FlashProfile

A Framework for Synthesizing Data Profiles

Saswat Padhi [1]† Prateek Jain [2] Daniel Perelman [3]

Oleksandr Polozov [4] Sumit Gulwani [3] Todd Millstein [1]

[1] University of California, Los Angeles, CA

[2] Microsoft Research Lab, India

[3] Microsoft Corporation, Redmond, WA

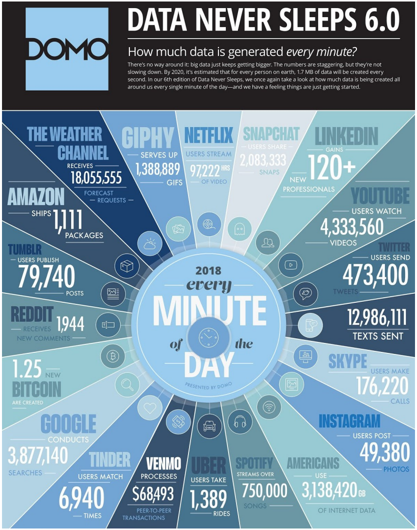
[4] Microsoft Research, Redmond, WA

SPLASH BOSTON 2018

(OOPSLA)

† Contributed during an internship with PROSE team at Microsoft

The Challenges of “Big” Data



The Challenges of “Big” Data

High *Volume*

> 2.5 M TB of data generated *every day*!

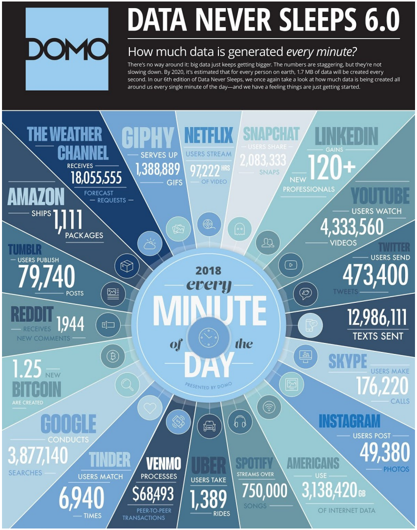
High *Velocity*

~ 4 M Google searches, ~ ½ M tweets,
> 1 K Amazon shipments … *per minute*!

High *Variety*

90 % of generated data is unstructured!

Data may be incomplete, inconsistent,
may contain multiple formats …



State of The Art

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

State of The Art

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

Microsoft SSDT: (does not describe all data formats)

- doi:␣$^+$10\.\d\d\d\d\d/\d$^+$ (110)
- .* (113)
- ISBN:␣0-\d\d\d-\d\d\d\d\d-\d (204)
- PMC\d$^+$ (1024)

State of The Art

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

MICROSOFT SSDT: (does not describe all data formats)

- `doi:␣`$^+$`10\.\d\d\d\d\d/\d`$^+$ (110)
- `.`* (113)
- `ISBN:␣0-\d\d\d-\d\d\d\d\d-\d` (204)
- `PMC\d`$^+$ (1024)

ATACCAMA ONE: (coarse grained, no constants & fixed-width patterns)

- `W_W` (5)
- `W: N.N/LN-N(N)N-D` (11)
- `W: D-N-N-L` (34)
- `W: N.N/N` (110)
- `W: D-N-N-D` (267)
- `WN` (1024)

State of The Art

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

MICROSOFT SSDT: (does not describe all data formats)

- doi:$␣^+$10\.\d\d\d\d/\d$^+$ (110)
- .* (113)
- ISBN:␣0-\d\d\d-\d\d\d\d\d-\d (204)
- PMC\d$^+$ (1024)

ATACCAMA ONE: (coarse grained, no constants & fixed-width patterns)

- W_W (5)
- W: N.N/LN-N(N)N-D (11)
- W: D-N-N-L (34)
- W: N.N/N (110)
- W: D-N-N-D (267)
- WN (1024)

FLASHPROFILE:

- ‘not_available’ (5)
- ‘doi:’ $␣^+$ ‘10.1016/’ U D^4 ‘-’ D^4 ‘(’ D^2 ‘)’ D^5 ‘-’ D (11)
- ‘ISBN:’ ␣ D ‘-’ D^3 ‘-’ D^5 ‘-X’ (34)
- ‘doi:’ $␣^+$ ‘10.13039/’ D^+ (110)
- ‘ISBN:’ ␣ D ‘-’ D^3 ‘-’ D^5 ‘-’ D (267)
- ‘PMC’ D^7 (1024)

Can We Do Even Better?

