GETTIN FUZZY WIT IT

APPROXIMATE STRING MATCHING IN PYTHON

INVESTIGATION PRESENTATION

WILL HUGUENIN

MAY 2, 2016

QUESTION: WHAT HAPPENS WHEN YOU TRY TO INNER JOIN THESE TWO TABLES?

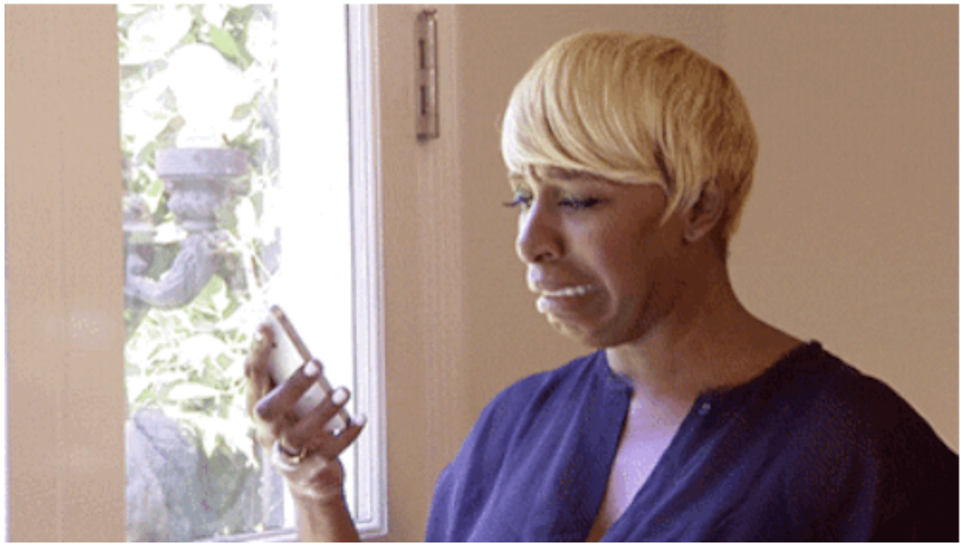
College	Some Data You Need
Harvard University	A
MIT	B
Princeton University	C
The Ohio State University	D

↔

College	Other Data You Need
Harvard College	1
The Massachusetts Institute of Tech	2
Princeton	3
Ohio State University	4

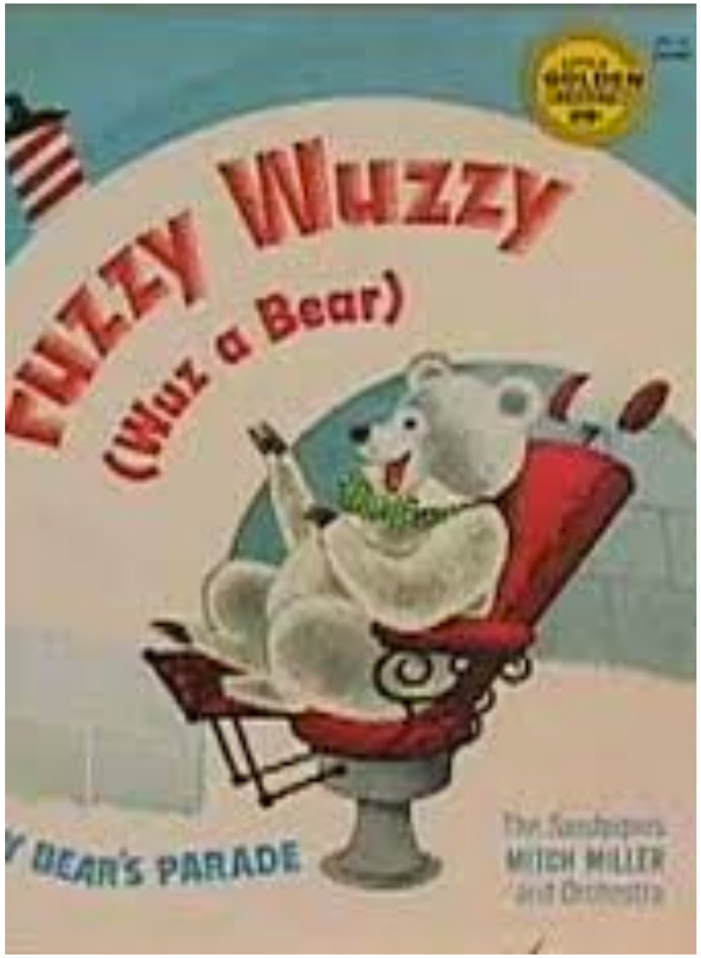
ANSWER: AGONY, RAGE AND DESPAIR

College	Some Data You Need	Some Other Data You Need
Nope	*Nothing*	*Here*



INTRODUCING: FUZZYWUZZY

- Python library for quantifying string similarity and matching *approximately* similar strings.
- Developed by SeatGeek
- Built on top of python difflib library.
- Easy Command-Line Installation:
  - pip install fuzzywuzzy



QUANTIFYING STRING SIMILARITY:
THE RATCLIFF-OBERSHELP ALGORITHM

$$R = \frac{2M}{T} X100$$

Where,

R is the string similarity ratio (as a %)

M is the number of matching characters

(i.e. number of characters in the longest matching substring plus all matching characters on both sides of longest matching substring)

T is the sum of the length of the two strings

EXAMPLE:
IS ANNE ACTUALLY ANNIE?

