Advanced Data Management (CSCI 490/680)

Python

Dr. David Koop
Data to Knowledge and Beyond

[D. Somerville, based on H. McLeod's original]
Data to Knowledge and Beyond

[D. Somerville, based on H. McLeod’s original]
Data to Knowledge and Beyond

[D. Somerville, based on H. McLeod’s original]

Require People
Data to Knowledge

Can computers do this for us?

[D. Somerville, based on H. McLeod's original]
FINDINGS

we got about the future of the data science, the most salient takeaway was how excited our respondents were about the evolution of the field. They cited things in their own practice, how they saw their jobs getting more interesting and less repetitive, all while expressing a real and broad enthusiasm about the value of the work in their organization.

As data science becomes more commonplace and simultaneously a bit demystified, we expect this trend to continue as well. After all, last year's respondents were just as excited about their work (about 79% were “satisfied” or better).

How a Data Scientist Spends Their Day

Here’s where the popular view of data scientists diverges pretty significantly from reality. Generally, we think of data scientists building algorithms, exploring data, and doing predictive analysis. That’s actually not what they spend most of their time doing, however.

As you can see from the chart above, 3 out of every 5 data scientists we surveyed actually spend the most time cleaning and organizing data. You may have heard this referred to as “data wrangling” or compared to digital janitor work. Everything from list verification to removing commas to debugging databases— that time adds up and it adds up immensely. Messy data is by far the more time-consuming aspect of the typical data scientist’s work flow. And nearly 60% said they simply spent too much time doing it.

Data scientist job satisfaction

- Building training sets: 3%
- Cleaning and organizing data: 60%
- Collecting data sets: 19%
- Mining data for patterns: 9%
- Refining algorithms: 4%
- Other: 5%

What data scientists spend the most time doing

How do data scientists spend their time?

[CrowdFlower Data Science Report, 2016]
Finding & Discovering Data (even data you already have!)

DATA WAREHOUSE vs DATA LAKE

- Data is processed and organized into a single schema before being put into the warehouse.
- The analysis is done on the cleansed data in the warehouse.

- Raw and unstructured data goes into a data lake.
- Data is selected and organized as and when needed.

[S. Dewan]
Data Wrangling

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>Transaction Date</td>
<td>Customer Name</td>
<td>Phone Numbers</td>
<td>Address</td>
</tr>
<tr>
<td>2 Wed, 12 Jan 2011</td>
<td>John K. Doe Jr.</td>
<td>(609)-993-3001</td>
<td>2196 184th Ave. NE, Redmond, 98052</td>
</tr>
<tr>
<td>3 Thu, 15 Sep 2011</td>
<td>Mr. Doe, John</td>
<td>609.993.3001 ext 2001</td>
<td>4297 148th Avenue NE, Bellevue, 98007</td>
</tr>
<tr>
<td>4 Mon, 17 Sep 2012</td>
<td>Jane A. Smith</td>
<td>+1-4250013981</td>
<td>2720 N Mesa St, El Paso, 79902, USA</td>
</tr>
<tr>
<td>6 2013-Jan-11 02:27:21</td>
<td>Smith, Jane</td>
<td>tel: 4250013981</td>
<td>4740 N 132nd St Apt 417, Omaha, 68164</td>
</tr>
<tr>
<td>7 2011-Jan-12</td>
<td>Anthony R Von Fange II</td>
<td>650-384-9911</td>
<td>10508 Prairie Ln, Oklahoma City</td>
</tr>
<tr>
<td>8 2010-Dec-24</td>
<td>Mr. Peter Tyson</td>
<td>(405)123-3981</td>
<td>525 1st St, Marysville, WA 95901</td>
</tr>
<tr>
<td>9 9/2/2011</td>
<td>Dan E. Williams</td>
<td>1-650-1234183</td>
<td>211 W Ridge Dr, Waukon,52172</td>
</tr>
<tr>
<td>10 7/11/2012</td>
<td>James Davis Sr.</td>
<td>+1-425-736-9999</td>
<td>1312 Five Mile Rd, Brainerd</td>
</tr>
<tr>
<td>11 2/12/2012</td>
<td>Mr. James J. Davis</td>
<td>425.736.9999 x 9</td>
<td>602 Highland Ave, Shinnston, 26431</td>
</tr>
<tr>
<td>12 3/31/2013</td>
<td>Donald Edward Miller</td>
<td>(206) 309-8381</td>
<td>840 W Star St, Greenville, 27384</td>
</tr>
<tr>
<td>13 6/1/2009 12:01</td>
<td>Miller, Donald</td>
<td>206 309 8381</td>
<td>25571 Elba, Redford, 48239</td>
</tr>
<tr>
<td>14 2/26/2007 18:37</td>
<td>Rajesh Krishnan</td>
<td>206 456 8500 extension 1</td>
<td>539 Co Hwy 48, Sikeston, USA</td>
</tr>
<tr>
<td>15 1/4/2011 14:33</td>
<td>Daniel Chen</td>
<td>425 960 3566</td>
<td>1008 Whitlock Ave NW, Marietta, 30064</td>
</tr>
</tbody>
</table>

Example

API is returned as the top result.