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

Default profile from FlashProfile:

- 'not_available' (5)
- 'doi:' $␣^{+}$ '10.1016/' U D^{4} '-' D^{4} '(' D^{2} ')' D^{5} '-' D (11)
- 'ISBN:' ␣ D '-' D^{3} '-' D^{5} '-X' (34)
- 'doi:' $␣^{+}$ '10.13039/' D^{+} (110)
- 'ISBN:' ␣ D '-' D^{3} '-' D^{5} '-' D (267)
- 'PMC' D^{7} (1024)

Can We Do Even Better?

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

Allowing domain-experts to profile with custom patterns:

- ‘not_available’ (5)
- ‘doi:’ $␣^+$ DOI (121)
- ‘ISBN:’ ISBN_{10} (301)
- ‘PMC’ D^7 (1024)

Default profile from FlashProfile:

- ‘not_available’ (5)
- ‘doi:’ $␣^+$ ‘10.1016/’ U D^4 ‘-’ D^4 ‘(’ D^2 ‘)’ D^5 ‘-’ D (11)
- ‘ISBN:’ ␣ D ‘-’ D^3 ‘-’ D^5 ‘-X’ (34)
- ‘doi:’ $␣^+$ ‘10.13039/’ D^+ (110)
- ‘ISBN:’ ␣ D ‘-’ D^3 ‘-’ D^5 ‘-’ D (267)
- ‘PMC’ D^7 (1024)

Can We Do Even Better?

Reference ID
PMC5079771
doi:␣10.1016/S1387-7003(03)00113-8
ISBN:␣2-287-34069-6
⋮ ⋮ ⋮
ISBN:␣0-006-08903-1
PMC9473786
⋮ ⋮ ⋮
doi:␣␣10.13039/100005795
ISBN:␣1-158-23466-X
not_available
PMC9035311

Allowing domain-experts to profile with custom patterns:

- 'not_available' (5)
- 'doi:' $␣^+$ DOI (121)
- 'ISBN:' ISBN_{10} (301)
- 'PMC' D^7 (1024)

Interactive refinement to gradually drill into data:

- 'not_available' (5)
- 'doi:' $␣^+$ '10.1016/' U D^4 '-' D^4 '(' D^2 ')' D^5 '-' D (11)
- 'doi:' $␣^+$ '10.13039/' D^+ (110)
- 'ISBN:' ISBN_{10} (301)
- 'PMC' D^7 (1024)

Default profile from FlashProfile:

- 'not_available' (5)
- 'doi:' $␣^+$ '10.1016/' U D^4 '-' D^4 '(' D^2 ')' D^5 '-' D (11)
- 'ISBN:' ␣ D '-' D^3 '-' D^5 '-X' (34)
- 'doi:' $␣^+$ '10.13039/' D^+ (110)
- 'ISBN:' ␣ D '-' D^3 '-' D^5 '-' D (267)
- 'PMC' D^7 (1024)

Key Challenges

The space of profiles is large and inherently ambiguous

Key Challenges

The space of profiles is large and inherently ambiguous

Should {'`1817`', '`1813?`'} be generalized to a pattern, or {'`1817`', '`1907`'}?

Key Challenges

The space of profiles is large and inherently ambiguous

Should {'`1817`', '`1813?`'} be generalized to a pattern, or {'`1817`', '`1907`'}?

- prior tools have a fixed bias for {'`1817`', '`1907`'}

Key Challenges

The space of profiles is large and inherently ambiguous

Should {'1817', '1813?'} be generalized to a pattern, or {'1817', '1907'}?

- prior tools have a fixed bias for {'1817', '1907'}

We allow users to disambiguate, by defining custom domain-specific patterns

Key Challenges

The space of profiles is large and inherently ambiguous

Should {'1817', '1813?'} be generalized to a pattern, or {'1817', '1907'}?

- prior tools have a fixed bias for {'1817', '1907'}

We allow users to disambiguate, by defining custom domain-specific patterns

- for example, a user may define a pattern `1800s` (= the regex `18.*`)

Key Challenges

The space of profiles is large and inherently ambiguous

Should {'1817', '1813?'} be generalized to a pattern, or {'1817', '1907'}?