(1)	Identify longest common substring	ANNE : ANNIE
(2)	Identify common characters on either side of LCS	ANNE: ANNIE
(3)	Compute M (Length of LCS+N common characters on either side of LCS	M=(3+1)=4
(4)	Compute similarity ratio - 2M/ T (combined length of strings)	R = (2x4)/(4+5) = 8/9 ~ **89%**

PROBLEM:

THE "TRADITIONAL" SIMILARITY RATIO FAILS TO BE OPTIMAL WHEN ONE STRING IS SIGNIFICANTLY LONGER THAN THE OTHER.

String 1 (Shorter)	String 2 (Longer)	Ratio
Harvard	Harvard University	56%

SOLUTION: THE PARTIAL RATIO

- **Start with the shorter of the two strings.**
- **Compare the shorter string to EACH substring in the longer string of the same length as the shorter string.**
- **Compute each similarity ratio**
- **Partial Ratio = Maximum ratio calculated**

Partial Ratio

String 1	String 2	Ratio
Harvard	**Harvard** University	100%
Harvard	H**arvard** University	86%
Harvard	Ha**rvard U**niversity	71%
	…	
Harvard	Harvard Uni**versity**	14%

DEALING WITH DISORDER: TOKEN SORT & PARTIAL TOKEN SORT

- *Sort* tokens (words) in strings before computing ratios or partial ratios
- Useful when similar strings are constructed differently (e.g. "Harvard University" vs. "University of Harvard")

	String 1	String 2	Ratio	Partial Ratio
Original:	Harvard University	University of Harvard	51%	56%
Sorted:	Harvard University	Harvard of University	92%	83%

DON’T SWEAT THE SMALL STUFF: TOKEN SET & PARTIAL TOKEN SET

- **When using token sort, small words (e.g. “of”) can get in the way.**
- **Instead we could use Token Set:**
  - Select an *intersection* of common tokens from both strings.
  - Return maximum ratio/ partial ratio for:
    - The sorted intersection vs.
    - The sorted intersection + Sorted remaining words from string 1 vs.
    - The sorted intersection + Sorted remaining words from string 2.

TAKE 2:
UNIVERSITY OF HARVARD VS. HARVARD UNIVERSITY

	String 1	String 2	Ratio	Partial Ratio
Original:	Harvard University	University of Harvard	51%	56%
I vs. I+R1:	Harvard University	Harvard University	100%	100%
I vs. I+R2:	Harvard University	Harvard University of	92%	100%
I+R1 vs. I+R2	Harvard University	Harvard University of	92%	100%

PYTHON IMPLEMENTATION:

Ratio	Python Code
Ratio	fuzzywuzzy.fuzz.ratio(<string 1>, <string 2>)
Partial Ratio	fuzzywuzzy.fuzz.partial_ratio(<string 1>, <string 2>)
Token Sort Ratio	fuzzywuzzy.fuzz.token_sort_ratio(<string 1>, <string 2>)
Token Sort Partial Ratio	fuzzywuzzy.fuzz.partial_token_sort_ratio(<string 1>, <sting 2>)
Token Set Ratio	fuzzywuzzy.fuzz.token_set_ratio(<string 1>, <string 2>)
Token Set Partial Ratio	fuzzywuzzy.fuzz.partial_token_set_ratio(<string 1>, <string 2>)

BUT HOW CAN I USE THIS?
(THE PROCESS MODULE)

- **FuzzyWuzzy's Process Module contains two useful functions for extracting the best matches from a list:**
  - extract
  - extractOne

- **Scores matches based on the maximum of:**
  - Standard ratio, weighted token sort ratio, and weighted token set ratio (for strings of similar length), or
  - Standard ratio, weighted partial ratio, weighted partial token sort ratio, and weighted partial token set ratio (for strings of discrepant lengths).

- **Let's turn out attention over to the Ipython Notebook to see this in practice →**

SOURCES CONSULTED

https://github.com/seatgeek/fuzzywuzzy

http://chairnerd.seatgeek.com/fuzzywuzzy-fuzzy-string-matching-in-python/

https://docs.python.org/2/library/difflib.html

http://www.morfoedro.it/doc.php?n=223&lang=en

http://stackoverflow.com/questions/13636848/is-it-possible-to-do-fuzzy-match-merge-with-python-pandas