Data science for the datacenter: processing logs with Apache Spark

William Benton
Red Hat Emerging Technology
Challenges of log data
Challenges of log data

SELECT hostname, DATEPART(HH, timestamp) AS hour, COUNT(msg)
FROM LOGS WHERE level='CRIT' AND msg LIKE '%failure%'
GROUP BY hostname, hour
Challenges of log data

SELECT hostname, DATEPART(HH, timestamp) AS hour, COUNT(msg)
FROM LOGS WHERE level='CRIT' AND msg LIKE '%failure%'
GROUP BY hostname, hour
Challenges of log data

postgres
INFO INFO WARN CRIT DEBUG INFO

httpd
GET GET GET POST GET (404)

syslog
WARN INFO WARN INFO INFO INFO

(ca. 2000)
Challenges of log data

(postgres) [INFO][WARN][CRIT][INFO]
(httpd) [GET][GET][GET][POST]
(syslog) [INFO][INFO][INFO][WARN]
(CouchDB) [INFO][INFO][CRIT][INFO]
(httpd) [GET][GET (404)][POST]
(Django) [INFO][INFO][INFO][WARN]

(Cassandra) [INFO][CRIT][INFO][INFO]
(nginx) [GET][POST][PUT][POST]
(Rails) [INFO][INFO][INFO][INFO]
(redis) [INFO][CRIT][INFO][INFO]
(httpd) [PUT (500)][GET][PUT]
(syslog) [INFO][WARN][INFO][INFO]

(haproxy) [INFO][INFO][WARN][DEBUG][CRIT]
(k8s) [WARN][WARN][INFO][INFO][INFO]

(ca. 2016)
Challenges of log data

How many services are generating logs in your datacenter today?
Apache Spark

Spark core (distributed collections, scheduler)

Graph  Query  ML  Streaming

Spark core (distributed collections, scheduler)

ad hoc  Mesos  YARN
DATA INGEST
Collecting log data

Ingesting live log data via rsyslog, logstash, fluentd
Collecting log data

Ingesting live log data via rsyslog, logstash, fluentd

Reconciling log record metadata across sources
Collecting log data

- **Collecting**: Ingesting live log data via rsyslog, logstash, fluentd
- **Normalizing**: Reconciling log record metadata across sources
- **Warehousing**: Storing normalized records in ES indices
Collecting log data

warehousing

Storing normalized records in ES indices
Collecting log data

warehousing

Storing normalized records in ES indices

analysis

cache warehoused data as Parquet files on Gluster volume local to Spark cluster
Schema mediation
Schema mediation
Schema mediation
Schema mediation

timestamp, level, host, IP addresses, message, &c.

rsyslog-style metadata, like app name, facility, &c.
Structured queries at scale

```
logs
  .select("level").distinct
  .as[String].collect
```
Structured queries at scale

```java
logs
   .select("level").distinct
   .as[String].collect
depbug, notice, emerg,
err, warning, crit, info,
severe, alert
```
Structured queries at scale

logs
  .select("level").distinct
  .as[String].collect
debug, notice, emerg,
err, warning, crit, info,
severe, alert

logs
  .groupBy("level", $"rsyslog.app_name")
  .agg(count("level").as("total"))
  .orderBy("total".desc)
Structured queries at scale

```
logs
  .select("level").distinct
  .as[String].collect

logs
  .groupBy("level", "$rsyslog.app_name")
  .agg(count("level").as("total"))
  .orderBy("total").desc
```

<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>INFO</td>
<td>kubelet</td>
<td>17933574</td>
</tr>
<tr>
<td>INFO</td>
<td>kube-proxy</td>
<td>10961117</td>
</tr>
<tr>
<td>ERR</td>
<td>journal</td>
<td>6867921</td>
</tr>
<tr>
<td>INFO</td>
<td>systemd</td>
<td>5184475</td>
</tr>
</tbody>
</table>

...
FEATURE ENGINEERING
From log records to vectors

What does it mean for two log records to be similar?
From log records to vectors

What does it mean for two log records to be similar?

red
green
blue
orange
From log records to vectors

What does it mean for two log records to be similar?

<table>
<thead>
<tr>
<th>Color</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>000</td>
</tr>
<tr>
<td>green</td>
<td>010</td>
</tr>
<tr>
<td>blue</td>
<td>100</td>
</tr>
<tr>
<td>orange</td>
<td>001</td>
</tr>
</tbody>
</table>
From log records to vectors

What does it mean for two log records to be similar?

<table>
<thead>
<tr>
<th>Item</th>
<th>Code</th>
<th>Item</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>000</td>
<td>pancakes</td>
<td></td>
</tr>
<tr>
<td>green</td>
<td>010</td>
<td>waffles</td>
<td></td>
</tr>
<tr>
<td>blue</td>
<td>100</td>
<td>aebliskiver</td>
<td></td>
</tr>
<tr>
<td>orange</td>
<td>001</td>
<td>omelets</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>bacon</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>hash browns</td>
<td></td>
</tr>
</tbody>
</table>
From log records to vectors

What does it mean for two log records to be similar?

