Ayudante: Identifying Undesired Variable Interactions

Irfan Ul Haq
IMDEA Software Institute
Madrid, Spain

Juan Caballero
IMDEA Software Institute
Madrid, Spain

Michael D. Ernst
University of Washington
Seattle, USA

Sometimes, a programmer uses variables erroneously.

```
dollars = euros;
```

```
array[ fd ] = value;
```

```
tax = itemPrice + miles;
```

Compiler **does not detect** these errors.

dollars = euros;

array[**fd**] = value;

tax = itemPrice + **miles**;

Compiler **does not detect** these errors.

```
float dollars, euros;              dollars = euros;

int index, fd;                     array[ fd ] = value;

float tax, itemPrice, miles;       tax = itemPrice + miles;
```

Compiler **does not detect** these errors.

```
float dollars, euros;        dollars = euros;
int index, fd;               array[ fd ] = value;
float tax, itemPrice, miles; tax = itemPrice + miles;

index = dollars
```

Compiler **does not detect** these errors.

```
float dollars, euros;
```

```
dollars = euros;
```

```
int index, fd;
```

```
array[ fd ] = value;
```

```
float tax, itemPrice
```

```
mPrice + miles;
```

Warning because of type specification.

```
index = dollars
```

Compiler **does not detect** these errors.

float dollars, euros;

dollars = euros;

int index, fd;

array[**fd**] = value;

float tax, itemPrice

mPrice + **miles**;

Warning because of type specification.

index = dollars

dollars = euros

Compiler **does not detect** these errors.

float dollars, euros;

dollars = euros;

int index, fd;

array[**fd**] = value;

float tax, itemPrice

mPrice +

Warning because of type specification.

Should be a warning or error.

index = dollars

dollars = euros

Natural Language in Source Code

Natural Language in Source Code

Natural language in 70% of the source code.

Natural Language in Source Code

Natural language in 70% of the source code.

Same for all programming languages.

Research Question

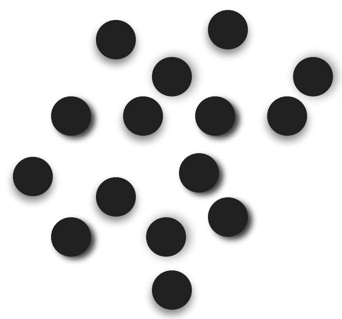
Can we identify undesired variable interactions automatically?

Related Variables

- Our goal is to find related variables.

Related Variables

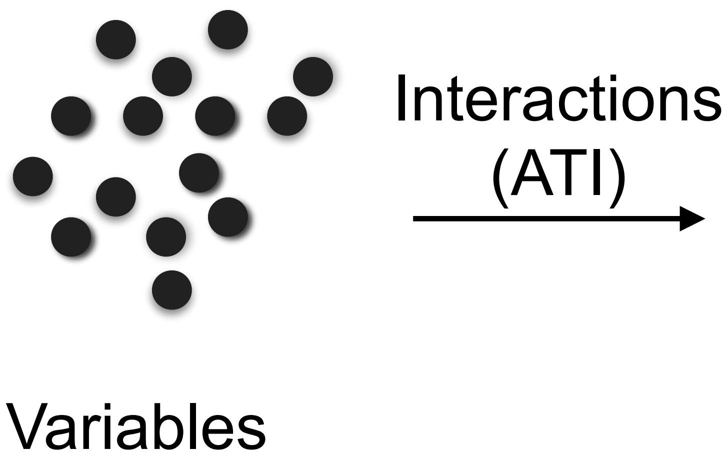
- Our goal is to find related variables.



Variables

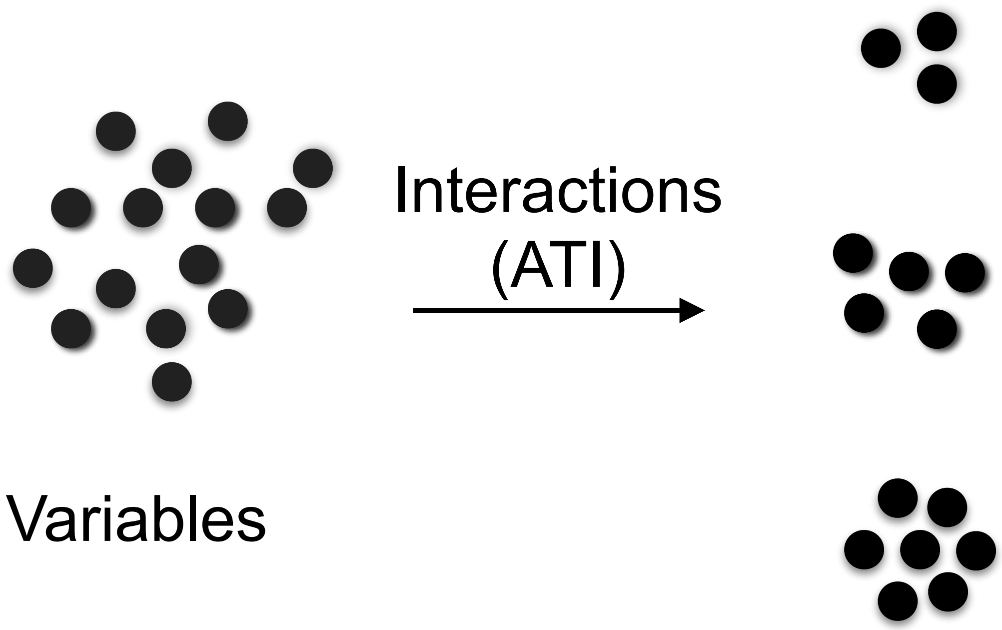
Related Variables

- Our goal is to find related variables.



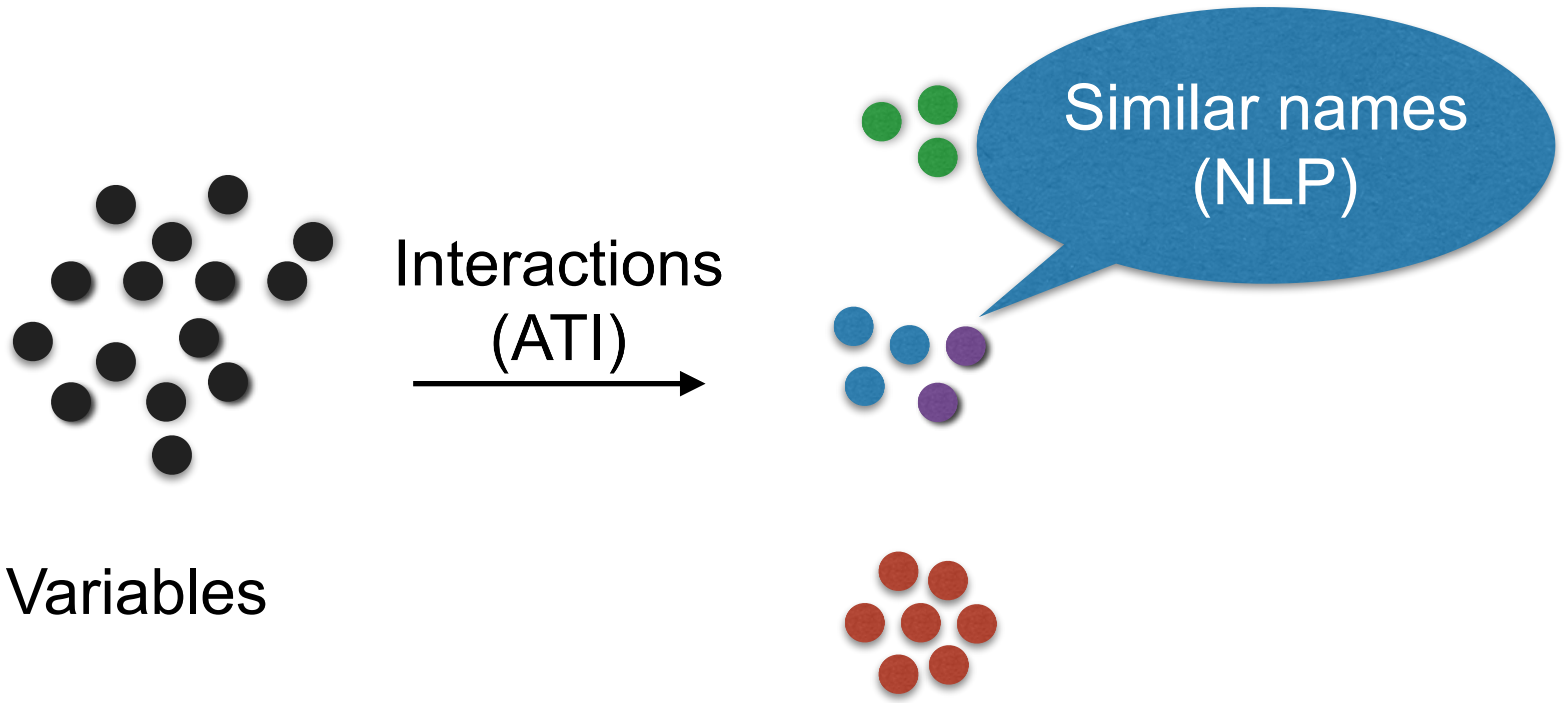
Related Variables

- Our goal is to find related variables.



