TRIFACTA

Programming by Example: Challenges and Opportunities

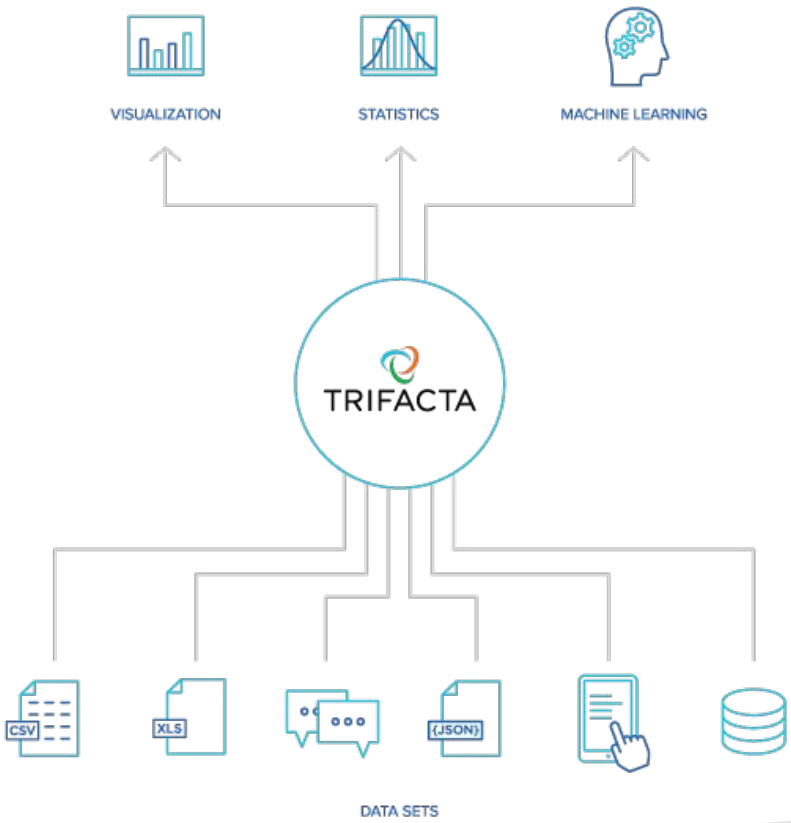
Anish Doshi

What this talk will cover

- What **programming by example** (PBE) is
- **Algorithms** for solving the PBE problem
- **Integrating** it into Trifacta, a production data application
- How we enable PBE to become a **user data-driven** feature

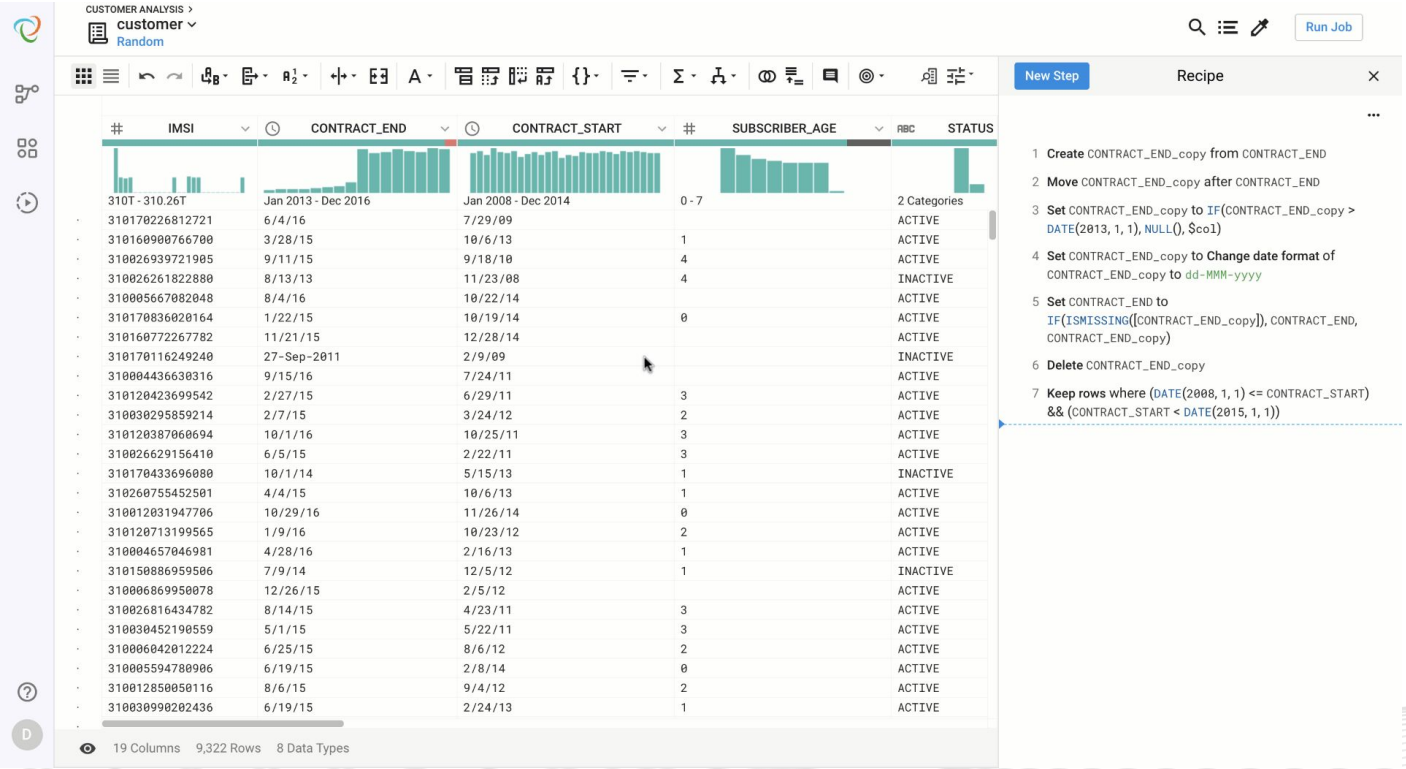
What Trifacta Is

- Data Preparation Platform - Focus on *Data Cleaning* for analytics/ML
- Data scientists can spend 80% of their time cleaning, validating, and preparing their data



