

Trabajo Práctico 3: Map-Reduce

Martín Arjovsky 683/12
Ezequiel Darío Gambaccini 715/13
Silvio Vileriño 106/12

Julio 2014

1 Encontrar las mejores películas

Se requería obtener la lista de las 12 películas mejor rankeadas con más de 20 reviews. Para esto, lo que decidimos hacer fue que en el map se emitiera el id del producto con un objeto que contenía el puntaje de la review y una variable count inicializada a 1.

En el siguiente paso, reduce, lo que se hace es sumar todos los objetos puntaje, cantidad y emitir el resultado.

Finalmente, en el finalize, se filtra aquellas películas que tienen count > 19. Para las películas que tienen más de 19 reseñas se devuelve la sumatoria de puntajes dividida la cantidad de reseñas.

Para las que tienen menos de 20 reseñas se devuelve -1.

Luego, usando un script de python para parsear la salida de runner.py, se filtra aquellas películas que su puntaje promedio es distinto de -1, y luego se las ordena en orden decreciente por puntaje, quedándose con las primeras 12.

Las películas que obtuvimos con sus respectivos puntajes y nombres (conseguidos a través del buscador de Amazon) fueron:

- B00000JSJ5, 5.0, All Creatures Great and Small, Series 2: Volumes 1-6
- B00004YKS6, 5.0, Genghis Blues
- B00004YKS7, 5.0, Genghis Blues
- B0002NY7UY, 5.0, Live in Concert (Dion)
- B0007GAEXK, 5.0, The Mole - The Complete First Season
- B0007Z4HAC, 5.0, Salsa Crazy Presents: Learn to Salsa Dance, Intermediate Series, Volume 1
- B000AOEPU2, 5.0, WWE: Bret "Hitman" Hart - The Best There Is, The Best There Was, The Best There Ever Will Be
- B000MCIADA, 5.0, A Reiki 1st, Aura and Chakra Attunement Performed

- B003YBGJ4S, 5.0, WELL worked out with Tannis
- B004LK24BI, 5.0, 50/50 Cardio and Weights with Angie Gorr
- B006JN87UC, 5.0, Transformers: Prime - Season One
- B008COIZHQ, 5.0, Genghis Blues
- B005FY0FPG, 4.987012987012987, Dream With Me in Concert
- B000M7XRC4, 4.977777777777778, The Venture Bros. - Season Two

A continuación se muestra el código de map, reduce y finalize respectivamente.

```

1 function () {
2   data = {};
3   data.score = parseFloat(this.score,10);
4   data.count = 1;
5   emit(this.productId, data);
6 }
7
8 function (key, values) {
9   var res = {};
10  res.score = 0;
11  res.count = 0;
12  for (var i = values.length - 1; i >= 0; i--) {
13    var data = values[i];
14    res.score += data.score;
15    res.count += data.count;
16  };
17  return res;
18 }
19
20 function (key, reducedVal) {
21   if (reducedVal.count > 19) {
22     return reducedVal.score/reducedVal.count;
23   } else{
24     return -1;
25   };
26 }

```

2 Palabras más frecuentes

Para obtener las 5 palabras más usadas de cada puntaje, en la función de map se realizan varias operaciones:

1. Se filtran los caracteres del texto de la review para que queden solo los alfanumericos, y se los pasa a minúscula.
2. Se divide el texto en un array de palabras y luego se aplica una función para filtrar aquellas palabras que están en el array de stop words.

3. Finalmente, se cuentan las apariciones de cada palabra en un objeto, y se emite el valor del puntaje y el objeto que cuenta ocurrencias.

En el reduce, simplemente se genera un nuevo objeto con las ocurrencias de cada palabra de ese puntaje particular.

En el finalize, se invierten los valores del objeto, es decir, se devuelve un objeto que contiene numeros como claves y listas de palabras como valor. Los numeros representan la cantidad de apariciones de las palabras de la lista.

