Proyecto Final aprendizaje automático sobre grandes volúmenes de datos

Germán E. Trouillet

5 de diciembre de 2014

Este trabajo está basado en el caso de estudio visto en clase de recomendación, se creo un repositorio en GitHub (https://github.com/gtrouillet/aprendizajengrande.git) para mantener el código del mismo.

Se crearon dos proyectos maven, *recommender*, donde se encuentra el programa para extraer los logs de un repositorio de Git (**GitLogExtractor**), la clase **PreferenceBuilder** para generar las preferencias a partir de los datos extraídos y la clase **Recommend** para generar las recomendaciones utilizando el **RecommenderJob**.

También se creo el proyecto *item-recommender* que utiliza la recomendación basada en items de **Mahout**.

1. Git Log Extractor

Está clase utiliza el comando de git-log, que permite extraer toda la información necesaria, con los siguientes parámetros:

```
\$> git log --pretty=format:'commit: "%H", author: "%an"' --name-status
```

La salida tiene la siguiente forma:

```
commit: "fc14f9c1272f62c3e8d01300f52467c0d9af50f9", author: "Linus Torvalds"
M       Makefile

commit: "e35c5a2759f360a45f052b422a87c47971a4e2d9", author: "Linus Torvalds"
commit: "435e46f5d35dbe5ea745822db29f16e003fa07d7", author: "Linus Torvalds"
commit: "0fbae13642fc35ba6a774e9f76d0c1946e8f5c35", author: "Linus Torvalds"
commit: "e899dbaf4898efdaf6d1a02ed4ac205d35f54df8", author: "Peter Rosin"
M       arch/arm/boot/dts/sama5d31.dtsi
M       arch/arm/boot/dts/sama5d33.dtsi
M       arch/arm/boot/dts/sama5d34.dtsi
M       arch/arm/boot/dts/sama5d35.dtsi
M       arch/arm/boot/dts/sama5d36.dtsi
M       arch/arm/boot/dts/sama5d3xcm.dtsi
```

donde, para cada commit tenemos la información del hash, autor y la lista de archivos modificada, en caso de no tener modificaciones en archivos, sólo tenemos la información de hash y autor.

1.1. Ejecución

Una vez compilado el proyecto con maven, en la carpeta target se encuentran, el jar *recommender-0.1.0-SNAPSHOT.jar*, y la carpeta lib con las dependencias necesarias. Para correr el programa, se debe ejecutar el siguiente comando:

```
\$> java -cp "<Path>/recommender-0.1.0-SNAPSHOT.jar:<Path>/lib/*"
net.aprendizajengrande.recommender.ExtractorMain <GitRepo/.git> <logDB
output Folder>
```

Una vez finalizada la ejecución obtendremos una salida similar a la que se muestra en la Figura 1