What Trifacta Is

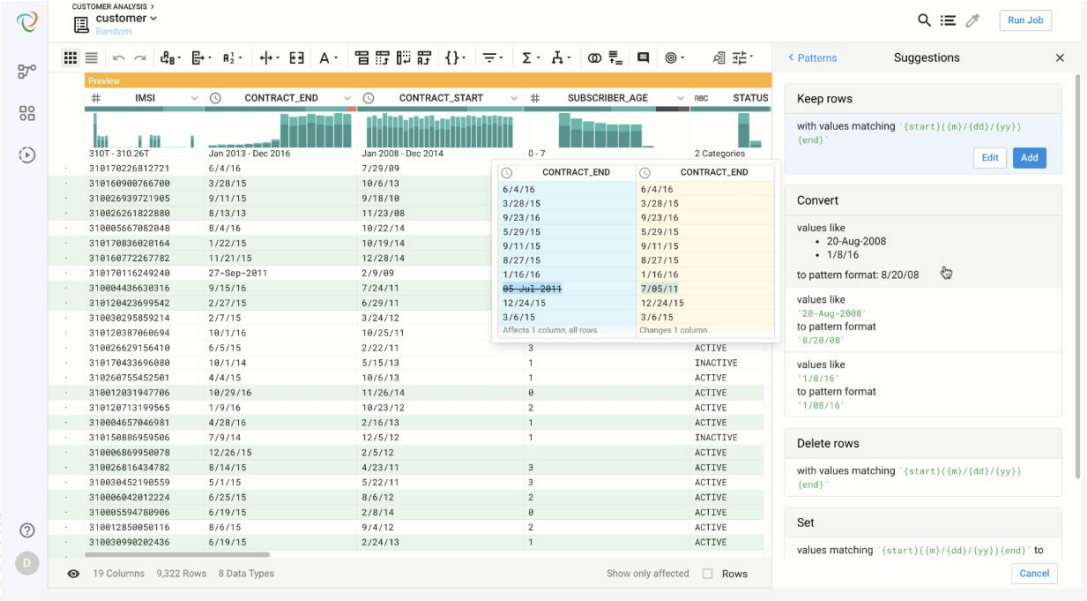
- Interactive, "Excel Like" page for seeing, visualizing, and transforming data



IMSI	CONTRACT_END	CONTRACT_START	SUBSCRIBER_AGE	STATUS
310T - 310.26T	Jan 2013 - Dec 2016	Jan 2008 - Dec 2014	0 - 7	2 Categories
310170226812721	6/4/16	7/29/09		ACTIVE
310160900766700	3/28/15	10/6/13	1	ACTIVE
310026939721905	9/11/15	9/18/10	4	ACTIVE
310026261822880	8/13/13	11/23/08	4	INACTIVE
310005667082048	8/4/16	10/22/14		ACTIVE
310170836020164	1/22/15	10/19/14	0	ACTIVE
310160772267782	11/21/15	12/28/14		ACTIVE
310170116249240	27-Sep-2011	2/9/09		INACTIVE
310004436630316	9/15/16	7/24/11		ACTIVE
310120423699542	2/27/15	6/29/11	3	ACTIVE
310030295859214	2/7/15	3/24/12	2	ACTIVE
310120387060694	10/1/16	10/25/11	3	ACTIVE
310026629156410	6/5/15	2/22/11	3	ACTIVE
310170433696080	10/1/14	5/15/13	1	INACTIVE
310260755452501	4/4/15	10/6/13	1	ACTIVE
310012031947706	10/29/16	11/26/14	0	ACTIVE
310120713199565	1/9/16	10/23/12	2	ACTIVE
310004657046981	4/28/16	2/16/13	1	ACTIVE
310150886959506	7/9/14	12/5/12	1	INACTIVE
310006869950078	12/26/15	2/5/12		ACTIVE
310026816434782	8/14/15	4/23/11	3	ACTIVE
310030452190559	5/1/15	5/22/11	3	ACTIVE
310006042012224	6/25/15	8/6/12	2	ACTIVE
310005594780906	6/19/15	2/8/14	0	ACTIVE
310012850050116	8/6/15	9/4/12	2	ACTIVE
310030990202436	6/19/15	2/24/13	1	ACTIVE

Data cleaning involves...Stuff with Strings

- Dates, Phone Numbers, Addresses, Currencies, Floats, Emails, URLs
- User often wants to *standardize* a column to a single format
- Existing solution is in regex transformations / limited pattern standardization



Cleaning messy data: Standardization

'949-727-6490'	+1 949 727 6490
'282-854-3328'	+1 282 854 3328
'972-399-4667'	+1 972 399 4667
'4046985388'	+1 404 698 5388
'6177590287'	+1 617 759 0287
'8133663103'	+1 813 366 3103
'(315) 732-3363'	+1 315 732 3363
'(315) 317-2248'	+1 315 317 2248

```sql
    CREATE FUNCTION [dbo].[ufn_FormatPhone]
    (@PhoneNumber VARCHAR(32))
RETURNS VARCHAR(32)
AS
  BEGIN
    DECLARE  @Phone CHAR(32)

    SET @Phone = @PhoneNumber

    -- cleanse phone number string
    WHILE PATINDEX('%[^0-9]%',@PhoneNumber) > 0
      SET @PhoneNumber = REPLACE(@PhoneNumber,
               SUBSTRING(@PhoneNumber,PATINDEX('%[^0-9]%',@PhoneNumber),1),'')

    -- skip foreign phones
    IF (SUBSTRING(@PhoneNumber,1,1) = '1'
         OR SUBSTRING(@PhoneNumber,1,1) = '+'
         OR SUBSTRING(@PhoneNumber,1,1) = '0')
       AND LEN(@PhoneNumber) > 11
      RETURN @Phone

    -- build US standard phone number
    SET @Phone = @PhoneNumber

    SET @PhoneNumber = '(' + SUBSTRING(@PhoneNumber,1,3) + ') ' +
             SUBSTRING(@PhoneNumber,4,3) + '-' + SUBSTRING(@PhoneNumber,7,4)

    IF LEN(@Phone) - 10 > 1
      SET @PhoneNumber = @PhoneNumber + ' X' + SUBSTRING(@Phone,11,LEN(@Phone) - 10)

    RETURN @PhoneNumber
  END
```

(taken from stackoverflow)

What if you could just tell it what you want it too look like?

