Lecture 11: CRISP-DM, Graph mining, and NLP

Meelis Kull

meelis.kull@ut.ee

Autumn 2022
✓ Sample vs population
✓ Example task with red and black cards
✓ Statistical terminology
✓ Permutation test and hypergeometric test
✓ Histogram on a sample vs population
✓ More statistical terminology
Lecture 11 – CRISP-DM, Graphs, NLP

- Demo: Data science mini-project
- CRISP-DM: cross-industrial standard process for data mining
- Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks
- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
Lecture 11 – CRISP-DM, Graphs, NLP

- **Demo: Data science mini-project**
- CRISP-DM: cross-industrial standard process for data mining
- Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks
- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
Warnings before the demo

• You will see **many ugly details** about file formats etc
  – This is intended! Similar problems are likely to happen in many data science projects
  – I hope it prepares a bit for when you do your own project
  – Uncommented Python code
  – I hope it shows you roughly how many lines of code are needed and where the complicated parts are
Demo: Data science mini-project
Data from clickers (1)

- Let’s analyse the data from previous lecture!
- What are my overall goals in this analysis?
  - Demonstrate a data science mini-project
  - Where are the problems in understanding?
  - Which questions attract less answers?
  - How many people stopped answering?
  - How many people started later?
  - Identify potential technical problems with clickers
Data from clickers (2)

• What data are available?
  – Public:
    • DS2018_lecture_01_intro_part_1.pdf
    • DS2018_lecture_01_intro_part_2.pdf
  – Private:
    • lecture_01_results_2018.xlsm
    • lecture_01_results_detail_2018.csv
  – No documentation with the data
Is it a good idea to use clickers in data science lectures?

A. Absolutely!
B. Probably good
C. Not sure yet
D. Probably bad
E. Bad idea
Is it a good idea to use clickers in data science lectures?

A. Absolutely!
B. Probably good
C. Not sure yet
D. Probably bad
E. Bad idea

Hard to programmatically extract information from PDF
### 2. Please tell me about yourself (Multiple Choice)

<table>
<thead>
<tr>
<th></th>
<th>Percent</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st year master student</td>
<td>21.88%</td>
<td>14</td>
</tr>
<tr>
<td>2nd year master student</td>
<td>23.44%</td>
<td>15</td>
</tr>
<tr>
<td>PhD student</td>
<td>7.81%</td>
<td>5</td>
</tr>
<tr>
<td>Bachelor student</td>
<td>43.75%</td>
<td>28</td>
</tr>
<tr>
<td>Other</td>
<td>3.13%</td>
<td>2</td>
</tr>
<tr>
<td><strong>Totals</strong></td>
<td><strong>100%</strong></td>
<td><strong>64</strong></td>
</tr>
</tbody>
</table>
2. Please tell me about yourself (Multiple Choice)

<table>
<thead>
<tr>
<th>Responses</th>
<th>Percent</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st year master student</td>
<td>21.88%</td>
<td>14</td>
</tr>
<tr>
<td>2nd year master student</td>
<td>23.44%</td>
<td>15</td>
</tr>
<tr>
<td>PhD student</td>
<td>7.81%</td>
<td>5</td>
</tr>
<tr>
<td>Bachelor student</td>
<td>43.75%</td>
<td>28</td>
</tr>
<tr>
<td>Other</td>
<td>3.13%</td>
<td>2</td>
</tr>
<tr>
<td>Totals</td>
<td>100%</td>
<td>64</td>
</tr>
</tbody>
</table>

More information than in PDF: both percents and counts
Results by Participant

Session does not contain standards

<table>
<thead>
<tr>
<th>Name</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>User Id</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Responding Device</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>396916</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Have you used classroom answering systems in any courses yet?</td>
<td>A. Yes, the same clickers!</td>
</tr>
</tbody>
</table>
Results by Participant

Session does not contain standards

Name
-

User Id
-

Responding Device
396916

<table>
<thead>
<tr>
<th>Question</th>
<th>Response</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. How many device IDs are in the sheet?</td>
<td>Name clickers!</td>
</tr>
</tbody>
</table>

Device ID – always 6-digit? Decimal? Hexadecimal?
### Results Detail

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
<th>Q10</th>
<th>Total Points</th>
<th>Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer Key</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>A</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1.00</td>
<td>100.00%</td>
</tr>
<tr>
<td>3956916</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>A</td>
<td>B</td>
<td>E</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>0.00</td>
<td>0.00%</td>
</tr>
<tr>
<td>398884</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>1.00</td>
<td>100.00%</td>
</tr>
<tr>
<td>6430F1</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>1.00</td>
<td>100.00%</td>
</tr>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>A</td>
<td>A</td>
<td>A</td>
<td>1.00</td>
<td>100.00%</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
<td>E</td>
<td>J</td>
<td>A</td>
<td>-</td>
<td>1.00</td>
<td>100.00%</td>
</tr>
<tr>
<td>643174</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>-</td>
<td>B</td>
<td>C</td>
<td>-</td>
<td>-</td>
<td>A</td>
<td>B</td>
<td>0.00</td>
<td>0.00%</td>
</tr>
</tbody>
</table>
### Results Detail

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer Key</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>396916</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>39B884</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>6430F1</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>643174</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>-</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
## Results Detail

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Answer Key</strong></td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>396916</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>39B88E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>6430F1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>3D762E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>643174</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>-</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>Device ID</td>
<td>Q1</td>
<td>Q2</td>
<td>Q3</td>
<td>Q4</td>
<td>Q5</td>
<td>Q6</td>
</tr>
<tr>
<td>-----------</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
<td>----</td>
</tr>
<tr>
<td>Answer Key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>396916</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>39B85</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>64300</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>643174</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td>-</td>
<td>B</td>
<td>C</td>
</tr>
</tbody>
</table>
### Results Detail

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer Key</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>396916</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>B</td>
</tr>
<tr>
<td>39B884</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>A</td>
</tr>
<tr>
<td>6430F1</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td></td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td></td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td></td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>643174</td>
<td>A</td>
<td>D</td>
<td>B</td>
<td></td>
<td>-</td>
<td>B</td>
</tr>
</tbody>
</table>

'-' seems to denote that the person did not answer this question.
Actually Q4 is the same as Q5 (I looked up the question text and it was identical). It must be the failed attempt where I closed the poll immediately.
# Results Detail

<table>
<thead>
<tr>
<th>Device ID</th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
</tr>
</thead>
<tbody>
<tr>
<td>Answer Key</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>A</td>
</tr>
<tr>
<td>396916</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>A</td>
<td>B</td>
</tr>
<tr>
<td>39B884</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>6430F1</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>-</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>643174</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>-</td>
<td>A</td>
<td>A</td>
</tr>
</tbody>
</table>

