Programming by Examples

Sumit Gulwani
Microsoft

ECML/PKDD Conference
Sep 2019
Example-based help-forum interaction

300_w30_aniSh_c1_b → w30
300_w5_aniSh_c1_b → w5

=A HUGE Thank you!!!!

=MID(B1,5,2)

=MID(B1,FIND(“_”,$B:$B)+1,
FIND(“_”,REPLACE($B:$B,1,FIND(“_”,$B:$B),””))-1)
Flash Fill (Excel feature)

Excel 2013’s coolest new feature that should have been available years ago

“Automating string processing in spreadsheets using input-output examples”
[POPL 2011] Sumit Gulwani
AI is going to take over the world... and this is what Excel auto-populated today.

<table>
<thead>
<tr>
<th>K</th>
<th>L</th>
<th>M</th>
<th>N</th>
</tr>
</thead>
<tbody>
<tr>
<td>DEC</td>
<td>December</td>
<td></td>
<td></td>
</tr>
<tr>
<td>NOV</td>
<td>November</td>
<td></td>
<td></td>
</tr>
<tr>
<td>OCT</td>
<td>October</td>
<td></td>
<td></td>
</tr>
<tr>
<td>APR</td>
<td>April</td>
<td></td>
<td>Aprember</td>
</tr>
<tr>
<td>AUG</td>
<td>August</td>
<td></td>
<td>Augember</td>
</tr>
<tr>
<td>FEB</td>
<td>February</td>
<td></td>
<td>Febember</td>
</tr>
<tr>
<td>JAN</td>
<td>January</td>
<td></td>
<td>Janember</td>
</tr>
<tr>
<td>JUL</td>
<td>July</td>
<td></td>
<td>Julember</td>
</tr>
<tr>
<td>JUN</td>
<td>June</td>
<td></td>
<td>Junember</td>
</tr>
<tr>
<td>MAR</td>
<td>March</td>
<td></td>
<td>Marember</td>
</tr>
<tr>
<td>MAY</td>
<td>May</td>
<td></td>
<td>Mayember</td>
</tr>
<tr>
<td>SEP</td>
<td>September</td>
<td></td>
<td>September</td>
</tr>
</tbody>
</table>
PROGRAM SYNTHESIS USING EXAMPLES

"AI is going to take over the world..."
October 23, 2018
<p>| | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>DEC</td>
<td>December</td>
</tr>
<tr>
<td>2</td>
<td>NOV</td>
<td>November</td>
</tr>
<tr>
<td>3</td>
<td>OCT</td>
<td>October</td>
</tr>
<tr>
<td>4</td>
<td>APR</td>
<td>April</td>
</tr>
<tr>
<td>5</td>
<td>AUG</td>
<td>August</td>
</tr>
<tr>
<td>6</td>
<td>FEB</td>
<td>February</td>
</tr>
<tr>
<td>7</td>
<td>JAN</td>
<td>January</td>
</tr>
<tr>
<td>8</td>
<td>JUL</td>
<td>July</td>
</tr>
<tr>
<td>9</td>
<td>JUN</td>
<td>June</td>
</tr>
<tr>
<td>10</td>
<td>MAR</td>
<td>March</td>
</tr>
<tr>
<td>11</td>
<td>MAY</td>
<td>May</td>
</tr>
<tr>
<td>12</td>
<td>SEP</td>
<td>September</td>
</tr>
<tr>
<td>13</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
## Number, DateTime Transformations

<table>
<thead>
<tr>
<th>Input</th>
<th>Output (round to 2 decimal places)</th>
<th>Excel/C#:</th>
<th>Python/C#:</th>
<th>Java:</th>
</tr>
</thead>
<tbody>
<tr>
<td>123.4567</td>
<td>123.46</td>
<td>#.00</td>
<td>.2f</td>
<td>#.##</td>
</tr>
<tr>
<td>123.4</td>
<td>123.40</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>78.234</td>
<td>78.23</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Input</th>
<th>Output (3-hour weekday bucket)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CEDAR AVE &amp; COTTAGE AVE; HORSHAM; 2015-12-11 @ 13:34:52;</td>
<td>Fri, 12PM - 3PM</td>
</tr>
<tr>
<td>RT202 PKWY; MONTGOMERY; 2016-01-13 @ 09:05:41-Station:STA18;</td>
<td>Wed, 9AM - 12PM</td>
</tr>
<tr>
<td>; UPPER GWYNEDD; 2015-12-11 @ 21:11:18;</td>
<td>Fri, 9PM - 12AM</td>
</tr>
</tbody>
</table>