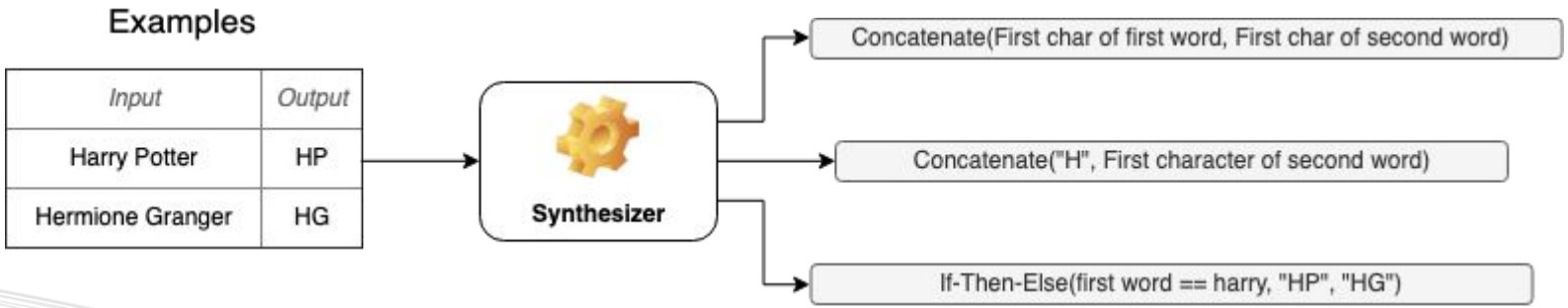
'949-727-6490'	+1 949 727 6490

In PBE, rather than specifying the program directly, the user specifies input/output **examples**, and the machine figures out the program the user would like to craft

Building a PBE Algorithm

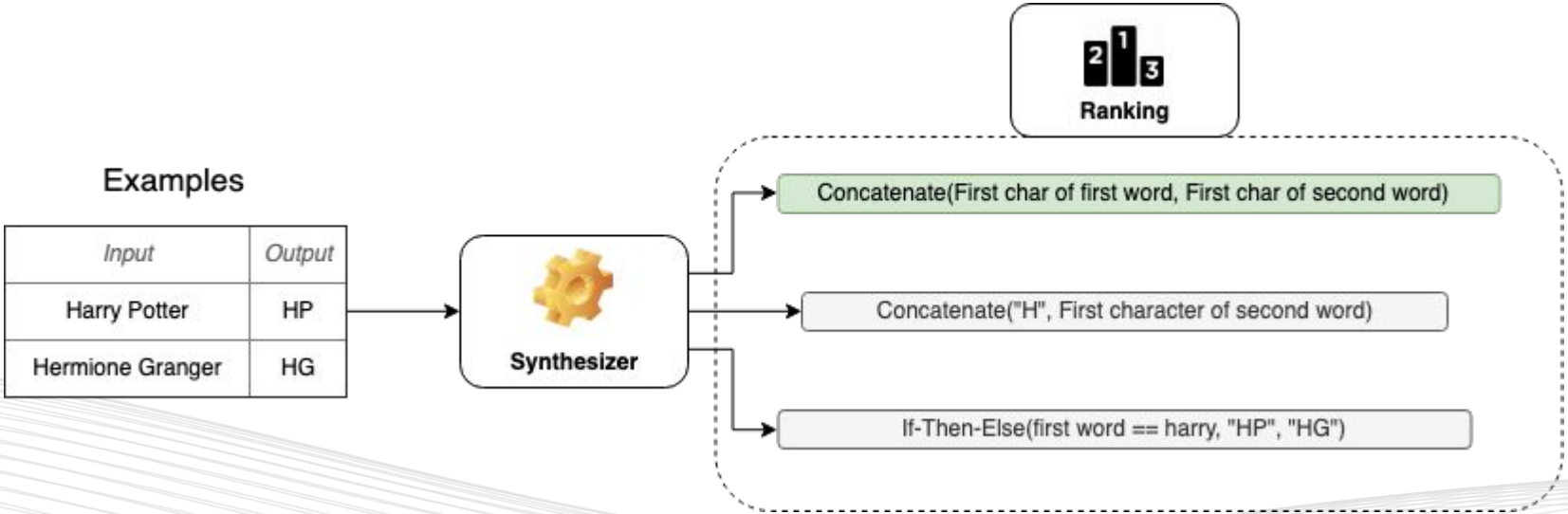
How it works

- General Idea: Given a set of input and output examples,
  - *synthesize* a set of programs that could represent that state



How it works

- General Idea: Given a set of input and output examples,
  - ***synthesize*** a set of programs that could represent that state
  - then ***rank*** them to pick the best one



Synthesis

- Domain specific languages (the language programs are written in, e.g. SQL) are usually too big to synthesize over
  - Large numbers of functions
  - Nesting
  - Multi-step programs
  - Numeric + String parameters
- Most PBE systems therefore restrict the DSL to something smaller, more task oriented
  - String Formatting DSL
  - Supports operations like Substring(), Concat(), Upper/Lowercasing

FlashFill (Gulwani 2011)

- First real software application of PBE (shipped in Microsoft Excel 2013)

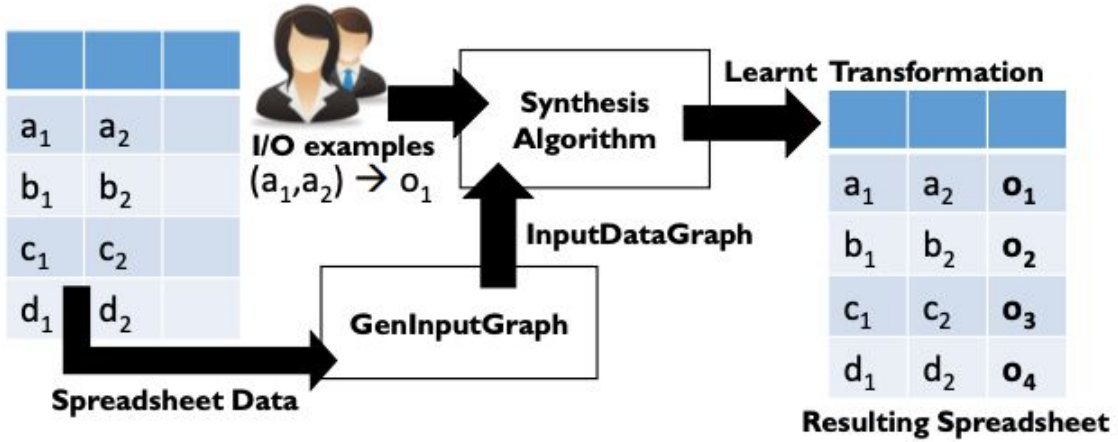
Input v_1	*Output*
(6/7)(4/5)(14/1)	*6/7 # 4/5 # 14/1 #*
49(28/11)(14/1)	*28/11 # 14/1 #*
() (28/11)(14/1)	*28/11 # 14/1 #*