Finalmente, las palabras más usadas para las reviews, divididas por puntaje, ocurrencias, y palabras, son:

- Puntaje: 1.0
268: film
271: time
297: dvd
325: one
372: movie
- Puntaje: 2.0
182 : very
189 : dvd
196 : one
247 : good
334 : movie
- Puntaje: 3.0
252 : dvd
264 : one
276 : great
328 : good
369 : movie
- Puntaje: 4.0
356 : very
368 : one
376 : good , great
381 : movie
- Puntaje: 5.0
381: dvd,really,film,love,see

A continuación se muestra el código de map, reduce y finalize respectivamente.

```
1 function () {  
2  
3   var data = this.text.slice(0);  
4  
5   data = data.replace(/\\W/g, ' ').toLowerCase(); //remueve  
6     caracteres no alfanumericos
```

```

7  var stop_words = ["", "a", "able", "about", "across", "after", "all", "almost", "also", "am", "among", "an",
8  "and", "any", "are", "as", "at", "be", "because", "been", "but", "by", "can",
9  "cannot",
10 "could", "dear", "did", "do", "does", "either", "else", "ever", "every", "for", "from",
11 "get", "got", "had", "has", "have", "he", "her", "hers", "him", "his", "how",
12 "however",
13 "i", "if", "in", "into", "is", "it", "its", "just", "least", "let", "like", "likely", "may",
14 "me", "might", "most", "must", "my", "neither", "no", "nor", "not", "of", "off", "often",
15 "on", "only", "or", "other", "our", "own", "rather", "said", "say", "says", "she", "should",
16 "since", "so", "some", "than", "that", "the", "their", "them", "then", "there", "these",
17 "they", "this", "tis", "to", "too", "twas", "us", "wants", "was", "we", "were", "what",
18 "when", "where", "which", "while", "who", "whom", "why", "will", "with", "would", "yet", "you", "your"];
19
20 var clear_stopwords = function (elem) {
21     return (stop_words.indexOf(elem) === -1);
22 }
23
24 var filtered = data.split(" ").filter(clear_stopwords);
25
26 var dict = new Object();
27
28 for (var i = filtered.length - 1; i >= 0; i--) {
29     var s = filtered[i];
30     if (s in dict) {
31         dict[s] += 1;
32     }
33     else {
34         dict[s] = 1;
35     }
36 };
37
38 emit(this.score, tojson(dict));
39 }
40
41 function (key, values) {
42
43     var res = new Object();
44
45     for (var i = values.length - 1; i >= 0; i--) {
46         var data = JSON.parse(values[i]);
47         for (var key in data){
48             if (data.hasOwnProperty(key)){
49                 if (key in res) {
50                     res[key] += 1;
51                 }
52                 else {
53                     res[key] = 1;
54                 }
55             }
56         }
57     }
58 }

```

```

54     }
55 };
56
57 return toJson(res);
58 }
59
60 function (key, reducedVal) {
61     reducedVal = JSON.parse(reducedVal);
62     var top_words = new Object();
63
64     for(var word in reducedVal){
65         if (reducedVal.hasOwnProperty(word)) {
66             var count = reducedVal[word];
67             if (count in top_words){
68                 top_words[count].push(word);
69             }
70             else{
71                 top_words[count] = [word];
72             }
73         }
74     }
75     return toJson(top_words);
76 }

```

3 Helpfulness y Longitud

Este fue el ejercicio más sencillo de los tres.

En el map, simplemente se emite el helpfulness y la longitud de la review.

Luego, en el reduce, se calcula la longitud promedio de las reviews, sumando sus longitudes y dividiendolas por la cantidad de valores en ese reduce.