- prior tools have a fixed bias for {'1817', '1907'}

We allow users to disambiguate, by defining custom domain-specific patterns

- for example, a user may define a pattern `1800s` (= the regex `18.*`)
- *Exponentially* many ways of partitioning a given set of strings

Key Challenges

The space of profiles is large and inherently ambiguous

Should { '1817', '1813?' } be generalized to a pattern, or { '1817', '1907' }?

- prior tools have a fixed bias for { '1817', '1907' }

We allow users to disambiguate, by defining custom domain-specific patterns

- for example, a user may define a pattern `1800s` (= the regex `18.*`)
- *Exponentially* many ways of partitioning a given set of strings
- *Exponentially* many ways of generalizing strings to a pattern

Key Challenges

The space of profiles is large and inherently ambiguous

Should { '1817', '1813?' } be generalized to a pattern, or { '1817', '1907' }?

- prior tools have a fixed bias for { '1817', '1907' }

We allow users to disambiguate, by defining custom domain-specific patterns

- for example, a user may define a pattern `1800s` (= the regex `18.*`)
- *Exponentially* many ways of partitioning a given set of strings
- *Exponentially* many ways of generalizing strings to a pattern

Inductive program synthesis to the rescue!

Key Challenges

The space of profiles is large and inherently ambiguous

Should {'`1817`', '`1813?`'} be generalized to a pattern, or {'`1817`', '`1907`'}?

- prior tools have a fixed bias for {'`1817`', '`1907`'}

We allow users to disambiguate, by defining custom domain-specific patterns

- for example, a user may define a pattern `1800s` (= the regex `18.*`)
- *Exponentially* many ways of partitioning a given set of strings
- *Exponentially* many ways of generalizing strings to a pattern
  - Efficient synthesis of complex patterns

Inductive program synthesis to the rescue!

Key Challenges

The space of profiles is large and inherently ambiguous

Should {‘1817’, ‘1813?’} be generalized to a pattern, or {‘1817’, ‘1907’}?

- prior tools have a fixed bias for {‘1817’, ‘1907’}

We allow users to disambiguate, by defining custom domain-specific patterns

- for example, a user may define a pattern `1800s` (= the regex `18.*`)
- *Exponentially* many ways of partitioning a given set of strings
  - Clustering, with similarity ≈ Pattern *score*
- *Exponentially* many ways of generalizing strings to a pattern
  - Efficient synthesis of complex patterns

Inductive program synthesis to the rescue!

Main Contributions

An application of a supervised learning technique (inductive program synthesis) to the unsupervised learning problem of syntactic profiling.

Main Contributions

An application of a supervised learning technique (inductive program synthesis) to the unsupervised learning problem of syntactic profiling.

We present:

- a definition for syntactic profiling as a pattern-aware clustering problem

Main Contributions

An application of a supervised learning technique (inductive program synthesis) to the unsupervised learning problem of syntactic profiling.

We present:

- a definition for syntactic profiling as a pattern-aware clustering problem
- a technique using inductive program synthesis

Main Contributions

An application of a supervised learning technique (inductive program synthesis) to the unsupervised learning problem of syntactic profiling.

We present:

- a definition for syntactic profiling as a pattern-aware clustering problem
- a technique using inductive program synthesis
- practical optimizations for fast, approximate profiling

Main Contributions

An application of a supervised learning technique (inductive program synthesis) to the unsupervised learning problem of syntactic profiling.

We present:

- a definition for syntactic profiling as a pattern-aware clustering problem
- a technique using inductive program synthesis
- practical optimizations for fast, approximate profiling
- FlashProfile, and evaluation of its performance and accuracy

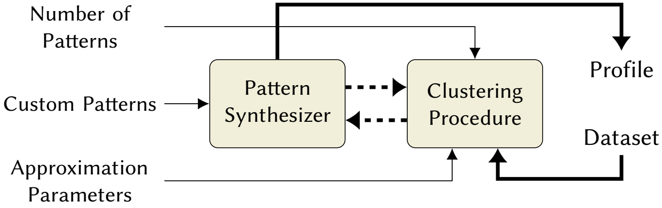
Main Contributions

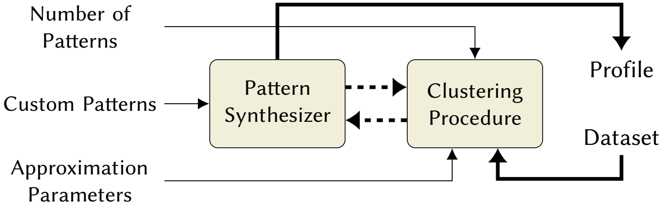
An application of a supervised learning technique (inductive program synthesis) to the unsupervised learning problem of syntactic profiling.

