INFERENCIA DE LUGARES ICÓNICOS DE UNA CIUDAD

GUÍA DE USO

Álvaro Sánchez Blasco

26/07/2015

GUIA DE INSTALACIÓN Y USO DEL PROYECTO DE INFERENCIA DE LUGARES ICÓNICOS DE UNA CIUDAD.

Contenido

1. Introducción.

El propósito de este documento es una guía de instalación y uso del algoritmo de inferencia desarrollado para inferir lugares icónicos de una ciudad.

Se detallan los pasos esenciales para llevar a cabo todo el proceso.

Los conocimientos de informática y de desarrollo de software están sobreentendidos.

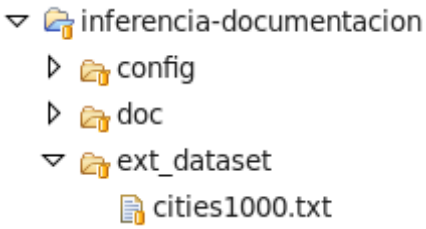
Para cualquier información adicional que se desee: alvaroschz@gmail.com

Muchas gracias.

2. Obtener los dataset.

Para ello, descargamos el dataset de geonames (http://download.geonames.org/export/dump/), y buscaremos este archivo.

Se encuentra también en el proyecto de documentación, en la carpeta ext_dataset ya descomprimido:



También debemos obtener el dataset de Flickr, que se encuentra almacenado en AWS, (hay que registrarse en Yahoo y en AWS para obtener la información de descarga): http://webscope.sandbox.yahoo.com/catalog.php?datatype=i&did=67.

Descomprimir el fichero descargado de geonames, porque vamos a leer un fichero txt, no un archivo comprimido.

3. Copiar los datos a HDFS

Crear en HDFS los directorios donde queremos que se encuentren nuestros datos inicialmente.

Mediante el comando

```
hdfsdfs –put *.bz2 [ruta_incial]/
```

copiar los archivos del dataset a HDFS.

Realizar la misma acción para el fichero txt de geonames a la ruta deseada.

4. Preparar el jar.

https://github.com/AlvaroSanchezBlasco

Se provee de un archivo ejecutable para la preparación del jar del proyecto mediante el uso de la herramienta Maven.

El archivo `package` se encuentra en la raíz proyecto.

```
[cloudera@quickstart inferencia-mapony]$ ls -l
total 24
drwxrwxr-x 9 cloudera cloudera 4096 Jul 26 00:25 bin
-rwxrwxr-x 1 cloudera cloudera  150 Jul 26 03:55 package
-rw-rw-r-- 1 cloudera cloudera 2922 Jul 25 17:15 pom.xml
drwxrwxr-x 2 cloudera cloudera 4096 Jul 26 01:13 properties
drwxrwxr-x 3 cloudera cloudera 4096 Jul 26 00:25 src
drwxrwxr-x 7 cloudera cloudera 4096 Jul 26 04:25 target
[cloudera@quickstart inferencia-mapony]$
```

Es un ejecutable que limpia, compila, crea los jar para poder ejecutar el job.

Además, prepara el proyecto para poder importarlo a Eclipse, si se desea.

```
mvn eclipse:clean
mvn clean
mvn compile
mvn package
mvn eclipse:eclipse
cd target
ls -l
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar
exit
```

Ejecutar por tanto

```
./package
```

desde la carpeta raíz del Proyecto.

En la carpeta target encontraremos el proyecto en formato jar

```
inferencia-mapony-1.0.0-ejecucion.jar
```

con las librerías dependientes incluidas, para poder llevarlo a la máquina que sea y poder ejecutarlo.

A esta carpeta habrá que copiar los ficheros de propiedades de cada Job.

Podemos ejecutar con la siguiente sentencia:

```
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar
```