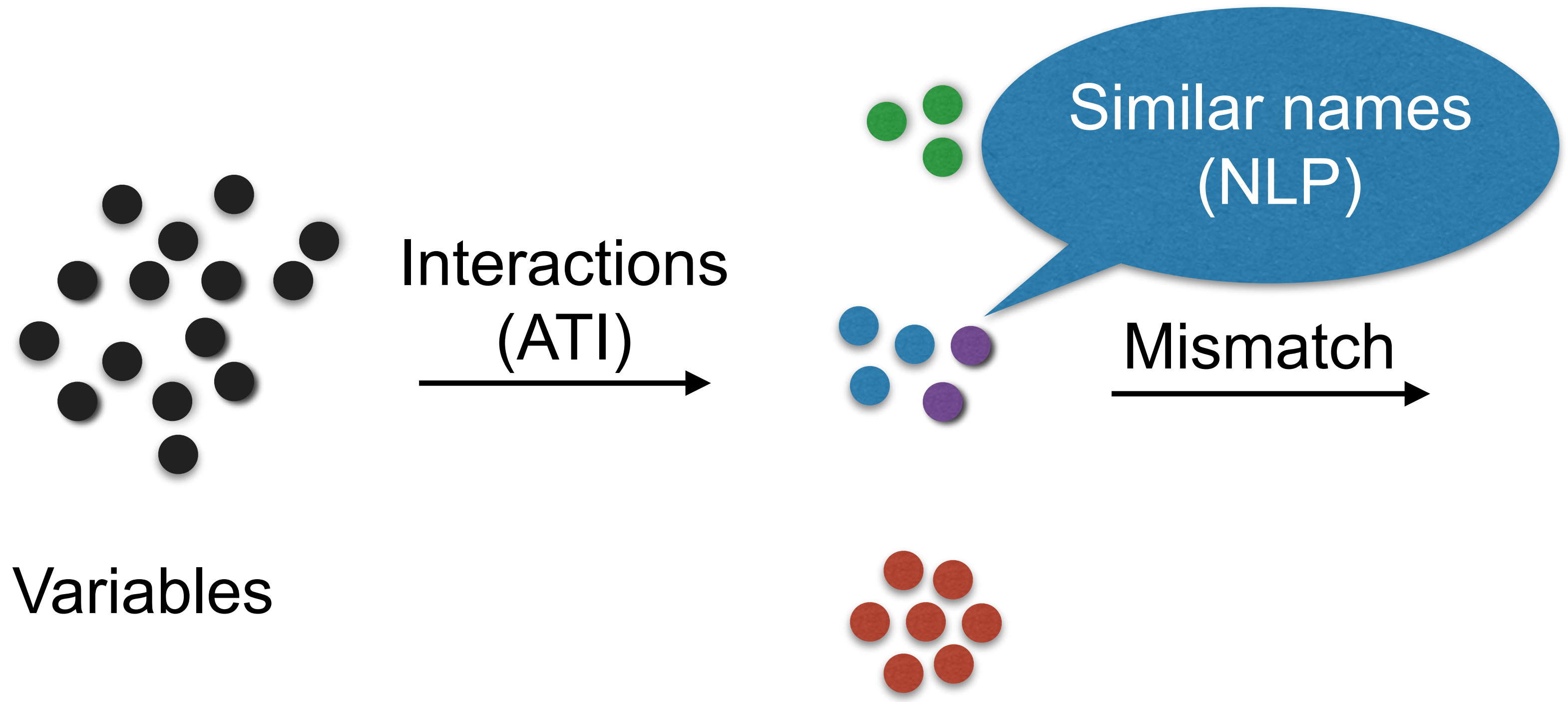
Related Variables

- Our goal is to find related variables.



Related Variables

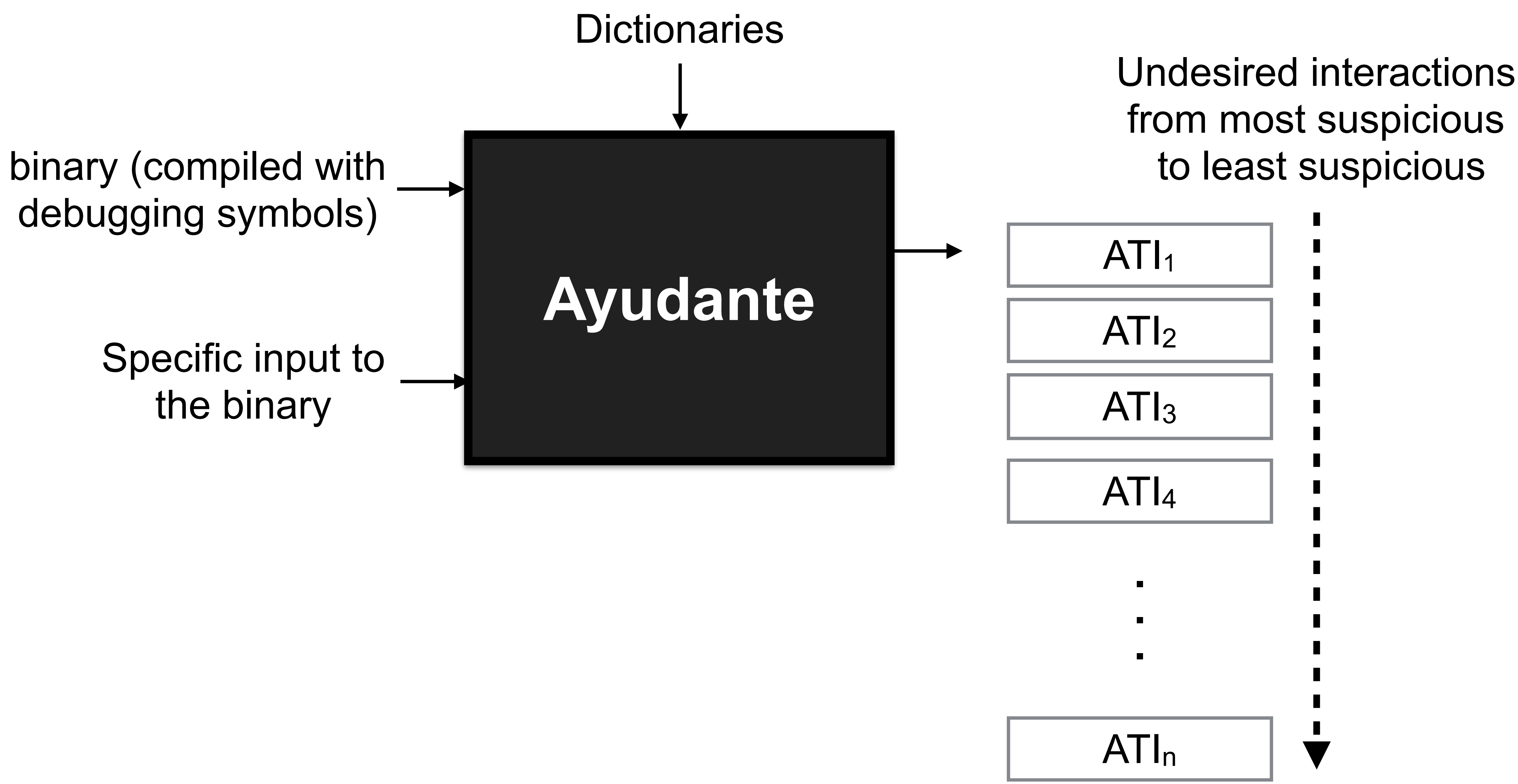
- Our goal is to find related variables.