String Program:

$\texttt{Loop}(\lambda w : \texttt{Concatenate}(\texttt{SubStr}(v_1, p_1, p_2), \texttt{ConstStr}(\text{“\# ”})))$
where $p_1 \equiv \texttt{Pos}(\mathit{LeftParenTok}, \texttt{TokenSeq}(\mathit{NumTok}, \mathit{SlashTok}), w))$
and $p_2 \equiv \texttt{Pos}(\texttt{TokenSeq}(\mathit{SlashTok}, \mathit{NumTok}), \mathit{RightParenTok}, w)$.

BlinkFill (Singh 2016)

- Idea: Programs should be semantically valid for the *whole column*, not just for input examples provided
- Space of such programs is also dramatically smaller, leading to increased performance (up to 40x as fast as FlashFill, according to authors)



Ranking: Heuristics

- Simplest: Occam's Razor (prefer simpler, shorter programs)

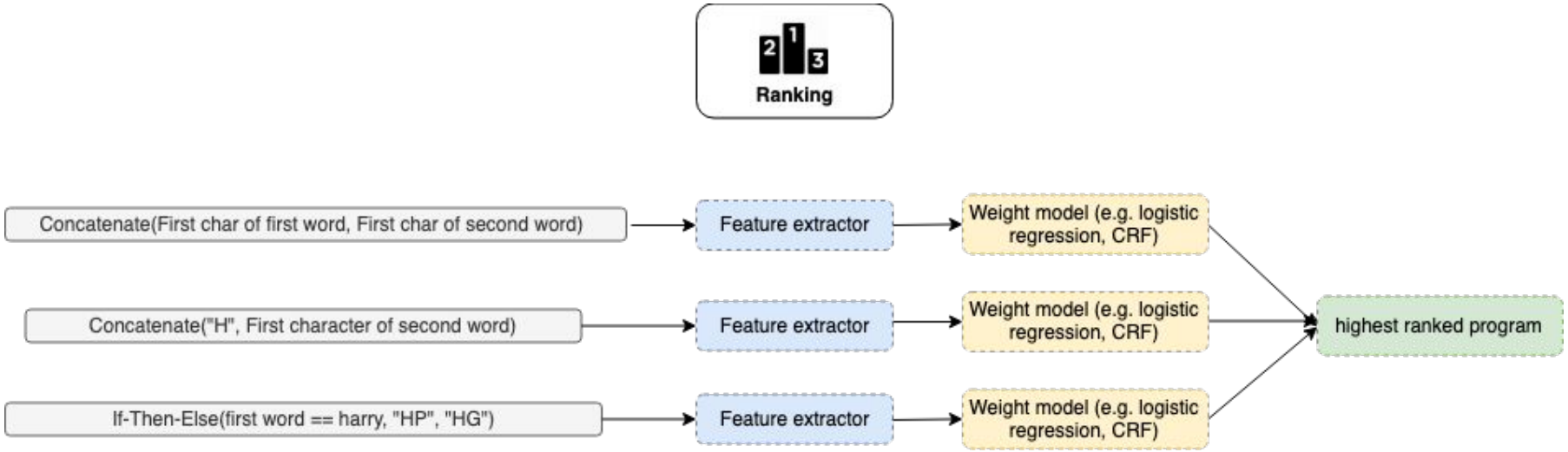
```
MERGE([SUBSTRING(column1, 0, 1),SUBSTRING(column1,
1, 2)], '')
```

```
SUBSTRING(column1, 0, 2)
```

Ranking

- More sophisticated:
  - Prefer certain functions (e.g. Propercase over UPPER + lower)
  - Prefer substring boundaries that end at delimiters
  - Use *metadata* about the column (e.g., use date formatting functions in a date column)
- Can we improve these heuristics by looking at user data?

Ranking with ML



mixture of hand tuned **heuristics** (feature extractor) and **ml** (weight models are trained on data)

Ranking with ML: Challenges in Production

- Training Data: simply look at hand crafted transformations!
- I.E. - save data before a transformation, data afterwards as a set of *input examples*, save the transformation itself as the *output program*
- Operations that people are doing on your product are a great source of training data
- Personalization potentially possible through transfer learning

Ranking with ML: Challenges in Production

- How do you train models on user data while respecting data privacy?
  - Ideal is online trained models, but those may be hard to deploy
  - Another strategy: Mask sensitive fields in analytics pipeline
    - Fields like SSN, credit card numbers, email addresses should be "masked" before saving