Green and red seem to stand for correct and wrong answer, white if none considered correct.
<table>
<thead>
<tr>
<th>Session Name: 05-09-2018, 21-40</th>
</tr>
</thead>
<tbody>
<tr>
<td>Date Created: 05.09.2018 16:09:10, Active Participants: 66 of 66</td>
</tr>
<tr>
<td>Average Score: 74,24%, Questions: 10</td>
</tr>
</tbody>
</table>

Results Detail

<table>
<thead>
<tr>
<th>Device ID, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Total Points, Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>396916, A, D, A, -, A, B, E, A, A, A, 0.00, 0.00%</td>
</tr>
<tr>
<td>39B884, A, D, A, -, B, A, E, A, B, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>6430F1, A, B, B, -, A, A, E, A, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>3D762E, A, D, A, -, B, A, E, A, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>3D766B, A, B, B, -, A, A, E, J, A, -, 1.00, 100.00%</td>
</tr>
<tr>
<td>643174, A, D, B, -, B, C, -, -, A, B, 0.00, 0.00%</td>
</tr>
<tr>
<td>398C81, A, D, A, -, B, A, E, I, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>3D75F9, A, D, B, -, B, A, E, I, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>64310B, A, B, A, -, B, A, E, A, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>3D7681, C, B, B, -, C, A, E, F, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>643130, A, C, B, -, B, -, E, J, A, A, 0.00, 0.00%</td>
</tr>
<tr>
<td>3D764A, A, A, C, -, B, A, E, G, A, A, 1.00, 100.00%</td>
</tr>
<tr>
<td>3968AA, B, A, B, -, B, A, E, A, E, D, 1.00, 100.00%</td>
</tr>
<tr>
<td>Device ID</td>
</tr>
<tr>
<td>-----------</td>
</tr>
<tr>
<td>396916</td>
</tr>
<tr>
<td>39B884</td>
</tr>
<tr>
<td>726150</td>
</tr>
<tr>
<td>643130</td>
</tr>
<tr>
<td>3D764A</td>
</tr>
<tr>
<td>3968AA</td>
</tr>
</tbody>
</table>

**Empty column at the beginning (before the comma)**
Session Name: 05-09-2018, 21-40
Date Created: 05.09.2018 16:09:10, Active Participants: 66 of 66
Average Score: 74.24%, Questions: 10

Results Detail
Device ID, Q1, Q2, Q3, Q4, Q5, Q6, Q7, Q8, Q9, Q10, Total Points, Score
Answer Key, -, -, -, -, -, -, -, 1.00, 100.00%,
396916, A, D, A, -, A, B, E, A, A, A, 0.00, 0.00%  
39B884, A, D, A, -, B, A, E, A, B, A, 1.00, 100.00%  
6430F1, A, B, B, -, A, A, E, A, A, A, 1.00, 100.00%  
3D762E, A, D, A, -, B, A, E, A, A, A, 1.00, 100.00%  
3D766B, A, B, B, -, A, A, E, J, A, -, 1.00, 100.00%  
643174, A, D, B, -, B, C, -, -, A, A, 1.00, 100.00%
643130, A, C, B, -, B, -, E, J, A, A, 0.00, 0.00%
3D764A, A, A, C, -, B, A, E, G, A, A, 1.00, 100.00%  
3968AA, B, A, B, -, B, A, E, A, E, D, 1.00, 100.00%
Due to locale the percentage 100.00% has been written as 100,00% and this confuses CSV readers, which interpret these as two different columns.
This extra comma is also confusing for some CSV readers
Data from clickers (3)

• Let’s prepare the data for analysis:

```python
In [1]: import pandas as pd

data = pd.read_csv('lecture_01_results_detail_2018.csv', skiprows=[0,1,2,3,4,5,7],
                     skipfooter=1, engine='python', index_col=1)
data = data.iloc[:,[1,2,3,5,6,7,8,9,10]]
data.columns = ['Q'+str(i) for i in range(1,10)]
data.head()
```

Out[1]:

<table>
<thead>
<tr>
<th></th>
<th>Q1</th>
<th>Q2</th>
<th>Q3</th>
<th>Q4</th>
<th>Q5</th>
<th>Q6</th>
<th>Q7</th>
<th>Q8</th>
<th>Q9</th>
</tr>
</thead>
<tbody>
<tr>
<td>396916</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>A</td>
<td>B</td>
<td>E</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>39B884</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>A</td>
<td>B</td>
<td>A</td>
</tr>
<tr>
<td>6430F1</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>E</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>E</td>
<td>J</td>
<td>A</td>
<td>-</td>
</tr>
</tbody>
</table>
• Let’s prepare the data for analysis:

In [1]: `import pandas as pd

data = pd.read_csv('lecture_01_results_detail_2018.csv',skiprows=[0,1,2,3,4,5,7],
                    skipfooter=1,engine='python',index_col=1)
data = data.iloc[:,[1,2,3,5,6,7,8,9,10]]
data.columns = ['Q'+str(i) for i in range(1,10)]
data.head()

Removed Q4, as the poll was closed accidentally too early.

<table>
<thead>
<tr>
<th>6430F1</th>
<th>A</th>
<th>B</th>
<th>B</th>
<th>A</th>
<th>A</th>
<th>E</th>
<th>A</th>
<th>A</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D762E</td>
<td>A</td>
<td>D</td>
<td>A</td>
<td>B</td>
<td>A</td>
<td>E</td>
<td>A</td>
<td>A</td>
<td>A</td>
</tr>
<tr>
<td>3D766B</td>
<td>A</td>
<td>B</td>
<td>B</td>
<td>A</td>
<td>A</td>
<td>E</td>
<td>J</td>
<td>A</td>
<td>-</td>
</tr>
</tbody>
</table>
Data from clickers (3)

• Let’s prepare the data for analysis:

```python
In [1]: import pandas as pd

data = pd.read_csv('lecture_01_results_detail_2018.csv', skiprows=[0, 1, 2, 3, 4, 5, 7],
                   skipfooter=1, engine='python', index_col=1)
data = data.iloc[:, [1, 2, 3, 5, 6, 7, 8, 9, 10]]
data.columns = ['Q' + str(i) for i in range(1, 11)]
data.head()
```

Removed Q4, as the poll was closed accidentally too early.

Renumbering, now old Q5, Q6, … become Q4, Q5, … respectively
Data from clickers (4)

• Let’s analyse the data!
• Goals revisited:
  – Where are the problems in understanding?
    • Proportion of correct answers?
  – Which questions attract less answers?
    • Number of answers per question
  – How many people stopped answering?
    • Clickers that are silent after some point?
  – How many people started later?
    • Clickers that do not answer the first question(s)?
  – Which questions attract more answers?
    • Distribution of answers per question
  – Identify potential technical problems with clickers
    • Distribution of answers per clicker
Number of answers?