We present:

- a definition for syntactic profiling as a pattern-aware clustering problem
- a technique using inductive program synthesis
- practical optimizations for fast, approximate profiling
- FlashProfile, and evaluation of its performance and accuracy
- profile-guided interaction for traditional PBE workflows

Overview of FlashProfile

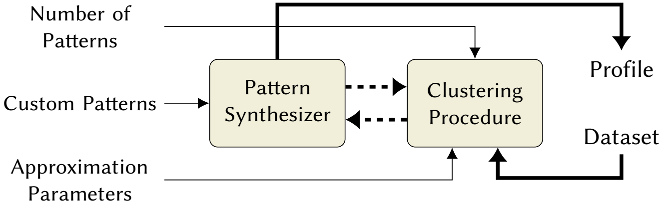




FlashProfile provides:

- Support for user-defined patterns

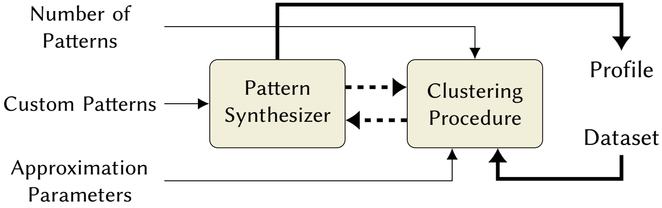
Overview of FlashProfile



FlashProfile provides:

- Support for user-defined patterns
- Support for arbitrary constants and fixed-width patterns

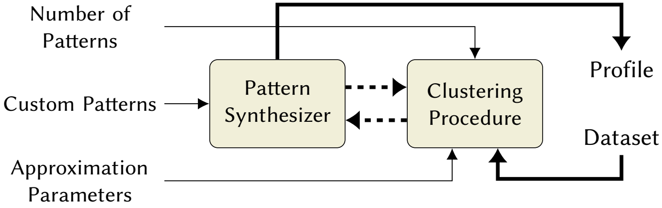
Overview of FlashProfile



FlashProfile provides:

- Support for user-defined patterns
- Support for arbitrary constants and fixed-width patterns
- Interactive refinement of profiles

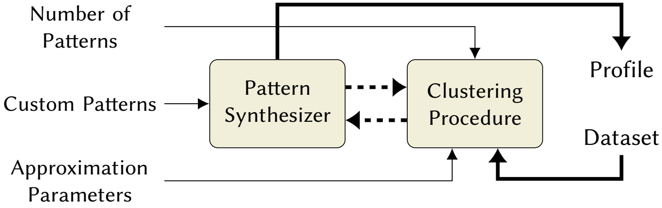
Overview of FlashProfile



FlashProfile provides:

- Support for user-defined patterns
- Support for arbitrary constants and fixed-width patterns
- Interactive refinement of profiles
- Control over accuracy *vs.* performance trade-off

Overview of FlashProfile

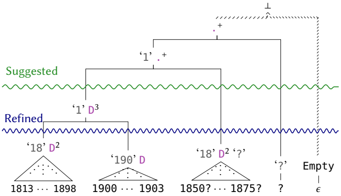


FlashProfile provides:

- Support for user-defined patterns
- Support for arbitrary constants and fixed-width patterns
- Interactive refinement of profiles
- Control over accuracy *vs.* performance trade-off

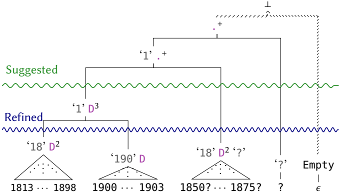
FlashProfile is publicly-available as a cross-platform C# library (`Matching.Text`), as part of the Microsoft PROSE SDK.