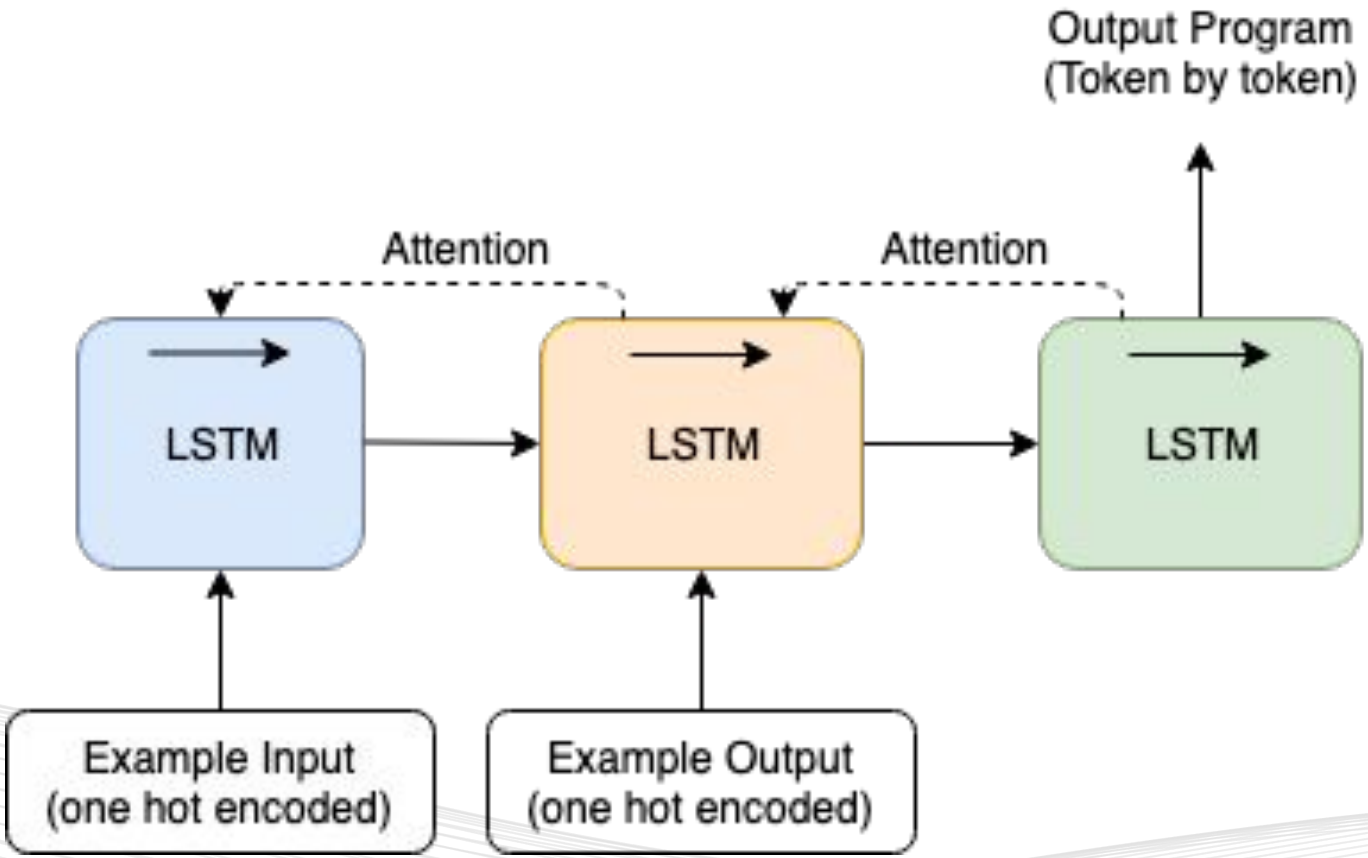
      original: 123-45-6789 -> 123 45 6789
      masked: 999-99-9999 -> 999 99 9999

    - Model still has access to the informational content of the pattern transformation