We have built a production-quality data-transformation engine that can leverage rich transformation logic in source code, DLLs, web services, and mapping tables, the transformation logic in these resources will be automatically extracted, and made immediately search-able. The way crawls relevant functions from such domains.

Surajit Chaudhuri (TDE), Kris Ganjam (TDE), Vivek Narasayya (TDE), Xu Chu (TDE), Yue Wang (TDE), Kukjin Lee (TDE) that can leverage rich transformation logic in source code, to allow users to search transformations by examples, a paradigm known as program-by-example (PBE).

Although existing PBE systems such as FlashFill that compose results by examples, a paradigm known as program-by-example allow end-users to search transformations by examples, a paradigm known as program-by-example that we believe are important.

Transform-Data-by-Example (TDE) was designed to be extensible where the key objective is to build a composed program using library CSharpNameParser that can synthesize desired programs using relevant transformation logic from sources such as code, to allow users to search transformations by examples, a paradigm known as program-by-example that we believe are important.

Chaudhuri, Kris Ganjam, Vivek Narasayya, Xu Chu, Yue Wang, Kukjin Lee (TDE) that can leverage rich transformation logic in source code, DLLs, web services, and mapping tables, the transformation logic in these resources will be automatically extracted, and made immediately search-able. The way crawls relevant functions from such domains.

The TDE source code is open source and is available at https://github.com/transform-data-by-example. We believe that this is the first system that can automatically synthesize new programs whose output are consistent with the given examples. Within a few seconds, a ranked list of programs are returned in the right pane. Hovering over the results (shaded in green).

The way crawls relevant functions from such domains. This can be performed ad-hoc analysis using diverse data sets, which however of-

From the table above, we can observe that date strings is non-trivial even for programmers, and the heterogeneity of data in these resources will be automatically extracted, and made immediately search-able. The way crawls relevant functions from such domains.

This is di

There is increasing momentum in the industry towards self-service data transformation. (More details of the system can be extracted from pages on StackOver)

Extensibility.

Our observation is that these domain-specific APIs are rarely used in an enterprise setting, which means that they are not designed to be extensible where the key objective is to build a composed program using library CSharpNameParser that can synthesize desired programs using relevant transformation logic from sources such as code, to allow users to search transformations by examples, a paradigm known as program-by-example that we believe are important.

This data set is obviously not ready for analysis yet – an analyst can synthesize desired programs using relevant transformation logic from sources such as code, to allow users to search transformations by examples, a paradigm known as program-by-example that we believe are important.

We have built an instance of TDE that can leverage rich transformation logic in source code, DLLs, web services, and mapping tables, the transformation logic in these resources will be automatically extracted, and made immediately search-able. The way crawls relevant functions from such domains.

Although existing PBE systems such as FlashFill that compose results by examples, a paradigm known as program-by-example allow end-users to search transformations by examples, a paradigm known as program-by-example that we believe are important.

Transform-Data-by-Example (TDE) was designed to be extensible where the key objective is to build a composed program using library CSharpNameParser that can synthesize desired programs using relevant transformation logic from sources such as code, to allow users to search transformations by examples, a paradigm known as program-by-example that we believe are important.

Chaudhuri, Kris Ganjam, Vivek Narasayya, Xu Chu, Yue Wang, Kukjin Lee (TDE) that can leverage rich transformation logic in source code, DLLs, web services, and mapping tables, the transformation logic in these resources will be automatically extracted, and made immediately search-able. The way crawls relevant functions from such domains.
Data Cleaning/Standardization (Aliases)

