Tarea 3

Sistemas Distribuídos

Alejandro Díaz

Docente: Nicolas Hidalgo

Índice

1. Introducción

El objetivo de este trabajo es introducir a los estudiantes a los sistemas de procesamientos para grandes volúmenes de datos. Para ello, los alumnos deber´an trabajar con Hadoop, un proyecto de software de código abierto que se puede utilizar para procesar de forma eficaz conjuntos de datos de gran tamaño. Los alumnos deberán aplicar la tecnología, aplicando un código enfocado a map-reduce y reconocer las ventajas de esta. Para lograr esto, se desea recopilar 10 artículos de wikipedia y hacer un mapa de las palabras encontradas con el respectivo documento donde aparecen y la frecuencia de aparición.

2. Desarrollo

En un principio, se crea el código para recopilar los textos a través de la api de wikipedia, este código fue creado en python 3

```
#!/usr/bin/env python
# -*-coding:utf-8 -*

import wikipedia as wiki
import regex as re
wiki.set_lang("es")

pages = []
pages.append('Inquisicion')
pages.append('Guerra santa')
pages.append('Brujeria')
pages.append('Herejia en el catolicismo')
pages.append('Color')
pages.append('Idioma griego')
pages.append('Caballeros Templarios')
pages.append('Orden militar')
pages.append('Orden de malta')
pages.append('Palacio magistral de la orden de malta')

counter = 0

for page in pages:
    a = wiki.page(page)

    page = page.strip().replace(" ", "_")
    array = a.content.split(" ")
    path = "./"

    if counter < 5:
        path = path + "docs1/" + page + ".txt"
    else:
        path = path + "docs2/" + page + ".txt"

    f = open(path, "w")
    f.write("Title: " + page + "\n")

    for i in array:
        f.write(i + " ")
    counter += 1
```

En este código se recorre un arreglo con 10 títulos de páginas, en el cual por cada página se creará un documento .txt con el contenido de las páginas conseguidos a través de la api de wikipedia, considerando un contador para separar los documentos en las carpetas docs1 y docs2.

Luego, se investiga sobre la estructura de código de map reduce, el cual consta de 5 puntos:

- Asignar los procesadores que mapearán los input a través de index's que se asociarán a cada procesador, y se pasará la data del los archivos a cada procesador dependiendo de la key que le corresponda.
- Se utiliza el código de Map que se le indica a Hadoop, en este caso examples/mapper.py y este clasifica Título, palabra y 1, de esta manera se escribe el output en un alamacenamiento temporal que será leído por los reducers.
- Se asignan los procesadors que reducirán el output anterior, a través de keys que asignarán la data que deberá trabajar cada procesador, y se provee a estos con la data necesaria.
- Se utiliza el código de reducer que se le indica a Hadoop, en este caso examples/reducer.py y este cuenta la frecuencia de las palabaras a través de un diccionario, para luego utilizar un diccionario de índice invertido con key: palabra y value: array de tuplas de [página donde se encuentra la palabra, frecuencia (cantidad) de aparición].
- Finalmente se junta todo el output de los reducers y se escribe un output final.

El código de mapper.py utilizado fue el siguiente:

```
#!/usr/bin/env python
# -*-coding:utf-8 -*

import sys
import re

title = sys.stdin.readline().split(":")[1].strip()

for line in sys.stdin:
    line = line.strip()

    if line.startswith("Title:"):
        title = line.split(":")[1].strip()

    words = re.split('\W+', line)

    words = [word for word in words if not word.isdigit() and len(word) > 1]
    words = [word.lower() for word in words]

    for word in words:

        if word!= '':
            print('%s\t%s\t%s' % (title, word, 1))
```

El código de reducer utilizado fue el siguiente :

```
#!/usr/bin/env python
# -*-coding:utf-8 -*

import sys
import re

def SaveWords (hashmap, lastTitle, wordMap):
    for word in hashmap.keys():
        value = hashmap[word]
        if word in wordMap:
            array = wordMap[word]
            array.append([lastTitle,value])
            wordMap[word] = array
        else:
            wordMap[word] = ([[lastTitle,value]])

wordMap = {}
hashmap = {}
title =""
lastTitle = ""
```

```
counter = 0

for line in sys.stdin:
    title, word, count = line.split('\t')

    if title != lastTitle and counter !=0:
        SaveWords(hashmap, lastTitle, wordMap)
        hashmap = {}

    lastTitle = title
    counter += 1
    if word in hashmap:
        hashmap[word] += 1
    else:
        hashmap[word] = 1

SaveWords(hashmap, lastTitle, wordMap)

for key, value in wordMap.items():
    print('%s\t%s' % (key,value))
```

