DASFAA 2015 Hanoi Tutorial

Scalable Learning Technologies for Big Data Mining

Gerard de Melo, Tsinghua University
http://gerard.demelo.org

Aparna Varde, Montclair State University
http://www.montclair.edu/~vardea/
Big Data

Alibaba: 31 million orders per day! (2014)
Big Data on the Web

400 MILLION TWEETS PER DAY

72 HOURS OF VIDEO UPLOADED PER MINUTE

Source: Coup Media 2013
Big Data on the Web

Source: Coup Media 2013
# Learning from Data

<table>
<thead>
<tr>
<th>Student</th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
</tbody>
</table>
# Learning from Data

<table>
<thead>
<tr>
<th>Student</th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>Student A</td>
<td>30%</td>
<td>20h</td>
<td>?</td>
</tr>
</tbody>
</table>
## Learning from Data

<table>
<thead>
<tr>
<th>Student</th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
<tr>
<td>Student A</td>
<td>30%</td>
<td>20h</td>
<td>?</td>
</tr>
<tr>
<td>Student B</td>
<td>80%</td>
<td>45h</td>
<td>?</td>
</tr>
</tbody>
</table>
Machine Learning

Unsupervised or Supervised Learning

Data with or without labels

Model

Prediction

Labels for Test Data

Probably Spam!

D1

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>D1</td>
<td>0.324</td>
<td>0.739</td>
<td>0.000</td>
<td>0.112</td>
</tr>
</tbody>
</table>
Data Mining

Unsupervised or Supervised Learning

Data with or without labels

Unsupervised or Supervised Analysis

Analysis Results

Visualization

Model

Prediction

Use of new Knowledge

Preprocessing + Feature Engineering

Data Acquisition

Raw Data

Data: 0100101

Data: 0010110

Data: 1110011

World

D1

0.324

0.739

0.000

0.112
Problem with Classic Methods: Scalability
Scaling Up
# Scaling Up: More Features

<table>
<thead>
<tr>
<th>Student</th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
</tbody>
</table>
## Scaling Up: More Features

<table>
<thead>
<tr>
<th>Student</th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
</tbody>
</table>

For example:
- words and phrases mentioned in exam response
- Facebook likes, Websites visited
- user interaction details in online learning

Could be many millions!
## Scaling Up: More Features

<table>
<thead>
<tr>
<th></th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Student 1</strong></td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Student 2</strong></td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Student 3</strong></td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Student 4</strong></td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td><strong>Student 5</strong></td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
</tbody>
</table>

Classic solution: Feature Selection
## Scaling Up: More Features

<table>
<thead>
<tr>
<th>Student</th>
<th>Previous Knowledge</th>
<th>Preparation Time</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>80%</td>
<td>48h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>50%</td>
<td>75h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>95%</td>
<td>24h</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>60%</td>
<td>24h</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>80%</td>
<td>10h</td>
<td>No</td>
</tr>
</tbody>
</table>

Scalable solution: Buckets with sums of original features

## Scaling Up: More Features

<table>
<thead>
<tr>
<th></th>
<th>F0</th>
<th>F1</th>
<th>F2</th>
<th>F3</th>
<th>...</th>
<th>...</th>
<th>Fn</th>
<th>Passed Exam?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student 1</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 2</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 3</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>Yes</td>
</tr>
<tr>
<td>Student 4</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>No</td>
</tr>
<tr>
<td>Student 5</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>No</td>
</tr>
</tbody>
</table>

**Feature Hashing:**

- Use fixed feature dimensionality n.
- Hash original Feature ID (e.g. “Clicked on http://...”) to a bucket number in 0 to n-1.
- Normalize features and use bucket-wise sums.

**Small loss of precision usually trumped by big gains from being able to use more features**.
Scaling Up: More Training Examples

Banko & Brill (2001): Word confusion experiments (e.g. “principal” vs. “principle”)
Scaling Up: More Training Examples

Banko & Brill (2001): Word confusion experiments (e.g. “principal” vs. “principle”)

More Data often trumps better Algorithms

Alon Halevy, Peter Norvig, Fernando Pereira (2009). The Unreasonable Effectiveness of Data
### Results: Linear SVM

$$\ell(\hat{y}, y) = \max\{0, 1 - y\hat{y}\} \quad \lambda = 0.0001$$

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Time</th>
<th>Primal cost</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>SVMLight</td>
<td>23,642 secs</td>
<td>0.2275</td>
<td>6.02%</td>
</tr>
<tr>
<td>SVMPerf</td>
<td>66 secs</td>
<td>0.2278</td>
<td>6.03%</td>
</tr>
<tr>
<td>SGD</td>
<td>1.4 secs</td>
<td>0.2275</td>
<td>6.02%</td>
</tr>
</tbody>
</table>

### Results: Log-Loss Classifier

$$\ell(\hat{y}, y) = \log(1 + \exp(-y\hat{y})) \quad \lambda = 0.00001$$

<table>
<thead>
<tr>
<th>Method</th>
<th>Training Time</th>
<th>Primal cost</th>
<th>Test Error</th>
</tr>
</thead>
<tbody>
<tr>
<td>LibLinear ($\epsilon = 0.01$)</td>
<td>30 secs</td>
<td>0.18907</td>
<td>5.68%</td>
</tr>
<tr>
<td>LibLinear ($\epsilon = 0.001$)</td>
<td>44 secs</td>
<td>0.18890</td>
<td>5.70%</td>
</tr>
<tr>
<td>SGD</td>
<td>2.3 secs</td>
<td>0.18893</td>
<td>5.66%</td>
</tr>
</tbody>
</table>
Background: Stochastic Gradient Descent

- Stochastic nature may help us escape local optima
- Stochastic nature may help us escape local optima

**Improved variants:**
- AdaGrad (Duchi et al. 2011)
- AdaDelta (Zeiler et al. 2012)

Images:
- [Hill climb](http://en.wikipedia.org/wiki/File:Hill_climb.png)
- [Local maximum](http://en.wikipedia.org/wiki/Hill_climbing#mediaviewer/File:Local_maximum.png)
Scaling Up: More Training Examples

Recommended Tool

VowPal Wabbit
By John Langford et al.
http://hunch.net/~vw/
Scaling Up: More Training Examples

Parallelization?