Profiling via Clustering



Profiling via Clustering

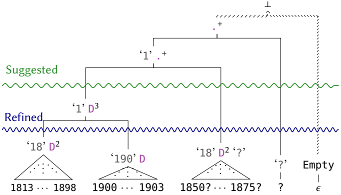
- Pattern-Aware Partitioning
  - *Clustering:* Agglomerative hierarchical clustering
  - *Objective:* Minimize the cost of describing partitions
  - *Similarity:* Minimum cost of describing 2 strings



(see our paper for details)

Profiling via Clustering

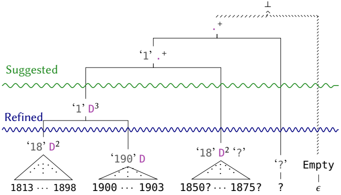
- Pattern-Aware Partitioning
  - *Clustering:* Agglomerative hierarchical clustering
  - *Objective:* Minimize the cost of describing partitions
  - *Similarity:* Minimum cost of describing 2 strings
- Dependencies
  - A pattern learner $\mathcal{L}$
  - A cost function $\mathcal{C}$



(see our paper for details)

Profiling via Clustering

- Pattern-Aware Partitioning
  - *Clustering:* Agglomerative hierarchical clustering
  - *Objective:* Minimize the cost of describing partitions
  - *Similarity:* Minimum cost of describing 2 strings
- Dependencies
  - A pattern learner $\mathcal{L}$
  - A cost function C
- Optimizations
  - Approximate similarity using previous patterns
  - Profiling small chunks → Full profile



(see our paper for details)

Pattern Synthesis

Pattern Synthesis

- A Language $\mathscr{L}_{\text{FP}}$:

$$
\begin{aligned}
\text{Pattern } P[s] \;&:=\; \texttt{Empty}(s) \\
&\;\;|\;\; P\left[\texttt{SuffixAfter}(s, \alpha)\right] \\
\text{Atom } \alpha \;&:=\; \text{Class}_c^n \;\;|\;\; \text{RegEx}_r \\
&\;\;|\;\; \text{Funct}_f \;\;|\;\; \text{Const}_s
\end{aligned}
$$

(see our paper for details)

Pattern Synthesis

- A Language $\mathscr{L}_{\text{FP}}$:

$$
\begin{aligned}
\text{Pattern } P[s] \;&:=\; \texttt{Empty}(s) \\
&\;\;|\;\; P\left[\texttt{SuffixAfter}(s, \alpha)\right] \\
\text{Atom } \alpha \;&:=\; \text{Class}_c^n \;\;|\;\; \text{RegEx}_r \\
&\;\;|\;\; \text{Funct}_f \;\;|\;\; \text{Const}_s
\end{aligned}
$$

- A Pattern Learner $\mathcal{L}_{\text{FP}}$
  - *recursively reduces* a synthesis problem
  - sound and complete over a given set of atoms

(see our paper for details)

Pattern Synthesis

- A Language $\mathscr{L}_{\mathsf{FP}}$:

$$
\begin{aligned}
\text{Pattern } P[s] \;&:=\; \texttt{Empty}(s) \\
&\;\;\mid\;\; P\left[\texttt{SuffixAfter}(s, \alpha)\right] \\
\text{Atom } \alpha \;&:=\; \text{Class}_c^n \;\mid\; \text{RegEx}_r \\
&\;\;\mid\;\; \text{Funct}_f \;\mid\; \text{Const}_s
\end{aligned}
$$

- A Pattern Learner $\mathcal{L}_{\mathsf{FP}}$
  - *recursively reduces* a synthesis problem
  - sound and complete over a given set of atoms
- A Cost Function C_{FP}
  - tradeoff between specificity and simplicity
  - *weighted sum* of costs of individual atoms

(see our paper for details)

Profile-Guided Interaction for PBE

Traditional PBE Interaction	*Profile-Guided Interaction*
Users typically provide their desired outputs *sequentially*	System *proactively requests* outputs for syntactically discrepant inputs

Birthdays	Years
`8/20 ’92`	
`1986 June 07`	
`3/24 ’88`	
`1994 November 23`	
⋮	
`13-08-83`	
`4/21 ’79`	
`24-11-91`	

Birthdays	Years
`8/20 ’92`	
`1986 June 07`	
`3/24 ’88`	
`1994 November 23`	
⋮	
`13-08-83`	
`4/21 ’79`	
`24-11-91`	

Profile-Guided Interaction for PBE

Traditional PBE Interaction	*Profile-Guided Interaction*

Users typically provide their desired outputs *sequentially*

Birthdays	Years	
8/20 ’92	1992	1
1986 June 07	1986	2
3/24 ’88	1988	3
1994 November 23	1994	
⋮		
13-08-83	1983	4
4/21 ’79	1979	
24-11-91	1991	