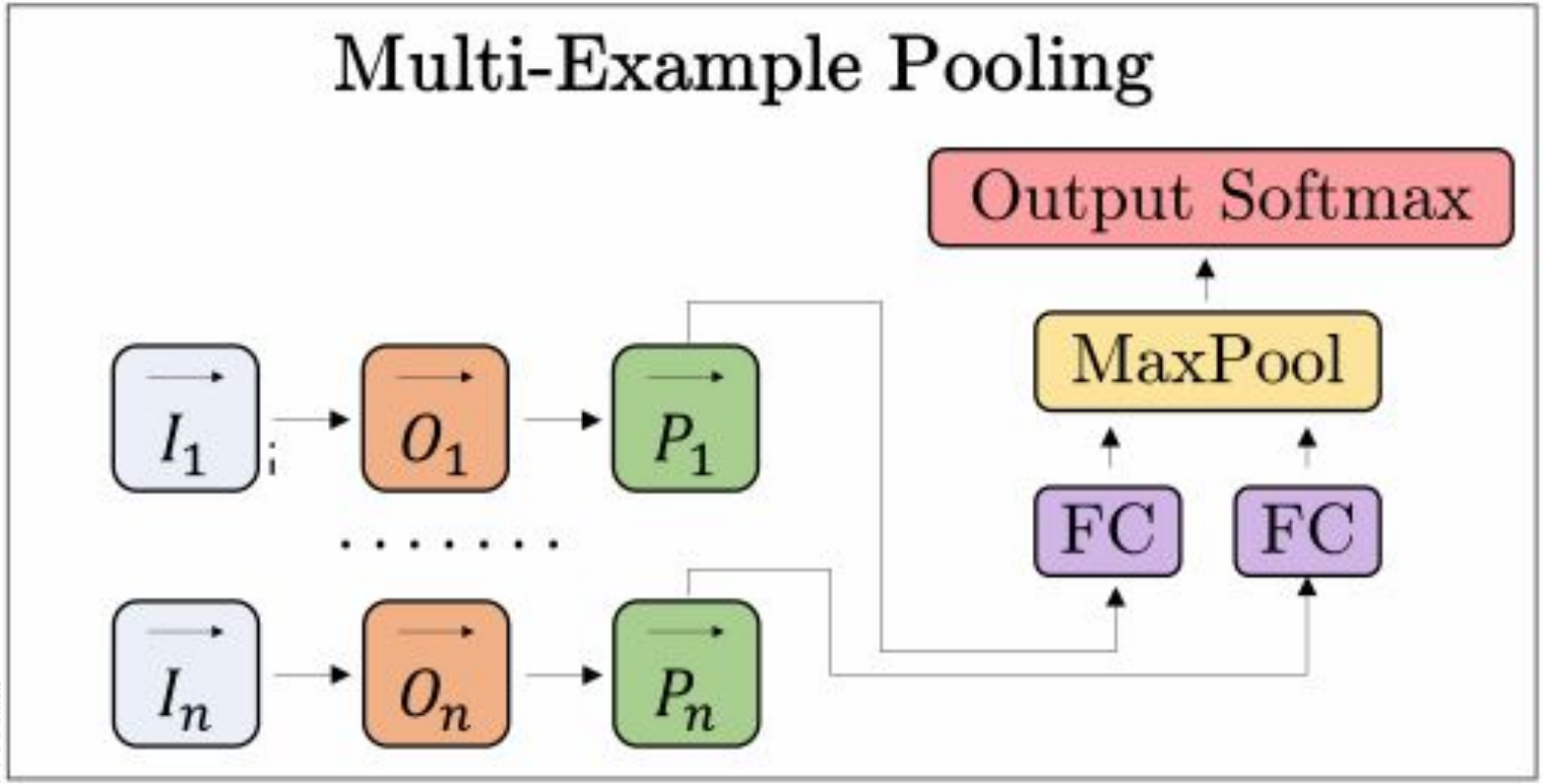
Neural Programming by Example

- Idea - Train a neural network directly to output a program given some encoding of input/output examples
- "Output a program" can mean a bunch of things:
  - Selecting a program from a preset list (a classification problem)
    - Hard to predict on such a large space - maybe prefilter to a threshold amount using heuristics, and then predict
  - Write out a program token by token (e.g. with an RNN)
  - Output a vector in some embedding space, and then find the closest valid program that satisfies the validity constraint
- Program Synthesis ≠ Program Induction

RobustFill (Devlin, Uesato et al. 2017)



RobustFill (Devlin, Uesato et al. 2017)



RobustFill (Devlin, Uesato et al. 2017)

- How do you make sure the generated program actually works?
  - Uses a **modified beam search** when outputting program tokens to make sure the program result is as consistent with the examples as possible.
  - Relies on nature of the DSL (String concatenation based DSL similar to FlashFill/BlinkFill)
- Pros
  - Continuous space, so tolerant to noise in examples (e.g. typos)
  - Could be trained on data directly, no need for custom heuristics
- Cons
  - Potentially hard to interpret results
  - Hard to verify determinism

Neural Programming by Example: Challenges in Production

- Deployment
  - How do you make sure the prediction step happens in a scalable way?
  - Where do you store the neural network's weights, which can be quite large?
- Testing
  - How do you make guarantees on an inherently probabilistic operation?
  - Can you make guarantees about the number of examples it takes to output a correct program?
- Usability
  - How would users provide feedback to the operation of the network?