Use lock-free approach to updating weight vector components

(HogWild! by Niu, Recht, et al.)
Labeled Data is expensive!

- Penn Chinese Treebank: 2 years for 4000 sentences
- Adaptation is difficult
  - Wall Street Journal ≠ Novels ≠ Twitter
  - For Speech Recognition, ideally need training data for each domain, voice/accent, microphone, microphone setup, social setting, etc.
Semi-Supervised Learning
Semi-Supervised Learning

- **Goal:** When learning a model, use unlabeled data in addition to labeled data.

- **Example:** Cluster-and-label
  - Run a *clustering* algorithm on labeled and unlabeled data.
  - Assign cluster *majority label* to unlabeled examples of every cluster.

Semi-Supervised Learning

• **Bootstrapping or Self-Training** (e.g. Yarowsky 1995)
  
  – Use **classifier to label** the unlabelled examples
  
  – Add the **labels with the highest confidence** to the training data and re-train
  
  – Repeat
Co-Training (Blum & Mitchell 1998)

- **Given:** multiple (ideally independent) views of the same data (e.g. left context and right context of a word)
- Learn **separate models** for each view
- Allow different views to **teach each other**: Model 1 can generate labels that will be helpful to improve model 2 and vice versa.
Semi-Supervised Learning: Transductive Setting
Semi-Supervised Learning: Transductive Setting

Algorithms:
Label Propagation (Zhu et al. 2003), Adsorption (Baluja et al. 2008), Modified Adsorption (Talukdar et al. 2009)
Distant Supervision

- Sentiment Analysis:
  - Look for Twitter tweets with emoticons like “:), “:(“
  - Remove emoticons. Then use as training data!
Representation Learning to Better Exploit Big Data
Representations

Representations

Inputs Bits: 0011001.....

Note sharing between classes

Images: Marc'Aurelio Ranzato
Massive improvements in image object recognition (human-level?), speech recognition.

Good improvements in NLP and IR-related tasks.

Representations

Inputs Bits: 0011001......

Images: Marc’Aurelio Ranzato
Example

Google's image
Source: Jeff Dean, Google
**Input:** delivered via dendrites from other neurons

**Processing:** Synapses may alter input signals. The cell then combines all input signals

**Output:** If enough activation from inputs, output signal sent through a long cable (“axon”)

Source: Alex Smola
Perceptron

Input: Features

Every feature $f_i$ gets a weight $w_i$.

<table>
<thead>
<tr>
<th>feature</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>dog</td>
<td>7.2</td>
</tr>
<tr>
<td>food</td>
<td>3.4</td>
</tr>
<tr>
<td>bank</td>
<td>-7.3</td>
</tr>
<tr>
<td>delicious</td>
<td>1.5</td>
</tr>
<tr>
<td>train</td>
<td>-4.2</td>
</tr>
</tbody>
</table>
Activation of Neuron
Multiply the feature values of an object $x$ with the feature weights.

$$a(x) = \sum_{i} w_i f_i(x) = w^T f(x)$$
Output of Neuron
Check if activation exceeds a threshold $t = -b$

$$output(x) = g(w^T f(x) + b)$$

- Feature $f_1$ with weight $w_1$
- Feature $f_2$ with weight $w_2$
- Feature $f_3$ with weight $w_3$
- Feature $f_4$ with weight $w_4$

Neuron

E.g., $g$ could return 1 (positive) if positive, -1 otherwise.

E.g., 1 for “spam”, -1 for “not-spam”
**Decision Surfaces**

- **Decision Trees**: Not max-margin, only straight decision surface.
- **Linear Classifiers (Perceptron, SVM)**: Only straight decision surface.
- **Kernel-based Classifiers (Kernel Perceptron, Kernel SVM)**: Multi-Layer Perceptron, any decision surface.

Images: Vibhav Gogate
Deep Learning: Multi-Layer Perceptron

Input Layer  Hidden Layer  Output Layer
Deep Learning: Multi-Layer Perceptron

Input Layer: Feature $f_1$, Feature $f_2$, Feature $f_3$, Feature $f_4$

Hidden Layer: Neuron 1, Neuron 2, Neuron 3

Output Layer: Neuron

Output
Deep Learning: Multi-Layer Perceptron

Input Layer  Hidden Layer  Output Layer

Neuron 1

Neuron 2

Neuron

Output 1

Output 2

Feature $f_1$

Feature $f_2$

Feature $f_3$

Feature $f_4$
Deep Learning: Multi-Layer Perceptron

Input Layer (Feature Extraction)

\[ f(x) \]

Single-Layer:

\[ \text{output}(x) = g(Wf(x) + b) \]

Three-Layer Network:

\[ \text{output}(x) = g_2(W_2g_1(W_1f(x) + b_1) + b_2) \]

Four-Layer Network:

\[ \text{output}(x) = g_3(W_3g_2(W_2g_1(W_1f(x) + b_1) + b_2) + b_3) \]
Deep Learning: Computing the Output

Simply evaluate the output function (for each node, compute an output based on the node inputs)
Deep Learning: Training

Compute error on output, if non-zero, do a stochastic gradient step on the error function to fix it

Backpropagation
The error is propagated back from output nodes towards the input layer
We are interested in the gradient, i.e. the partial derivatives for the output function $z=g(y)$ with respect to all inputs and weights, including those at a deeper part of the network.

