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.

\[
\text{dollars} = \text{euros};
\]

\[
\text{array[ fd ]} = \text{value};
\]

\[
\text{tax} = \text{itemPrice} + \text{miles};
\]
Compiler does not detect these errors.

\[
\text{dollars} = \text{euros};
\]

\[
\text{array}[\text{fd}] = \text{value};
\]

\[
\text{tax} = \text{itemPrice} + \text{miles};
\]
Compiler does not detect these errors.

float dollars, euros;

int index, fd;

float tax, itemPrice, miles;

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;`

- `index = dollars;`
Compiler does not detect these errors.

```
float dollars, euros;
int index, fd;
float tax, itemPrice, miles;
dollars = euros;
array[fd] = value;
tax = itemPrice + miles;
index = dollars
```
Compiler does not detect these errors.

Warning because of type specification.

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

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

float tax, itemPrice;
tax = itemPrice + miles;

index = dollars;

dollars = euros;
```
Compiler does not detect these errors.

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

**Warning** because of type specification.

`index = dollars`

`dollars = euros`

Should be a warning or error.
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.
Can we identify undesired variable interactions automatically?
• 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.
Related Variables

- Our goal is to find 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

Ayudante

Dictionaries

Specific input to the binary

binary (compiled with debugging symbols)

Undesired interactions from most suspicious to least suspicious

\[ \text{ATI}_1, \text{ATI}_2, \text{ATI}_3, \text{ATI}_4, \ldots, \text{ATI}_n \]
Approach Overview

Find variables that interacts (ATI Clusters)

Find similar variable names (NLP Clusters)

Find Mismatch between ATI and NLP

Step 1

Step 2

Step 3
Approach Overview

1. **Step 1**: Find variables that interact (ATI Clusters)
2. **Step 2**: Find similar variable names (NLP Clusters)
3. **Step 3**: Find Mismatch between ATI and NLP

**Variable Name Similarity**
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’.

```c
int a = 0;
int x = 10;
int y = 100;
```
ATI Clusters

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

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

Diagram:
```
  a
→   →
  a   a
  x   x
   a
```

Green box: `x, y`
ATI Clusters

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

```c
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

1. Find variables that interacts (ATI Clusters)
2. Find similar variable names (NLP Clusters)
3. Find Mismatch between ATI and NLP
Variable Name Similarity

- \( \text{varsim}(v1, v2) \rightarrow [0,1] \)
Variable Name Similarity

• $\text{varsim}(v_1, v_2) \rightarrow [0, 1]$

Running Example:
‘in_authskey’ and ‘maxDepth’
Tokenization

Word Similarities
(wordsim(w1, w2))

Variable Similarities
(varsim(v1, v2))
Tokenization

Word Similarities \( (\text{wordsim}(w_1, w_2)) \)

Variable Similarities \( (\text{varsim}(v_1, v_2)) \)

\text{in_authskey} \rightarrow \text{Tokenization} \rightarrow
Tokenization

Word Similarities
\((\text{wordsim}(w_1, w_2))\)

Variable Similarities
\((\text{varsim}(v_1, v_2))\)

Plain case

in_authskey → Tokenization →
Tokenization

Word Similarities (\texttt{wordsim(w1, w2)})

Variable Similarities (\texttt{varsim(v1, v2)})

Plain case

\texttt{in_authskey} \rightarrow \textbf{Tokenization} \rightarrow

Abbreviations
Tokenization

Word Similarities
\(\text{wordsim}(w_1, w_2)\)

Variable Similarities
\(\text{varsim}(v_1, v_2)\)

in_authskey → Tokenization → [in, authentications, key]
Tokenization

Word Similarities
\(\text{wordsim}(w_1, w_2)\)

Variable Similarities
\(\text{varsim}(v_1, v_2)\)

Plain case

in_authskey → Tokenization → [in, authentications, key]

Abbreviations

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

Tokenization → Word Similarities \( \text{wordsim}(w_1, w_2) \) → Variable Similarities \( \text{varsim}(v_1, v_2) \)

Plain case

Abbreviations

Expansion

\[ \text{in\_authskey} \rightarrow \text{Tokenization} \rightarrow \text{[in, authentications, key]} \]
- **Solution:**
  - A tokenization algorithm.
  - Common programming abbreviations as an input.

\[
in\_authskey \rightarrow \text{Tokenization} \rightarrow [\text{in, authentications, key}] \]

\[
[\text{in, authentications, key}] \rightarrow [\text{maximum, Depth}] 
\]
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] ➔ wordsim(‘authentications’, ‘Depth’) ➔ [maximum, Depth]
Which sense to use?