- Per person? Per question?

```python
In [2]:
    n_answered_per_person = data.ne('\-').sum(axis=1)
    n_answered_per_person.value_counts()

Out[2]:
   9  50
   8  10
   7   3
   6   2
   5   1
dtype: int64
```

```python
In [3]:
    n_answered_per_question = data.ne('\-').sum()
    n_answered_per_question

Out[3]:
   Q1   63
   Q2   64
   Q3   64
   Q4   65
   Q5   63
   Q6   65
   Q7   63
   Q8   61
   Q9   60
dtype: int64
```
Number of answers?

- Per person? Per question?

```python
In [2]:
    n_answered_per_person = data.ne(' - ').sum(axis=1)
    n_answered_per_person.value_counts()
```

```
Out[2]:
   9    50
   8    10
   7     3
   6     2
   5     1
dtype: int64
```

- Per question?

```python
In [3]:
    n_answered_per_question = data.ne(' - ').sum()
    n_answered_per_question
```

```
Out[3]:
Q1    63
Q2    64
Q3    64
Q4    65
Q5    63
Q6    65
Q7    63
Q8    61
Q9    60
dtype: int64
```

50 people answered all 9 questions
Number of answers?

- Per person? Per question?

In [2]:
`n_answered_per_person = data.ne(' ').sum(axis=1)
n_answered_per_person.value_counts()`

Out[2]:
```
50 people answered all 9 questions
```

In [3]:
`n_answered_per_person = data.ne(' ').sum()`

Out[3]:
```
Q1    63
Q2    64
Q3    64
Q4    65
Q5    63
Q6    65
Q7    63
Q8    61
Q9    60
dtype: int64
```

1 person answered only 5. Technical problems? Deliberately? Left early?
Number of answers?

• Per person? Per question?

In [2]:
```python
n_answered_per_person = data.ne('-').sum(axis=1)
n_answered_per_person.value_counts()
```

Out[2]:
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>50</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>7</td>
<td>3</td>
</tr>
<tr>
<td>6</td>
<td>2</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>

dtype: int64
```

In [3]:
```python
n_answered_per_question = data.ne('-').sum()
n_answered_per_question
```

Out[3]:
```

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Q1</td>
<td>63</td>
</tr>
<tr>
<td>Q2</td>
<td>64</td>
</tr>
<tr>
<td>Q3</td>
<td>64</td>
</tr>
<tr>
<td>Q4</td>
<td>65</td>
</tr>
<tr>
<td>Q5</td>
<td>63</td>
</tr>
<tr>
<td>Q6</td>
<td>65</td>
</tr>
<tr>
<td>Q7</td>
<td>63</td>
</tr>
<tr>
<td>Q8</td>
<td>61</td>
</tr>
<tr>
<td>Q9</td>
<td>60</td>
</tr>
</tbody>
</table>

dtype: int64
```

Slightly smaller number of answers. A hard or confusing question?
Number of correct answers?

- Per person? Per question?

```python
In [4]:
correct_q5 = data['Q5'].eq('A')*1
correct_q6 = data['Q6'].eq('E')*1
n_correct_per_person = correct_q5 + correct_q6
n_correct_per_person.value_counts()

Out[4]:
2     41
1     19
0      6
dtype: int64
```

```python
In [5]:
n_correct_per_question = {'Q5':sum(correct_q5), 'Q6':sum(correct_q6),
                         'n_people':len(correct_q5)}
n_correct_per_question

Out[5]:
{'Q5': 49, 'Q6': 52, 'n_people': 66}
```
Number of correct answers?

- Per person? Per question?

```python
In [4]:
correct_q5 = data['Q5'].eq('A')*1
correct_q6 = data['Q6'].eq('E')*1
n_correct_per_person = correct_q5 + correct_q6
n_correct_per_person.value_counts()
```

```
Out[4]:
2    41
1    19
0     6
dtype: int64
```

41 people got both 2 answers correct

```python
In [5]:
n_correct_per_question = {'Q5':sum(correct_q5),'Q6':sum(correct_q6),
                         'n_people':len(correct_q5)}
n_correct_per_question
```

```
Out[5]:
{'Q5': 49, 'Q6': 52, 'n_people': 66}
```
Number of correct answers?

• Per person? Per question?

In [4]:
```python
correct_q5 = data['Q5'].eq('A')*1
correct_q6 = data['Q6'].eq('E')*1
n_correct_per_person = correct_q5 + correct_q6
n_correct_per_person.value_counts()
```

Out[4]:
```
        2    41
          1    19
          0    6
dtype: int64
```

49 out of 66 people answered Q5 correctly

Out[5]:
```
{'Q5': 49, 'Q6': 52, 'n_people': 66}
```
First / last question answered

- Which question was the first that you answered? The last?

```python
In [6]:
import numpy as np

first_question_answered = data.ne('-').apply(lambda row:
                                             min(np.where(row)[0])
                                             ,axis=1)

first_question_answered.value_counts()

Out[6]:
0   63
1   2
4   1
dtype: int64

In [7]:
last_question_answered = data.ne('-').apply(lambda row:
                                             max(np.where(row)[0])
                                             ,axis=1)

last_question_answered.value_counts()

Out[7]:
8   60
7   4
6   1
5   1
dtype: int64
```
First / last question answered

- Which question was the first that you answered? The last?

```python
In [6]:
import numpy as np
first_question_answered = data.ne('-').apply(lambda row:
                        min(np.where(row)[0])
                        ,axis=1)
first_question_answered.value_counts()

Out[6]:
0    63
1     2
4     1
dtype: int64

In [7]:
last_question_answered = data.ne('-').apply(lambda row:
                                    max(np.where(row)[0])
                                    ,axis=1)
last_question_answered.value_counts()

Out[7]:
8    60
7     4
6     1
5     1
dtype: int64
```

63 people started answering immediately from Q1
First / last question answered

- Which question was the first that you answered? The last?

```python
import numpy as np
first_question_answered = data.ne('-').apply(lambda row:
    min(np.where(row[0])
    ,axis=1)
first_question_answered.value_counts()
```

```
Out[6]:
0    63
1     2
4     1
dtype: int64
```

- 63 people started answering immediately from Q1

```python
last_question_answered
last_question_answered.value_counts()
```

```
Out[7]:
8    60
7     4
6     1
5     1
dtype: int64
```

- One person started at Q5 – late arrival to the lecture?
First / last question answered

• Which question was the first that you answered? The last?