Exploit the chain rule to compute the gradient:

$$\frac{\partial z}{\partial x} = \frac{\partial z}{\partial y} \frac{\partial y}{\partial x}$$

Compute error on output, if non-zero, do a stochastic gradient step on the error function to fix it.

Backpropagation

The error is propagated back from output nodes towards the input layer.
DropOut Technique

**Basic Idea**
While training, randomly drop inputs (make the feature zero)

**Effect**
Training on variations of original training data (artificial increase of training data size). Trained network relies less on the existence of specific features.

Reference: Hinton et al. (2012)

Also: Maxout Networks by Goodfellow et al. (2013)
Deep Learning: Convolutional Neural Networks

Reference: Yann LeCun's work

Deep Learning: Recurrent Neural Networks

Source: Bayesian Behavior Lab, Northwestern University
Deep Learning: Recurrent Neural Networks

Then can do backpropagation. Challenge: Vanishing/Exploding gradients

Source: Bayesian Behavior Lab, Northwestern University
Long Short-Term Memory

1. Input
2. Input gate
3. “Remember” gate
4. Output gate

Somewhat complicated, lots of parameters

Source: Bayesian Behavior Lab, Northwestern University
Deep Learning: Long Short Term Memory Networks

Deep LSTMs for Sequence-to-sequence Learning

Suskever et al. 2014 (Google)
French Original:
La dispute fait rage entre les grands constructeurs aéronautiques à propos de la largeur des sièges de la classe touriste sur les vols long-courriers, ouvrant la voie à une confrontation amère lors du salon aéronautique de Dubaï qui a lieu de mois-ci.

LSTM's English Translation:
The dispute is raging between large aircraft manufacturers on the size of the tourist seats on the long-haul flights, leading to a bitter confrontation at the Dubai Airshow in the month of October.

Ground Truth English Translation:
A row has flared up between leading plane makers over the width of tourist-class seats on long-distance flights, setting the tone for a bitter confrontation at this Month's Dubai Airshow.

Suskever et al. 2014 (Google)
Deep Learning: Neural Turing Machines

Source: Bayesian Behavior Lab, Northwestern University
Deep Learning: Neural Turing Machines

Source: Bayesian Behavior Lab, Northwestern University
Deep Learning: Neural Turing Machines

Source: Bayesian Behavior Lab, Northwestern University
Deep Learning: Neural Turing Machines

Read from memory ("blurry")

\[ r_t \leftarrow \sum_i w_t(i) M_t(i), \]

Write to memory ("blurry")

\[ \tilde{M}_t(i) \leftarrow M_{t-1}(i) \left[ 1 - w_t(i)e_t \right], \]
\[ M_t(i) \leftarrow \tilde{M}_t(i) + w_t(i)a_t. \]

Source: Bayesian Behavior Lab, Northwestern University
### Deep Learning: Neural Turing Machines

**Addressing by content (similarity)**

\[
\tilde{w}_t(i) \leftarrow \frac{\exp \left( \beta_t K [k_t, M_t(i)] \right)}{\sum_j \exp \left( \beta_t K [k_t, M_t(j)] \right)}
\]

\[
K[u, v] = \frac{u \cdot v}{||u|| \cdot ||v||}
\]

**Addressing by location (shift)**

\[
\tilde{w}_t(i) \leftarrow \sum_{j=0}^{N-1} \tilde{w}_t^g(j) s_t(i - j) \quad w_t(i) \leftarrow \frac{\tilde{w}_t(i) \gamma_t}{\sum_j \tilde{w}_t(j) \gamma_t}
\]

Source: Bayesian Behavior Lab, Northwestern University
Deep Learning: Neural Turing Machines

Learning to sort!

Vectors for numbers are random
Big Data in Feature Engineering and Representation Learning
Web Semantics: Statistics from Big Data as Features

- Language Models for Autocompletion
Word Segmentation

Source: Wang et al. An Overview of Microsoft Web N-gram Corpus and Applications
Parsing: Ambiguity

- NP Coordination

Source: Bansal & Klein (2011)
They considered running the ad during the Super Bowl.

Source: Bansal & Klein (2011)
Adjective Ordering

- Lapata & Keller (2004): The Web as a Baseline (also: Bergsma et al. 2010)
- “big fat Greek wedding” but not “fat Greek big wedding”

Source: Shane Bergsma
Coreference Resolution

When Obama met Jobs, the president discussed ...
Coreference Resolution

Source: Bansal & Klein 2012
Data Sparsity:
E.g. most words are rare (in the “long tail”)
→ Missing in training data