\([\text{in}, \text{ authentications}, \text{ key}], \text{ maximum, Depth}\]

\(\text{wordsim('authentications', 'Depth')}\)
Which sense to use?

\[ \text{in, authentications, key} ]\rightarrow \text{wordsim(‘authentications’, ‘Depth’)}

\rightarrow \text{max (6 * 2 combination)}
Tokenization

Word Similarities
\( \text{wordsim}(w_1, w_2) \)

Variable Similarities
\( \text{varsim}(v_1, v_2) \)

\([\text{in, authentications, key}]\) \rightarrow \text{wordsim}(\text{‘authentications’, ‘Depth’}) \rightarrow \max (6 \times 2 \text{ combination})

\text{wordsim}(\text{‘authentications’, ‘Depth’}) = 0.36
Tokenization

Word Similarities
(wordsim(w1, w2))

Variable Similarities
(varsim(v1, v2))

[in, authentications, key]  [maximum, Depth]
Tokenization

Word Similarities
\( \text{wordsim}(w_1, w_2) \)

Variable Similarities
\( \text{varsim}(v_1, v_2) \)

\[
\text{[in, authentications, key] \quad \text{[maximum, Depth]}}
\]

\[
\text{wordsim(‘authentications’, ‘Depth’) = 0.36}
\]
Tokenization

Word Similarities
 \((\text{wordsim}(w_1, w_2))\)

Variable Similarities
 \((\text{varsim}(v_1, v_2))\)

\([\text{in, authentications, key}]\) \hspace{1cm} [\text{maximum, Depth}]

\[\text{wordsim('authentications', 'Depth')} = 0.36\]
\[\text{wordsim('authentications', 'maximum')} = 0.31\]
Tokenization

Word Similarities
\(\text{wordsim}(w_1, w_2)\)

Variable Similarities
\(\text{varsim}(v_1, v_2)\)

\[\text{in, authentications, key} \quad \quad \quad \text{[maximum, Depth]}\]

\[
\begin{align*}
\text{wordsim}(\text{‘authentications’, ‘Depth’}) &= 0.36 \\
\text{wordsim}(\text{‘authentications’, ‘maximum’}) &= 0.31 \\
\text{maxwordsim}(\text{‘authentications’}) &= 0.36
\end{align*}
\]
[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 \((\text{wordsim}(w_1, w_2))\) | Variable Similarities \((\text{varsim}(v_1, v_2))\)

\[
\begin{align*}
\text{wordsim(‘authentications’, ‘Depth’) } &= 0.36 \\
\text{wordsim(‘authentications’, ‘maximum’) } &= 0.31 \\
\text{maxwordsim(‘authentications’) } &= 0.36 \\
\text{varsim(‘in_authskey’, ’maxDepth’) } &= \text{Avg} \\
\end{align*}
\]
Approach Overview

Find variables that interacts. (ATI Clusters)

Variable Name Similarity.

Find similar variable names (NLP Clusters)

Find Mismatch between ATI and NLP.

Step1

Step2

Step3
NLP Clusters

• For each ATI cluster find similar variables.

ATI Cluster with m variables
NLP Clusters

• For each ATI cluster find similar variables.

• An $m \times m$ symmetric matrix

ATI Cluster with $m$ variables
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.

- An $m \times m$ symmetric matrix
- K-means clustering algorithm
NLP Clusters

• For each ATI cluster find similar variables.

  • An m*m symmetric matrix
  • K-means clustering algorithm
Approach Overview

Variable Name Similarity.

Step1: Find variables that interacts. (ATI Clusters)

Step2: Find similar variable names (NLP Clusters)

Step3: Find Mismatch between ATI and NLP.
ATI and NLP Mismatch

ATI Cluster

NLP Clusters
ATI and NLP Mismatch

ATI Cluster

NLP Clusters

indicates

Suspicious ATI cluster  ---  Cohesive NLP clusters
Rank Mismatch ATI Clusters
Rank Mismatch ATI Clusters

Suspicious ATI cluster indicates Cohesive NLP clusters
Rank Mismatch ATI Clusters

Suspicious ATI cluster \( \rightarrow \) Cohesive NLP clusters

Less Suspicious ATI cluster \( \rightarrow \) Less Cohesive NLP clusters
Rank Mismatch ATI Clusters

- Suspicious ATI cluster indicates cohesive NLP clusters.
- Less Suspicious ATI cluster indicates less cohesive NLP clusters.
- ATI cluster with cohesive NLP clusters at the top.
Evaluation

- Tokenization Algorithm
- End-to-end Testing
- Manually established ground truth.
- **2500** variable names from 4 programs.

<table>
<thead>
<tr>
<th></th>
<th>Exim</th>
<th>Grep</th>
<th>Valgrind</th>
<th>Putty</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>With Abbreviations</strong></td>
<td>95%</td>
<td>87%</td>
<td>81%</td>
<td>76%</td>
</tr>
<tr>
<td><strong>Without Abbreviations</strong></td>
<td>91%</td>
<td>75%</td>
<td>77%</td>
<td>66%</td>
</tr>
</tbody>
</table>
- 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`.

```c
delta[tree->label] = depth;
```
Two programs; Exim and Grep.

Analysed 5 top-ranked clusters.

One mistake in \texttt{grep} that assigns integer value to an unsigned char.

Variable in top-ranked cluster: \texttt{delta}, \texttt{depth}, \texttt{tree}, and \texttt{eolbyte}.

\begin{verbatim}
  delta[tree->label] = depth;
\end{verbatim}
• 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:
  \[ \text{delta, depth, tree, } \text{Unsigned char} \text{ and eolbyte.} \text{ int} \]

  \[ \text{delta[tree->label]} = \text{depth}; \]
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.