```python
In [6]:
import numpy as np
first_question_answered = data.ne('-').apply(lambda row:
min(np.where(row)[0])
,axis=1)
first_question_answered.value_counts()
```

```
Out[6]:
0    63
1     2
4     1
dtype: int64
```

```python
In [7]:
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
```
First / last question answered

• Which question was the first that you answered? The last?

```python
In [6]:
import numpy as np
first_question_answered = data.ne('‐').apply(lambda row:
            min(np.where(row)[0])
        , axis=1)

first_question_answered.value_counts()
```

Out[6]:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>63</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
<tr>
<td>4</td>
<td>1</td>
</tr>
</tbody>
</table>
dtype: int64

```python
In [7]:

Out[7]:

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td>7</td>
<td>4</td>
</tr>
<tr>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>5</td>
<td>1</td>
</tr>
</tbody>
</table>
dtype: int64
```

60 people answered the last question Q9

One stopped after Q6. Bored? Sleeping? Left?
Data from clickers (5)

• Conclusions
  – Too few questions with correct/wrong answers to evaluate understanding properly
  – Good participation in answering
  – No (obvious) technical problems other than one poll closed too early and then repolled
End of Demo
Steps of this data science mini-project

• Understanding the goals
• Understanding the data
  – Which files available? Format?
  – Meanings of fields? Errors?
• Preparing the data
  – Making the table of all answers
• Analysis of data
• Evaluation of results
Steps in data science projects

• It turns out that other data science projects have similar steps!
• Being aware of these steps helps to make the project successful
• This inspired development of the standard: CRISP-DM
Demo: Data science mini-project

- **CRISP-DM**: cross-industrial standard process for data mining

- Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks

- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
CRISP-DM

- CRoss-Industrial Standard Process for Data Mining

CRISP-DM

• Published in 1999
• Many companies were involved
• Several data science project management tools support it (e.g. IBM SPSS Modeler)
• IBM is the primary corporation that currently embraces the CRISP-DM process model
• In online polls in 2002, 2004, 2007 and 2014 CRISP-DM was the leading methodology used by industry data miners
• In 2015 IBM published ASUM-DM, an advanced process based on CRISP-DM
Why to use CRISP-DM?

CRISP-DM – a Standard Methodology to Ensure a Good Outcome
Posted by William Vorhies on July 26, 2016 at 9:15am

Summary: To ensure quality in your data science group, make sure you’re enforcing a standard methodology. This includes not only traditional data analytic projects but also our most advanced recommenders, text, image, and language processing, deep learning, and AI projects.

About the author: Bill Vorhies is Editorial Director for Data Science Central and has practiced as a data scientist and commercial predictive modeler since 2001.

Source:
CRISP-DM is data mining – what about data science?

• Data science projects have sometimes less specific goals
  – “Here are the data, make money out of it”

• However, thinking in terms of CRISP-DM can still be beneficial
CRISP-DM

- CRoss-Industrial Standard Process for Data Mining

Source:
CRISP-DM: Business understanding

• Subtasks:
  – Understand what you want to accomplish
  – Assess the situation (uncover important factors that could influence the outcome of the project)
  – Translate the business goal in a data mining objective
  – Develop a project plan

• In our mini-project to analyse clicker data:
  – List of goals
CRISP-DM: Data understanding

• Subtasks:
  – Collect initial data
  – Describe the data
  – Explore the data
  – Verify data quality

• In our mini-project:
  – Extracted the files
  – Studied the contents of files
  – Found the problems in formatting
  – Found a column with useless data
CRISP-DM: Data preparation

• Subtasks:
  – Select the data
  – Clean the data
  – Construct new data
  – Integrate the data
  – Format the data

• In our mini-project:
  – Choose two files to be used
  – Extract the matrix of answers
CRISP-DM: Modelling

• Subtasks:
  – Select modelling techniques
  – Generate a test design
  – Build the models
  – Assess the models

• In our mini-project:
  – Simple counting
  – No formal modelling
CRISP-DM: Evaluation

• Subtasks:
  –Evaluate the results
  –Review the process
  –Determine the next steps

• In our mini-project:
  –Conclusions from the analysis results
  –Identified hard / potentially confusing questions
CRISP-DM: Deployment

• Subtasks:
  – Plan for deployment
  – Plan monitoring and maintenance
  – Produce a final report
  – Conduct a final project review

• In our mini-project:
  – Plan how to improve the lecture material
  – Plan future use of clickers
CRISP-DM

- CRoss-Industrial Standard Process for Data Mining

Which CRISP-DM step comes first?

A. Business understanding
B. Data preparation
C. Data understanding
D. Deployment
E. Evaluation
F. Modelling

Response Counter

- Business understanding: 1
- Data preparation: 1
- Data understanding: 1
- Deployment: 1
- Evaluation: 1
- Modelling: 1
Which CRISP-DM step comes 2nd?

A. Business understanding
B. Data preparation
C. Data understanding
D. Deployment
E. Evaluation
F. Modelling

![Bar chart showing the CRISP-DM steps with Data understanding as the second step marked with a checkmark.]
Which CRISP-DM step comes 3rd?

A. Business understanding
B. Data preparation
C. Data understanding
D. Deployment
E. Evaluation
F. Modelling
Which CRISP-DM step comes 4th?

A. Business understanding
B. Data preparation
C. Data understanding
D. Deployment
E. Evaluation
F. Modelling

Correct answer: A. Business understanding
Which CRISP-DM step comes 5th?

A. Business understanding
B. Data preparation
C. Data understanding
D. Deployment
E. Evaluation
F. Modelling

![Bar chart showing the CRISP-DM steps]

Response Counter

Business understanding
Data preparation
Data understanding
Deployment
Evaluation
Modelling

Correct answer: E. Evaluation
Which CRISP-DM step comes 6th?

A. Business understanding
B. Data preparation
C. Data understanding
D. Deployment
E. Evaluation
F. Modelling

![Response Counter Diagram]
CRISP-DM and data science

• CRISP-DM is still useful when the goal and task are clearly specified and data analysis is the core

• However, data science projects are sometimes much more open-ended:
  – We suspect that there is value in the data, but how to unlock it? (task is not clear)
  – Data mining – we know what and where to mine