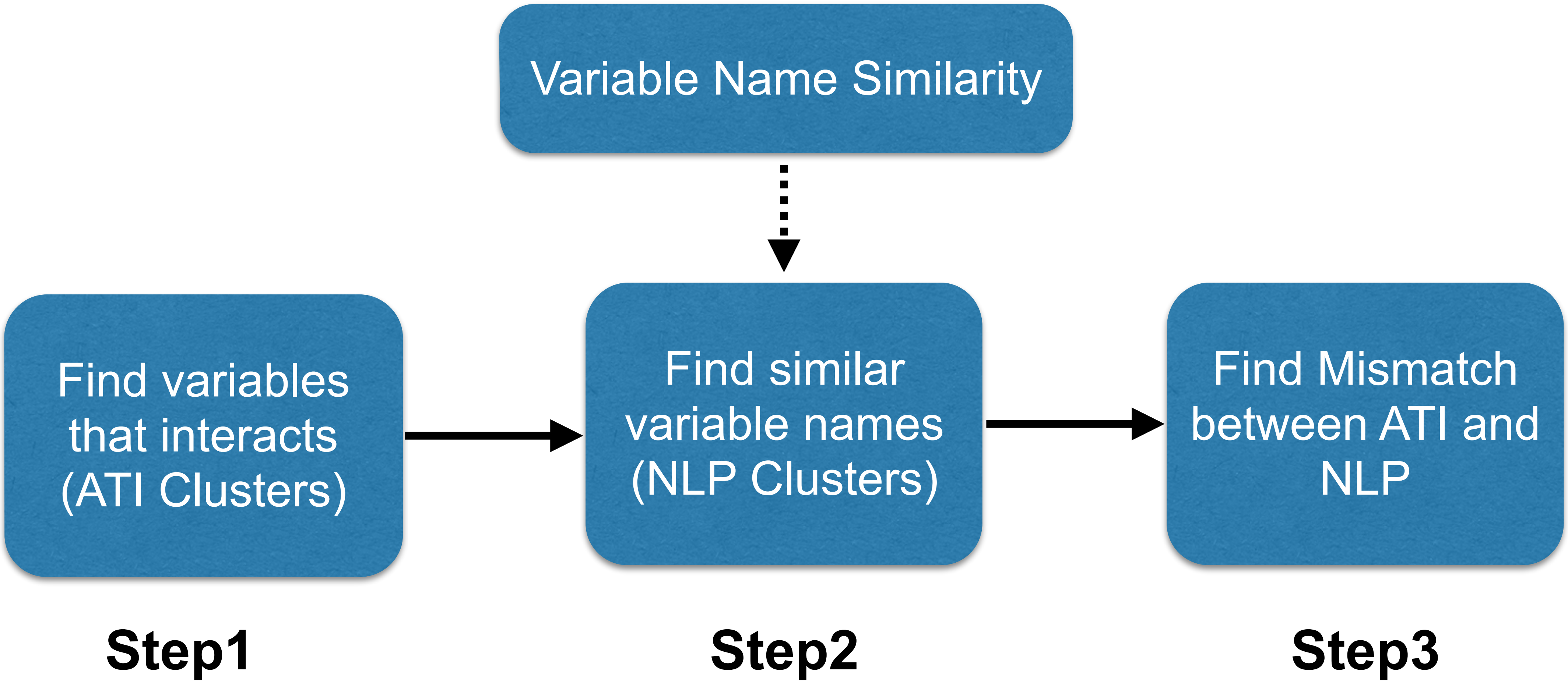
Contributions

- Automatically report suspicious variable interactions.
- A novel technique to use semantics embedded in variable names.
- A tool called Ayudante.
- Evaluation
  - Found an undesired interaction in grep.

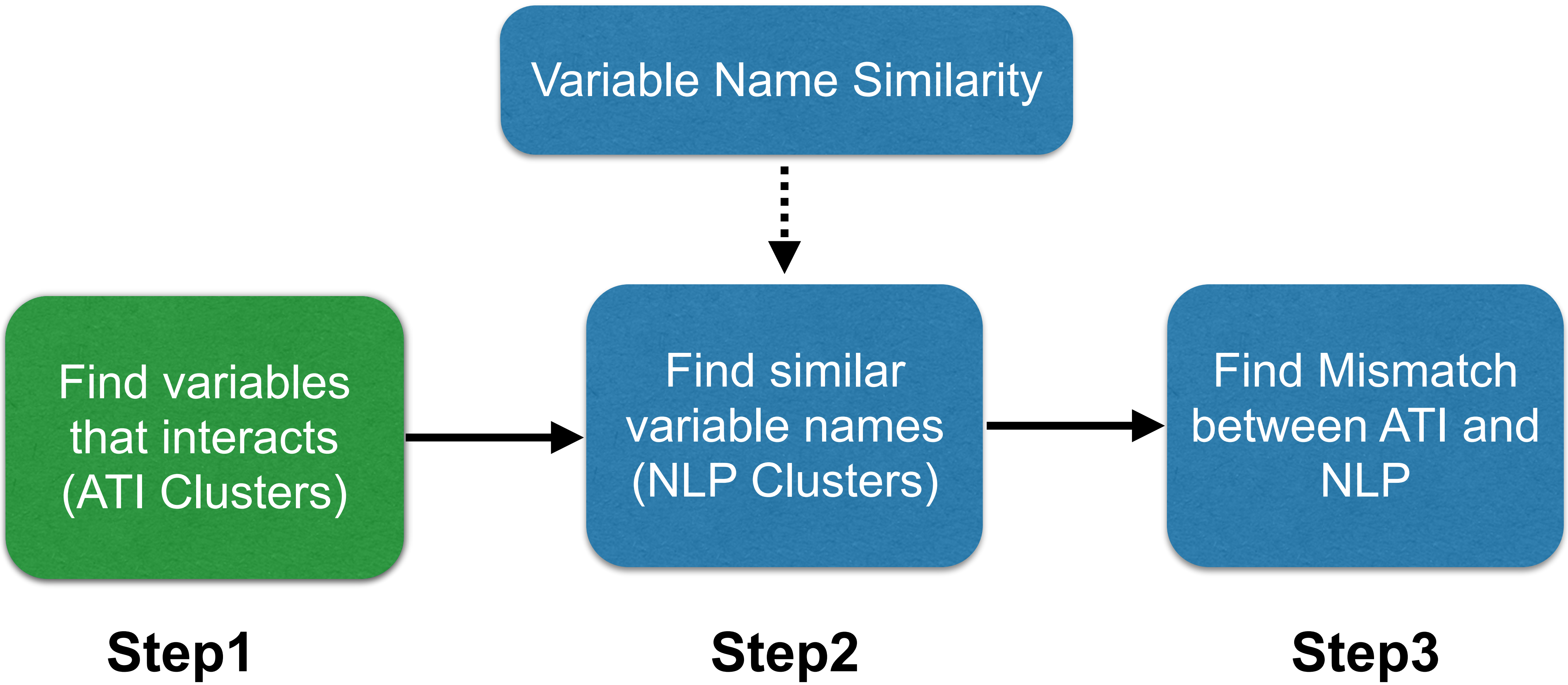
Ayudante as a Black Box



Approach Overview



Approach Overview



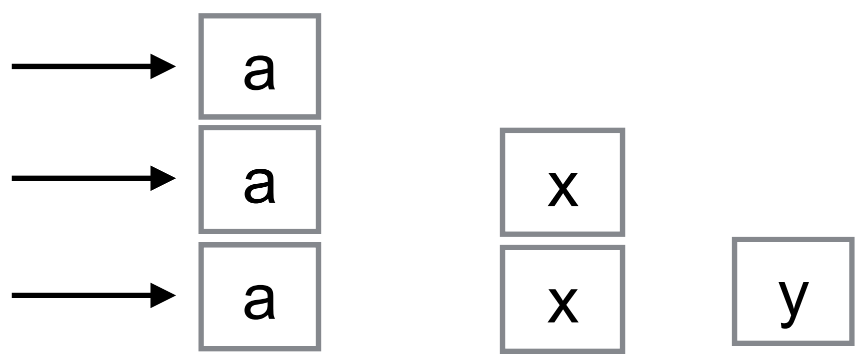
ATI Clusters

- Use interaction between variables to group them.
- Interactions, e.g, ’comparison’, ‘addition’.

ATI Clusters

- Use interaction between variables to group them.
- Interactions, e.g, ’comparison’, ‘addition’.

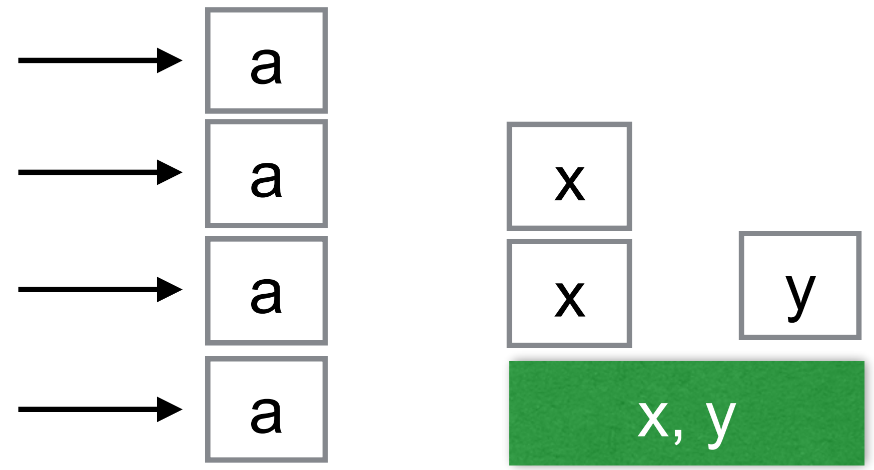
```
int a = 0;
int x = 10;
int y = 100;
```



ATI Clusters

- Use interaction between variables to group them.
- Interactions, e.g, 'comparison', 'addition'.

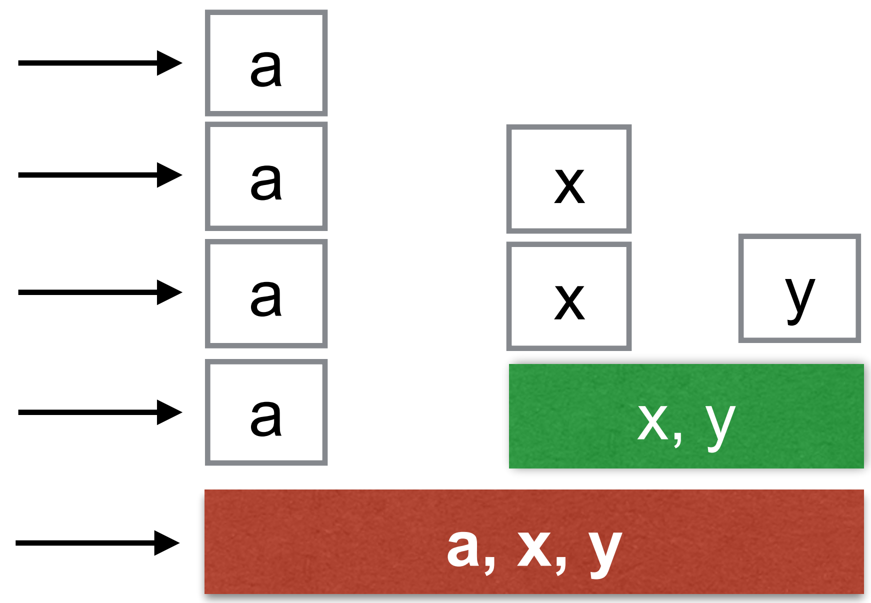
```
int a = 0;
int x = 10;
int y = 100;
if (x < y)
```



ATI Clusters

- Use interaction between variables to group them.
- Interactions, e.g, ’comparison’, ‘addition’.