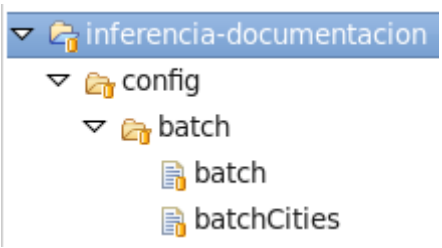
Nos mostrará las opciones de ejecución.

```
[cloudera@quickstart target]$ hadoop jar inferencia-mapony-1.0.0-ejecucion.jar
An example program must be given as the first argument.
Valid program names are:
  groupNear:    Group By Near Places Job.
                Primer Job a ejecutar. Agrupa los datos por ciudades.
                First Job to run. Arrange data by populated cities.
  groupNearCities:      Group By Near Places Selected Cities Job.
                Primer Job a ejecutar. Agrupa los datos de las ciudades: Madrid, Londres, Berlin, Roma, Paris, Nueva York.
                First Job to run. Arrange data by the cities: Madrid, London, Berlin, Rome, Paris, New York.
  load:         Load Job.
                Tercer Job a ejecutar. Carga los datos en un cluster de ElasticSearch.
                Third Job to run. Loads data into a ElasticSearch cluster.
  precission:   Precission Job.
                Segundo Job a ejecutar. Agrupa con mayor precision los datos.
                Second Job to run. Gives more accuracy to arrange data.
  precissionCities:     Precission Job.
                Segundo Job a ejecutar. Agrupa con mayor precision los de las ciudades: Madrid, Londres, Berlin, Roma, Paris, Nueva York.
                Second Job to run. Gives more accuracy to arrange data of the cities: Madrid, London, Berlin, Rome, Paris, New York.
```

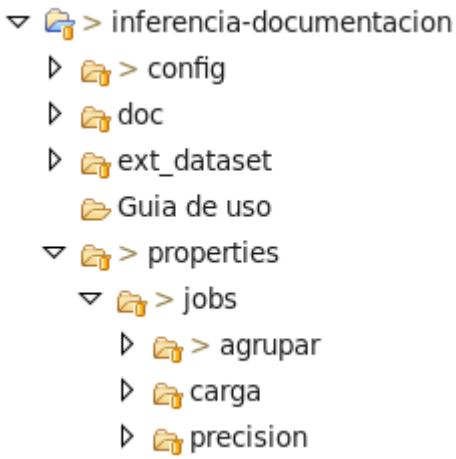
Todos los Jobs que pueden lanzarse, necesitan de un fichero de propiedades en el que definiremos diferentes parámetros de ejecución.

```
[cloudera@quickstart target]$ hadoop jar inferencia-mapony-1.0.0-ejecucion.jar groupNear
Uso: <config.properties file>
[cloudera@quickstart target]$
```

Se provee también, en el proyecto de documentación, de una serie de lanzadores para cada job de agrupación: `batch` y `batchCities`.



Están preparados para lanzar en modo batch la ejecución de cada Job, asociado a un `properties` concreto. Dichos `properties` se encuentran en la carpeta `properties` del proyecto de documentación, en `jobs/NOMBRE_JOB`. Habrá que copiarlos en el directorio target.




```
batch:
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar groupNear gn0.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precission pre00.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precission pre10.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precission pre20.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar groupNear gn1.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precission pre01.properties
....
exit


batchCities:
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar groupNearCities gn0.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precissionCities pre00.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precissionCities pre10.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar precissionCities pre20.properties
hadoop jar inferencia-mapony-1.0.0-ejecucion.jar groupNearCities gn1.properties
....
exit
```

5. Propiedades de los job.

Cada Job necesita de unos parámetros para ser ejecutado. Han de incluirse todos, ya que de no existir, obtendremos errores de ejecución.

GroupNear:

```
#Ruta en HDFS donde tenemos que copiar el fichero obtenido de geonames
#(se puede encontrar en la carpeta ext_dataset del proyecto de
#documentacion).

#Cuando se realice la ejecución por ciudades, esta propiedad no será
#necesaria.

ruta_paises=ext_dataset/cities5000.txt


#Patron de busqueda de ficheros en la carpeta /data/ de nuestro HDFS+indice
#(concatenado en Java).

ruta_inicial_fichero=data/*


#Ruta donde se almacena la salida del primer Job+indice (concatenado en
java).

ruta_salida_job=groupNearJobOut


#Indica el fichero del dataset que queremos procesar. Modificar a medida
que ejecutamos, aumentando para procesar cada fichero del dataset.

indice_archivo=0


#Precision para el cálculo del geohash del job de agrupacion

precision=3


#Numero de reducer de nuestro Job.

numero_reducer=10


#Tamanyo del reservoir
#Dependiendo del algoritmo que se vaya a ejecutar, aumentar el tamanyo
#a 7500000 en el caso de inferencia de ciudades seleccionadas
reservoir=500000
```

Precission.

