

## Learning to program in Apache spark framework

What did I do?

I took the Lab exercise 3 from the online course “Introduction to Big data with Apache Spark” offered on edx.org. I followed the instructions in the tutorial wrote the code in Python and executed it in three environments.

1. On my machine in local mode
2. On my machine in standalone mode
3. On AWS cluster in standalone mode

### Brief description of the problem

Entity Resolution refers to the task of finding records in a dataset that refer to the same entity across different data sources (e.g., data files, books, websites, databases).

I have two datasets in this problem. The goal is to identify records that refer to the same product in both the datasets. For doing so, I define a cosine similarity metric which returns the similarity between two records from each dataset. I optimized the algorithm to calculate the cosine similarities so that it is scalable and efficient for big data sets.

Once we have the similarities between each pair of records from both the datasets, it is important to determine a threshold similarity. Above this “threshold similarity” we can say that the two records are statistically similar to each other.

To get the true duplicates, we use the gold standard mapping file provided as part of the dataset. And run the results through standard statistical functions of precision recall and F-measure to determine the threshold similarity for this example.

In this experiment the maximum F-measure is about 40% at similarity threshold of 0.26. This means that all the records having similarity greater than 0.26 will be considered similar to each other.

### Input data description

All the data for this assignment has been downloaded from <https://code.google.com/archive/p/metric-learning/>

The three files I worked with:

1. Amazon dataset – contains 1363 products
2. Google dataset - contains 3226 products
3. Gold standard contains 1300 ideally mapped records

### Brief description of algorithm

Each record is treated as vector. The more tokens the two vectors share in common, the closer they will be in space.

Similarity between two vectors =  $\cos\theta = \frac{a \cdot b}{(|a|) \cdot (|b|)}$

Problem breaks down into two parts now:

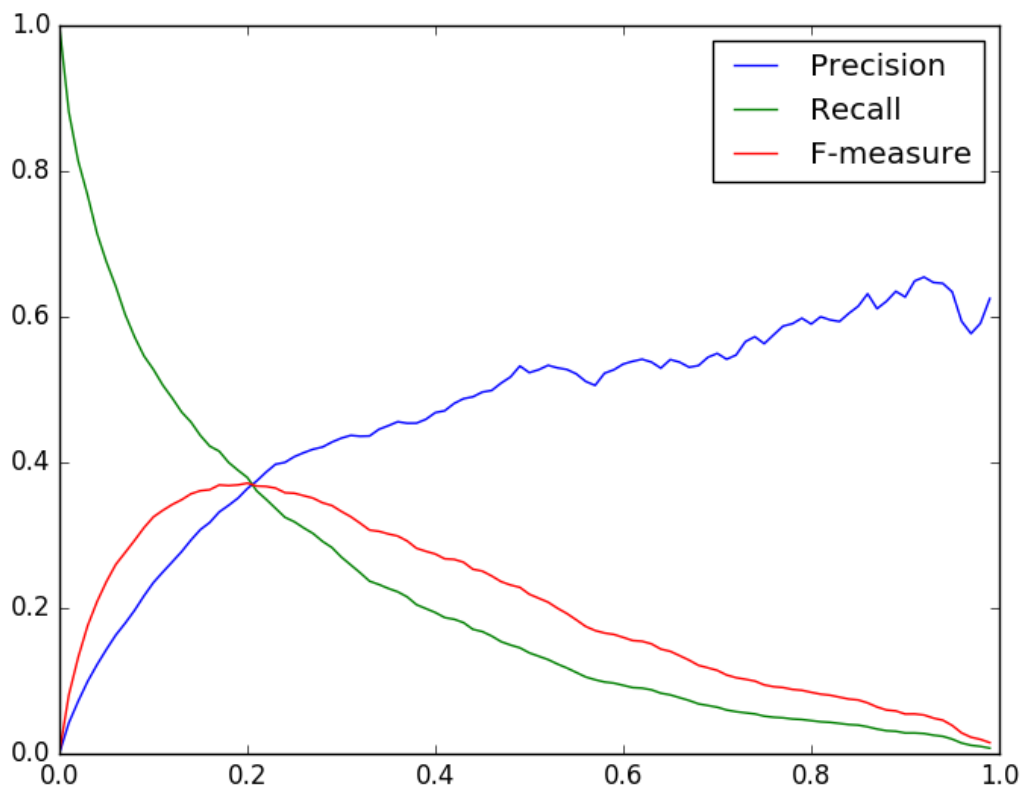
1. Calculating magnitude of the vector in each dimension.
  - Magnitude of vector in each dimension =  $TF \cdot IDF$
  - TF (Token frequency): Local weight
  - IDF: Inverse document frequency: Global weight