- 'google brain resident': 'google',
- 'google brain': 'google',
- 'google inc': 'google',
- 'google inc.': 'google',
- 'google research nyc': 'google',
- 'google research': 'google',
- 'google, inc.': 'google',
- 'deepmind @ google': 'deepmind',
- 'deepmind technologies': 'deepmind',
- 'google deepmind': 'deepmind',
- 'ibm research - china': 'ibm',
- 'ibm research': 'ibm',
- 'ibm research, ny': 'ibm',
- 'ibm research, usa': 'ibm',
- 'ibm t.j. watson research center': 'ibm',
- 'ibm t.j. watson research': 'ibm',
- 'ibm t.j. watson research center': 'ibm',
- 'ibm t.j. watson research center': 'ibm',
- 'ibm thomas j. watson research center': 'ibm',
- 'ibm tj watson research center': 'ibm',
- 'microsoft research cambridge': 'microsoft',
- 'microsoft research india': 'microsoft',
- 'microsoft research maluuba': 'microsoft',
- 'microsoft research new england': 'microsoft',
- 'microsoft research': 'microsoft',
- 'microsoft research, redmond, w': 'microsoft',
- 'microsoft research, redmond, wa': 'microsoft',
- 'mililcosearch': 'microsoft',

[ICML, NIPS, EMNLP, NAACL, EACL, ACL]

Georgia Institute of Technology
Chinese Academy of Sciences
Harbin Institute of Technology
Peking University
Tsinghua University
Cornell University
University College London
University of Toronto
University of Maryland
INRIA
University of Pennsylvania
Princeton University
UT Austin
Johns Hopkins University
University of Cambridge
University of Edinburgh
IBM
University of Washington
Massachusetts Institute of Technology
UC Berkeley
Columbia University
Google
Stanford University
Microsoft
Carnegie Mellon University

[NLP Publishing Stats, M. Rei & R. Allen]
Data Integration

- **Google Thinks I’m Dead** (I know otherwise.) [R. Abrams, NYTimes, 2017]
- **Not only Google, but also Alexa:**
  - "Alexa replies that Rachel Abrams is a sprinter from the Northern Mariana Islands (which is true of someone else)."
  - "He asks if Rachel Abrams is deceased, and Alexa responds yes, citing information in the Knowledge Graph panel."
Data Storage

SQL DATABASES

NoSQL DATABASES

Relational

Column

Graph

Key-Value

Document
Provenance and Reproducibility

Data Management

Visualization

Computation

Paper

DATA

DATA
Provenance and Reproducibility

DATA

Data Management

Computation

Visualization

Paper

Provenance

Our research has been funded by the National Science Foundation.
About this course

- Course web page is authoritative:
  - http://faculty.cs.niu.edu/~dakoop/cs680-2021sp
  - Schedule, Readings, Assignments will be posted online
  - Check the web site before emailing me

- Course is meant to be more "cutting edge"
  - Still focus on building skills related to data management
  - Tune into current research and tools

- Requires student participation: readings and discussions

- Exam Dates: Feb. 17, Mar. 29, and Final on Apr. 26
In this Notebook we explore the Lorenz system of differential equations:

\[
\begin{align*}
\dot{x} &= \sigma(y - x) \\
\dot{y} &= x\rho - y - xz \\
\dot{z} &= -\beta z + xy
\end{align*}
\]

Let's call the function once to view the solutions. For this set of parameters, we see the trajectories swirling around two points, called attractors.

```python
In [4]: from lorenz import solve_lorenz
t, x, z = solve_lorenz(N=10)
```
JupyterLab

- An interactive, configurable programming environment
- Supports many activities including notebooks
- Runs in your web browser
- Notebooks:
  - Originally designed for Python
  - Supports other languages, too
  - Displays results (even interactive maps) inline
  - You decide how to divide code into executable cells
  - Shift+Enter to execute a cell
Installing Python & JupyterLab

- [www.anaconda.com/download/](http://www.anaconda.com/download/)
- Anaconda has Jupyter Lab
- Use Python 3.8 version (*not* 2.7)
- Anaconda Navigator
  - GUI application for managing Python environment
  - Can install packages
  - Can start JupyterLab
- Can also use the shell to do this:
  - `$ jupyter lab`
  - `$ conda install <pkg_name>`
JupyterLab Notebook Tips

• Starts with a directory view
• Create new notebooks using the Launcher (+ icon on the left)
  - New notebooks have the name "Untitled"
  - File → Rename Notebook… (or right-click) to change the name
• Save a notebook using the command under the File menu
• Shutting down the notebook requires quitting the kernel
  - Web browser is interface to display code and results
  - Kernel actually runs the code: may see messages in a console/terminal window
  - Closing the browser window does not stop Jupyter
JupyterLab Notebooks

• Open a notebook using the left panel like you would in a desktop view
• Past results are displayed—does not mean they are loaded in memory
• Use "Run All" or "Run All Above" to re-execute past work
  - If you shut down the kernel, all of the data and variables you defined need to be redefined (so you need to re-run all)
  - **Watch Out—Order Matters**: If you went back and re-executed cells in a different order than they are shown, doing "Run All" may not produce the same results!
• Edit mode (green) versus Command mode (blue == Be Careful)
JupyterLab Notebooks