System *proactively requests* outputs for syntactically discrepant inputs

Birthdays	Years
8/20 ’92	
1986 June 07	
3/24 ’88	
1994 November 23	
⋮	
13-08-83	
4/21 ’79	
24-11-91	

Profile-Guided Interaction for PBE

Traditional PBE Interaction	*Profile-Guided Interaction*
Users typically provide their desired outputs *sequentially*	System *proactively requests* outputs for syntactically discrepant inputs

Birthdays	Years	
8/20 ’92	1992	1
1986 June 07	1986	2
3/24 ’88	1988	3
1994 November 23	1994	
⋮		
13-08-83	1983	4
4/21 ’79	1979	
24-11-91	1991	

Birthdays	Years
8/20 ’92	
1986 June 07	
3/24 ’88	
1994 November 23	
⋮	
13-08-83	
4/21 ’79	
24-11-91	

Profile-Guided Interaction for PBE

Traditional PBE Interaction	*Profile-Guided Interaction*
Users typically provide their desired outputs *sequentially*	System *proactively requests* outputs for syntactically discrepant inputs

Birthdays	Years	
8/20 ’92	1992	1
1986 June 07	1986	2
3/24 ’88	1988	3
1994 November 23	1994	
⋮		
13-08-83	1983	4
4/21 ’79	1979	
24-11-91	1991	

Birthdays	Years
8/20 ’92	
1986 June 07	
3/24 ’88	
1994 November 23	
⋮	
13-08-83	
4/21 ’79	
24-11-91	

Profile-Guided Interaction for PBE

Traditional PBE Interaction	*Profile-Guided Interaction*
Users typically provide their desired outputs *sequentially*	System *proactively requests* outputs for syntactically discrepant inputs

Birthdays	Years	
8/20 ’92	1992	1
1986 June 07	1986	2
3/24 ’88	1988	3
1994 November 23	1994	
⋮		
13-08-83	1983	4
4/21 ’79	1979	
24-11-91	1991	

Birthdays	Years	
8/20 ’92	1992	3
1986 June 07	1986	2
3/24 ’88	1988	
1994 November 23	1994	
⋮		
13-08-83	1983	1
4/21 ’79	1979	
24-11-91	1991	

Profile-Guided Interaction for PBE

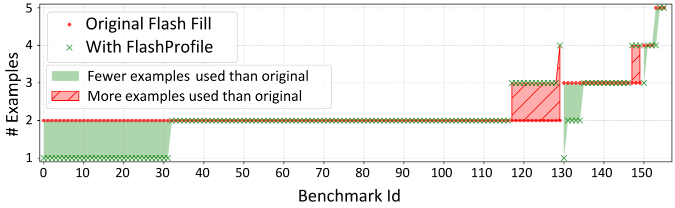
Traditional PBE Interaction	*Profile-Guided Interaction*

Users typically provide their desired outputs *sequentially*

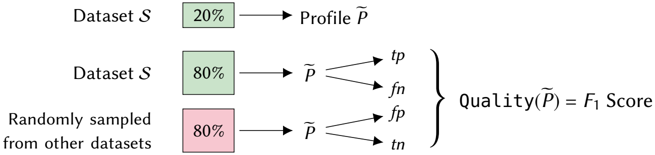
Birthdays	Years	
8/20 '92	1992	1
1986 June 07	1986	2
3/24 '88	1988	3
1994 November 23	1994	
⋮		
13-08-83	1983	4
4/21 '79	1979	
24-11-91	1991	

System *proactively requests* outputs for syntactically discrepant inputs

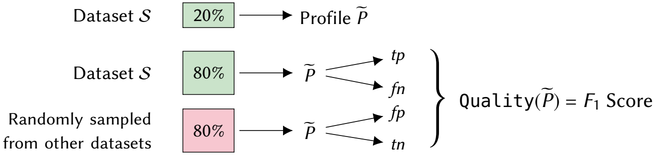
Birthdays	Years	
8/20 '92	1992	3
1986 June 07	1986	2
3/24 '88	1988	
1994 November 23	1994	
⋮		
13-08-83	1983	1
4/21 '79	1979	
24-11-91	1991	