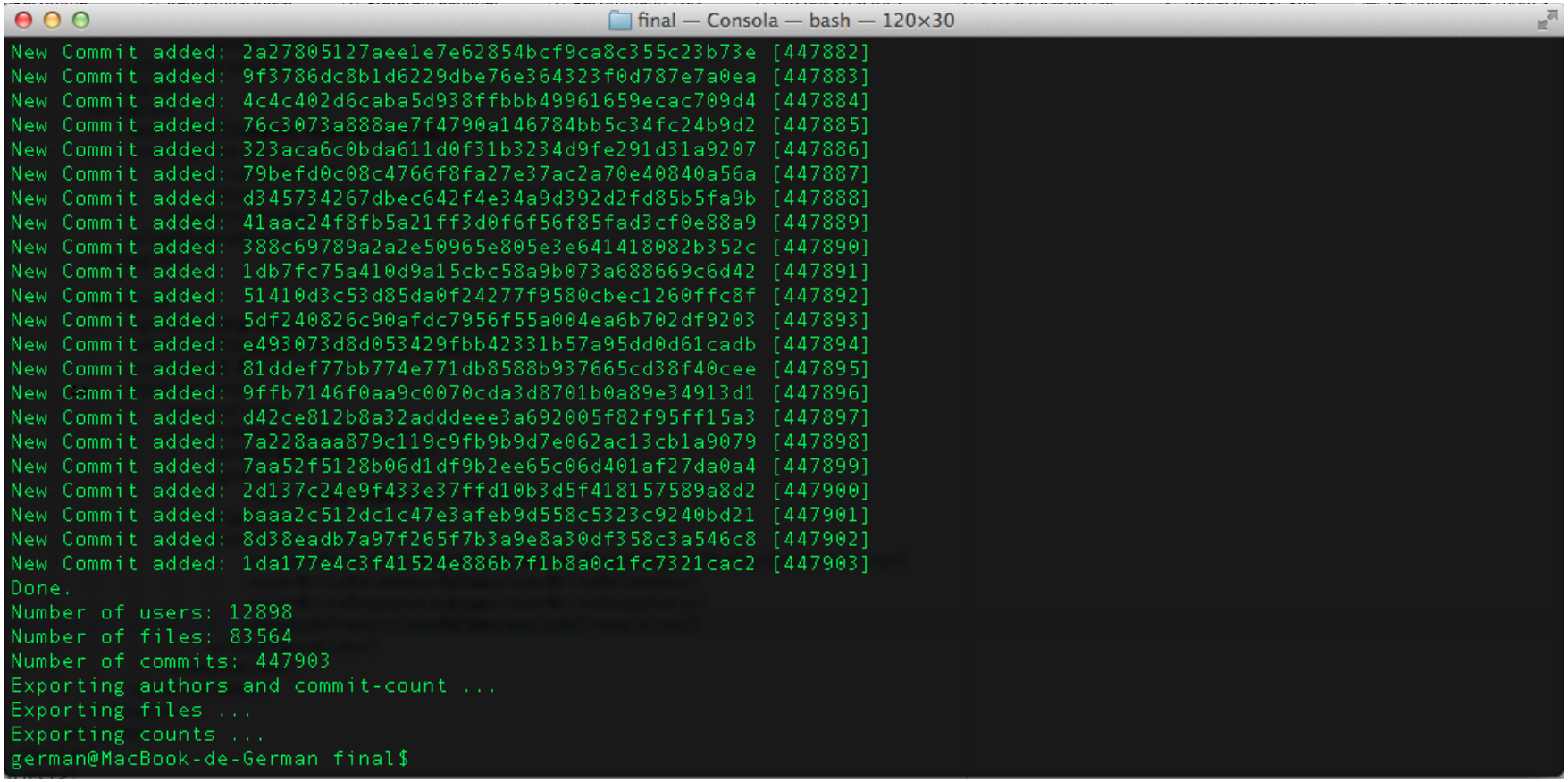

Figura 1: Log Extractor

1.2. Control del código

Para controlar el proceso de extracción de logs de git, se realizo un Unit Test (UT) de la clase **GitLogExtractor**, que controla la extracción de información de la salida del comando. Además se seleccionaron un número de usuarios al azar, para controlar la información generada por el programa.

A continuación se detallan los controles realizados para dos de los usuarios seleccionados:

Del archivo commit-counts.tsv se selecciona el usuarion 7 (con siete commits), y 28 (con 5 commits). Podemos ver en el archivo authors.txt que dichos usuarios son:

- Usuario 7: Ulrik De Bie
- Usuario 28: William Cohen

Se ejecuto el siguiente comando para obtener los el log de los commits realizados :

```
\$> git log --pretty=format:'commit: "%H", author: "%an"' --name-status
--author='<author name>'
```

1.2.1. Usuario 7

```
\$> git log --pretty=format:'commit: "%H", author: "%an"' --name-status
--author='Ulrik De Bie'
commit: "c6c748ef85c342d2726fc3c91c6a2af313af2360", author: "Ulrik De Bie"
M       Documentation/input/elantech.txt

commit: "2d9eb81fdb9f08df3a4b1638c1270a4453b40ac2", author: "Ulrik De Bie"
M       drivers/input/mouse/elantech.c

commit: "f386474e12a560e005ec7899e78f51f6bdc3cf41", author: "Ulrik De Bie"
M       drivers/input/mouse/elantech.c

commit: "0dc1587905a50f8f61bbc29e850aa592821e4bea", author: "Ulrik De Bie"
M       drivers/input/mouse/elantech.c

commit: "caeb0d37fa3e387eb0dd22e5d497523c002033d1", author: "Ulrik De Bie"
M       drivers/input/mouse/elantech.c

commit: "a2418fc4a13b5da8d007a038c0a6a50a54edfabd", author: "Ulrik De Bie"
M       drivers/input/mouse/elantech.c
M       drivers/input/mouse/elantech.h

commit: "ac84eba220c401f7616237ee6e5b73f66afb3590", author: "Ulrik De Bie"
M       drivers/input/mouse/elantech.c
```