Solution (Blitzer et al. 2006, Koo & Collins 2008, Huang & Yates 2009, etc.)
- Cluster together similar features
- Use clustered features instead of / in addition to original features
<table>
<thead>
<tr>
<th>488941</th>
<th>britney spears</th>
<th>29 brittent spears</th>
<th>9 brittty spears</th>
</tr>
</thead>
<tbody>
<tr>
<td>40134</td>
<td>brittany spears</td>
<td>29 brittny spears</td>
<td>9 brittny spears</td>
</tr>
<tr>
<td>36315</td>
<td>brittney spears</td>
<td>29 britttny spears</td>
<td>9 brittny spears</td>
</tr>
<tr>
<td>24342</td>
<td>brittany spears</td>
<td>29 btrine spears</td>
<td>9 britn spears</td>
</tr>
<tr>
<td>7331</td>
<td>britny spears</td>
<td>26 birttney spears</td>
<td>9 birt new spears</td>
</tr>
<tr>
<td>6633</td>
<td>brittney spears</td>
<td>26 birttney spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>2696</td>
<td>britteny spears</td>
<td>26 britteny spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1807</td>
<td>briney spears</td>
<td>26 brittenay spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1635</td>
<td>brittny spears</td>
<td>26 britney spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1479</td>
<td>britney spears</td>
<td>26 brittnay spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1479</td>
<td>brittanny spears</td>
<td>26 brittny spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1338</td>
<td>britiny spears</td>
<td>26 britten spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1211</td>
<td>britnet spears</td>
<td>26 britten spears</td>
<td>9 britis spears</td>
</tr>
<tr>
<td>1096</td>
<td>britiney spears</td>
<td>24 birtteny spears</td>
<td>8 britis spears</td>
</tr>
<tr>
<td>991</td>
<td>brittany spears</td>
<td>24 brightney spears</td>
<td>8 brittany spears</td>
</tr>
<tr>
<td>991</td>
<td>brittny spears</td>
<td>24 brightnay spears</td>
<td>8 brittany spears</td>
</tr>
<tr>
<td>811</td>
<td>britthney spears</td>
<td>24 brittante spears</td>
<td>8 breteny spears</td>
</tr>
<tr>
<td>811</td>
<td>britney spears</td>
<td>24 brittney spears</td>
<td>8 brightny spears</td>
</tr>
<tr>
<td>664</td>
<td>birtney spears</td>
<td>24 britni spears</td>
<td>8 brittay spears</td>
</tr>
<tr>
<td>664</td>
<td>britney spears</td>
<td>24 brittini spears</td>
<td>8 brittay spears</td>
</tr>
<tr>
<td>601</td>
<td>bittney spears</td>
<td>24 brittini spears</td>
<td>8 brittay spears</td>
</tr>
<tr>
<td>601</td>
<td>brinty spears</td>
<td>21 britney spears</td>
<td>8 brittley spears</td>
</tr>
<tr>
<td>544</td>
<td>brittaney spears</td>
<td>21 brittney spears</td>
<td>8 britney spears</td>
</tr>
<tr>
<td>544</td>
<td>brittnay spears</td>
<td>21 Britn spears</td>
<td>8 britney spears</td>
</tr>
<tr>
<td>364</td>
<td>britey spears</td>
<td>21 britney spears</td>
<td>8 britnty spears</td>
</tr>
<tr>
<td>364</td>
<td>brittyny spears</td>
<td>21 britani spears</td>
<td>8 brittners spears</td>
</tr>
</tbody>
</table>

Even worse: Arnold Schwarzenegger
Vector Representations

Put words into a vector space (e.g. with d=300 dimensions)
Word Vector Representations

Tomas Mikolov et al.

Available from https://code.google.com/p/word2vec/
Hydroxyl radical

This article needs additional citations for verification. Please help improve this article by adding citations to reliable sources. Unsourced material may be challenged and removed. (May 2010)

The hydroxyl radical, 'HO', is the neutral form of the hydroxide ion (HO⁻). Hydroxyl radicals are highly reactive and consequently short-lived; however, they form an important part of radical chemistry. Most notably hydroxyl radicals are produced from the decomposition of hydroperoxides (ROHO) or, in atmospheric chemistry, by the reaction of excited atomic oxygen with water. It is also an important radical formed in radiation chemistry, since it leads to the formation of hydrogen peroxide and oxygen, which can enhance corrosion and SCC in coolant systems subjected to radioactive environments. Hydroxyl radicals are also produced during UV-light dissociation of H₂O₂ (suggested in 1879) and likely in Fenton chemistry, where trace amounts of reduced transition metals catalyze peroxide-mediated oxidations of organic compounds.

In organic synthesis hydroxyl radicals are most commonly generated by photolysis of 1-Hydroxy-2(1H)-pyridinethione.

The hydroxyl radical is often referred to as the "detergent" of the troposphere because it reacts with many pollutants, often acting as the first step to their removal. It also has an important role in eliminating some greenhouse gases like methane and ozone. The rate of reaction with the hydroxyl radical often determines how long many pollutants last in the atmosphere, if they do not undergo photolysis or are rained out. For instance methane, which reacts relatively slowly with hydroxyl radical, has an average lifetime of 66 years and many CFCs have lifetimes of 80+ years. Pollutants, such as larger hydrocarbons, can have very short average lifetimes of less than a few hours.

The first reaction with many volatile organic compounds (VOCs) is the removal of an hydrogen atom, forming water and an alkyl radical (R').

'HO + RH → H₂O + R'

The alkyl radical will typically react rapidly with oxygen forming a peroxy radical.
Text Simplification

- Exploit **edit history**, especially on Simple English Wikipedia
- “collaborate” → “work together”
  “stands for” → “is the same as”
Answering Questions

IBM's Jeopardy!-winning Watson system
Knowledge Integration
UWN/MENTA

multilingual extension of WordNet for word senses and taxonomical information over 200 languages

www.lexvo.org/uwn/
WebChild: Common-Sense Knowledge

- **WebChild**
  - AAAI 2014
  - WSDM 2014
  - AAAI 2011

Examples:
- **pop-singer-n^1**
  - hasAppearance
  - hot-a^3

- **chili-n^1**
  - has Taste
  - hot-a^9

- **volcano-n^1**
  - hasTemperature
  - hot-a^1
Challenge: From Really Big Data to Real Insights
Big Data Mining in Practice
Scalable Learning Technologies for Big Data Mining