  – Data science also includes prospecting: searching for deposits of precious metals to start mining
Data Science Trajectories

Data Science Trajectories

Data Science Trajectories

Start of CRISP-DM activities

Data Science Trajectories

Data management activities

Open-ended exploration activities

Demo: Data science mini-project
CRISP-DM: cross-industrial standard process for data mining

- **Graph mining**
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks

- **Natural language processing**
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
Acknowledgements

• Most of the following slides have been adapted from:
  – Joseph E. Gonzalez (University of California, Berkeley)
    https://bcourses.berkeley.edu/courses/1267848/files/52373801
  – Natalia Vanetik
    http://www.cs.bgu.ac.il/~orlovn/presentations/graph_mining_seminar_2009.ppt
  – John Canny (University of California, Berkeley)
    https://bcourses.berkeley.edu/courses/1267848/files/50935031
Lecture 11 – CRISP-DM, Graphs, NLP

- Demo: Data science mini-project
- CRISP-DM: cross-industrial standard process for data mining
- Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks
- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
Social Network

Vertices
- Users
- Posts / Images

Edges
- Social Relationships
  - Directed: Twitter followers
  - Undirected: Facebook friends
- Likes

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Web Graphs

- Vertices: Web-pages
- Edges: Links (Directed)

Generated Content:
- Click-streams

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Semantic Networks

- Organized knowledge
- Vertices: Subject, Object
- Edges: Predicates
- Example: Google Knowledge Graph
  - 570M Vertices
  - 18B Edges

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014

http://wiki.dbpedia.org
Trade Networks

Supply Chain:
Vertices: Suppliers/Consumers
Edges: Exchange of Goods

Transaction Networks (e.g., Bitcoin):
Vertices: Users
Edges: Exchange of Currency


Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Biological Networks

Protein-Protein Interaction Networks (Interactomes)
Vertices: Proteins
Edges: Interactions

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
2.2 Paths and Connectivity

We now turn to some of the fundamental concepts and definitions surrounding graphs. Perhaps because graphs are so simple to define and work with, an enormous range of graph-theoretic notions have been studied; the social scientist John Barnes once described graph theory as a "terminological jungle, in which any newcomer may plant a tree" [45]. Fortunately, for our purposes, we will be able to get underway with just a brief discussion of some of the most central concepts.

Vertices: Devices, Routers
Directed Edges: Network Flows

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Co-Authorship Network

Vertices: Authors
Edges: Co-authorship

Example: Erdős Number

**Erdős number 4** (Trail: Paul Erdős [1913-1996], Bela Bollobas [1943-...], Heikki Mannila, Jaak Vilo, Meelis Kull)

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Lecture 11 – CRISP-DM, Graphs, NLP

✓ Demo: Data science mini-project
✓ CRISP-DM: cross-industrial standard process for data mining
  • Graph mining
    ✓ Examples of applications
      • Properties of graphs
      • Graph algorithms
      • Graph pattern mining
      • Graph neural networks
  • Natural language processing
    • Bag-of-words and N-grams
    • Parts of speech, grammars and parsing
    • Dependencies
    • Neural networks for NLP
More than $10^8$ vertices have one neighbor.

High-Degree Vertices

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Giant Connected Component

Do my Facebook friends know each other?

Vertices: My friends

Edges: Direct friendship

Densification

Average distance between nodes reduces over time.

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Community Structure

Conductance of a community: proportion of outwards links among all links

Community profile plot: lowest conductance for each community size

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Community Structure

Dolphins social network . . .

Linked-In

Messenger

Different colours = different algorithms to find communities

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014

Meelis Kull - Autumn 2022 - LTAT.02.002 – Intro to Data Science - Lecture 11
Lecture 11 – CRISP-DM, Graphs, NLP

- Demo: Data science mini-project
- CRISP-DM: cross-industrial standard process for data mining

- Graph mining
  - Examples of applications
  - Properties of graphs
    - **Graph algorithms**
    - Graph pattern mining
    - Graph neural networks

- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
PageRank (Centrality Measures)

• Recursive Relationship:

\[ R[i] = \alpha + (1 - \alpha) \sum_{(j,i) \in E} \frac{1}{L[j]} R[j] \]

• Where:
  – \( \alpha \) is the random reset probability (typically 0.15)
  – \( L[j] \) is the number of links on page \( j \)

\[ R[5] = \alpha + (1 - \alpha) \left( \frac{1}{3} R[1] + \frac{1}{1} R[4] \right) \]


Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Ratings

Users

Items

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Recommending Products

Low-Rank Matrix Factorization:

User Factors (U) \( \approx \) Movie Factors (M)

User Factors (U) \( \times \) Movies

Slide adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Finding Communities

• Count triangles passing through each vertex:

• Measure “cohesiveness” of local community

\[
\text{ClusterCoeff}[i] = \frac{2 \times \#\text{Triangles}[i]}{\text{Deg}[i] \times (\text{Deg}[i] - 1)}
\]
**Connected Components**

- Every vertex starts out with a unique component id (typically it’s vertex id):

\[ CC[i] = \text{arg min}_{\{i,j\} \in E} CC[j] \]
Putting it All Together

Raw Wikipedia

XML

Discussion Table

Text Table

Term-Doc Graph

Community Detection

Editor Graph

Hyperlinks

PageRank

Top 20 Pages

Topic Model (LDA)

Word Topics

Term - Doc

Graph

Community

User Community

User

Com.

Disc.

Community

Topic

Com.

Table adapted from Joseph E. Gonzalez, University of California, Berkeley, 2014
Lecture 11 – CRISP-DM, Graphs, NLP

✓ Demo: Data science mini-project
✓ CRISP-DM: cross-industrial standard process for data mining
  • Graph mining
    ✓ Examples of applications
    ✓ Properties of graphs
    ✓ Graph algorithms
    • Graph pattern mining
    • Graph neural networks
  • Natural language processing
    • Bag-of-words and N-grams
    • Parts of speech, grammars and parsing
    • Dependencies
    • Neural networks for NLP
Graph Pattern Mining

• Frequent subgraphs
  – A (sub)graph is frequent if its support (occurrence frequency) in a given dataset is no less than a minimum support threshold