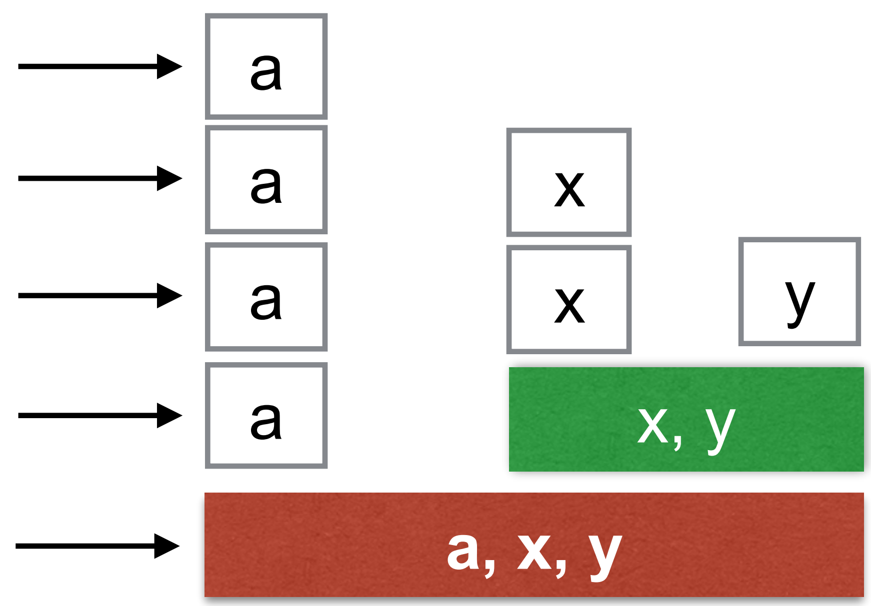
```
int a = 0;
int x = 10;
int y = 100;
if (x < y)
    a = x + y;
```



ATI Clusters

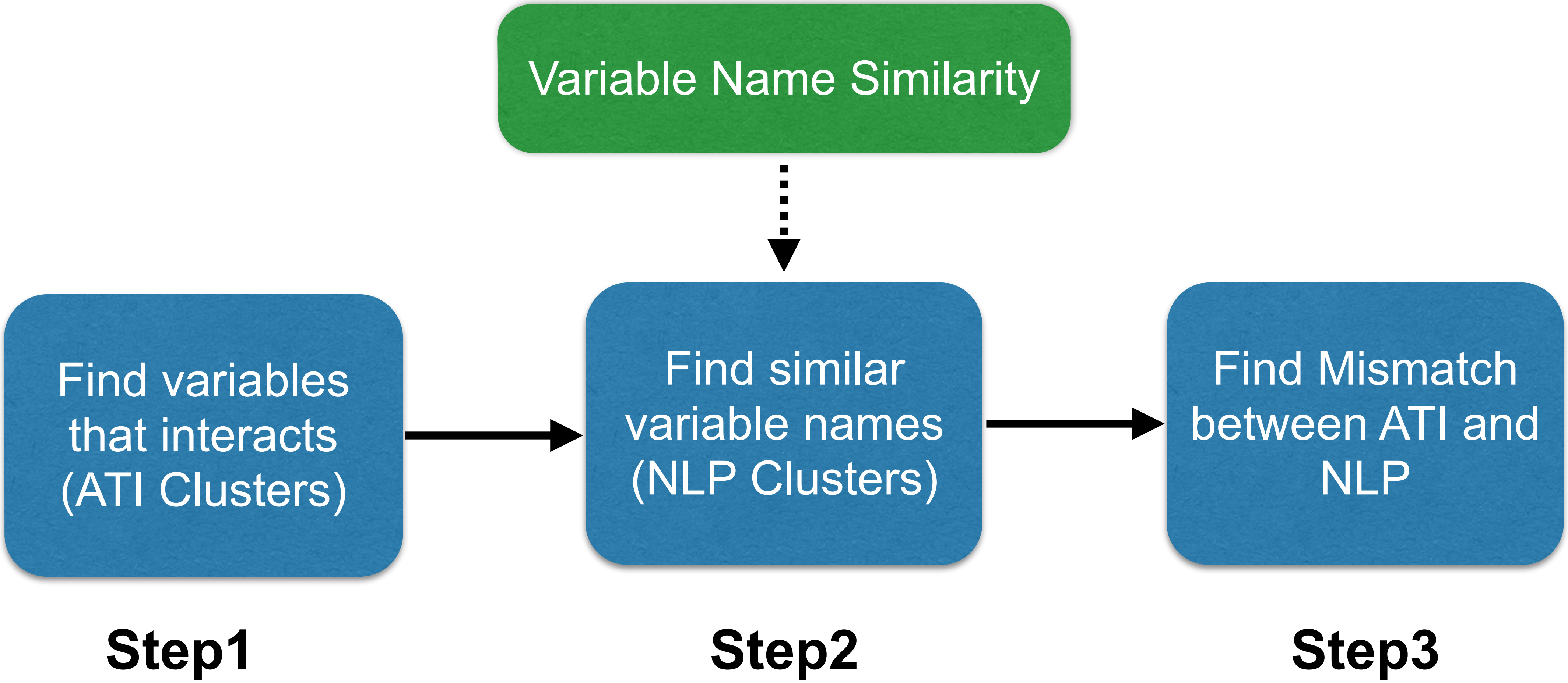
- Use interaction between variables to group them.
- Interactions, e.g, ’comparison’, ‘addition’.

```
int a = 0;
int x = 10;
int y = 100;
if (x < y)
    a = x + y;
```



- Static or Dynamic

Approach Overview

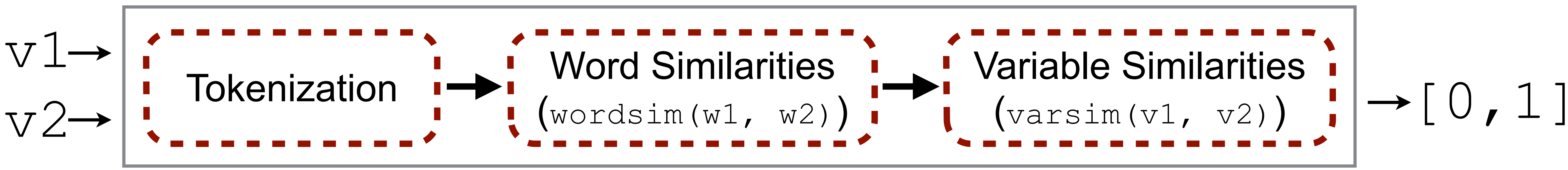


Variable Name Similarity

- varsim(v1, v2) → [0,1]

Variable Name Similarity

- varsim(v1, v2) → [0,1]