Se puede observar que el usuario Ulrik De Bie, efectivamente tiene 7 commits, además dichos commits son sobre los archivos *Documentation/input/elantech.txt* en una ocasión, *drivers/input/mouse/elantech.c* en 6 ocasiones y *drivers/input/mouse/elantech.h* en una ocasión. Por lo tanto si controlamos el archivo *counts.tsv* para el usuario 7 se obtiene:

```
7
     12    1
     13    6
     8728    1
```

Del archivo *files.txt* se obtiene que los archivos 12, 13 y 8728 son: *Documentation/input/elantech.txt*, *drivers/input/mouse/elantech.c* y *drivers/input/mouse/elantech.h* respectivamente.

1.2.2. Usuario 28

```
    \$> git log --pretty=format:'commit: "%H", author: "%an"' --name-status
--author='William Cohen'
    commit: "899d5933b2dd2720f2b20b01eaa07871aa6ad096", author: "William Cohen"
    M       arch/arm64/kernel/insn.c

    commit: "cfadf9d4ac4be940595ab73d3def24c23c8b875f", author: "William Cohen"
    M       tools/perf/Documentation/perf-kvm.txt

    commit: "881245dcff29df992d8431392a41fb81549129f9", author: "William Cohen"
    M       Documentation/DocBook/tracepoint.tmpl
    M       include/trace/events/block.h

    commit: "3d337c653c94be50f11a45fb14a2afa8a8a1a618", author: "William Cohen"
    M       arch/x86/oprofile/nmi_int.c

    commit: "97dc32cdb1b53832801159d5f634b41aad9d0a23", author: "William Cohen"
    M       fs/proc/array.c
    M       include/linux/sched.h
```

Se puede observar que el usuario William Cohen, efectivamente tiene 5 commits, sobre 7 archivos. Del archivo *counts.tsv* para el usuario 28 se obtiene:

```
12391
     1254    1
     1765    1
     11938    1
     8852    1
     22873    1
     60212    1
     45    1
```

Controlando en el archivo files.txt se observa que dichos archivos se corresponden a los encontrados en el log de git.

2. PreferenceBuilder

Este main creara las preferencias de los usuarios sobre los archivos, utilizando la cantidad de veces que el usuario realizó un commit sobre dicho archivo de la misma manera que el caso de estudio visto en clase.

Se debe correr en un entorno Hadoop en modo pseudo-cluster (como fue visto en clase), ejecutando el siguiente comando:

```
\$> hadoop jar recommender/target/recommender-0.1.0-SNAPSHOT-jar-with-dependencies.jar
net.aprendizajengrande.recommender.PreferenceBuilder ./logdb/ /user/gtrouillet/input
```

Una vez finalizada la ejecución podemos encontrar en el HDFS el archivo ratings con las preferencias de los usuarios como se puede observar en la Figura 2.

Figura 2: Archivo Ratings

3. Recomendaciones con RecommenderJob

La última clase main que se encuentra en el proyecto **recommender**, es **Recommend**, que se utiliza para calcular las recomendaciones para los usuarios utilizando el **RecommenderJob**.

Para ejecutar este main se debe correr el siguiente comando:

```
\$> hadoop jar recommender/target/recommender-0.1.0-SNAPSHOT-jar-with-dependencies.jar
net.aprendizajengrande.recommender.Recommend ./logdb/ /user/gtrouillet/input/ratings
/user/gtrouillet/output recommendations
```

Durante la ejecución de dicho comando se observan los distintos jobs de MapReduce que utiliza el **RecommenderJob**, como puede observarse en la Figura 3 y la Figura 4.

Una vez finalizada la ejecución, encontraremos que se crearon los archivos con las recomendaciones, como puede observarse en la Figura 5.