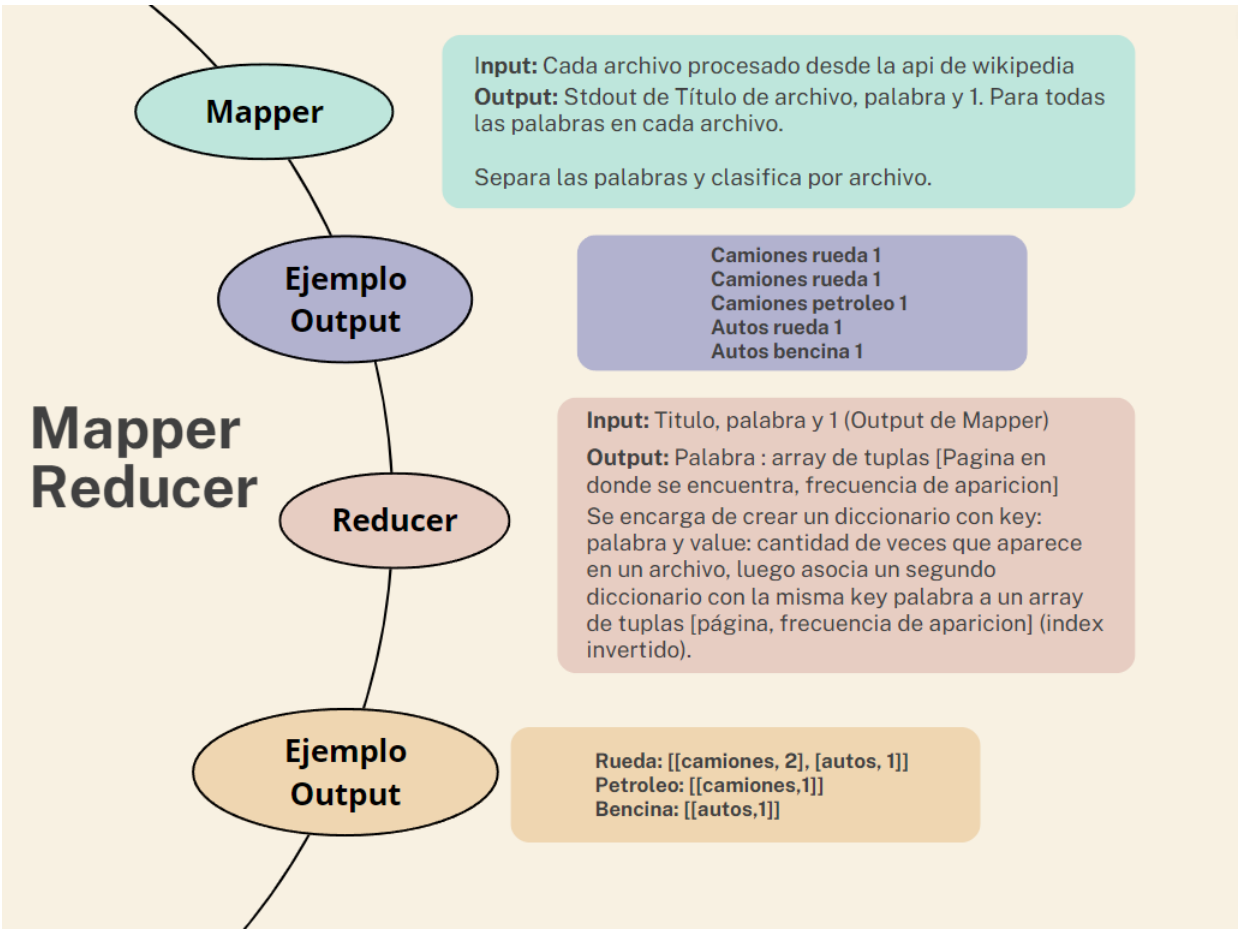

Figura 1: Diagrama de mapper reducer.

Finalmente, para el buscador, se crea un código que busca al ingresar una palabra buscará en el output de los reducer si existe la palabra y entregará los documentos en los cuales fue encontrada, con la cantidad de apariciones y la url de wikipedia del documento.

```
#!/usr/bin/env python
# -*-coding:utf-8 -*
```

```
3
4  import wikipedia as wiki
5  import regex as re
6  import sys
7  from unidecode import unidecode
8  wiki.set_lang("es")
9
10 def get_url(title):
11     return wiki.page(title).url
12
13 f = open('data.txt', 'r')
14 lines = f.readlines()
15
16 print('Ingrese palabras a buscar separadas por espacio')
17
18 # #read line from sys stdin and split by space
19 words = sys.stdin.readline().split(' ')
20 # #remove \n from last word
21 words[-1] = words[-1].strip()
22
23
24 for line in lines:
25     line = line.strip()
26     # split by tab
27     line = line.split('\t')
28     palabra = line[0]
29     #palabra to unidecode
30     palabra = unidecode(palabra)
31     #if palabra in words:
32     if palabra in words:
33         data = line[1]
34         # data remove first and last character
35         data = data[1:-1]
36         # split by comma ignoring commas inside brackets
37         data = re.split(r',\s*(?![^()]*\))', data)
38         #remove special characters from every string on data
39         data = [re.sub(r'[^\w\s]','',i) for i in data]
40         counter = 0
41
42         titles = []
43         q = []
44         for i in data:
45             if counter % 2 == 0:
46                 titles.append(i)
47             else:
48                 q.append(i)
49             counter += 1
50
51         print('Palabra: ' + palabra)
52         for i in range(len(titles)):
53             url = get_url(titles[i])
54             toPrint = '|Titulo: ' + titles[i] + '| Cantidad: ' + q[i] + '| URL: ' + url + '|\n'
55             for i in range(len(toPrint)):
56                 print('-', end ='')
57             print('\n', end = '')
58             print(toPrint, end='')
59             for i in range(len(toPrint)):
60                 print('-', end ='')
61             print('\n')
```