• Applications of graph pattern mining:
  – Mining biochemical structures
  – Program control flow analysis
  – Mining XML structures or Web communities
  – Building blocks for graph classification, clustering, compression, comparison, and correlation analysis

Slide adapted from Natalia Vanetik, 2009
Example 1: Sub-parts in molecules

GRAPH DATASET

(T1)  

(T2)  

(T3)  

FREQUENT PATTERNS  
(MIN SUPPORT IS 2)

(1)  

(2)  

Slide adapted from Natalia Vanetik, 2009
Example 2: Function call graph during program execution

**GRAPH DATASET**

![Graph dataset with nodes and arrows indicating function calls.]

**FREQUENT PATTERNS**

(MIN SUPPORT IS 2)

Slide adapted from Natalia Vanetik, 2009
Typical scenarios in graph mining

• Transaction scenario:
  – Given a dataset of many graphs, find patterns which occur in sufficiently many of those graphs
  – Called transaction setting because each graph is one transaction

• Single graph scenario:
  – Given a single graph, find patterns which occur sufficiently many times in this graph
Finding Frequent Subgraphs: Input and Output

Input
- Database of graph transactions.
- Undirected simple graph (no loops, no multiples edges).
- Each graph transaction has labels associated with its vertices and edges.
- Transactions may not be connected.
- Minimum support threshold $\sigma$.

Output
- Frequent subgraphs that satisfy the minimum support constraint.
- Each frequent subgraph is connected.

Slide adapted from Natalia Vanetik, 2009
Apriori-Based Approach

Frequent k-graphs

\[ G \]

\[ G' \]

\[ G'' \]

Candidate frequent \((k+1)\)-graphs

\[ G_1 \]

\[ G_2 \]

\[ \ldots \]

\[ G_n \]

join

Slide adapted from Natalia Vanetik, 2009
frequent 1-subgraphs

frequent 2-subgraphs

3-candidates

frequent 3-subgraphs

4-candidates

frequent 4-subgraphs
**FSG Algorithm**

[M. Kuramochi and G. Karypis. Frequent subgraph discovery. ICDM 2001]

**Notation:** $k$-subgraph is a subgraph with $k$ edges.

**Init:** Scan the transactions to find $F_1$, the set of all frequent 1-subgraphs and 2-subgraphs, together with their counts;

For $(k=3; F_{k-1} \neq \emptyset ; k++)$

1. **Candidate generation** - $C_k$, the set of candidate $k$-subgraphs, from $F_{k-1}$, the set of frequent $(k-1)$-subgraphs;

2. **Candidates pruning** - a necessary condition of candidate to be frequent is that each of its $(k-1)$-subgraphs is frequent.

3. **Frequency counting** - Scan the transactions to count the occurrences of subgraphs in $C_k$;

4. $F_k = \{ c \in C_K | c$ has counts no less than $\#minSup \}$

5. **Return** $F_1 \cup F_2 \cup \ldots \cup F_k$ ($= F$)

Slide adapted from Natalia Vanetik, 2009
Trivial operations are complicated with graphs

- **Candidate generation**
  - To determine two candidates for joining, we need to check for graph isomorphism.

- **Candidate pruning**
  - To check downward closure property, we need graph isomorphism.

- **Frequency counting**
  - Subgraph isomorphism for checking containment of a frequent subgraph.

*Slide adapted from Natalia Vanetik, 2009*
Graph mining software
Overview

GraphX is a new component in Spark for graphs and graph-parallel computation. At a high level, GraphX extends the Spark RDD by introducing a new Graph abstraction: a directed multigraph with properties attached to each vertex and edge. To support graph computation, GraphX exposes a set of fundamental operators (e.g., subgraph, joinVertices, and aggregateMessages) as well as an optimized variant of the Pregel API. In addition, GraphX includes a growing collection of graph algorithms and builders to simplify graph analytics tasks.

GraphX Programming Guide

- Overview
- Getting Started
- The Property Graph
  - Example Property Graph
- Graph Operators
  - Summary List of Operators
  - Property Operators
  - Structural Operators
  - Join Operators
  - Neighborhood Aggregation
    - Aggregate Messages (aggregateMessages)
    - Map Reduce Triplets Transition Guide (Legacy)
    - Computing Degree Information
    - Collecting Neighbors
  - Caching and Uncaching
- Pregel API
- Graph Builders
- Vertex and Edge RDDs
  - VertexRDDs
  - EdgeRDDs
- Optimized Representation
- Graph Algorithms
  - PageRank
  - Connected Components
  - Triangle Counting
- Examples
Welcome to Apache Giraph!

*Apache Giraph* is an iterative graph processing system built for high scalability. For example, it is currently used at Facebook to analyze the social graph formed by users and their connections. Giraph originated as the open-source counterpart to *Pregel*, the graph processing architecture developed at Google and described in a 2010 paper. Both systems are inspired by the *Bulk Synchronous Parallel* model of distributed computation introduced by Leslie Valiant. Giraph adds several features beyond the basic Pregel model, including master computation, sharded aggregators, edge-oriented input, out-of-core computation, and more. With a steady development cycle and a growing community of users worldwide, Giraph is a natural choice for unleashing the potential of structured datasets at a massive scale. To learn more, consult the *User Docs* section above.
Gephi – a visualization framework

Features

Gephi is a tool for data analysts and scientists keen to explore and understand graphs. Like Photoshop™ but for graph data, the user interacts with the representation, manipulate the structures, shapes and colors to reveal hidden patterns. The goal is to help data analysts to make hypothesis, intuitively discover patterns, isolate structure singularities or faults during data sourcing. It is a complementary tool to traditional statistics, as visual thinking with interactive interfaces is now recognized to facilitate reasoning.

This is a software for Exploratory Data Analysis, a paradigm appeared in the Visual Analytics field of research.

Real-time visualization
Graph Database Technologies

- Property graph data-model for storing and retrieving graph structured data.
- **Neo4j**: popular commercial graph database
- **Titan**: open-source distributed graph database
Lecture 11 – CRISP-DM, Graphs, NLP

✓ Demo: Data science mini-project
✓ CRISP-DM: cross-industrial standard process for data mining
  • Graph mining
    ✓ Examples of applications
    ✓ Properties of graphs
    ✓ Graph algorithms
    ✓ Graph pattern mining
      • Graph neural networks
  • Natural language processing
    • Bag-of-words and N-grams
    • Parts of speech, grammars and parsing
    • Dependencies
    • Neural networks for NLP
Graph neural networks

- A Graph neural network (GNN) is a class of artificial neural networks for processing data that can be represented as graphs.

Basic building blocks of a Graph neural network (GNN):