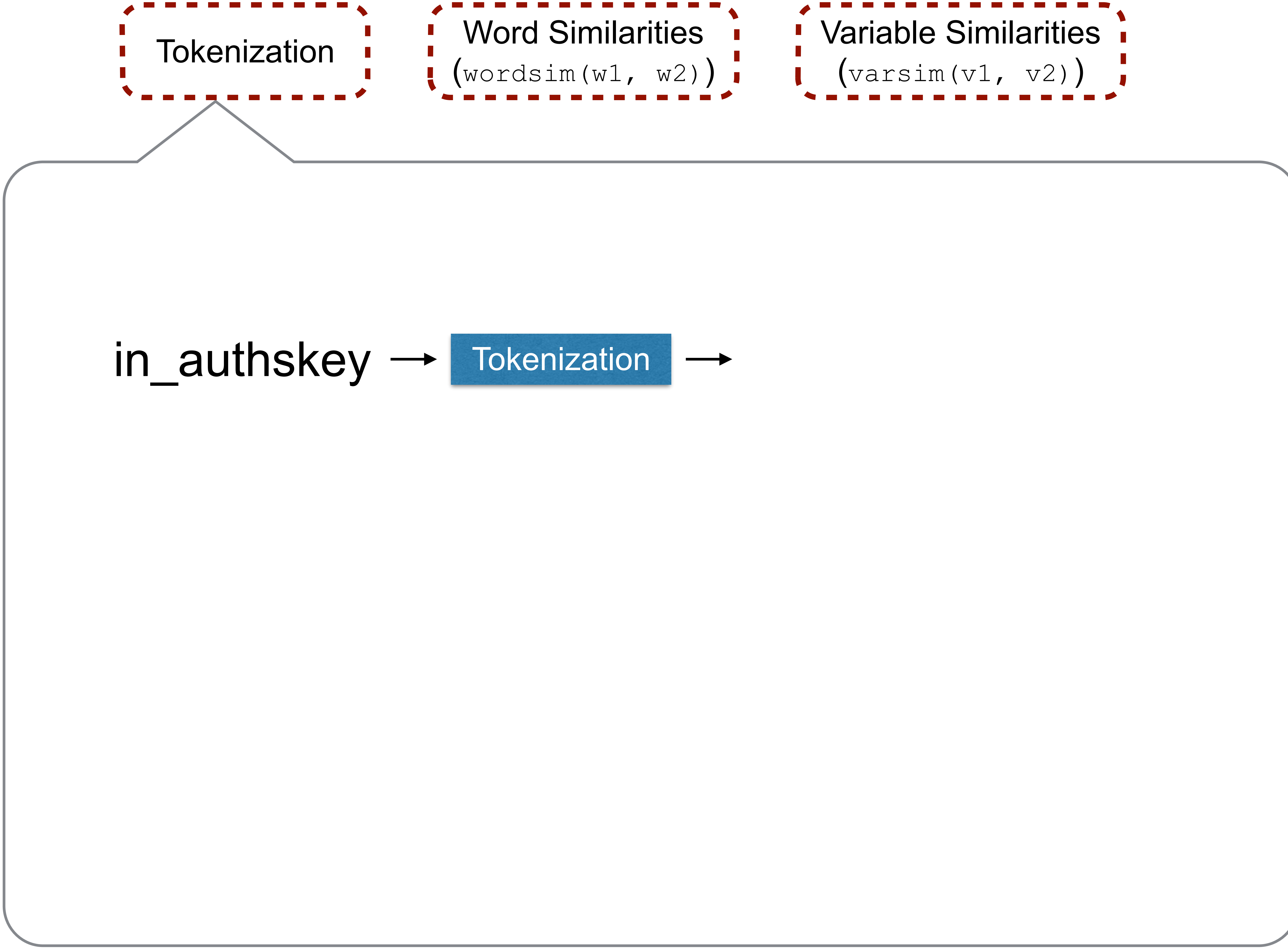


Running Example:
'in_authskey' and **'maxDepth'**

Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)


Tokenization
Word Similarities
(wordsim(w1, w2))
Variable Similarities
(varsim(v1, v2))
in_authskey
Tokenization

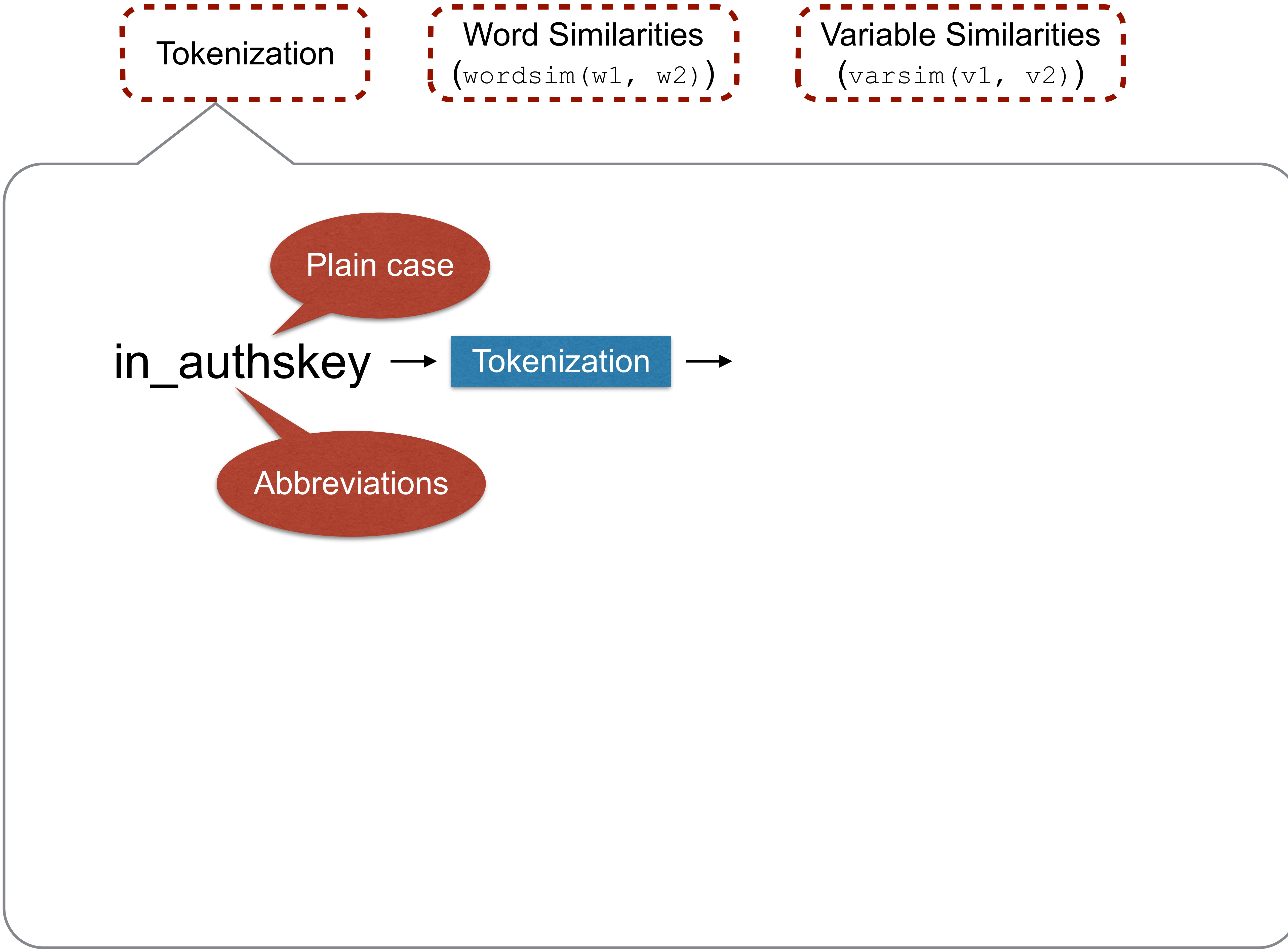
Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

Plain case

in_authskey → Tokenization →


Tokenization
Word Similarities
(wordsim(w1, w2))
Variable Similarities
(varsim(v1, v2))
Plain case
in_authskey
Tokenization
Abbreviations

Tokenization

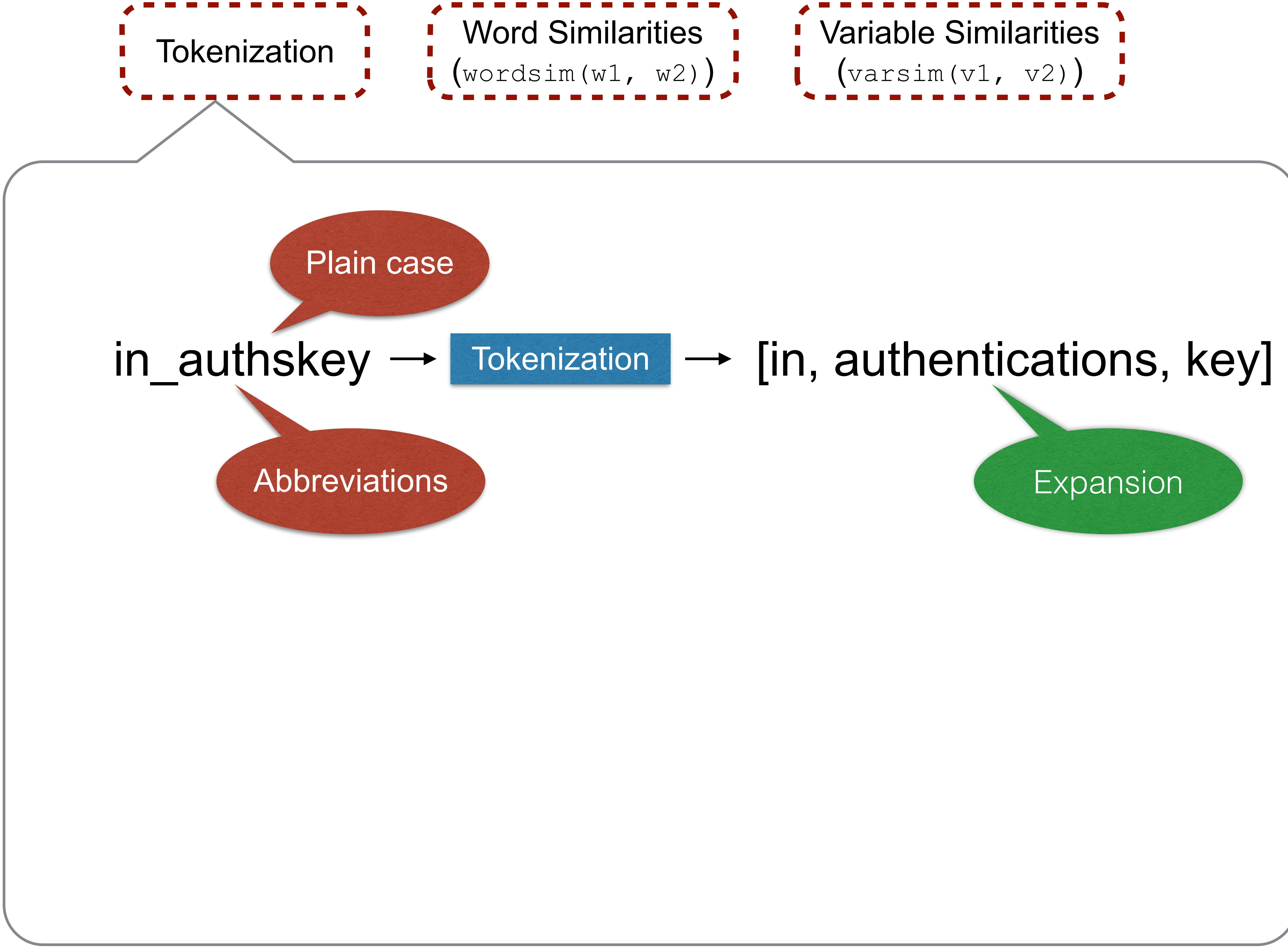
Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