[CAV 2012] “Synthesizing Number Transformations from Input-Output Examples”; Singh, Gulwani
[POPL 2015] “Transforming Spreadsheet data types using Examples”; Singh, Gulwani
FlashExtract: A Framework for data extraction by examples

[PLDI 2014] Vu Le, Sumit Gulwani
Table Reshaping

<table>
<thead>
<tr>
<th>Bureau of I.A.</th>
<th>Regional Dir.</th>
<th>Numbers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Niles C.</td>
<td>Tel: (800)645-8397</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (907)586-7252</td>
<td></td>
</tr>
<tr>
<td>Jean H.</td>
<td>Tel: (918)781-4600</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (918)781-4604</td>
<td></td>
</tr>
<tr>
<td>Frank K.</td>
<td>Tel: (615)564-6500</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Fax: (615)564-6701</td>
<td></td>
</tr>
</tbody>
</table>

FlashRelate
From few examples of rows in output table

<table>
<thead>
<tr>
<th>Tel</th>
<th>Fax</th>
</tr>
</thead>
<tbody>
<tr>
<td>Niles C.</td>
<td>(800)645-8397</td>
</tr>
<tr>
<td></td>
<td>(907)586-7252</td>
</tr>
<tr>
<td>Jean H.</td>
<td>(918)781-4600</td>
</tr>
<tr>
<td></td>
<td>(918)781-4604</td>
</tr>
<tr>
<td>Frank K.</td>
<td>(615)564-6500</td>
</tr>
<tr>
<td></td>
<td>(615)564-6701</td>
</tr>
</tbody>
</table>

50% spreadsheets are semi-structured.
KPMG, Deloitte budget millions of dollars for normalization.

“FlashRelate: Extracting Relational Data from Semi-Structured Spreadsheets Using Examples”
Huge search space
- Prune using Logical reasoning
- Guide using Machine learning

Under-specification
- Guess using Ranking (PL features, ML models)
- Interact: leverage extra inputs (clustering) and programs (execution)
Flash Fill DSL

\[
\text{Tuple}(\text{String } x_1, \ldots, \text{String } x_n) \rightarrow \text{String}
\]

top-level expr \( T := C \mid \text{ifThenElse}(B, C, T) \)

condition-free expr \( C := A \mid \text{Concat}(A, C) \)

atomic expression \( A := \text{SubStr}(X, P, P) \mid \text{ConstantString} \)

input string \( X := x_1 \mid x_2 \mid \ldots \)

position expression \( P := K \mid \text{Pos}(X, R_1, R_2, K) \)

\( K \)th position in \( X \) whose left/right side matches with \( R_1/R_2 \).
Search Idea 1: Deduction

Let \([G \models \phi]\) denote programs in grammar G that satisfy spec \(\phi\). \(\phi\) is a Boolean constraint over (input state \(i \mapsto\) output value \(o\)).

Divide-and-conquer style problem reduction

\[
[G \models \phi_1 \land \phi_2] = \text{Intersect}([G \models \phi_1], [G \models \phi_2])
\]

\[
= [G_1 \models \phi_2] \text{ where } G_1 = [G \models \phi_1]
\]

Let \(G := G_1 \mid G_2\)

\[
[G \models \phi] = [G_1 \models \phi] \mid [G_2 \models \phi]
\]
Search Idea 1: Deduction

Inverse Set: $F^{-1}(o) \overset{\text{def}}{=} \{ (u, v) \mid F(u, v) = o \}$

E.g. $\text{Concat}^{-1}("Abc") = \{ ("A", "bc"), ("Ab", "c"), \ldots \}$

Let $G := F(G_1, G_2)$
Let $F^{-1}(o)$ be $\{ (u, v), (u', v') \}$

$$[G \models (i \mapsto o)] = F([G_1 \models (i \mapsto u)], [G_2 \models (i \mapsto v)])$$
$$\mid F([G_1 \models (i \mapsto u')], [G_2 \models (i \mapsto v')])$$

“FlashMeta: A Framework for Inductive Program Synthesis”
[OOPSLA 2015] Alex Polozov, Sumit Gulwani
Search Idea 2: Learning

Machine Learning for ordering search
• Which grammar production to try first?
• Which sub-goal resulting from inverse semantics to try first?