1. Permutation equivariant layer.
2. Local pooling layer.
3. Global pooling (or readout) layer.
Colors indicate features.

Node representation update in a Message Passing Neural Network (MPNN) layer.

https://en.wikipedia.org/wiki/Graph_neural_network
Demo: Data science mini-project
CRISP-DM: cross-industrial standard process for data mining
Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks
• **Natural language processing**
  • Bag-of-words and N-grams
  • Parts of speech, grammars and parsing
  • Dependencies
  • Neural networks for NLP
Natural Language Processing (NLP)

• In a recent survey (KDNuggets blog) of data scientists, 62% reported working “mostly or entirely” with data about people. Much of this data is text.

• NLP is a central part of mining large datasets.
Natural Language Processing

Some basic terms:

- **Syntax**: the allowable structures in the language: sentences, phrases, affixes (-ing, -ed, -ment, etc.).
- **Semantics**: the meaning(s) of texts in the language.
- **Part-of-Speech (POS)**: the category of a word (noun, verb, preposition etc.).
- **Bag-of-words (BoW)**: a featurization that uses a vector of word counts (or binary) ignoring order.
- **N-gram**: for a fixed, small N (2-5 is common), an n-gram is a consecutive sequence of words in a text.
✓ Demo: Data science mini-project
✓ CRISP-DM: cross-industrial standard process for data mining
✓ Graph mining
  ✓ Examples of applications
  ✓ Properties of graphs
  ✓ Graph algorithms
  ✓ Graph pattern mining
  ✓ Graph neural networks
• Natural language processing
  • **Bag-of-words and N-grams**
  • Parts of speech, grammars and parsing
  • Dependencies
  • Neural networks for NLP
Bag of words Featurization

Assuming we have a dictionary mapping words to a unique integer id, a bag-of-words featurization of a sentence could look like this:

Sentence: The cat sat on the mat
word id-s: 1 12 5 3 1 14

The BoW featurization would be the vector:

Vector 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1
position 1 3 5 12 14

In practice this would be stored as a sparse vector of (id, count) pairs:

(1,2),(3,1),(5,1),(12,1),(14,1)

Note that the original word order is lost, replaced by the order of id’s.

Slide adapted from John Canny, University of California, Berkeley, 2014
Bag-of-words lose word order

Because word order is lost, the sentence meaning is weakened. This sentence has quite a different meaning but the same BoW vector:

Sentence: The mat sat on the cat

word id-s: 1 14 5 3 1 12

The BoW featurization would be the vector:

Vector: 2, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 1, 0, 1

position: 1 3 5 12 14

But word order is important, especially the order of nearby words.

N-grams capture this, by modeling tuples of consecutive words.

Slide adapted from John Canny, University of California, Berkeley, 2014
N-grams

Sentence: The cat sat on the mat
2-grams: the-cat, cat-sat, sat-on, on-the, the-mat
Notice how even these short n-grams “make sense” as linguistic units. For the other sentence we would have different features:
Sentence: The mat sat on the cat
2-grams: the-mat, mat-sat, sat-on, on-the, the-cat
We can go still further and construct 3-grams:
Sentence: The cat sat on the mat
3-grams: the-cat-sat, cat-sat-on, sat-on-the, on-the-mat
Which capture still more of the meaning:
Sentence: The mat sat on the cat
3-grams: the-mat-sat, mat-sat-on, sat-on-the, on-the-cat

Slide adapted from John Canny, University of California, Berkeley, 2014
N-grams Features

Typically, it's advantageous to use multiple n-gram features in machine learning models with text, e.g.

unigrams + bigrams (2-grams) + trigrams (3-grams).

The unigrams have higher counts and are able to detect influences that are weak, while bigrams and trigrams capture strong influences that are more specific.

e.g. “the white house” will generally have very different influences from the sum of influences of “the”, “white”, “house”.

Slide adapted from John Canny, University of California, Berkeley, 2014
N-grams size

N-grams pose some challenges in feature set size.
If the original vocabulary size is $|V|$, the number of 2-grams is $|V|^2$
While for 3-grams it is $|V|^3$

Luckily natural language n-grams (including single words) have a **power law** frequency structure. This means that most of the n-grams you see are common. A dictionary that contains the most common n-grams will cover most of the n-grams you see.
Power laws for N-grams

N-grams follow a power law distribution:

Slide adapted from John Canny, University of California, Berkeley, 2014
N-grams size

Because of this you may see values like this:

- Unigram dictionary size: 40,000
- Bigram dictionary size: 100,000
- Trigram dictionary size: 300,000

With coverage of > 80% of the features occurring in the text.
N-gram Language Models

N-grams can be used to build statistical models of texts.

When this is done, they are called **n-gram language models**.

An n-gram language model associates a probability with each n-gram, such that the sum over all n-grams (for fixed n) is 1.

You can then determine the overall likelihood of a particular sentence:

The cat sat on the mat

Is much more likely than

The mat sat on the cat

Slide adapted from John Canny, University of California, Berkeley, 2014
We can also analyze the meaning of a particular word by looking at the contexts in which it occurs.

The context is the set of words that occur near the word, i.e. at displacements of ..., -3, -2, -1, +1, +2, +3, ... in each sentence where the word occurs.

A **skip-gram** is a set of non-consecutive words (with specified offset), that occur in some sentence.

We can construct a BoSG (bag of skip-gram) representation for each word from the skip-gram table.
Lecture 11 – CRISP-DM, Graphs, NLP

- Demo: Data science mini-project
- CRISP-DM: cross-industrial standard process for data mining
- Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks
- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
Parts of Speech

Original list by Dionysios Thrax (~ 100 B.C.):

- Noun (boat, plane)
- Verb (goes, spun, hunted)
- Pronoun (She, Her)
- Preposition (in, on)
- Adverb (quietly, then)
- Conjunction (and, but)
- Participle (eaten, running)
- Article (the, a)

Slide adapted from John Canny, University of California, Berkeley, 2014
# Parts of Speech (Penn Treebank 2014)

<table>
<thead>
<tr>
<th></th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>CC</td>
</tr>
<tr>
<td>2.</td>
<td>CD</td>
</tr>
<tr>
<td>3.</td>
<td>DT</td>
</tr>
<tr>
<td>4.</td>
<td>EX</td>
</tr>
<tr>
<td>5.</td>
<td>FW</td>
</tr>
<tr>
<td>6.</td>
<td>IN</td>
</tr>
<tr>
<td>7.</td>
<td>JJ</td>
</tr>
<tr>
<td>8.</td>