TF of a token = # of times the token occurs in the record / Total # of records in the set

IDF of a token = Total # of records in the set / # of records in which the token is present

2. Calculating  $\cos\theta$  to find cosine similarity between the vectors

### Output



## On local machine in stand alone mode

System specifications:

RAM: 8GB

Number of physical cores: 2

Number of logical cores: 4


### *Starting spark in standalone mode on your local machine*

Give the following commands in the bin folder of the directory where you have installed spark

```
sharus-MacBook-Pro:bin sharu$ ./start-master.sh
sharus-MacBook-Pro:bin sharu$ ./start-slave.sh spark://sharus-MacBook-Pro.local:7077
```

These are some screen shots I collected at the web UI while the application was running and after it finished successfully.

---

 **Spark Master at spark://sharus-MacBook-Pro.local:7077**

URL: spark://sharus-MacBook-Pro.local:7077  
REST URL: spark://sharus-MacBook-Pro.local:6066 (cluster mode)  
Alive Workers: 1  
Cores in use: 4 Total, 0 Used  
Memory in use: 7.0 GB Total, 0.0 B Used  
Applications: 0 Running, 1 Completed  
Drivers: 0 Running, 0 Completed  
Status: ALIVE

**Workers**

Worker Id	Address	State	Cores	Memory
<a href="#">worker-20160310142604-10.0.0.122-55272</a>	10.0.0.122:55272	ALIVE	4 (0 Used)	7.0 GB (0.0 B Used)

**Running Applications**

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
----------------	------	-------	-----------------	----------------	------	-------	----------

**Completed Applications**

Application ID	Name	Cores	Memory per Node	Submitted Time	User	State	Duration
<a href="#">app-20160310142800-0000</a>	<a href="#">ER</a>	4	1024.0 MB	2016/03/10 14:28:00	sharu	FINISHED	2.7 min

---

Spark Jobs (?)

Total Uptime: 36 s  
Scheduling Mode: FIFO  
Active Jobs: 1  
Completed Jobs: 12

▶ Event Timeline

Active Jobs (1)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
12	<a href="#">foreach at /Users/sharu/Documents/Sharu/spark/src/ER.py:295</a>	2016/03/10 14:28:12	22 s	1/3	<div><div></div>4/12</div>

Completed Jobs (12)

Job Id	Description	Submitted	Duration	Stages: Succeeded/Total	Tasks (for all stages): Succeeded/Total
11	<a href="#">collectAsMap at /Users/sharu/Documents/Sharu/spark/src/ER.py:283</a>	2016/03/10 14:28:11	0.4 s	1/1	<div><div></div>2/2</div>
10	<a href="#">collectAsMap at /Users/sharu/Documents/Sharu/spark/src/ER.py:282</a>	2016/03/10 14:28:10	0.6 s	1/1	<div><div></div>2/2</div>
9	<a href="#">collectAsMap at /Users/sharu/Documents/Sharu/spark/src/ER.py:281</a>	2016/03/10 14:28:10	0.4 s	1/1	<div><div></div>2/2</div>
8	<a href="#">collectAsMap at /Users/sharu/Documents/Sharu/spark/src/ER.py:280</a>	2016/03/10 14:28:09	0.7 s	1/1	<div><div></div>2/2</div>
7	<a href="#">collectAsMap at /Users/sharu/Documents/Sharu/spark/src/ER.py:264</a>	2016/03/10 14:28:08	0.1 s	1/1 (2 skipped)	<div><div></div>4/4 (8 skipped)</div>
6	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER.py:260</a>	2016/03/10 14:28:08	0.2 s	2/2 (1 skipped)	<div><div></div>8/8 (4 skipped)</div>
5	<a href="#">sortBy at /Users/sharu/Documents/Sharu/spark/src/ER.py:132</a>	2016/03/10 14:28:08	0.2 s	1/1 (1 skipped)	<div><div></div>4/4 (4 skipped)</div>