Building a User Interface for PBE

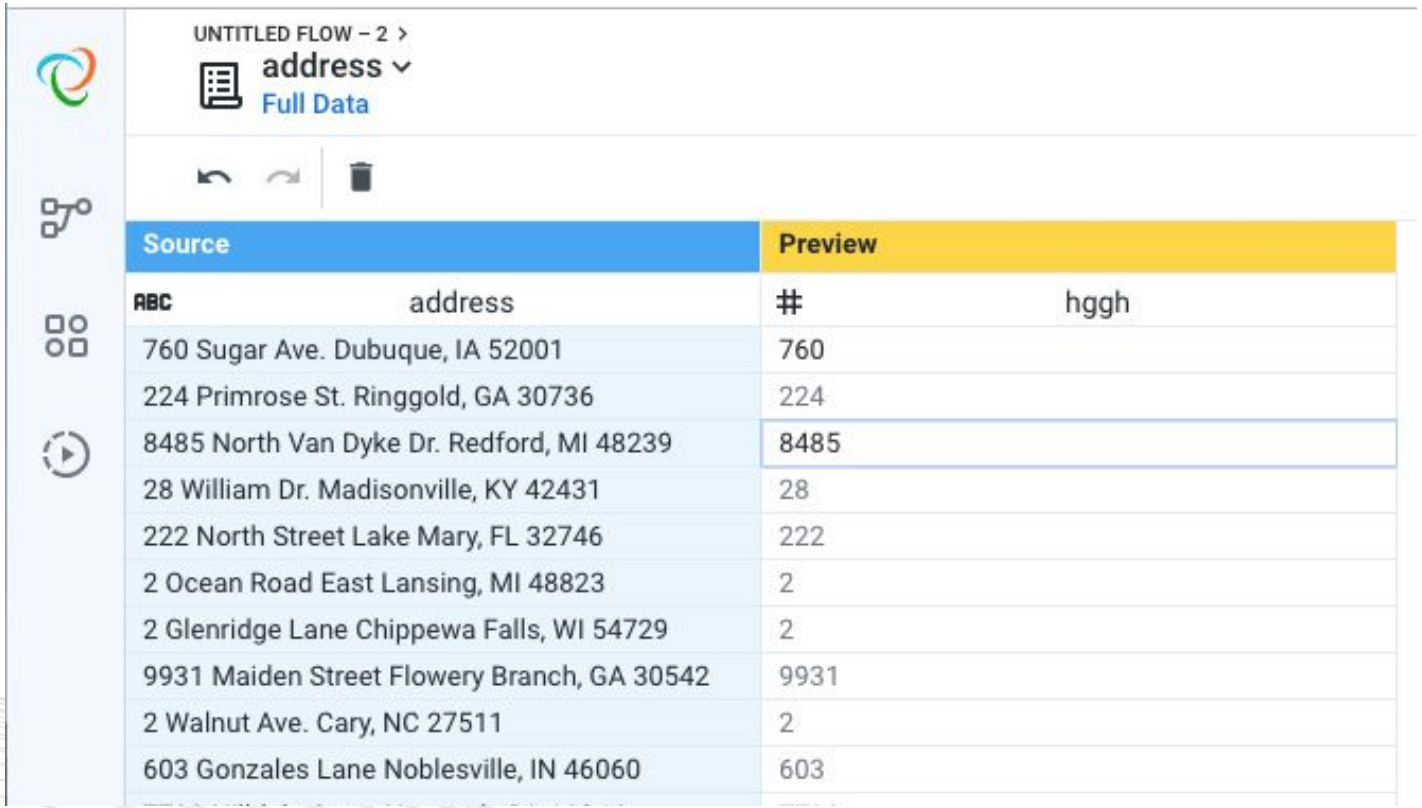
Started with a prototype

PBE Demo Upload file Phone Numbers (example) Cluster

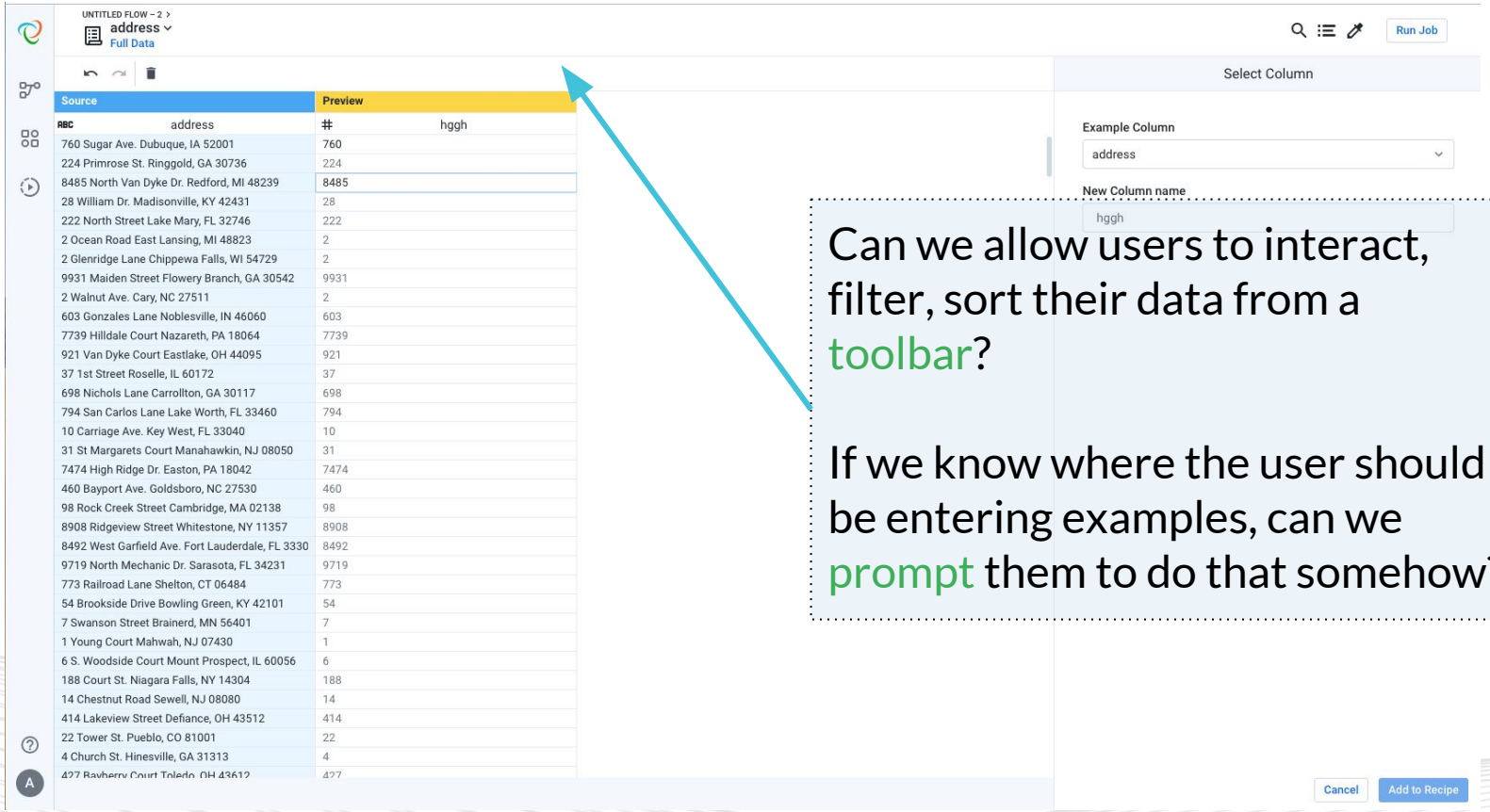
Inputs	Outputs
(908) 902-5338	
(302) 012-3489	
(124) 533-1207	
(128) 001-0123	
(963) 456-2039	

Interactivity and **Previewing** are important

Same basic idea applied in our main application...