• Can write code or plain text (can be styled Markdown)
  - Choose the type of cell using the dropdown menu
• Cells break up your code, but all data is **global**
  - Defining a variable `a` in one cell means it is available in **any** other cell
  - This includes cells above the cell `a` was defined in!
• Remember **Shift+Enter** to execute
• Enter just adds a new line
• Use `?<function_name>` for help
• Use Tab for **auto-complete** or suggestions
• Tab also indents, and **Shift+Tab** unindents
JupyterLab Outputs

- **stdout**: where print commands go
- **stderr**: where error messages go
- **display**: special output channel used to show rich outputs
- **output**: same as display but used to display the value of the last line of a cell
Other JupyterLab Features

• Terminal
  - Similar to what you see on turing/hopper but for your local machine

• File Viewers
  - CSV
  - Plugins available

• Console
  - Can be linked to notebooks
JupyterLab Documentation

- JupyterLab Tutorial Video
- JupyterLab Documentation
Jupyter Notebook

- Original Notebook Interface
- Just notebooks
- Same rich representations and text
- Same cell structure
- Same notebook files .ipynb
- Web-based
Python

- Started in December 1989 by Guido van Rossum
- “Python has surpassed Java as the top language used to introduce U.S. students to programming…” (ComputerWorld, 2014)
- Python and R are the two top languages for data science
- High-level, interpreted language
- Supports multiple paradigms (OOP, procedural, imperative)
- Help programmers write **readable** code, Use less code to do more
- Lots of libraries for python
  - Designed to be extensible
  - Easy to wrap code from other languages like C/C++
- Open-source with a large, passionate community
Learning Python Resources

- Python for Programmers
- http://www.pythontutor.com
- https://www.python-course.eu
- https://software-carpentry.org/lessons/
Python Compared to C++ and Java

• Dynamic Typing
  - A variable does not have a fixed type
  - Example: `a = 1; a = "abc"

• Indentation
  - Braces define blocks in Java, good style is to indent but not required
  - Indentation is critical in Python
    ```python
    z = 20
    if x > 0:
      if y > 0:
        z = 100
    else:
      z = 10
    ```
In-Class Notebook

• Try out the examples from the following slides:
  - Download
  - View
Print function

- `print("Hello World")`
- Can also print variables:
  ```
  name = "Jane"
  print("Hello,", name)
  ```
Python Variables and Types

• No type declaration necessary
• Variables are names, not memory locations
  
  ```python
  a = 0
  a = "abc"
  a = 3.14159
  ```

• Don't worry about types, but think about types
• Strings are a type
• Integers are as big as you want them
• Floats can hold large numbers, too (double-precision)
Python Math and String "Math"

- Standard Operators: +, -, *, /, %
- Division "does what you want" (new in v3)
  - 5 / 2 = 2.5
  - 5 // 2 = 2 # use // for integer division
- Shortcuts: +=, -=, *=
- No ++, --
- Exponentiation (Power): **
- Order of operations and parentheses: (4 - 3 - 1 vs. 4 - (3 - 1))
- "abc" + "def"
- "abc" * 3
Python Strings

• Strings can be delimited by single or double quotes
  - "abc" and 'abc' are exactly the same thing
  - Easier use of quotes in strings: "Joe's" or 'He said "Stop!"'

• String concatenation: "abc" + "def"

• Repetition: "abc" * 3

• Special characters: \n \t like Java/C++
Python Strings

• Indexing:
  
a = "abcdef"
  a[0]

• Slicing: a[1:3]

• Format:
  
  name = "Jane"
  print("Hello, {}".format(name))

  - or
  
  print(f"Hello, {name}"")
Loops

• while <condition>:
  <indented block>
  # end of while block (indentation done)

• Remember the colon!

• a = 5
  while a > 0:
    print(a)
    a -= 2

• a > 0 is the condition

• Python has standard boolean operators (<, >, <=, >=, ==, !=)
  - What does a boolean operation return?
  - Linking boolean comparisons (and, or)
Conditionals

- **if, else**
  - Again, indentation is required
- **elif**
  - Shorthand for `else: if:
- **Same type of boolean expressions (and or)**
break and continue

- **break** stops the execution of the loop
- **continue** skips the rest of the loop and goes to the next iteration

```python
• a = 7
  while a > 0:
    a -= 2
    if a < 4:
      break
  print(a)

• a = 7
  while a > 0:
    a -= 2
    if a < 4 and a > 2:
      continue
    print(a)
```
True and False