Prediction based on supervised training
• standard LSTM architecture
• Training: 100s of tasks, 1 task yields 1000s of sub-problems.
• Results: Up to 20x speedup with average speedup of 1.67
Ranking Idea 1: Program Features

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>Vasu Singh</td>
<td>v.s.</td>
</tr>
<tr>
<td>Stuart Russell</td>
<td>s.r.</td>
</tr>
</tbody>
</table>

P1: $\text{Lower}(1^{\text{st}} \text{ char}) + \text{".s."}$
P2: $\text{Lower}(1^{\text{st}} \text{ char}) + \text{"."} + 3^{\text{rd}} \text{ char} + \text{"."}$
P3: $\text{Lower}(1^{\text{st}} \text{ char}) + \text{"."} + \text{Lower}(1^{\text{st}} \text{ char after space}) + \text{"."}$

Prefer programs (P3) with simpler Kolmogorov complexity
- Fewer constants
- Smaller constants

“Predicting a correct program in Programming by Example”
[CAV 2015] Rishabh Singh, Sumit Gulwani
Ranking Idea 2: Output Features

<table>
<thead>
<tr>
<th>Input</th>
<th>Output</th>
<th>Output of P1</th>
</tr>
</thead>
<tbody>
<tr>
<td>[CPT-123]</td>
<td>[CPT-123]</td>
<td>[CPT-123]</td>
</tr>
<tr>
<td>[CPT-456]</td>
<td>[CPT-456]</td>
<td>[CPT-456]</td>
</tr>
</tbody>
</table>

P1: Input + “]”
P2: Prefix of input upto 1st number + “]”

Examine features of outputs of a program on extra inputs:
- IsYear, Numeric Deviation, # of characters, IsPerson

“Learning to Learn Programs from Examples: Going Beyond Program Structure”
[IJCAI 2017] Kevin Ellis, Sumit Gulwani
Disambiguation

Communicate actionable information back to user.

Program-based disambiguation

• Enable effective navigation between top-ranked programs.
• Highlight ambiguity based on *distinguishing inputs*.

Heuristics that can be machine learned

• Highlight ambiguity based on clustering of inputs/outputs.
• When to stop highlighting ambiguity?

ML in PBE

Advantages
- Better models
- Less time to author
- Online adaptation, personalization

PBE Component → Logical strategies + Creative heuristics → Features + Model

Written by developers
Can be learned and maintained by ML-backed runtime

“Programming by Examples: PL meets ML”
[APLAS 2017] Sumit Gulwani, Prateek Jain
Mode-less Synthesis

Non-intrusively watch, learn, and make suggestions

Advantages: Usability, Avoids Discoverability

Applications: Document Editing, Code Refactoring, Robotic Process Automation

Key Idea: Identify related examples within noisy action traces

“On the Fly Synthesis of Edit Suggestions”
[OOPSLA 2019] Miltner, Gulwani, Le, Luang, Radhakrishna, Soares, Tiwari, Udupa
Predictive Synthesis

Synthesis of intended programs from just the input.

Predictive Synthesis: PBE :: Unsupervised : Supervised ML

Applications: Tabular data extraction, Join, Sort, Split

Key Idea: Structure inference over inputs
Synthesis of Readable Code

Synthesis in target language of choice.
• Python, R, Scala, PySpark

Advantages:
• Transparency
• Education
• Integration with existing workflows in IDEs, Notebooks

Challenges: Quantify readability, Quantitative PBE

Key Idea: Observationally-equivalent (but non-semantic preserving) transformation of an intended program
A match made in heaven!

PS can synthesize **small code fragments**. Sufficient for notebook cell-based programming.

PS can synthesize code in different languages. A good solution for **polyglot challenge** in notebooks.

PS needs **interactivity**. Notebooks provide that.
Other Topics in Program Synthesis

- **Search methodology**: Code repositories [Murali et al., ICLR 2018]

- **Language**: Neural program induction
  - [Graves et al., 2014; Reed & De Freitas, 2016; Zaremba et al., 2016]

- **Intent specification**:
  - Natural language [Huang et al., NAACL-HLT 2018; Gulwani, Marron SIGMOD 2014, Shin et al. NeurIPS 2019]
  - Conversational pair programming

- **Applications**:
  - Super-optimization for model training/inference
  - Personalized Learning [Gulwani; CACM 2014]
Conclusion

*Program Synthesis*: key to next-generational programming

- Future: Multi-modal programming with Examples and NL
- 100x more programmers
- 10-100x productivity increase in several domains.

Next-generational AI techniques under the hood

- Logical Reasoning + Machine Learning

Questions/Feedback: Contact me at sumitg@microsoft.com