<table>
<thead>
<tr>
<th>color</th>
<th>vector</th>
<th>item</th>
<th>vector</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>000</td>
<td>pancakes</td>
<td>10000</td>
</tr>
<tr>
<td>green</td>
<td>010</td>
<td>waffles</td>
<td>01000</td>
</tr>
<tr>
<td>blue</td>
<td>100</td>
<td>aebiskiver</td>
<td>00100</td>
</tr>
<tr>
<td>orange</td>
<td>001</td>
<td>omelets</td>
<td>00001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bacon</td>
<td>00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hash browns</td>
<td>00010</td>
</tr>
</tbody>
</table>
From log records to vectors

What does it mean for two log records to be similar?

<table>
<thead>
<tr>
<th>Color</th>
<th>Code</th>
<th>Item</th>
<th>Code</th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td>000</td>
<td>pancakes</td>
<td>10000</td>
</tr>
<tr>
<td>green</td>
<td>010</td>
<td>waffles</td>
<td>01000</td>
</tr>
<tr>
<td>blue</td>
<td>100</td>
<td>aebliskiver</td>
<td>00100</td>
</tr>
<tr>
<td>orange</td>
<td>001</td>
<td>omelets</td>
<td>00001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bacon</td>
<td>00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hash browns</td>
<td>00010</td>
</tr>
</tbody>
</table>

red pancakes
orange waffles
From log records to vectors

What does it mean for two log records to be similar?

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>red</td>
<td></td>
<td>pancakes</td>
<td>10000</td>
</tr>
<tr>
<td>green</td>
<td></td>
<td>waffles</td>
<td>01000</td>
</tr>
<tr>
<td>blue</td>
<td></td>
<td>aebliskiver</td>
<td>00100</td>
</tr>
<tr>
<td>orange</td>
<td></td>
<td>omelets</td>
<td>00001</td>
</tr>
<tr>
<td></td>
<td></td>
<td>bacon</td>
<td>00000</td>
</tr>
<tr>
<td></td>
<td></td>
<td>hash browns</td>
<td>00010</td>
</tr>
</tbody>
</table>

red pancakes -> 00010000
orange waffles -> 00101000
From log records to vectors

What does it mean for two log records to be similar?
From log records to vectors

What does it mean for two log records to be similar?

```json
{level: INFO, hostname: fred, timestamp: "2016-05-06 00:00:01", ...}
{level: DEBUG, hostname: barney, timestamp: "2016-05-06 00:00:02", ...}
{level: INFO, hostname: fred, timestamp: "2016-05-06 00:00:03", ...}
{level: ERR, hostname: barney, timestamp: "2016-05-06 00:00:03", ...}
{level: ALERT, hostname: wilma, timestamp: "2016-05-06 00:00:03", ...}
```
From log records to vectors

What does it mean for two log records to be similar?

{level: INFO, hostname: fred, timestamp: "2016-05-06 00:00:01", ...}  -> 00000100
{level: DEBUG, hostname: barney, timestamp: "2016-05-06 00:00:02", ...}  -> 00100000
{level: INFO, hostname: fred, timestamp: "2016-05-06 00:00:03", ...}  -> 00000100
{level: ERR, hostname: barney, timestamp: "2016-05-06 00:00:03", ...}  -> 00001000
{level: ALERT, hostname: wilma, timestamp: "2016-05-06 00:00:03", ...}  -> 10000000
Similarity and distance
Similarity and distance
Similarity and distance

\[ \sqrt{(q - p) \cdot (q - p)} \]
Similarity and distance

\[ \sum_{i=1}^{n} |P_i - q_i| \]
Similarity and distance

$$\sum_{i=1}^{n} |P_i - q_i|$$
Similarity and distance
Similarity and distance
Similarity and distance
Similarity and distance
Similarity and distance

\[
\frac{10 | \bar{E} | - | \bar{E} | 3}{10 | \bar{E} |} = .7
\]
Other interesting features

host01

host02

host03
Other interesting features

- **host01**: WARN, INFO
- **host02**: DEBUG, INFO, WARN, DEBUG
- **host03**: WARN, INFO, INFO
Other interesting features

- host01:
  - INFO
  - DEBUG
  - INFO
  - INFO
  - INFO
  - WARN
  - INFO

- host02:
  - WARN
  - INFO
  - INFO
  - WARN
  - INFO
  - INFO
  - INFO
  - WARN
  - INFO

- host03:
  - WARN
  - INFO
  - INFO
  - WARN
  - INFO
  - INFO
  - INFO
  - INFO
  - WARN
Other interesting features

INFO: Everything is great! Just checking in to let you know I’m OK.
Other interesting features

INFO: Everything is great! Just checking in to let you know I’m OK.
CRIT: No requests in last hour; suspending running app containers.
Other interesting features

INFO: Everything is great! Just checking in to let you know I’m OK.
CRIT: No requests in last hour; suspending running app containers.
INFO: Phoenix datacenter is on fire; may not rise from ashes.
NATURAL LANGUAGE and LOG DATA
Introducing word2vec

Madrid
Spain
France
Introducing word2vec

![Diagram showing Madrid, Spain, and France in a 2D space. Madrid is closer to Spain.]
Introducing word2vec
Introducing word2vec
Introducing word2vec
Introducing word2vec

$v("Madrid") - v("Spain") + v("France") \approx v("Paris")$
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?