Gerard de Melo (Tsinghua University, Beijing China)
Aparna Varde (Montclair State University, NJ, USA)

DASFAA, Hanoi, Vietnam, April 2015
Cloud Data Analytics & Machine Learning

Dr. Aparna Varde
Cloud Computing

- Internet-based computing - shared resources, software & data provided on demand, like the electricity grid
- Follows a pay-as-you-go model
Several technologies, e.g., MapReduce & Hadoop

MapReduce: Data-parallel programming model for clusters of commodity machines
  • Pioneered by Google
  • Processes 20 PB of data per day

Hadoop: Open-source framework, distributed storage and processing of very large data sets
  • HDFS (Hadoop Distributed File System) for storage
  • MapReduce for processing
  • Developed by Apache
Goals of MapReduce

- **Scalability**
  - To large data volumes
  - Scan 100 TB on 1 node @ 50 MB/s = 24 days
  - Scan on 1000-node cluster = 35 minutes

- **Cost-efficiency**
  - Commodity nodes (cheap, but unreliable)
  - Commodity network (low bandwidth)
  - Automatic fault-tolerance (fewer admins)
  - Easy to use (fewer programmers)
MapReduce Principle

- **Data type**: key-value records

- **Map function**: 
  \[(K_{in}, V_{in}) \rightarrow \text{list}(K_{inter}, V_{inter})\]

- **Reduce function**: 
  \[(K_{inter}, \text{list}(V_{inter})) \rightarrow \text{list}(K_{out}, V_{out})\]
MapReduce Example

Input
- the quick brown fox
- the fox ate the mouse
- how now brown cow

Map
- the, 1
- brown, 1
- fox, 1

Shuffle & Sort
- the, 1
- fox, 1
- the, 1
- how, 1
- now, 1
- brown, 1

Reduce
- quick, 1
- ate, 1
- mouse, 1
- cow, 1

Output
- brown, 2
- fox, 2
- how, 1
- now, 1
- the, 3
- ate, 1
- cow, 1
- mouse, 1
- quick, 1
40 nodes/rack, 1000-4000 nodes in cluster
1 Gbps bandwidth in rack, 8 Gbps out of rack
Node specs (Facebook):
8-16 cores, 32 GB RAM, 8 × 1.5 TB disks, no RAID
Files split into 128MB blocks
Blocks replicated across several data nodes (often 3)
Name node stores metadata (file names, locations, etc)
Optimized for large files, sequential reads
Files are append-only
Hive:
Relational D/B on Hadoop developed at Facebook

Provides SQL-like query language
Knowledge Discovery using Hive

- Supports table partitioning, complex data types, sampling, some query optimization

- These help discover knowledge by various tasks, e.g.,
  - Search for relevant terms
  - Operations such as word count
  - Aggregates like MIN, AVG
Example of SELECT in Hive

/* Find documents of enron table with word frequencies within range of 75 and 80 */

SELECT DISTINCT D.DocID
FROM docword_enron D
WHERE D.count > 75 and D.count < 80 limit 10;

OK
1853...
11578
16653
Time taken: 64.788 seconds
Example of CREATE in Hive

/* Create a view to find the count for WordID=90 and docID=40, for the nips table */

CREATE VIEW Word_Freq
AS SELECT D.DocID, D.WordID, V.word, D.count
FROM docword_Nips D JOIN vocabNips V
ON D.WordID=V.WordID AND D.DocId=40 and D.WordId=90;
OK

Time taken: 1.244 seconds
/* Find documents which use word "rational" from nips table */

SELECT D.DocID,V.word
FROM docword_Nips D JOIN vocabnips V
ON D.wordID=V.wordID and V.word="rational"
LIMIT 10;

OK

434  rational
275  rational
158  rational
....

290  rational
422  rational

Time taken: 98.706 seconds
Example of AVG in Hive

/* Find average frequency of all words in the enron table */

SELECT AVG(count) FROM docWord_enron;

OK

1.728152608060543

Time taken: 68.2 seconds
## Original Hive Query Language (HQL) versus Traditional SQL for Querying

<table>
<thead>
<tr>
<th></th>
<th>STAND-ALONE</th>
<th>CLOUD</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>HQL</td>
<td>MySQL</td>
</tr>
<tr>
<td>CREATE</td>
<td>0.61 sec</td>
<td>4.89 sec (Including indexes)</td>
</tr>
<tr>
<td>LOAD</td>
<td>0.93 sec</td>
<td>21.35 sec</td>
</tr>
<tr>
<td>SELECT</td>
<td>121.525 sec</td>
<td>0.89 sec</td>
</tr>
<tr>
<td>AGGREGATE FUNCTIONS</td>
<td>337.041 sec</td>
<td>1.81 sec</td>
</tr>
</tbody>
</table>

Query Execution Time for HQL & MySQL on big data sets
Similar claims for other SQL packages
Hive v/s SQL for Storage

Max Storage per instance

Server Storage Capacity
## Hive v/s SQL (Contd.)