En este Job no vamos a trabajar ya con el dataset de geonames, por lo que no es necesario añadirlo como propiedad del Job.

```
#Patron de busqueda de ficheros en la carpeta de nuestro HDFS definida en el anterior Job como salida.
ruta_inicial_fichero=groupNearJobOut
#Segunda ejecucion
#ruta_inicial_fichero=agrupaPrimeroOut
#Tercera ejecucion
#ruta_inicial_fichero=agrupaSegundoOut

#Patron de ficheros a buscar en el directorio de entrada
patron_ficheros=/part-r-*

#Ruta donde se almacena la salida del Job+indice (concatenado en java).
ruta_salida_job=agrupaPrimeroOut
#Segunda ejecucion
#ruta_salida_job=agrupaSegundoOut
#Tercera ejecucion
#ruta_salida_job=agrupaFinOut

#Indice del directorio a utilizar como entrada y para nombrar la salida
indice_archivo=0

#Precision para el cálculo del geohash del job de agrupacion
precision=5
#Segunda ejecucion
#precision=7
#Tercera ejecucion
#precision=9

#Numero de reducer de nuestro Job.
numero_reducer=10

#Tamanyo del reservoir
#Dependiendo del algoritmo que se vaya a ejecutar, aumentar el tamanyo
#a 7500000 en el caso de inferencia de ciudades seleccionadas
reservoir=500000
```

Carga.

```
#Patron de busqueda de ficheros de entrada de nuestro HDFS.
#No hay indices de archivos.
#Cargamos toda la informacion obtenida de las ejecuciones previas de los
#Job groupNear y precission
ruta_inicial_fichero=agrupaFinOut*/*-r-*

#Numero de reducers
numero_reducer=10

#ELASTICSEARCH

#IP del cluster
ip=localhost
#Puerto de escucha del cluster
port=9200

#Nombre del cluster. Si se varía, variar también en los ficheros
#/config/ElasticSearch/Nodo_Maestro/elasticsearch.yml del proyecto
#config/ElasticSearch/Primer_Nodo_Secundario/elasticsearch.yml
#config/ElasticSearch/Segundo_Nodo_Secundario/elasticsearch.yml
#y cambiarlo en la instancia de ElasticSearch que sea el nodo maestro de
#nuestro cluster
clusterName=InferenciaES

#Nombre del indice que queremos crear en nuestro cluster.
#El proceso de carga en ES está preparado para crear el indice si no
 existe, o borrar y crear de nuevo el indice cada vez que se ejecuta.
#Si se varia el nombre habiendo ejecutado anteriormente el proceso, el
 antiguo indice no se borrara.
index_name=mapony

#Nombre de la coleccion
type_name=proyecto
```

En las configuraciones del cluster de ElasticSearch proporcionadas (`master`, `node1` y `node2`), se han definido además otra serie de parámetros necesarios:

```
# Note, that for development on a local machine, with small indices, it
# usually makes sense to "disable" the distributed features:
#
index.number_of_shards: 3
index.number_of_replicas: 2
```

Estos parámetro sólo han de tenerse en cuenta en las configuraciones de los nodos esclavos (aquellos que van a contener datos).

6. Obtener ElasticSearch.

Navegar a la página de ES (https://www.elastic.co/downloads/elasticsearch), descargar, usar los parámetros de configuración existentes en la carpeta config/cluster del proyecto. Está preparado para generar un clúster de ElasticSearch compuesto por un nodo maestro que no guarda información, y dos nodos esclavos, por lo que una vez descomprimida la carpeta, habrá que realizar dos copias más de la misma.

Copiar el fichero de configuración aportado en el proyecto de para cada instancia de ES que se va a arrancar, y ejecutar

```
./bin/elasticsearch
```

comenzando por el nodo maestro, y después los esclavos.

Esto arrancará el clúster y se mantendrá a la espera de que ejecutemos el Job de carga. Este paso no es necesario realizarlo hasta que tengamos los datos ya procesados.

7. Obtener kibana.

Navegar a la página de ES (https://www.elastic.co/downloads/kibana), descargar,

Descomprimir el archivo en la ruta deseada.

Para arrancar kibana, navegar a la carpeta donde se ha descomprimido, y ejecutar

```
./bin/kibana
```

con el clúster de ElasticSearch levantado.

Para acceder a kibana, en un navegador web escribimos

```
[host_cluster_elastic_search]:5601
```

```
localhost:5601 en nuestro caso.
```

Aquí seleccionaremos el índice que queremos consultar y podemos comenzar a visualizar resultados.

8. Ejecución del procesado de los datos.

Ya hemos visto que cada Job necesita de una serie de ficheros de propiedades para el correcto funcionamiento.