Executors (2)

Memory: 22.5 MB Used (1060.6 MB Total)  
Disk: 0.0 B Used

Executor ID	Address	RDD Blocks	Storage Memory	Disk Used	Active Tasks	Failed Tasks	Complete Tasks	Total Tasks	Task Time	Input	Shuffle Read	Shuffle Write	Logs
0	10.0.0.122:49385	16	22.5 MB / 530.3 MB	0.0 B	0	0	1288	1288	10.9 m	12.0 MB	0.0 B	357.5 MB	<a href="#">stdout</a> <a href="#">stderr</a>
driver	10.0.0.122:49378	0	0.0 B / 530.3 MB	0.0 B	0	0	0	0	0 ms	0.0 B	0.0 B	0.0 B	

Storage

RDDs

RDD Name	Storage Level	Cached Partitions	Fraction Cached	Size in Memory	Size in ExternalBlockStore	Size on Disk
<a href="#">PythonRDD</a>	Memory Serialized 1x Replicated	4	100%	21.0 MB	0.0 B	0.0 B
<a href="#">PythonRDD</a>	Memory Serialized 1x Replicated	2	100%	40.0 KB	0.0 B	0.0 B
<a href="#">PythonRDD</a>	Memory Serialized 1x Replicated	2	100%	982.9 KB	0.0 B	0.0 B
<a href="#">PythonRDD</a>	Memory Serialized 1x Replicated	2	100%	527.7 KB	0.0 B	0.0 B
<a href="#">PythonRDD</a>	Memory Serialized 1x Replicated	6	100%	966.0 B	0.0 B	0.0 B

## Stages for All Jobs

Completed Stages: 228

### Completed Stages (228)

Stage Id	Description	Submitted	Duration	Tasks: Succeeded/Total	Input	Output	Shuffle Read	Shuffl Write
849	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:17	40 ms	<div>6/6</div>	966.0 B			
845	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	36 ms	<div>6/6</div>	966.0 B			
841	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	37 ms	<div>6/6</div>	966.0 B			
837	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	42 ms	<div>6/6</div>	966.0 B			
833	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	51 ms	<div>6/6</div>	966.0 B			
829	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	41 ms	<div>6/6</div>	966.0 B			
825	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	65 ms	<div>6/6</div>	966.0 B			
821	<a href="#">count at /Users/sharu/Documents/Sharu/spark/src/ER_backup.py:268</a>	2016/03/10 10:49:16	44 ms	<div>6/6</div>	966.0 B			

## On AWS cluster in standalone mode

System specifications:

RAM: 2GB on each instance (1 GB per instance used by the OS)

Number of node: 2 (1 master + 1 slave)

Type of nodes: t2.small

Cost: ~10 cents per hour for this configuration

t2.micro instances are free but not good for us because they have only 1GB RAM. (All of which is used by the OS itself)

Some Remarks:

I used the `./spark-ec2` script in the `spark/ec2` directory. This script launces a cluster on AWS with the number and type of instances you want on your cluster. After setting up the cluster, it also installs hdfs, spark and Yarn for you. All this set up takes about 15-20 minutes. The input data for your applications is always read from `/hdfs` partition on the cluster.

Use the same script `./spark-ec2` with `destroy` action to destroy your cluster.

I will add screenshots for this part soon...