<table>
<thead>
<tr>
<th></th>
<th>Relational DBMS (MySQL)</th>
<th>Hadoop-Hive</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Cost</strong></td>
<td>Mostly Proprietary - Expensive</td>
<td>Open source – Less Expensive</td>
</tr>
<tr>
<td><strong>Data per query</strong></td>
<td>GBs</td>
<td>PBs</td>
</tr>
<tr>
<td><strong>Data capacity</strong></td>
<td>TB+ (may require sharding)</td>
<td>PB+</td>
</tr>
<tr>
<td><strong>Read/Write</strong></td>
<td>Uses Random Read/Write. Great for speedy indexed lookups</td>
<td>Great for massive full data sequential scans</td>
</tr>
<tr>
<td><strong>Query Language</strong></td>
<td>Deep support for relational semantics</td>
<td>Indirect support for relational semantics</td>
</tr>
<tr>
<td><strong>Support for complex data structures</strong></td>
<td>Indirect</td>
<td>Deep support</td>
</tr>
<tr>
<td><strong>Data loading</strong></td>
<td>Schema-on-write</td>
<td>Schema-on-read</td>
</tr>
<tr>
<td><strong>Latency</strong></td>
<td>Seconds</td>
<td>Minutes to hours</td>
</tr>
<tr>
<td><strong>Update Capabilities</strong></td>
<td>INSERT, UPDATE, and DELETE</td>
<td>INSERT but no UPDATE or DELETE</td>
</tr>
<tr>
<td><strong>Transactions</strong></td>
<td>Deep support</td>
<td>Little or no support</td>
</tr>
</tbody>
</table>
Hive supports rich data types: Map, Array & Struct; Complex types

It supports queries with SQL Filters, Joins, Group By, Order By etc.

Here is when (original) Hive users miss SQL....

- No support in Hive to update data after insert
- No (or little) support in Hive for relational semantics (e.g., ACID)
- No "delete from" command in Hive - only bulk delete is possible
- No concept of primary and foreign keys in Hive
Ensure dataset is already compliant with integrity constraints before load

Ensure that only compliant data rows are loaded SELECT & temporary staging table

Check for referential constraints using Equi-Join and query on those rows that comply
Further Advancements

Providing more power than Hive, Hadoop & MR
Cloudera’s Impala: More efficient SQL-compliant analytic database

Hortonworks’ Stinger: Driving the future of Hive with enterprise SQL at Hadoop scale

Apache’s Mahout: Machine Learning Algorithms for Big Data

Spark: Lightning fast framework for Big Data

MLlib: Machine Learning Library of Spark

MLbase: Platform Base for MLlib in Spark
Fully integrated, state-of-the-art analytic D/B to leverage the flexibility & scalability of Hadoop

Combines benefits
- Hadoop: flexibility, scalability, cost-effectiveness
- SQL: performance, usability, semantics
Impala Architecture

MPP: Massively Parallel Processing
Query Example: NDV Function

- **NDV:** function for counting
  - Table w/ 1 billion rows

- **COUNT(DISTINCT)**
  - precise answer
  - slow for large-scale data

- **NDV()** function
  - approximate result
  - much faster
How much faster are Impala queries?

Hardware Configuration
- Generates less CPU load than Hive
- Typical performance gains: 3x-4x
- Impala cannot go faster than hardware permits!

Query Complexity
- Single-table aggregation queries: less gain
- Queries with at least one join: gains of 7-45X

Main Memory as Cache
- Data accessed by query is in cache, speedup is more
- Typical gains with cache: 20x-90x
Is Impala a replacement for Hive?

- No - Many viable use cases for MR & Hive
  - Long-running data transformation workloads & traditional DW frameworks
  - Complex analytics on limited, structured data sets

- Impala is a complement to the approaches
  - Supports cases with very large data sets
  - Especially to get focused result sets quickly
Drive future of Hive with enterprise SQL at Hadoop scale

3 main objectives

- **Speed**: Sub-second query response times
- **Scale**: From GB to TB & PB
- **SQL**: Transactions & SQL:2011 analytics for Hive
Wider use cases with modifications to data
BEGIN, COMMIT, ROLLBACK for multi-stmt transactions
Sub-second queries with Hive LLAP

- Hybrid engine with LLAP (Live Long and Process)
  - Caching & data reuse across queries
  - Multi-threaded execution
  - High throughput I/O
  - Granular column level security
Common Table Expressions
Sub-queries: correlated & uncorrelated
Rollup, Cube, Standard Aggregates
Inner, Outer, Semi & Cross Joins
Non Equi-Joins
Set Functions: Union, Except & Intersect
Most sub-queries, nested and otherwise
From Traditional Hive to Stinger.next

- **Hive 0.10**
  - BATCH
  - Read-only Data
  - HiveQL
  - MR

- **Hive 0.13**
  - INTERACTIVE
  - Read-only Data
  - Substantial SQL
  - MR, Tez

- **Stinger.next**
  - SUB-SECOND
  - Modify w/ Transactions
  - SQL:2011 Analytics
  - MR, Tez, Spark

**Enterprise SQL at Hadoop Scale**
Data Mining Beyond Querying
Supervised and Unsupervised Learning Algorithms
ML algorithms on distributed frameworks good for mining big data on the cloud

- Supervised Learning: e.g. Classification
- Unsupervised Learning: e.g. Clustering

*The word Mahout means “elephant rider” in Hindi (from India), an interesting analogy 😊*
Algorithms in Mahout

- **Clustering (Unsupervised)**
  - K-means, Fuzzy k-means, Streaming k-means etc.

- **Classification (Supervised)**
  - Random Forest, Naïve Bayes etc.

- **Collaborative Filtering (Semi-Supervised)**
  - Item Based, Matrix Factorization etc.

- **Dimensionality Reduction (For Learning)**
  - SVD, PCA etc.