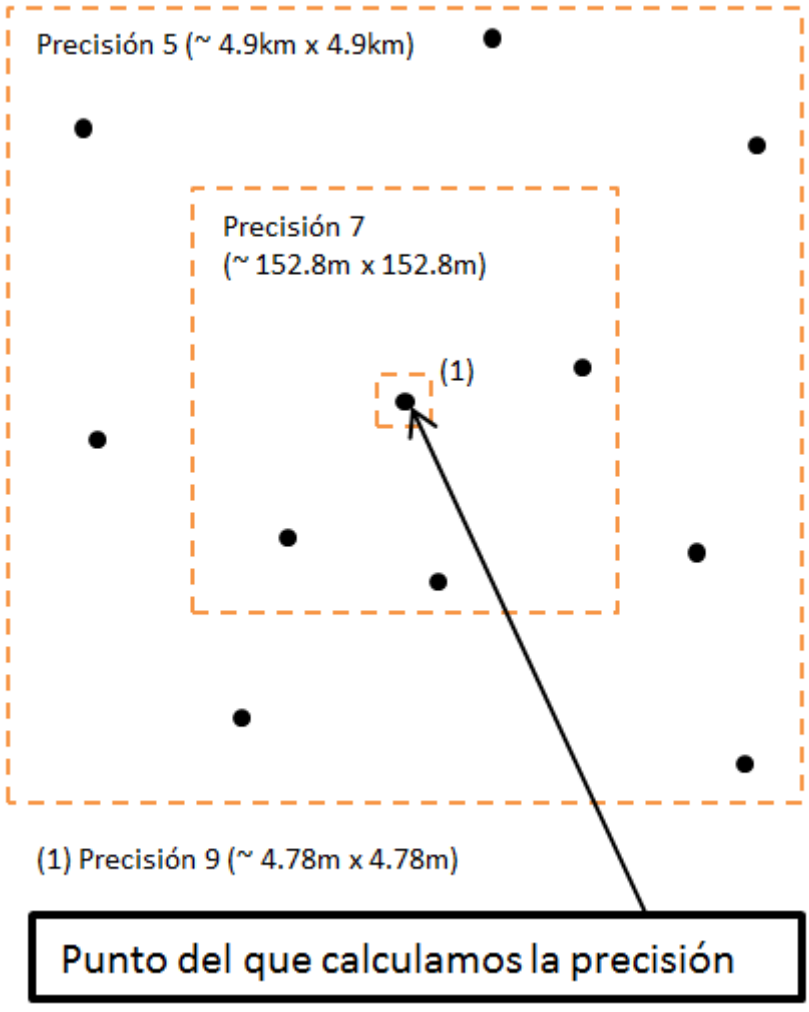
Habiendo preparado adecuadamente cada `properties`, hay que ejecutar el Job groupNear / groupNearCities al menos con un fichero del dataset como parámetro.

Seguidamente, el Job de precission / precissionCities habrá que lanzarlo al menos dos veces, modificando la precisión y las rutas de entrada/salida de datos.

Por defecto, la precisión es 5. En la segunda ejecución, aumentar a 7, y en una deseable tercera ejecución, aumentar a 9.

Se eligen estas precisiones por tratarse de números que representan áreas cuadradas, centradas en un punto.

De tal forma que, partiendo de una precisión 3 (~ 156km x 156km), que habíamos obtenido en el Job de agrupación inicial, vayamos reduciendo el área de búsqueda de elementos cercanos.



Por último, con el clúster de ElasticSearch arrancado, ejecutar el Job load, de carga, definiendo el directorio donde se encuentran los datos de entrada correctamente.

En este último paso, no hace falta definir índices de archivos para procesar. La propiedad de ficheros de entrada leerá en todas las carpetas definidas como salida de la última ejecución del Job precission / precissionCities, procesará los datos y los cargará en el clúster de Elastic Search.

La idea del proceso de carga de datos en Elastic Search es que cuando se lance su ejecución, vamos a insertar datos bien preparados, por lo que se ha decidido que, cada vez que se ejecuta el Job de carga (load), se elimine y se cree nuevamente el índice en el clúster, si es que existía previamente, o bien, lo cree de nuevas si no existiese.

Esto es importante tenerlo en cuenta, si cambiamos los nombres de los índices, por error, o por probar. Quedará información que no nos interesa mantener en el clúster de Elastic Search, que puede con el tiempo llegar a dar problemas de espacio, por ejemplo.