Single machine | Multiple machines |
|---|---|---|---|
| 2 (bi-gram) | 12 | 1m3.493s | 1m1.583s |
| 3 (tri-gram) | 36 | 15.861s | 15.746s |

In particular the results for the bigrams are:

```
('19th-century_venezuelan_people', '19th-century_venezuelan_peaple')    0.12903225806451613
('churches_in_kenton_county,_kentucky', 'churchs_in_kenton_county,_kentucky')   0.08571428571428574
('emporia_state_university_navigational_boxes', 'emporia_state_universtiy_navigational_boxes')  0.1333333333333333
('festivals_in_australia', 'festivals_in_austrailia')   0.13043478260869568
('history_of_kanagawa_prefecture', 'hstory_of_kanagawa_prefecture')     0.09999999999999998
('music_festival_templates_by_festival_type', 'music_festival_tempaltes_by_festival_type')
0.13953488372093026
('political_advocacy_groups_in_italy', 'polical_advocacy_groups_in_italy')      0.06060606060606055
('radio_control', 'radio_controll')     0.07692307692307687
('schools_in_the_central_african_republic', 'schools_in_the_central_african_repblic')   0.07692307692307687
('transportation_in_harrison_county,_indiana', 'transporation_in_harrison_county,_indiana')     0.0714285714285714
('wikipedia_a-class_vital_articles_by_level', 'wikipedia_a-class_vital_articles_by_levle')
0.09523809523809523
('wikipedia_articles_incorporating_a_citation_from_the_american_cyclopaedia_with_an_unnamed_parameter',
'wikipedia_articles_incorporating_a_citation_frome_the_american_cyclopaedia_with_an_unnamed_parameter')
0.030000000000000027
```

The results for the trigrams are:

```
('1868_establishments_in_chile', '1868_establishments_in_chile')        0.0714285714285714
('1906_establishments_in_austria-hungary', '1906_establishments_in_austria-hungary')    0.052631578947368474
('1989_in_austrian_sport', '1989_in_austrian_sport')    0.09090909090909094
('2007_swedish_open', '2007_swedish_open')      0.11764705882352944
('basketball_in_oceania', 'basketball_in_oceania')      0.09523809523809523
('borders_of_spain', 'borders_of_spain')        0.125
('chicago_cubs_articles', 'chicago_cubs_articles')      0.09523809523809523
('cities_in_anderson_county,_kentucky', 'cities_in_anderson_county,_kentucky')  0.05714285714285716
('education_in_mecca', 'education_in_mecca')    0.11111111111111116
('education_in_nagaland', 'education_in_nagaland')      0.09523809523809523
('education_in_orange_county,_florida', 'education_in_orange_county,_florida')  0.05714285714285716
('films_based_on_works_by_austrian_writers', 'films_based_on_works_by_austrian_writers')
0.050000000000000044
('history_of_software', 'history_of_software')  0.10526315789473684
('houses_in_lafourche_parish,_louisiana', 'houses_in_lafourche_parish,_louisiana')      0.05405405405405406
('islands_of_aichi_prefecture', 'islands_of_aichi_prefecture')  0.07407407407407407
('kotoko_(singer)_albums', 'kotoko_(singer)_albums')    0.09090909090909094
('la_roux_albums', 'la_roux_albums')    0.1428571428571429
('later_liang_(five_dynasties)_jiedushi_of_ningyuan_circuit',
'later_liang_(five_dynasties)_jiedushi_of_ningyuan_circuit')    0.03508771929824561
('media_in_baghdad', 'media_in_baghdad')        0.125
('nigerian_athletes', 'nigerian_athletes')      0.11764705882352944
('norwegian_society', 'norwegian_society')      0.11764705882352944
('novels_by_brian_moore', 'novels_by_brian_moore')      0.09523809523809523
('olympic_sailors_of_slovenia', 'olympic_sailors_of_slovenia')  0.07407407407407407
('pakistani_beverages', 'pakistani_beverages')  0.10526315789473684
('romeo_santos_albums', 'romeo_santos_albums')  0.10526315789473684
('silurian_oceania', 'silurian_oceania')        0.125
('sport_in_papua_new_guinea', 'sport_in_papua_new_guinea')      0.07999999999999996
('transport_in_larne', 'transport_in_larne')    0.11111111111111116
```

# References


[1] Jacob Bank and Benjamin Cole. Calculating the jaccard similarity coefficient with map reduce for entity pairs in wikipedia. *Wikipedia Similarity Team*, pages 1–18, 2008.

[2] Tamer Elsayed, Jimmy Lin, and Douglas W Oard. Pairwise document similarity in large collections with mapreduce. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics on Human Language Technologies: Short Papers*, pages 265–268. Association for Computational Linguistics, 2008.