- **Others**
  - LDA for Topic Models, Sparse TF-IDF Vectors from Text etc.
Example Application: Text Classification - Naïve Bayes & TF-IDF

- **Input:** Big Data from Emails
  - Goal: automatically classify text in various categories

- **Prior to classifying text, TF-IDF applied**
  - Term Frequency – Inverse Document Frequency
  - TF-IDF increases with frequency of word in doc, offset by frequency of word in corpus

- **Naïve Bayes used for classification**
  - Simple classifier using posteriori probability
  - Each attribute is distributed independently of others
Building the Model

**Training Data**

- Historical data with reference decisions:
  - Collected a set of e-mails organized in directories labeled with predictor categories:
    - Mahout
    - Hive
    - Other
  - Stored email as text in HDFS on an EC2 virtual server

**Pre-Process**

- Using Apache Mahout:
  - Convert text files to HDFS Sequential File format
  - Create TF-IDF weighted Sparse Vectors

**Training Algorithm**

- Build and evaluate the model with Mahout’s implementation of Naïve Bayes Classifier

**Model**

- Classification Model which takes as input vectorized text documents and assigns one of three document topics:
  - Mahout
  - Hive
  - Other
Using Model to Classify New Data

New Data

- Store a set of new email documents as text in **HDFS** on an EC2 virtual server
- Pre-process using Apache Mahout’s Libraries

Model

Use the existing model to predict topics for new text files.

Output

For each input document, the model returns one of the following categories:
- Mahout
- Hive
- Other

This was implemented in Java
- With Apache Mahout Libraries and
- Apache Maven to manage dependencies and build the project
- The executable JAR file is submitted with the project documentation
- The program works with data files stored in **HDFS**
Possible uses
- Automatic email sorting
- Automatic news classification
- Topic modeling

Input:

```plaintext
From: Marcin Cylke <mcl.h...@touk.pl>
Sent: Friday, January 18, 2013 5:01 AM
To: user@hive.apache.org
Subject: jdbc connection breaking on ALTER

Hi

I'm experiencing a strange behaviour with hive-0.9.0. Accessing insert statements causes the session to break. Specifically, the first step to reproduce:

alter table xyz add partition (dt='1');
alter table xyz add partition (dt='2');

the alter commands should be successful, if the partition exists call it many times.

Is this a known issue? Is there some kind of workaround?

Regards
Marcin
```
The Spark Framework

- Lightning fast processing for Big Data
- Open-source cluster computing developed in AMPLab at UC Berkeley
- Advanced DAG execution engine for cyclic data flow & in-memory computing
- Very well-suited for large-scale Machine Learning
Spark runs much faster than Hadoop & MR
- 100x faster in memory
- 10x faster on disk
# Data Sorting: Hadoop v/s Spark

<table>
<thead>
<tr>
<th></th>
<th>Hadoop MR Record</th>
<th>Spark Record</th>
<th>Spark 1 PB</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Size</td>
<td>102.5 TB</td>
<td>100 TB</td>
<td>1000 TB</td>
</tr>
<tr>
<td>Elapsed Time</td>
<td>72 mins</td>
<td>23 mins</td>
<td>234 mins</td>
</tr>
<tr>
<td># Nodes</td>
<td>2100</td>
<td>206</td>
<td>190</td>
</tr>
<tr>
<td># Cores</td>
<td>50400 physical</td>
<td>6592 virtualized</td>
<td>6080 virtualized</td>
</tr>
<tr>
<td>Cluster disk throughput</td>
<td>3150 GB/s (est.)</td>
<td>618 GB/s</td>
<td>570 GB/s</td>
</tr>
<tr>
<td>Sort Benchmark Daytona Rules</td>
<td>Yes</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>Network</td>
<td>dedicated data center, 10Gbps</td>
<td>virtualized (EC2) 10Gbps network</td>
<td>virtualized (EC2) 10Gbps network</td>
</tr>
<tr>
<td>Sort rate</td>
<td>1.42 TB/min</td>
<td>4.27 TB/min</td>
<td>4.27 TB/min</td>
</tr>
<tr>
<td>Sort rate/node</td>
<td>0.67 GB/min</td>
<td>20.7 GB/min</td>
<td>22.5 GB/min</td>
</tr>
</tbody>
</table>
 Reasons for Efficiency of Spark

- Uses TimSort (derived from merge-sort & insertion-sort), faster than quick-sort
- Exploits cache locality due to in-memory computing
- Fault-tolerant when scaling, well-designed for failure-recovery
- Deploys power of cloud through enhanced N/W & I/O intensive throughput
Spark Core: Distributed task dispatching, scheduling, basic I/O

Spark SQL: Has SchemaRDD (Resilient Distributed Databases); SQL support with CLI, ODBC/JDBC

Spark Streaming: Uses Spark’s fast scheduling for stream analytics, code written for batch analytics can be used for streams

MLlib: Distributed machine learning framework (10x of Mahout)

Graph X: Distributed graph processing framework with API
MLBase - tools & interfaces to bridge gap b/w operational & investigative ML

Platform to support MLlib - the distributed Machine Learning Library on top of Spark
Components of MLbase

- **MLlib**: Distributed ML library for classification, regression, clustering & collaborative filtering
- **MLI**: API for feature extraction, algorithm development, high-level ML programming abstractions
- **ML Optimizer**: Simplifies ML problems for end users by automating model selection
MLlib Algorithms

- **Classification**
  - Support Vector Machines (SVM), Naive Bayes, decision trees

- **Regression**
  - Linear regression, regression trees

- **Collaborative Filtering**
  - Alternating Least Squares (ALS)

- **Clustering**
  - k-means

- **Optimization**
  - Stochastic gradient descent (SGD), Limited-memory BFGS

- **Dimensionality Reduction**
  - Singular value decomposition (SVD), principal component analysis (PCA)
Classification with Decision Trees: Supervised Learning