Apache® Oozie™  Java HotSpot™
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?

Apache® Oozie™

systemd

Java HotSpot™

etcd

ZooKeeper
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?

Apache® Oozie™

systemd

/dev/null

Kernel::exec

Java HotSpot™

OutOfMemoryError

etcd

ZooKeeper
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?
Preprocessing text

What is a word? How are “log words” different from words found in conventional prose?
// assume messages is an RDD of log message texts

val oneletter = new scala.util.matching.Regex("\.*([A-Za-z]).*")

def tokens(s: String, post: String=>String): Seq[String] =
  collapseWhitespace(s)
  .split(" ")
  .map(s => post(stripPunctuation(s)))
  .collect { case token @ oneletter(_) => token }

val tokenSeqs = messages.map(line => tokens(line), identity[String])

val w2v = new org.apache.spark.mllib.feature.Word2Vec

val model = w2v.fit(tokenSeqs)
Insights from log messages
Insights from log messages

nova
Insights from log messages

- nova
- containers
- glance
- images
- instances
- VM

graph showing relationships between log message insights.
Insights from log messages

password

containers

nova

glance

vm

images

instances
Insights from log messages

- password
- publickey
- IPMI
- instances
- nova
- opened
- accepted
- containers
- glance
- vm
- images
Insights from log messages

- password
- nova
- containers
- glance
- images
- instances
- opened
- accepted
- publickey
- IPMI
- vm
- sh
Insights from log messages

- password
- nova
- opened
- accepted
- containers
- glance
- instances
- images
- IPMI
- vm
- /usr/bin/bash
- sh
- AUDIT_SESSION
VISUALIZING STRUCTURE and FINDING OUTLIERS
Multidimensional data
Multidimensional data
Multidimensional data

[4, 7]
Multidimensional data

[4,7]

[2,3,5]
Multidimensional data

[4,7]

[2,3,5]
Multidimensional data

[4,7]

[2,3,5]

[7,1,6,5,12,8,9,2,2,4,7,11,6,1,5]
Multidimensional data

[4,7]  
[2,3,5]  
[7,1,6,5,12,8,9,2,2,4,7,11,6,1,5]
A linear approach: PCA
A linear approach: PCA
A nonlinear approach: t-SNE
A nonlinear approach: t-SNE
A nonlinear approach: t-SNE

\[ p(\mathbb{E} | \mathbb{X}) \approx p(\mathbb{G} | \mathbb{D}) \]
Tree-based approaches
Tree-based approaches

- if orange
  - if red: yes
  - if !red: no
- if !orange
  - if !gray: yes
  - if gray: no

- if !gray
Tree-based approaches
Tree-based approaches

If orange:
  - If red: yes
  - If !red:
    - If !orange: yes
    - If !gray: no
  - If !gray: yes

If red:
  - Yes

If !red:
  - No

If !gray:
  - Yes

If orange:
  - No
Self-organizing maps
Self-organizing maps
Training SOMs
Training SOMs
Training SOMs
Training SOMs
Self-organizing maps
Self-organizing maps
Self-organizing maps
Self-organizing maps
Self-organizing maps
Self-organizing maps
Self-organizing maps
Self-organizing maps
Self-organizing maps
Finding outliers with SOMs
Finding outliers with SOMs
Finding outliers with SOMs
Finding outliers with SOMs
Finding outliers with SOMs
Out of 310 million log records, we identified 0.0012% as outliers.
Thirty most extreme outliers

10 Can not communicate with power supply 2.
9 Power supply 2 failed.
8 Power supply redundancy is lost.
1 Drive A is removed.
1 Can not communicate with power supply 1.
1 Power supply 1 failed.
LESSONS LEARNED and KEY TAKEAWAYS
Spark and ElasticSearch

Data locality is an issue and caching is even more important than when running from local storage.

The best practice is to warehouse ES indices to Parquet files and write Spark jobs against them.
Structured queries in Spark

Always program defensively: mediate schemas, explicitly convert null values, etc.

Use the Dataset API whenever possible to minimize boilerplate and benefit from query planning without (entirely) forsaking type safety.
Natural language tips

Conventional preprocessing pipelines probably won’t get you the best results on log message texts.

Consider the sorts of “words” and word variants you’ll want to recognize with your preprocessing pipeline.

Use an approximate whitelist for unusual words.
Memory and partitioning

Large JVM heaps can lead to appalling GC pauses.

You probably won’t be able to use all of your memory in a single JVM without executor timeouts.

Consider memory budget at the driver when choosing a partitioning strategy.
Feature engineering

Design your feature engineering pipeline so you can translate feature vectors back to factor values.

Favor feature engineering effort over complex or novel learning algorithms.

Prefer approaches that train interpretable models.