<td>JJR</td>
</tr>
<tr>
<td>9.</td>
<td>JJS</td>
</tr>
<tr>
<td>10.</td>
<td>LS</td>
</tr>
<tr>
<td>11.</td>
<td>MD</td>
</tr>
<tr>
<td>12.</td>
<td>NN</td>
</tr>
<tr>
<td>13.</td>
<td>NNS</td>
</tr>
<tr>
<td>14.</td>
<td>NNP</td>
</tr>
<tr>
<td>15.</td>
<td>NNPS</td>
</tr>
<tr>
<td>16.</td>
<td>PDT</td>
</tr>
<tr>
<td>17.</td>
<td>POS</td>
</tr>
<tr>
<td>18.</td>
<td>PRP</td>
</tr>
<tr>
<td>19.</td>
<td>PRP$</td>
</tr>
<tr>
<td>20.</td>
<td>RB</td>
</tr>
<tr>
<td>21.</td>
<td>RBR</td>
</tr>
<tr>
<td>22.</td>
<td>RBS</td>
</tr>
<tr>
<td>23.</td>
<td>RP</td>
</tr>
<tr>
<td>24.</td>
<td>SYM</td>
</tr>
<tr>
<td>25.</td>
<td>TO</td>
</tr>
<tr>
<td>26.</td>
<td>UH</td>
</tr>
<tr>
<td>27.</td>
<td>VB</td>
</tr>
<tr>
<td>28.</td>
<td>VBD</td>
</tr>
<tr>
<td>29.</td>
<td>VBG</td>
</tr>
<tr>
<td>30.</td>
<td>VBN</td>
</tr>
<tr>
<td>31.</td>
<td>VBP</td>
</tr>
<tr>
<td>32.</td>
<td>VBZ</td>
</tr>
<tr>
<td>33.</td>
<td>WDT</td>
</tr>
<tr>
<td>34.</td>
<td>WP</td>
</tr>
<tr>
<td>35.</td>
<td>WP$</td>
</tr>
<tr>
<td>36.</td>
<td>WRB</td>
</tr>
</tbody>
</table>

*Slide adapted from John Canny, University of California, Berkeley, 2014*
Grammars

Grammars comprise rules that specify acceptable sentences in the language: (S is the sentence or root node) “the cat sat on the mat”

- $S \rightarrow NP \ VP$
- $S \rightarrow NP \ VP \ PP \ (the \ cat) \ (sat) \ (on \ the \ mat)$
- $NP \rightarrow DT \ NN \ (the \ cat), \ (the \ mat)$
- $VP \rightarrow VB \ NP$
- $VP \rightarrow VBD$
- $PP \rightarrow IN \ NP$
- $DT \rightarrow “the”$
- $NN \rightarrow “mat”, \ “cat”$
- $VBD \rightarrow “sat”$
- $IN \rightarrow “on”$
“The cat sat on the mat”

Parse Trees

Slide adapted from John Canny, University of California, Berkeley, 2014
Grammars

There are typically multiple ways to produce the same sentence. Consider the statement by Groucho Marx:

“While I was in Africa, I shot an elephant in my pajamas”
“How he got into my pajamas, I don’t know”
Parse Trees

“…,I shot an elephant in my pajamas” -what people hear first

Slide adapted from John Canny, University of California, Berkeley, 2014
Parse Trees

Groucho’s version

Slide adapted from John Canny, University of California, Berkeley, 2014
Grammars

Recursion is common in grammar rules, e.g.

\[ \text{NP} \rightarrow \text{NP RC} \]

Because of this, sentences of arbitrary length are possible.
Recursion in Grammars

“Buffalo buffalo Buffalo buffalo buffalo buffalo Buffalo buffalo”.

https://en.wikipedia.org/wiki/Buffalo_buffalo_Buffalo_buffalo_buffalo_buffalo_buffalo_buffalo_buffalo

Slide adapted from John Canny, University of California, Berkeley, 2014
Recursion in Grammars

“Nero played his lyre while Rome burned”.

It's also possible to have “sentences” inside other sentences...

\[
S \rightarrow \text{NP } \text{VP} \\
\text{VP} \rightarrow \text{VB } \text{NP} \\
\text{SBAR} \\
\text{SBAR} \rightarrow \text{IN } S
\]

Slide adapted from John Canny, University of California, Berkeley, 2014
PCFGs

Complex sentences can be parsed in many ways, most of which make no sense or are extremely improbable (like Groucho’s example).

Probabilistic Context-Free Grammars (PCFGs) associate and learn probabilities for each rule:

\[ S \rightarrow \text{NP VP} \quad 0.3 \]
\[ S \rightarrow \text{NP VP PP} \quad 0.7 \]

The parser then tries to find the most likely sequence of productions that generate the given sentence. This adds more realistic “world knowledge” and generally gives much better results.

Many parsers these days use PCFGs.

Slide adapted from John Canny, University of California, Berkeley, 2014
Systems

• **NLTK**: Python-based NLP system. Many modules, good visualization tools.

• **Stanford Parser**: Another comprehensive suite of tools (also POS tagger)

• **Berkeley Parser**: Another parser system.

• Note: high-quality parsing is usually very slow, but see: [https://github.com/dlwh/puck](https://github.com/dlwh/puck)

• [https://estnltk.github.io/](https://estnltk.github.io/) - Estonian Natural Language ToolKit
Demo: Data science mini-project

CRISP-DM: cross-industrial standard process for data mining

Graph mining
- Examples of applications
- Properties of graphs
- Graph algorithms
- Graph pattern mining
- Graph neural networks

Natural language processing
- Bag-of-words and N-grams
- Parts of speech, grammars and parsing
  - Dependencies
  - Neural networks for NLP
Dependencies

In a constituency parse, there is no direct relation between the constituents and words from the sentence (except for leaf nodes which produce a single word).

In dependency parsing, the idea is to decompose the sentence into relations directly between words.

This is an older, and some argue more natural, decomposition of the sentence. It also often makes semantic interpretation (based on the meanings of the words) easier.

Let's look at a simple example:

Slide adapted from John Canny, University of California, Berkeley, 2014
Dependencies

“The cat sat on the mat”

dependency tree

parse tree

constituency labels of leaf nodes

Slide adapted from John Canny, University of California, Berkeley, 2014
“Brave Merida prepared for a long, cold winter”
“Russell reveals himself here as a supremely gifted director of actors”
Dependencies

In Stanford Parser the dependencies are constructed from the output of a constituency parser (so you can in principle use other parsers).

The mapping is based on hand-written regular expressions.

Dependency grammars have been widely used for sentiment analysis and for semantic embeddings of sentences.

Slide adapted from John Canny, University of California, Berkeley, 2014
Lecture 11 – CRISP-DM, Graphs, NLP

- Demo: Data science mini-project
- CRISP-DM: cross-industrial standard process for data mining
- Graph mining
  - Examples of applications
  - Properties of graphs
  - Graph algorithms
  - Graph pattern mining
  - Graph neural networks
- Natural language processing
  - Bag-of-words and N-grams
  - Parts of speech, grammars and parsing
  - Dependencies
- Neural networks for NLP
Modern NLP techniques

• Most modern NLP techniques are based on artificial neural networks

• Large language models
  – Typically trained to predict a word from earlier text or from the surrounding context
  – The neural network models usually have the \textit{transformer} architecture which uses many self-attention modules
Demo: Data science mini-project
CRISP-DM: cross-industrial standard process for data mining
Graph mining
  Examples of applications
  Properties of graphs
  Graph algorithms
  Graph pattern mining
  Graph neural networks
Natural language processing
  Bag-of-words and N-grams
  Parts of speech, grammars and parsing
  Dependencies
  Neural networks for NLP