- Easily interpretable
- Ensembles are top performers
- Support for categorical variables
- Can handle missing data
- Distributed decision trees
- Scale well to massive datasets
Clustered in MLlib: Unsupervised Learning

- Uses modified version of k-means

- Feature extraction & selection
  - Extraction: lot of time and tools
  - Selection: domain expertise
  - Wrong selection of features: bad quality clusters

- Glassbeam’s SCALAR platform
  - SPL (Semiotic Parsing Language)
  - Makes feature engineering easier & faster
High-D data: Not all features IMP to build model & answer Qs

Many applications: Reduce dimensions before building model

MLlib: 2 algorithms for dimensionality reduction
  - Principal Component Analysis (PCA)
  - Singular Value Decomposition (SVD)
Reducing Data to 2-D
import org.apache.spark.mllib.linalg.Matrix
import org.apache.spark.mllib.linalg.distributed.RowMatrix
import org.apache.spark.mllib.clustering.KMeans
import org.apache.spark.mllib.linalg.Vectors

// Load and parse the data
val data = sc.textFile("sample_data.txt")
val inputData = data.map(s => Vectors.dense(s.split(' ')).map(_.toDouble))

val mat: RowMatrix = new RowMatrix(inputData)

// Compute the top 2 principal components.
val pc: Matrix = mat.computePrincipalComponents(2)

// Cluster the data into two classes using KMeans
val numClusters = 4
val numIterations = 20
val clusters = KMeans.train(pc, numClusters, numIterations)

val newEvent: Vector = Vectors.dense(9.3,1.8)
val clusterID = clusters.predict(newEvent)
print("Event belongs to cluster ") + clusterID + ", ClusterCenter: " +
Future of MLlib

- Grow into unified platform for data scientists
- Reduce time to market with platforms like Glassbeam’s SCALAR for feature engineering
- Include more ML algorithms
- Introduce enhanced filters
- Improve visualization for better performance
Big Data Streams
Processing Streaming Data on the Cloud
Stream-based Alternatives

- **Apache Storm**
  - Reliably process unbounded streams, do for real-time what Hadoop did for batch processing

- **Apache Flink**
  - Fast & reliable large scale data processing engine with batch & stream based alternatives
Integrates queueing & D/B

**Nimbus node**
- Upload computations
- Distribute code on cluster
- Launch workers on cluster
- Monitor computation

**ZooKeeper nodes**
- Coordinate the Storm cluster

**Supervisor nodes**
- Interacts w/ Nimbus through Zookeeper
- Starts & stops workers w/ signals from Nimbus
Tuple: ordered list of elements, e.g., a “4-tuple” (7, 1, 3, 7)

Stream: unbounded sequence of tuples

Spout: source of streams in a computation (e.g. Twitter API)

Bolt: process I/P streams & produce O/P streams to run functions

Topology: overall calculation, as N/W of spouts and bolts
Exploits in-memory data streaming & adds iterative processing into system

Makes system super fast for data-intensive & iterative jobs
Requires few config parameters

Built-in optimizer finds best way to run program

Supports all Hadoop I/O & data types

Runs MR operators unmodified & faster

Reads data from HDFS
Conclusions

Summary and Ongoing Work
MapReduce & Hadoop: Pioneering tech

Hive: SQL-like, good for querying

Impala: Complementary to Hive, overcomes its drawbacks

Stinger: Drives future of Hive w/ advanced SQL semantics

Mahout: ML algorithms (Sup & Unsup)

Spark: Framework more efficient & scalable than Hadoop

MLlib: Machine Learning Library of Spark

MLbase: Platform supporting MLlib

Storm: Stream processing for cloud & big data

Flink: Both stream & batch processing
Recommendations – What to use?

**Store & process Big Data?**
- MR & Hadoop - Classical technologies
- Spark – Very large data, Fast & scalable

**Query over Big Data?**
- Hive – Fundamental, SQL-like
- Impala – More advanced alternative
- Stinger - Making Hive itself better

**ML supervised / unsupervised?**
- Mahout - Cloud based ML for big data
- MLlib - Super large data sets, super fast

**Mine over streaming big data?**
- Only streams – Storm
- Batch & Streams – Flink
Big Data in Academia & Industry

- **Big Data Skills**
  - Cloud Technology
  - Deep Learning
  - Business Perspectives
  - Scientific Domains

- **Salary $100k +**
  - Big Data Programmers
  - Big Data Analysts

- **Univ Programs & concentrations**
  - Data Analytics
  - Data Science
Big Data Initiatives at Montclair State Univ, NJ, USA

- Big Data Concentration being developed in CS Dept
  - http://cs.montclair.edu/
  - Data Mining, Remote Sensing, HCI, Parallel Computing, Bioinformatics, Software Engineering

- Global Education Programs available for visiting and exchange students
  - http://www.montclair.edu/global-education/

- Please contact me for details
Future Research Issues

- Include more **cloud intelligence** in big data analytics.
- Further bridge gap between **Hive & SQL**.
- Add features from standalone ML packages to **Mahout, MLlib ...**
- Extend big data capabilities to focus more on **PB & higher scales**.
- Enhance mining of **big data streams** w/ cloud services & deep learning.
- Address **security & privacy** issues on a deeper level for cloud & big data.
- Build **lucrative applications** w/ scalable technologies for big data.
- Conduct **domain-specific research**, e.g., Cloud & GIS, Cloud & Green Computing.


[27] https://github.com/twitter/scalding


[31] https://storm.apache.org
Contact Information

Gerard de Melo (gdm@demelo.org) [http://gerard.demelo.org]

Aparna Varde (vardea@montclair.edu) [http://www.montclair.edu/~vardea]