Figura 3: Map Reduce

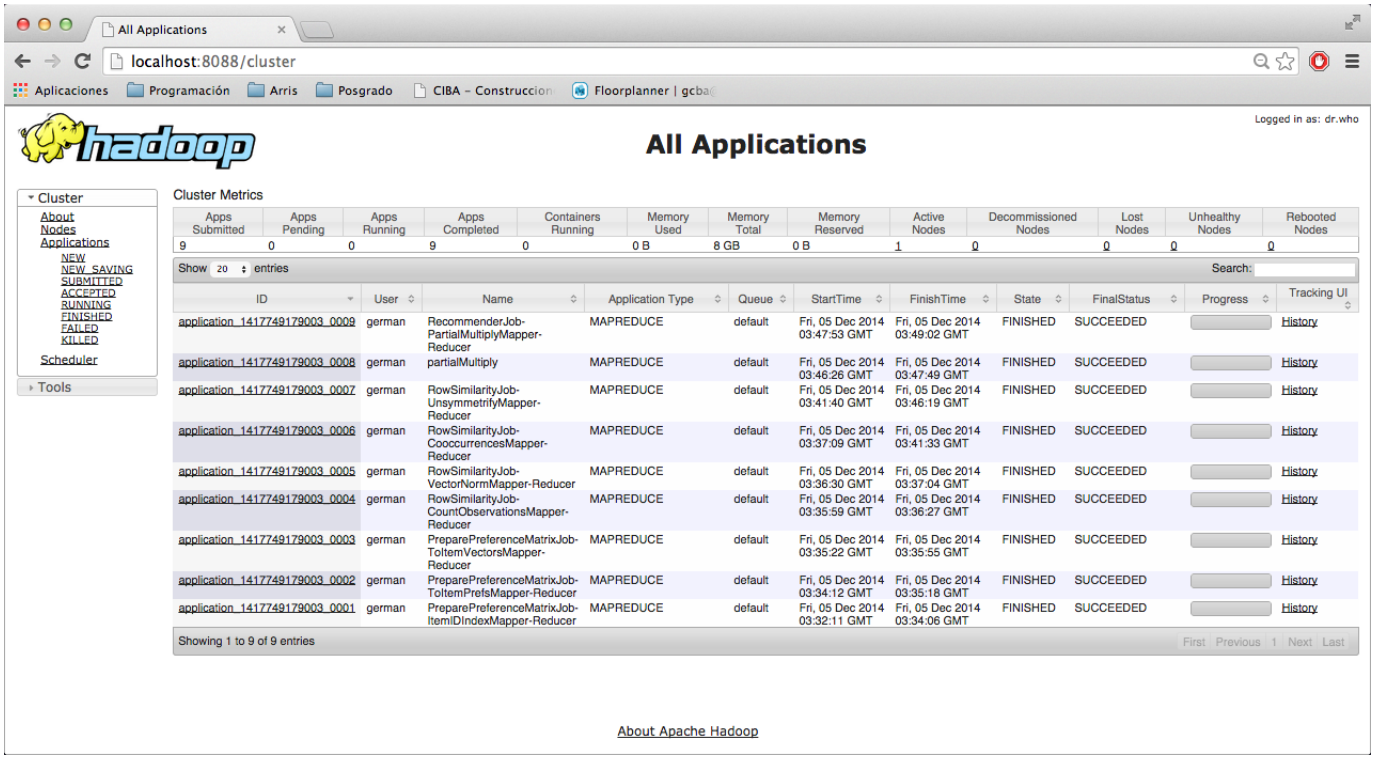

Figura 4: Trabajos Map Reduce

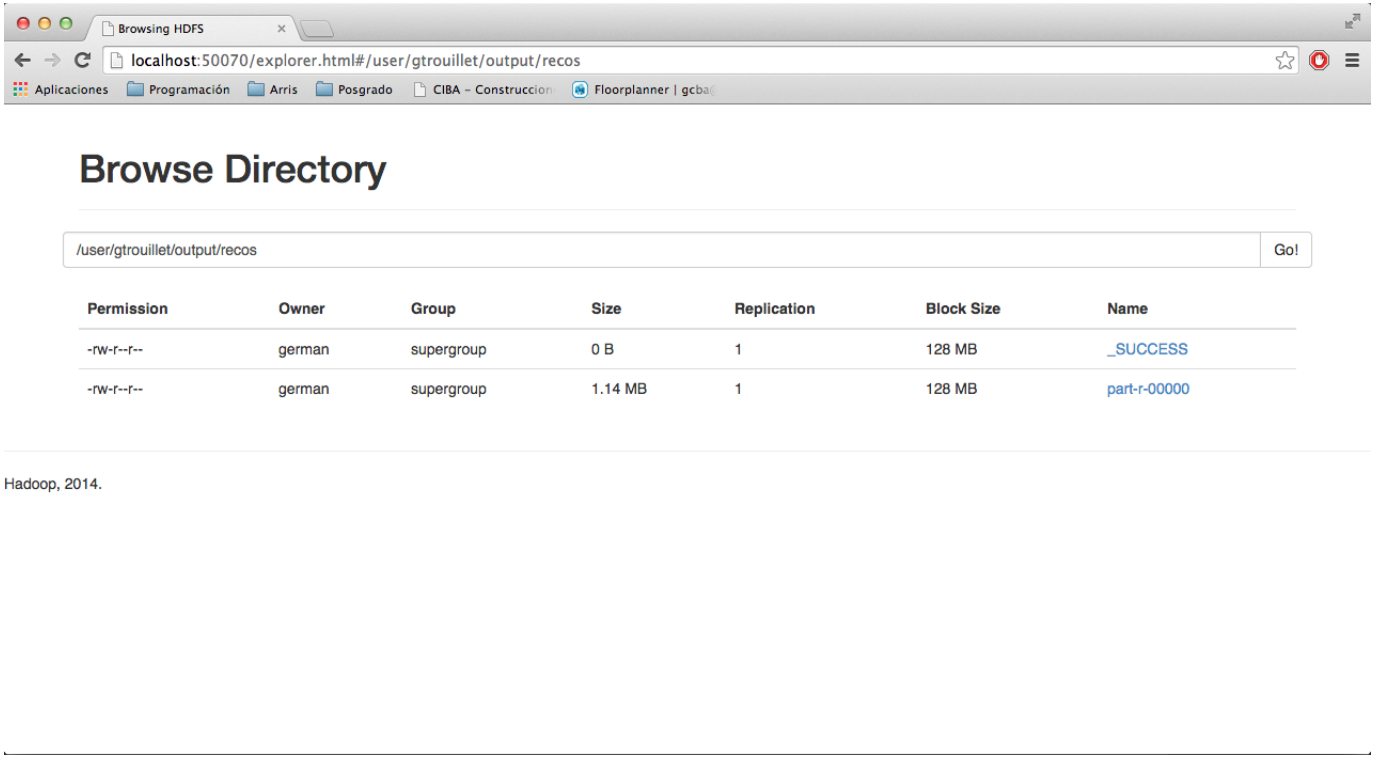

Figura 5: Archivo Recomendaciones

4. ItemBaseRecommender

En el proyecto **item-recommender**, la clase main permite ejecutar recomendaciones basada en items sobre el mismo archivo de preferencias creado con **PreferenceBuilder**.

Para la ejecución de este recommender, hay un parámetro que determina la implementación de la interfaz **ItemSimilarity** que se quiere utilizar. Por el momento, los valores posibles son, *Euclidean*, para utilizar una implementación que trae Mahout, o *File*, para ejecutar una implementación propia (**FileItemSimilarity**) que mide la similitud entre dos items teniendo en cuenta el nombre del archivo.

Para ejecutar este recommender se utiliza el siguiente comando:

```
\$> java -cp item-recommender/target/item-recommender-0.1.0-SNAPSHOT-jar-with-dependencies.jar
net.aprendizajengrande.recommender.ItemBasedRecommender ./logdb/ ./ratings
./itemBasedRecommendations <Euclidean|File>
```

Al ejecutar este comando obtendremos una salida somo la que se muestra en la Figura 6.

Figura 6: Item Based Recommender

5. Trabajo Futuro

Uno de los problemas observados durante la ejecución de estas tareas, es que la implementacón basada en items, es muy lenta por calcular la distancia entre los items. Una posible mejora del código podría ser pre-calcular las matriz de distancias y luego obtener las recomendaciones a partir de dicha matriz. Este cálculo podría ser realizado utilizando trabajos MapReduce, de similar manera a lo realizado por el RecommenderJob.

Otro aspecto de posible mejora es la utilización de la base de datos para obtener los datos para el calculo de las preferencias, y así evitar la creación de los archivos en la carpeta logDB.