Plain case

in_authskey → Tokenization → [in, authentications, key]

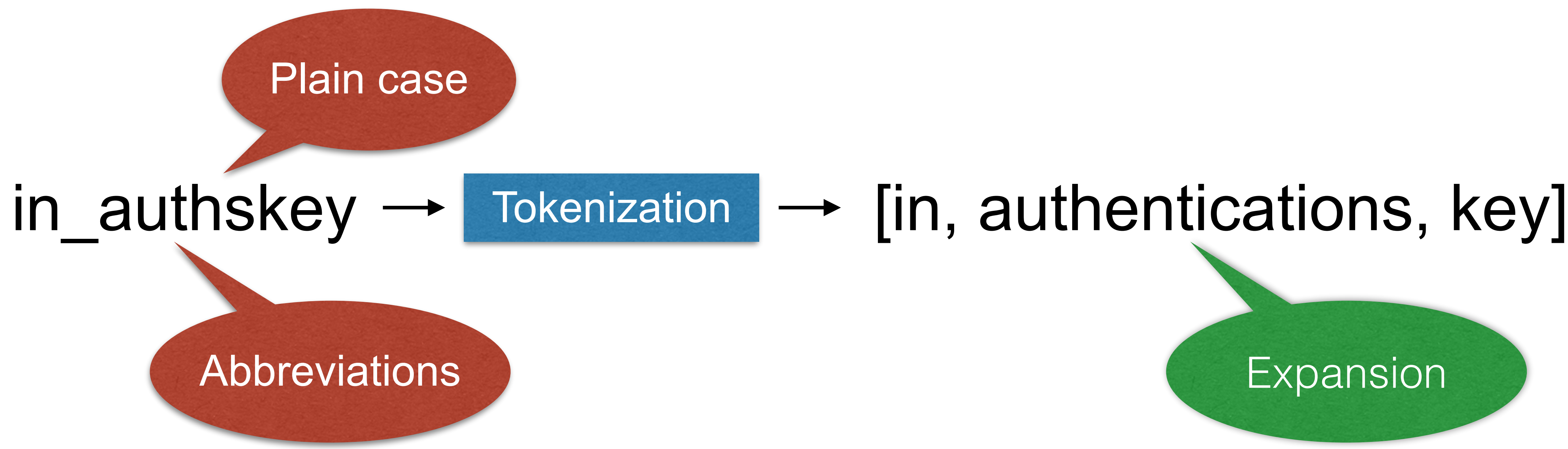
Abbreviations


Tokenization
Word Similarities
(wordsim(w1, w2))
Variable Similarities
(varsim(v1, v2))
Plain case
in_authskey
Tokenization
[in, authentications, key]
Abbreviations
Expansion

Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)



- Solution:
  - A tokenization algorithm.
  - Common programming abbreviations as an input.

Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

Plain case

in_authskey → Tokenization → [in, authentications, key]

Abbreviations

Expansion

- Solution:
  - A tokenization algorithm.
  - Common programming abbreviations as an input.

`[in, authentications, key]` **`[maximum, Depth]`**

Tokenization

Word Similarities
(`wordsim(w1, w2)`)

Variable Similarities
(`varsim(v1, v2)`)

`[in, authentications, key]` **`[maximum, Depth]`**

Tokenization

Word Similarities
(`wordsim(w1, w2)`)

Variable Similarities
(`varsim(v1, v2)`)

`[in, authentications, key]`

`[maximum, Depth]`

wordsim('authentications', 'Depth')

Tokenization

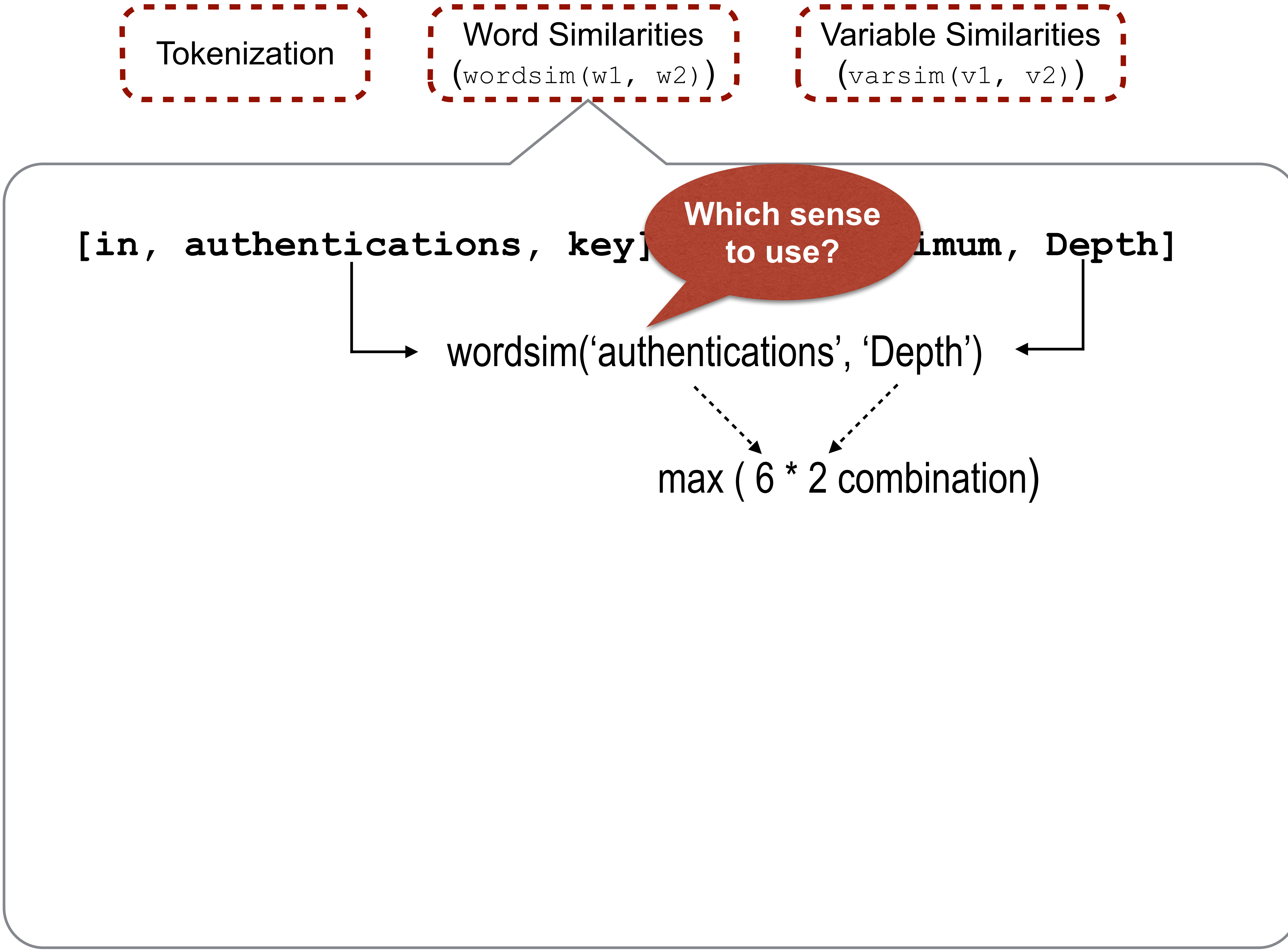
Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

`[in, authentications, key]` … **`imum, Depth]`**

Which sense to use?

wordsim('authentications', 'Depth')


Tokenization
Word Similarities
(wordsim(w1, w2))
Variable Similarities
(varsim(v1, v2))
[in, authentications, key]
imum, Depth]
Which sense to use?
wordsim('authentications', 'Depth')
max (6 * 2 combination)

Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

`[in, authentications, key]`

`[maximum, Depth]`

wordsim('authentications', 'Depth')

max (6 * 2 combination)

wordsim('authentications', 'Depth') = 0.36

Tokenization

Word Similarities
(`wordsim(w1, w2)`)

Variable Similarities
(`varsim(v1, v2)`)

`[in, authentications, key]`

`[maximum, Depth]`

Tokenization

Word Similarities
(`wordsim(w1, w2)`)

Variable Similarities
(`varsim(v1, v2)`)

`[in, authentications, key]` **`[maximum, Depth]`**

wordsim('authentications', 'Depth') = 0.36

Tokenization

Word Similarities
(`wordsim(w1, w2)`)

Variable Similarities
(`varsim(v1, v2)`)

`[in, authentications, key]` **`[maximum, Depth]`**

wordsim('authentications', 'Depth') = 0.36

wordsim('authentications', 'maximum') = 0.31

Tokenization

Word Similarities
(`wordsim(w1, w2)`)

Variable Similarities
(`varsim(v1, v2)`)

`[in, authentications, key]` **`[maximum, Depth]`**

wordsim('authentications', 'Depth') = 0.36
wordsim('authentications', 'maximum') = 0.31