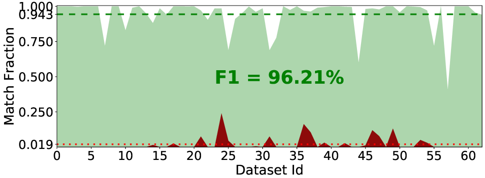


Quality of Generated Profiles

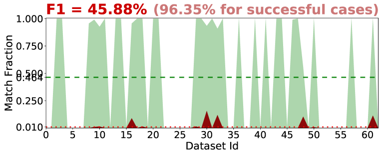


Quality of Generated Profiles

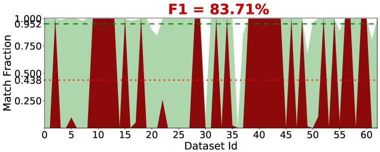




Quality of profiles generated by FlashProfile

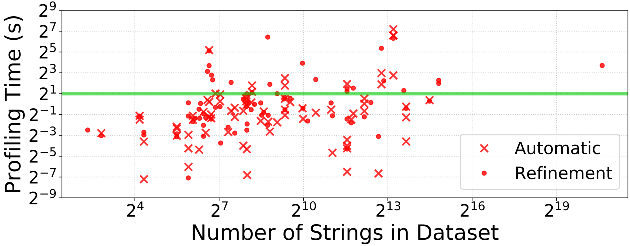


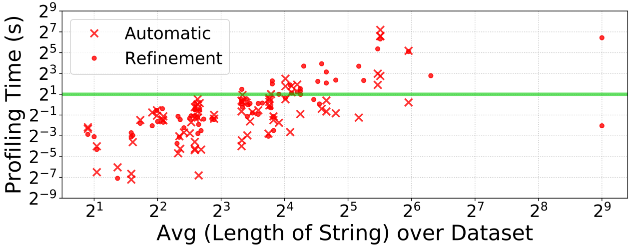
Ataccama One



Microsoft SSDT

End-to-End Profiling Performance





Related Work

- Microsoft SQL Server Data Tools (SSDT) [https://docs.microsoft.com/en-us/sql/ssdt]
  - Recognizes constants and fixed-width atoms.
  - Not extensible. No refinement. Profiles are sometimes not comprehensive.
- Ataccama One [https://one.ataccama.com/]
  - Comprehensive profiles. Recognizes fixed-width atoms.
  - A small fixed set of atoms. No refinement. Does not recognize constants.
- Trifacta Wrangler [https://cloud.trifacta.com]
  - Recognizes fixed-width atoms. Generates readable profiles.
  - Not extensible. No refinement. Does not recognize constants.
- Google OpenRefine [http://openrefine.org/]
  - No patterns, only clusters based on character-wise similarity.
- Potter's Wheel [Vijayshankar Raman and Joseph M. Hellerstein. *VLDB 2001*]
  - Extensible set of atoms.
  - Only learns the most-frequent pattern and shows outliers, not a profile.
- LearnPADS++ [Kathleen Fisher *et al. SIGMOD 2008* ; Kenny Q. Zhu *et al. PADL 2012*]
  - Not extensible. No refinement. Generates C-style structures.

Conclusion & Future Work

A novel composition of hierarchical clustering and program synthesis techniques for efficient pattern-based data profiling

Conclusion & Future Work

A novel composition of hierarchical clustering and program synthesis techniques for efficient pattern-based data profiling

Future Work:

- Automatically selecting costs for atoms
  - Machine-learnt costs to maximize the *quality* of profiles

Conclusion & Future Work

A novel composition of hierarchical clustering and program synthesis techniques for efficient pattern-based data profiling

Future Work:

- Automatically selecting costs for atoms
  - Machine-learnt costs to maximize the *quality* of profiles
- Alternate approximation strategies with better guarantees
  - Compute the overall "goodness" of an approximation and refine if needed

Conclusion & Future Work

A novel composition of hierarchical clustering and program synthesis techniques for efficient pattern-based data profiling

Future Work:

- Automatically selecting costs for atoms
  - Machine-learnt costs to maximize the *quality* of profiles
- Alternate approximation strategies with better guarantees
  - Compute the overall "goodness" of an approximation and refine if needed
- Identify and classify semantic entities as well
  - For example, combine with *named-entity recognition* (NER) techniques

Publicly-Available Artifacts

- The Matching.Text NuGet package:
  https://www.nuget.org/packages/Microsoft.ProgramSynthesis.Matching.Text/
- Documentation for Matching.Text library:
  https://microsoft.github.io/prose/documentation/matching-text/intro/
- OOPSLA artifacts (a C# app showing Matching.Text API usage):
  https://github.com/SaswatPadhi/FlashProfileDemo
- Contact: padhi@cs.ucla.edu