A continuación se muestra el código de map y reduce respectivamente.

```

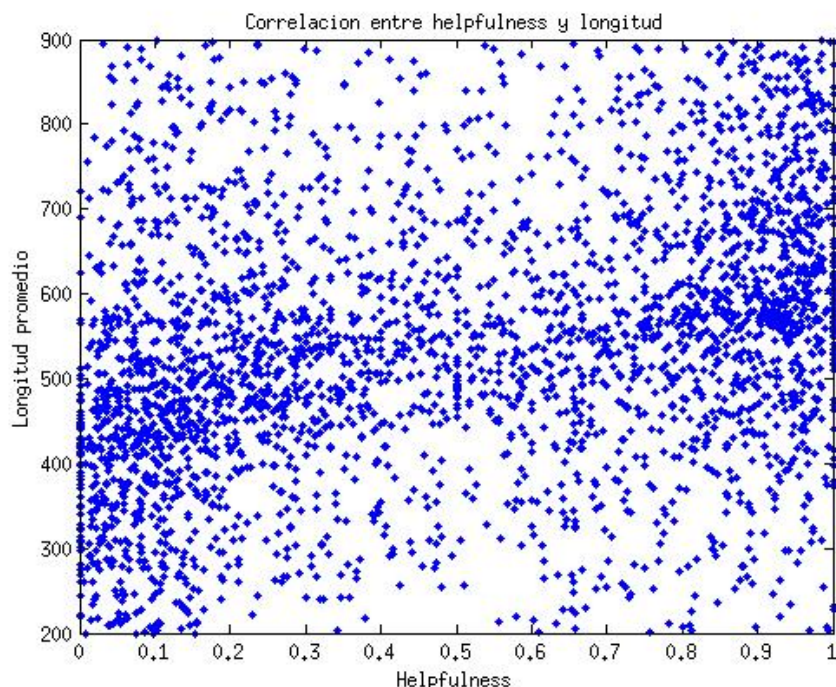
1 function () {
2     emit(this.helpfulness, this.text.length);
3 }
4
5 function (key, values) {
6     avg_length = 0;
7     for (var i=0; i < values.length; i++){
8         avg_length += values[i];
9     }
10    return avg_length/values.length;
11 }

```

La data luego fue parseada para realizar el análisis del punto 4. En este caso se midió la helpfulness como el cociente entre los votos de que una review era helpful y la cantidad total de votos. Se evaluó el coeficiente de correlación lineal de Pearson entre la helpfulness y la longitud promedio de las reviews, dando un valor de 0.3159. Esto indica que las variables están positivamente correlacionadas, indicando que es probable que cuando una crezca, la otra lo haga también.

Sin embargo es importante mostrar que no parece haber una correlación lineal entre estos factores, como se evidencia en el siguiente gráfico, por esto es que

concluimos que otras métricas pueden ser más útiles para medir la influencia de un dato sobre el otro. De todas maneras este coeficiente nos da la indicación (soportada por el gráfico) de que estas variables no son independientes, y de que se puede extraer información de una a partir de la otra.



4 Escalabilidad

4.1 Cloud Computing - Haas: Definicion

La computacion en la nube son servidores - o granjas de servidores - repartidos por todo el mundo que alojan servicios de manera remota de forma tal que se puede acceder a un servicio contratado desde cualquier parte del mundo donde se disponga de una conexion a internet

La infraestructura como servicio (IaaS) o hardware as a service (HaaS) provee almacenamiento básico y soluciones de cómputo como servicios contratables en la red. Servidores, sistemas de almacenamiento, conexiones, routers, y otros sistemas se concentran a través de la tecnología de virtualización para manejar tipos específicos de cargas de trabajo. El ejemplo comercial mejor conocido es Amazon Web Services, cuyos servicios EC2 y S3 ofrecen cómputo y servicios de almacenamiento esenciales a clientes, de forma que puedan accederlos de forma sencilla y con alto grado de estabilidad.

4.2 Implementacion de nuestra solucion en la nube

Existe un framework llamado Hadoop, que puede ayudarte a almacenar y procesar grandes volúmenes de datos a cualquier escala, a través de la distribución de estos datos sobre una granja de servidores. El problema es que el lanzamiento, aprovisionamiento, manejo y configuración de Hadoop puede ser complicado, costoso y puede tomar mucho tiempo valioso que puede aprovecharse analizando los datos. La forma más común de implementar esta idea, es comprar el servicio de **hardware as a service** (Haas), aprovisionar el hardware, instalar y finalmente lanzar el software sobre el hardware en la nube obtenido, todo esto sin poder siquiera haber procesado ningún dato aun.