• True and False (captialized) are defined values in Python
• \( v == 0 \) will evaluate to either True or False
Functions

• Calling functions is as expected:
  
  \texttt{mul(2,3)} \# \text{computes} 2 \times 3 \text{ (mul from operator package)}

  - Values passed to the function are parameters
  
  - May be variables!
    
    \begin{verbatim}
    a = 5
    b = 7
    mul(a,b)
    \end{verbatim}

• \texttt{print} is a function
  
  \begin{verbatim}
  print("This line doesn't end.", end=" ")
  print("See it continues")
  \end{verbatim}

  - \texttt{end} is also a parameter, but this has a different syntax (keyword argument!)
Defining Functions

• `def` keyword

• Arguments have names but **no types**
  ```python
def hello(name):
    print(f"Hello {name}\n")
  ```

• Can have defaults:
  ```python
def hello(name="Jane Doe"):
    print(f"Hello {name}\n")
  ```

• With defaults, we can skip the parameter: `hello()` or `hello("John")`

• Also can pick and choose arguments:
  ```python
def hello(name1="Joe", name2="Jane"):  
    print(f"Hello {name1} and {name2}\n")
  hello(name2="Mary")
  ```
Return statement

• Return statement gives back a value:
  ```python
def mul(a, b):
    return a * b
  ```

• Variables changed in the function won't be updated:
  ```python
def increment(a):
    a += 1
    return a

b = 12
print(b, increment(b))
```
Python Containers

• Container: store more than one value
• Mutable versus immutable: Can we update the container?
  - Yes → mutable
  - No → immutable
  - Lists are mutable, tuples are immutable
• Lists and tuples may contain values of different types:
  • List: [1,"abc",12.34]
  • Tuple: (1, "abc", 12.34)
• You can also put functions in containers!
  • len function: number of items: len(l)
Indexing and Slicing

• Just like with strings

• Indexing:
  - Where do we start counting?
  - Use brackets [ ] to retrieve one value
  - Can use negative values (count from the end)

• Slicing:
  - Use brackets plus a colon to retrieve multiple values:
    \[<\text{start}>:<\text{end}>\]
  - Returns a new list \( b = a[::] \)
  - Don't need to specify the beginning or end
Tuples

- months = ('January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October', 'November', 'December')

- Useful when you know you're not going to change the contents or add or delete values

- Can index and slice

- Also, can create new tuples from existing ones:
  - t = (1, 2, 3)
  - u = (4, 5, 6)
  - v = t + u # v points to a new object
  - t += u # t is a new object
Modifying Lists

• Add to a list l:
  - `l.append(v)`: add one value (v) to the end of the list
  - `l.extend(vlist)`: add multiple values (vlist) to the end of l
  - `l.insert(i, v)`: add one value (v) at index i

• Remove from a list l:
  - `del l[i]`: deletes the value at index i
  - `l.pop(i)`: removes the value at index i (and returns it)
  - `l.remove(v)`: removes the first occurrence of value v (careful!)

• Changing an entry:
  - `l[i] = v`: changes the value at index i to v (Watch out for IndexError!)
Modifying a list

- \( v = [1, 2, 3] \)
  \( w = [4, 5, 6] \)

- \( x = v + w \) # \( x \) is a new list [1, 2, 3, 4, 5, 6]
- \( v.extend(w) \) # \( v \) is mutated to [1, 2, 3, 4, 5, 6]
- \( v += w \) # \( v \) is mutated to [1, 2, 3, 4, 5, 6]
- \( v.append(w) \) # \( v \) is mutated to [1, 2, 3, [4, 5, 6]]
- \( x = v + 4 \) # error
- \( v += 4 \) # error
- \( v += [4] \) # \( v \) is mutated to [1, 2, 3, 4]
in: Checking for a value

• The `in` operator:
  - `'a' in l`
  - `'a' not in l`
• Not very fast for lists
For loops

- Used much more frequently than while loops
- Is actually a "for-each" type of loop
- In Java, this is:
  ```java
  for (String item : someList) {
    System.out.println(item);
  }
  ```
- In Python, this is:
  ```python
  for item in someList:
    print(item)
  ```
- Grabs each element of `someList` in order and puts it into `item`
- Be careful modifying container in a for loop! (e.g. `someList.append(new_item)`)

D. Koop, CSCI 680/490, Spring 2021
What about counting?

• In C++:
  ```cpp
  for(int i = 0; i < 100; i++) {
    cout << i << endl;
  }
  ```

• In Python:
  ```python
  for i in range(0,100): # or range(100)
    print(i)
  ```

• range(100) vs. list(range(100))

• What about only even integers?