Log-Parsing

Initially take the .log files, example for a log line: (say this n'th log)

081109 203615 148 INFO dfs.DataNode$PacketResponder: PacketResponder 1 for block blk\_38865049064139660 terminating

given log template as: '<Date> <Time> <Pid> <Level> <Component>: <Content>' # HDFS log format

to separate the components of log file using regular expression using pattern matching, example:

**LineId 1(here n=1)**

**Date 081109**

**Time 203615**

**Pid 148 (process id)**

**Level INFO**

**Component dfs.DataNode$PacketResponder**

**Content PacketResponder 1 for block blk\_38865049064139660 terminating**

2 types of log-parsers: (Logparser’s Documentation: https://logparser.readthedocs.io/en/latest/README.html)

- Spell Parser (<https://logparser.readthedocs.io/en/latest/tools/Spell.html>)

- Drain Parser (<https://logparser.readthedocs.io/en/latest/tools/Drain.html>)

reference papers:

Min Du, Feifei Li. Spell: Streaming Parsing of System Event Logs, IEEE International Conference on Data Mining (ICDM), 2016.

Pinjia He, Jieming Zhu, Zibin Zheng, and Michael R. Lyu. Drain: An Online Log Parsing Approach with Fixed Depth Tree, IEEE International Conference on Web Services (ICWS), 2017.

brief description: in the resp links

<https://github.com/logpai/logparser/tree/master/logparser/Spell>

<https://github.com/logpai/logparser/tree/master/logparser/Drain>

out of which we will be using drain parser is it is more optimal (detailed working of these parser have coded as a debugging option under the packages)

finally, we obtain “Event Template” column containing the Log Templates and their resp hashed values under “EventID” column.

EventID: hashlib.md5(template\_str.encode('utf-8')).hexdigest()[0:8]

(Notice we are taking only first 8 digits for simplicity)

MD5 Hash

This hash function accepts sequence of bytes and returns 128-bit hash value, usually used to check data integrity but has security issues. Functions associated:

encode() : Converts the string into bytes to be acceptable by hash function.

digest() : Returns the encoded data in byte format.

hexdigest() : Returns the encoded data in hexadecimal format.

https://www.geeksforgeeks.org/md5-hash-python/

They are also used in sending messages over network for security or storing messages in databases.

There are many hash functions defined in the “hashlib” library in python.