`maxwordsim('authentications') = 0.36`

Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

`[in, authentications, key]` **`[maximum, Depth]`**

wordsim('authentications', 'Depth') = 0.36

wordsim('authentications', 'maximum') = 0.31

`maxwordsim('authentications') = 0.36`

varsim('in_authskey', 'maxDepth') = **Avg** (
maxwordsim('authentications') = 0.11
maxwordsim('in') = 0.36
maxwordsim('key') = 0.62
maxwordsim('maximum') = 0.53
maxwordsim('Depth') = 0.62
)

Tokenization

Word Similarities (`wordsim(w1, w2)`)

Variable Similarities (`varsim(v1, v2)`)

`[in, authentications, key]` **`[maximum, Depth]`**

wordsim('authentications', 'Depth') = 0.36

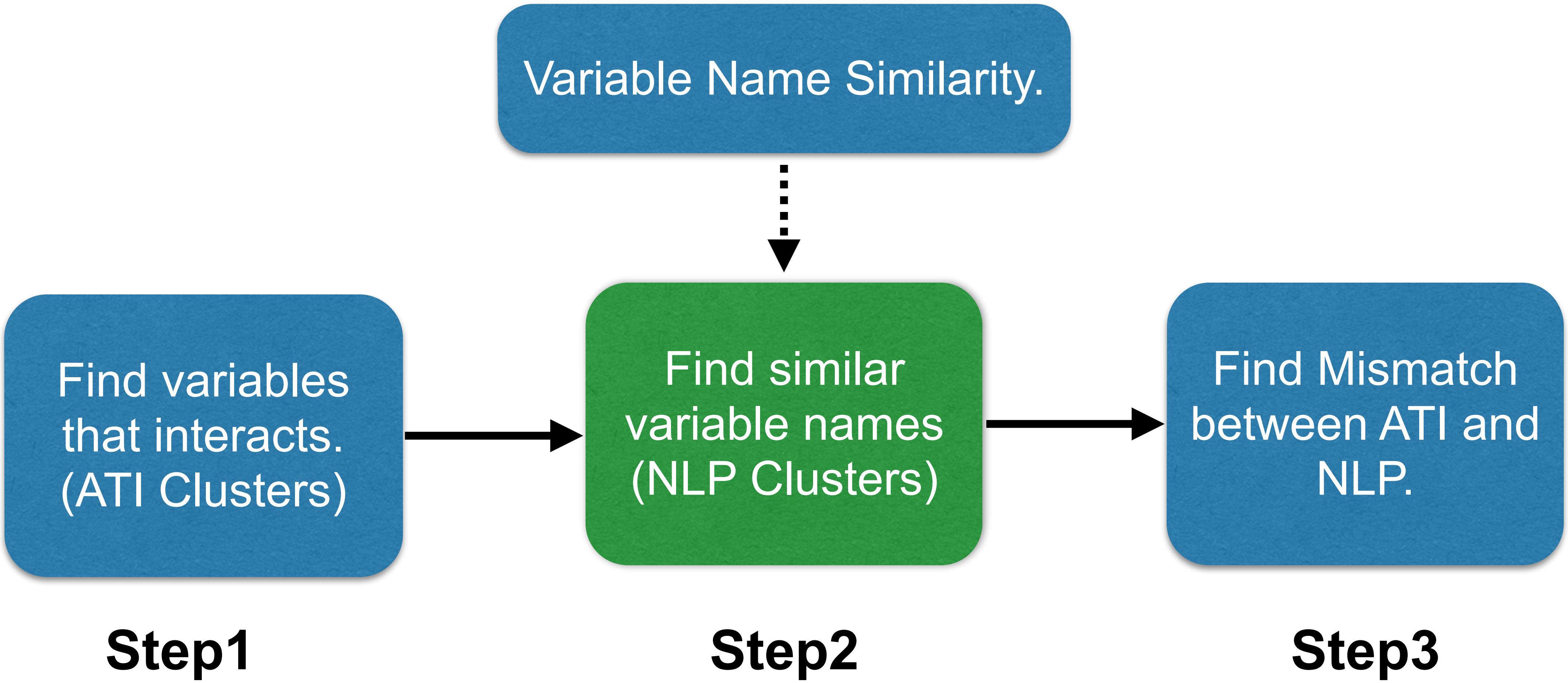
wordsim('authentications', 'maximum') = 0.31

`maxwordsim('authentications') = 0.36`

varsim('in_authskey', 'maxDepth') = **Avg** (
maxwordsim('authentications') = 0.11
maxwordsim('in') = 0.36
maxwordsim('key') = 0.62
maxwordsim('maximum') = 0.53
maxwordsim('Depth') = 0.62
)

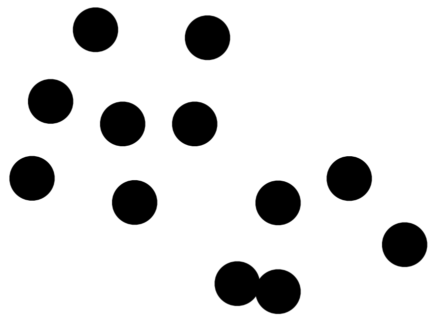
varsim('in_authskey', 'maxDepth') = 0.45

Approach Overview



NLP Clusters

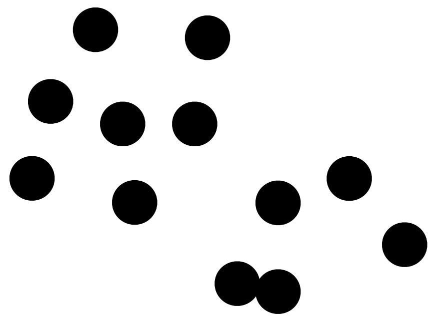
- For each ATI cluster find similar variables.



ATI Cluster with
m variables

NLP Clusters

- For each ATI cluster find similar variables.

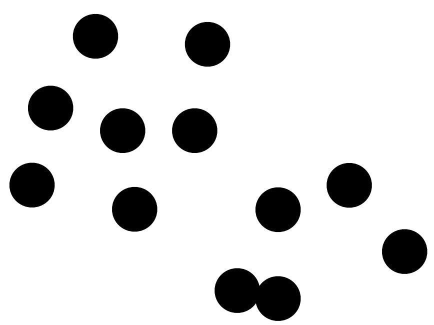


ATI Cluster with
m variables

- An m*m symmetric matrix

NLP Clusters

- For each ATI cluster find similar variables.

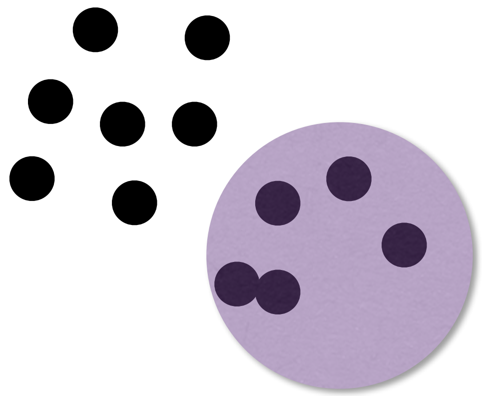


ATI Cluster with
m variables

- An m*m symmetric matrix
- K-means clustering algorithm

NLP Clusters

- For each ATI cluster find similar variables.

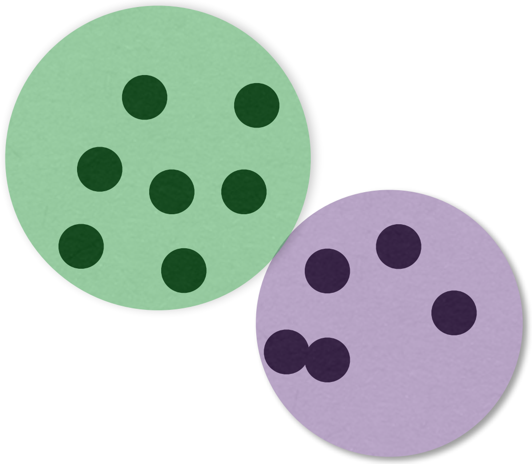


ATI Cluster with
m variables

- An m*m symmetric matrix
- K-means clustering algorithm

NLP Clusters

- For each ATI cluster find similar variables.