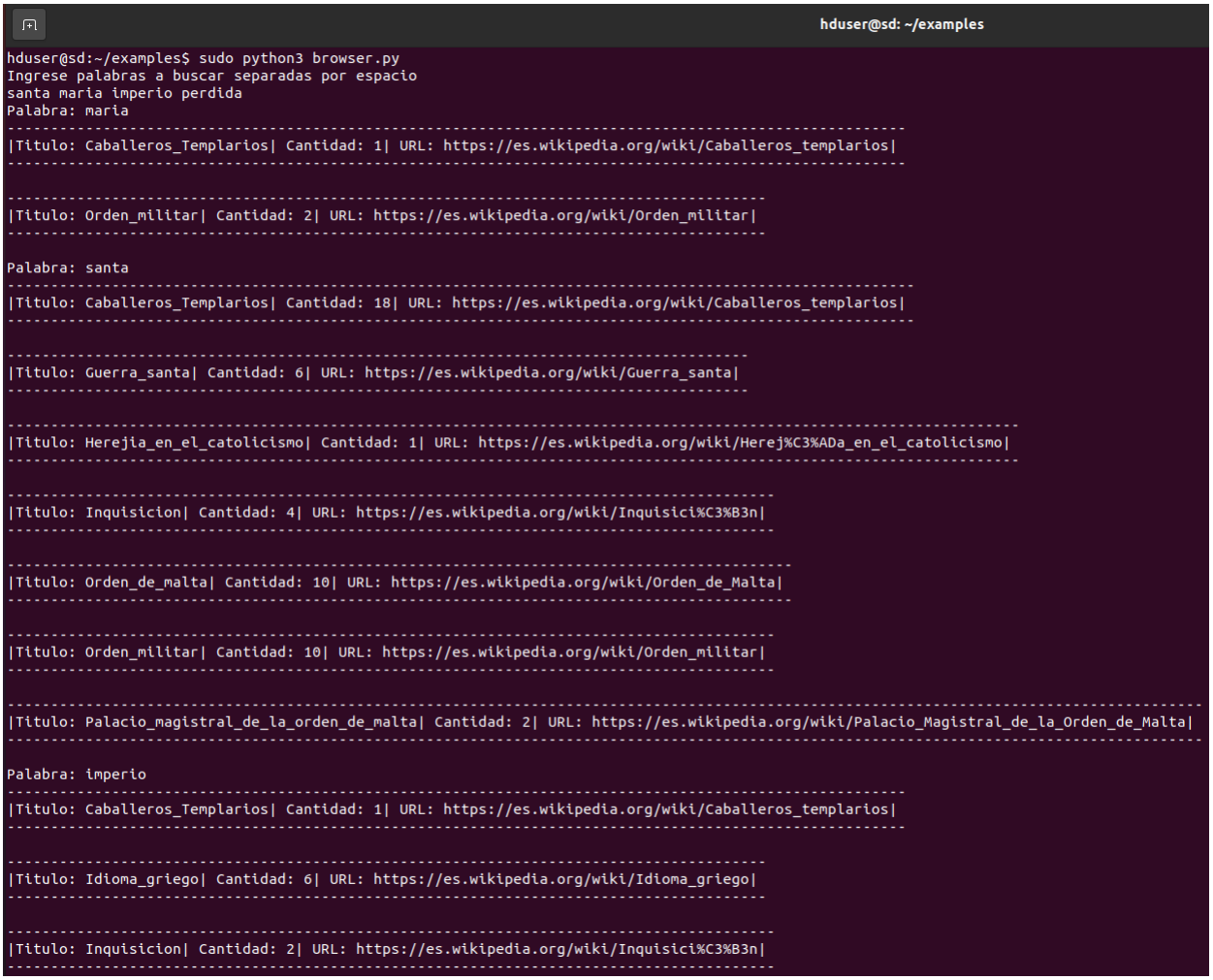

Figura 2: Funcionamiento del buscador browser.py