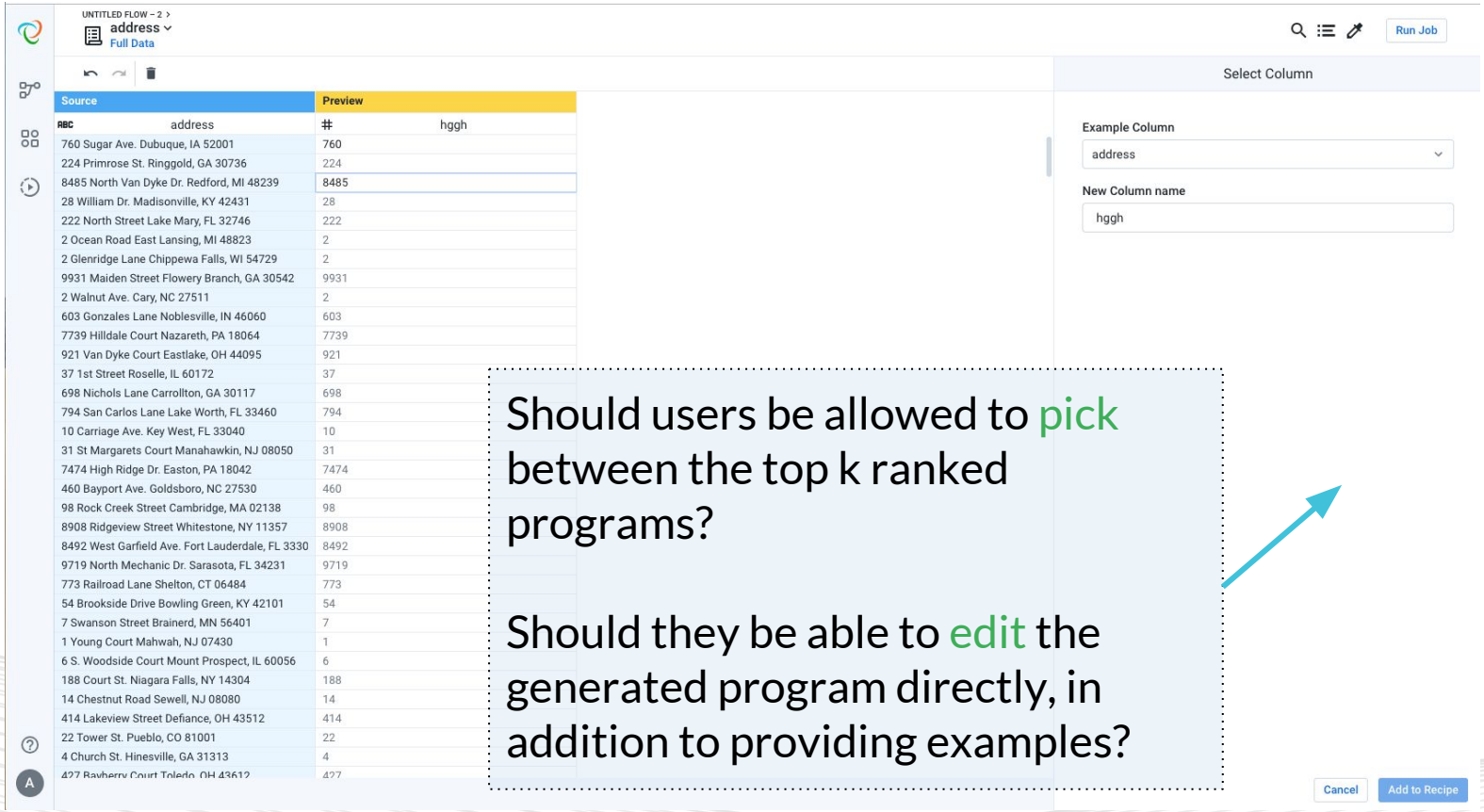
...but that raised a lot more questions



Can we allow users to interact, filter, sort their data from a toolbar?

If we know where the user should be entering examples, can we prompt them to do that somehow?

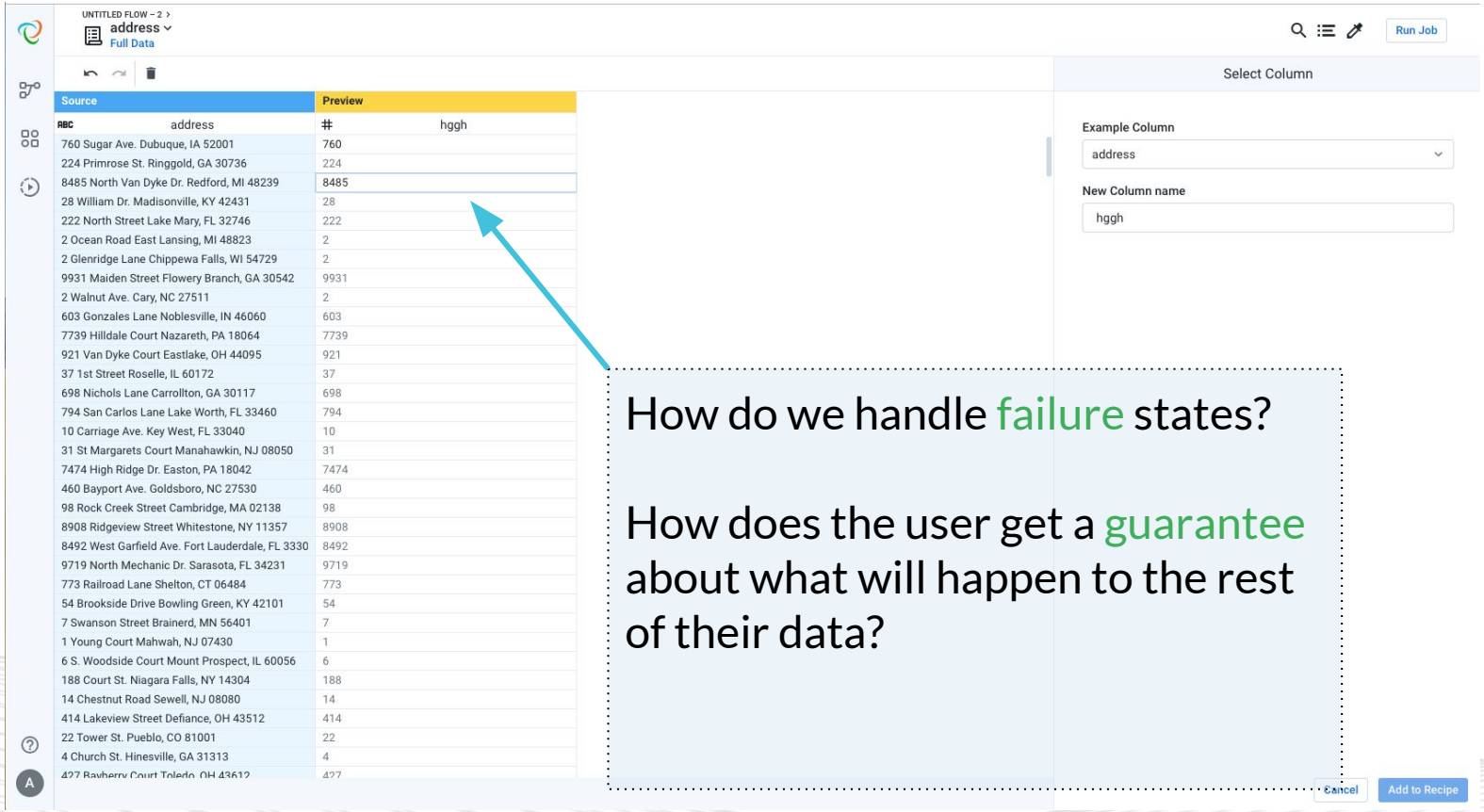
...but that raised a lot more questions



Should users be allowed to pick between the top k ranked programs?

Should they be able to edit the generated program directly, in addition to providing examples?

...but that raised a lot more questions



How do we handle failure states?

How does the user get a guarantee about what will happen to the rest of their data?

Key Takeaways

- Programming by Example is a methodology for users to interact with data in new way
- Tradeoffs between ML and heuristics, in expressibility and determinism
- Building it requires full stack, cross-disciplinary thought

Darin Friedrichs
@crushspread

Follow

AI is going to take over the world... and this is what Excel auto-populated today.

DEC	December
NOV	November
OCT	Octember
APR	Aprember
AUG	Augember
FEB	Febember
JAN	Janember
JUL	Julember
JUN	Junember
MAR	Marember
MAY	Mayember
SEP	Sepember

5:00 AM - 23 Oct 2018

Questions + Thanks!

www.trifacta.com

adoshi@trifacta.com