Amazon EMR provee soluciones de fácil uso, migración y escalabilidad para el procesamiento de grandes cantidades de datos. Amazon EMR, provee una forma simple, segura y rápida de solucionar este inconveniente: Aprovechando las plataformas **Amazon EC2** y **Amazon S3** se proveen soluciones de almacenamiento y procesamiento en la nube, brindando un entorno Hadoop que distribuye el procesamiento de los datos sobre múltiples instancias de **Amazon EC2**. Dado que **Amazon EMR** permite trabajar con código **map-reduce** personalizable, una vez obtenido el servicio, para comenzar, deben realizarse los siguientes pasos:

1. Cargar los datos y aplicaciones de procesamiento dentro de **Amazon S3**
2. Distribuir los datos en la instancia de **Amazon EC2**
3. **Amazon EMR** comienza a procesar los datos!
4. Al finalizar el procesamiento, los resultados estarán disponibles en la instancia de **Amazon S3**

4.3 Flexibilidad y costos

Amazon EMR monitorea el trabajo en todo momento, para desalojar los recursos en la nube al finalizar este, de esta forma, solo se paga por lo que se usa. De manera fácil, se puede incrementar o decrementar la cantidad de recursos para diferentes necesidades asociadas a volúmenes de datos o necesidad de procesamiento más rápido. El costo del servicio es flexible, puede pagarse por hora **on-demand**, por capacidad fija reservada, entre otros.

4.4 Presupuesto y escalabilidad

Dado que los volúmenes de datos crezcan a un ritmo lineal anual, y que **map-reduce** tiene como premisa la escalabilidad lineal (por ejemplo: doble procesamiento - mitad de tiempo), Sean X el volumen de datos, Y la capacidad de procesamiento, Z el rendimiento obtenido en tiempo (mientras Z decrece, mayor rendimiento) y $k, t \in \mathbb{N}$, se tiene:

Ecuacion de escalabilidad lineal para tamao fijo de datos:

$$X \text{ datos en } t.Y \text{ hardware} \rightarrow \frac{Z}{t} \text{ tiempo}$$

Asumiendo que el tiempo de procesamiento crece linealmente junto al tamao de los datos, tenemos:

Ecuacion de escalabilidad lineal generalizada:

$$k.X \text{ datos en } t.Y \text{ hardware} \rightarrow \frac{k.Z}{t} \text{ tiempo}$$

Si queremos mantener el rendimiento Z constante, debemos incrementar linealmente en un factor k el hardware Y de forma anual de forma que la ecuacion se reestablezca:

Ecuacion de escalabilidad balanceada:

$$k.X \text{ datos en } (k.t).Y \text{ hardware} \rightarrow \frac{k.Z}{(k.t)} \text{ tiempo} \rightarrow \frac{Z}{t} \text{ tiempo}$$

Respecto a la mejora obtenida respecto a utilizar una sola pc y utilizar los servicios en la nube es similar, dada la relacion de escalabilidad lineal, el rendimiento se mejorara linealmente a medida que incrementemos de forma lineal la cantidad de nodos en los clusteres de Amazon contratados.

En un caso hipotetico, dado que Amazon ofrece un clster de Hadoop de 10 nodos por 0,15 USD la hora, habria que calcular los tiempos promedio de procesamiento requeridos, ajustar la cantidad de nodos segun el volumen, el tiempo necesario de procesamiento y hacer los calculos del gasto mensual, anual, etc del servicio requerido a Amazon. Siempre teniendo en cuenta que debe duplicarse anualmente la cantidad de nodos para mantener estable el rendimiento por lo mencionado en el parrafo anterior.