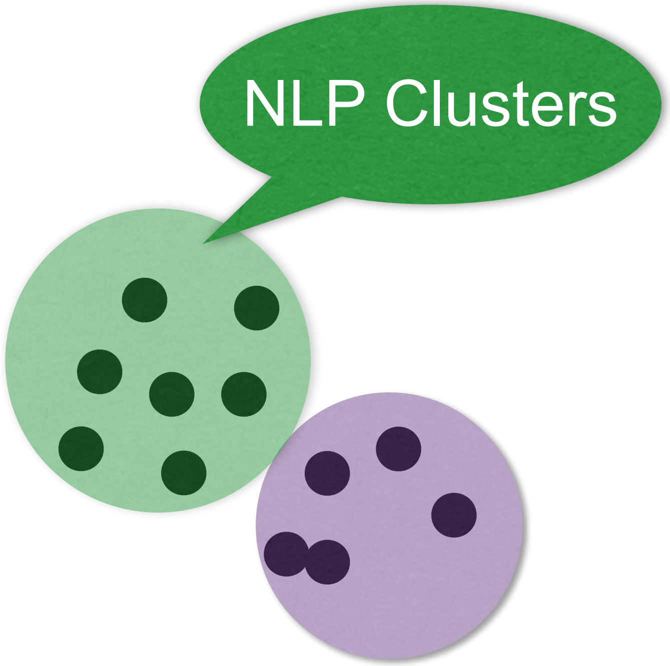


ATI Cluster with
m variables

- An m*m symmetric matrix
- K-means clustering algorithm

NLP Clusters

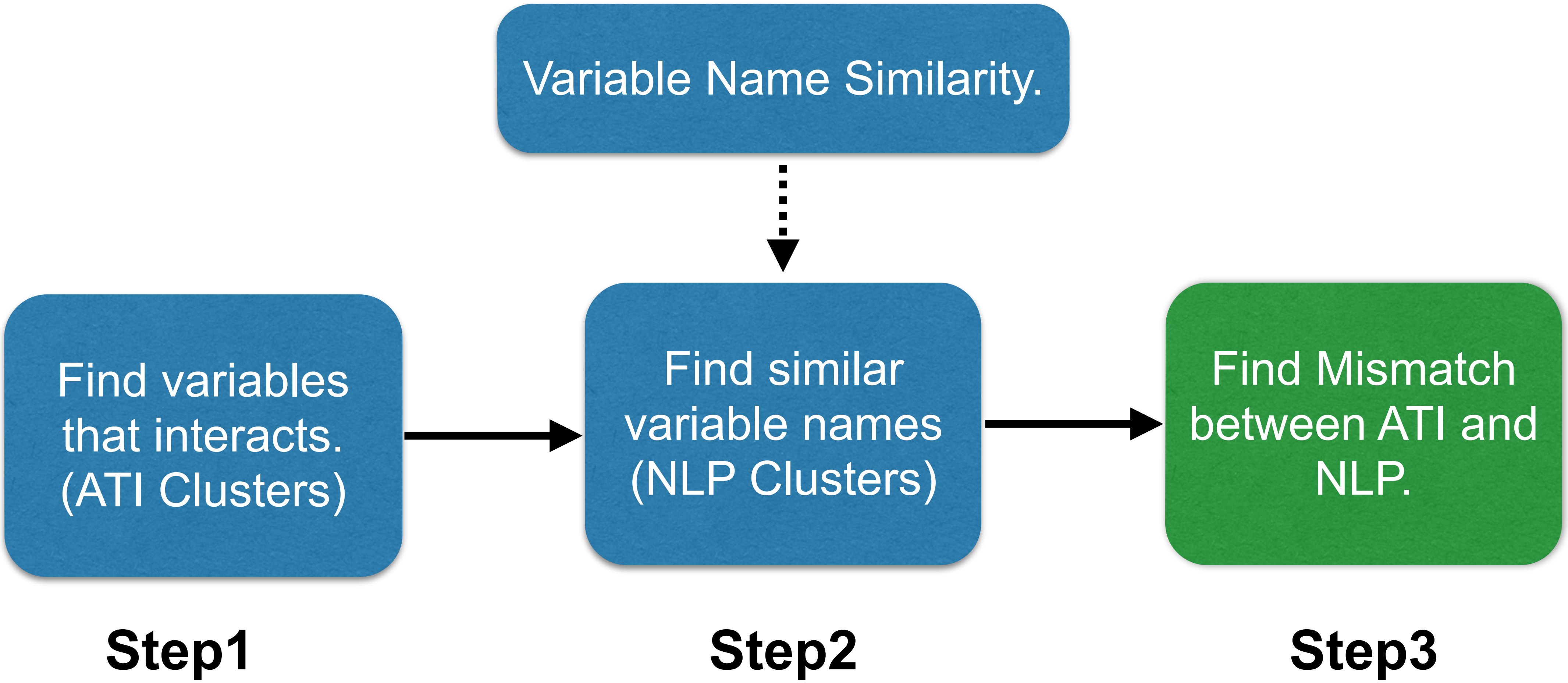
- For each ATI cluster find similar variables.



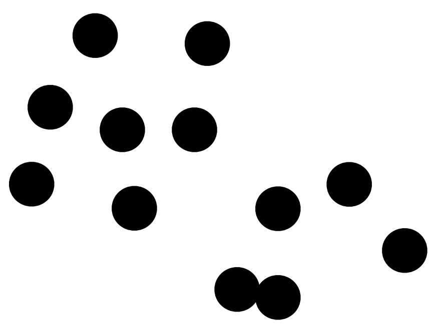
ATI Cluster with
m variables

- An m*m symmetric matrix
- K-means clustering algorithm

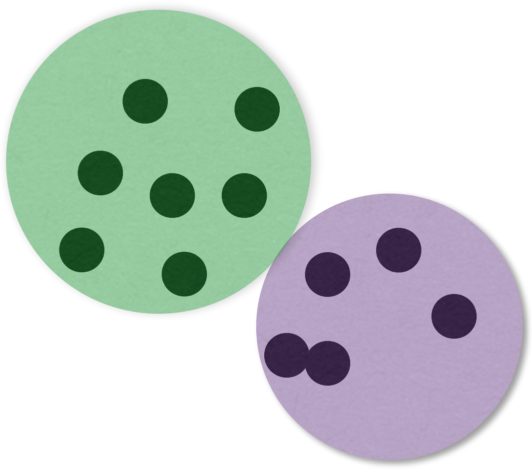
Approach Overview



ATI and NLP Mismatch

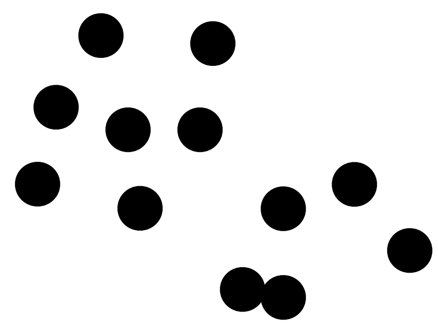


ATI Cluster

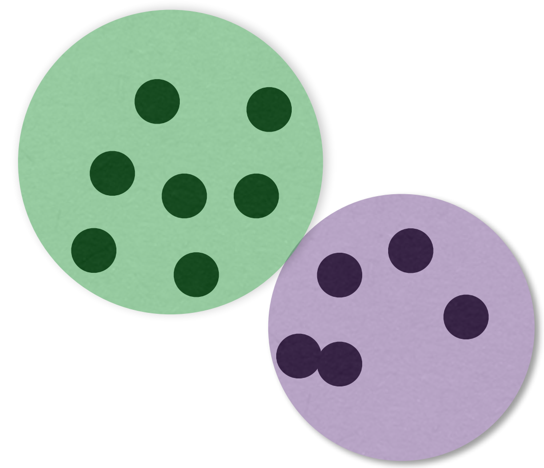


NLP Clusters

ATI and NLP Mismatch



ATI Cluster



NLP Clusters

Suspicious ATI cluster $\xleftarrow{\text{indicates}}$ Cohesive NLP clusters

Rank Mismatch ATI Clusters

Rank Mismatch ATI Clusters

Suspicious ATI cluster $\xleftarrow{\text{indicates}}$ Cohesive NLP clusters

Rank Mismatch ATI Clusters

Suspicious ATI cluster $\xleftarrow{\text{indicates}}$ Cohesive NLP clusters

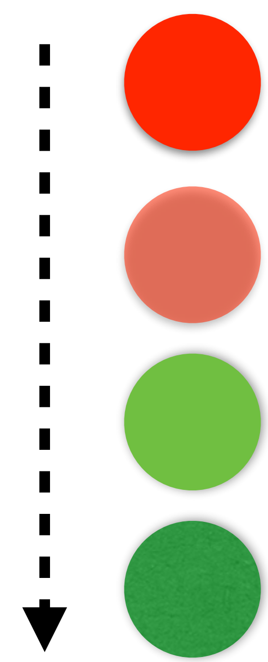
Less Suspicious ATI cluster $\xleftarrow{\text{indicates}}$ Less Cohesive NLP clusters

Rank Mismatch ATI Clusters

Suspicious ATI cluster $\xleftarrow{\text{indicates}}$ Cohesive NLP clusters

Less Suspicious ATI cluster $\xleftarrow{\text{indicates}}$ Less Cohesive NLP clusters

ATI cluster with cohesive
NLP clusters at the top



Evaluation

Tokenization Algorithm

End-to-end Testing

Tokenization Algorithm

End-to-end Testing

- Manually established ground truth.
- **2500** variable names from 4 programs.

	Exim	**Grep**	**Valgrind**	**Putty**
With Abbreviations	95%	87%	81%	76%
Without Abbreviations	91%	75%	77%	66%

Tokenization Algorithm

End-to-end Testing

- Two programs; Exim and Grep.
- Analysed 5 top-ranked clusters.

- One mistake in **grep** that assigns integer value to an unsigned char.
- Variable in top-ranked cluster: `delta`, `depth`, `tree`, and `eolbyte`.

```
delta[tree->label] = depth;
```

Tokenization Algorithm

End-to-end Testing

- Two programs; Exim and Grep.
- Analysed 5 top-ranked clusters.

- One mistake in **grep** that assigns integer value to an unsigned char.
- Variable in top-ranked cluster: `delta`, **Unsigned char** and `eolbyte`.

```
delta[tree->label] = depth;
```

Tokenization Algorithm

End-to-end Testing

- Two programs; Exim and Grep.
- Analysed 5 top-ranked clusters.

- One mistake in **grep** that assigns integer value to an unsigned char.
- Variable in top-ranked cluster: `delta`, and `eolbyte`.

Unsigned char

int

```
delta[tree->label] = depth;
```

Conclusion

- Automatically find suspicious variable interactions.
- A novel technique to use semantics embedded in variable names.
- A tool called Ayudante.
- Evaluation
    - Found an undesired interaction in grep.