Recovering Clear, Natural Identifiers from Obfuscated (JavaScript) Names
var geom2d = function() {
  var t = numeric.sum;
  function r(n, r) {
    this.x = n;
    this.y = r;
  }

  u(r, {
    P: function e(n) {
      return t([this.x * n.x,
                 this.y * n.y ]));
    }
  });

  function u(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
  }

  return {
    V: r
  }
}();
var geom2d = function() {
    var t = numeric.sum;
    function r(n, r) {
        this.x = n;
        this.y = r;
    }
    u(r, {
        P: function e(n) {
            return t([this.x * n.x,
                        this.y * n.y ]);
        }
    });
    function u(n, r) {
        for (var t in r) n[t] = r[t];
        return n;
    }
    return {
        V: r
    };
}();
Today

Data-driven method + tool

```javascript
var geom2d = function() {
    var t = numeric.sum;
    function r(n, r) {
        this.x = n;
        this.y = r;
    }
    u(r, {
        P: function e(n) {
            return t([ this.x * n.x,
                        this.y * n.y ]);
        }
    });
    function u(n, r) {
        for (var t in r) n[t] = r[t];
        return n;
    }
    return { V: r }
};
}();
```

```javascript
var geom2d = function() {
    var sum = numeric.sum;
    function Vector2d(x, y) {
        this.x = x;
        this.y = y;
    }
    mix(Vector2d, {
        P: function dotProduct(vector) {
            return sum([ this.x * vector.x,
                        this.y * vector.y ]);
        }
    });
    function mix(dest, src) {
        for (var k in src) dest[k] = src[k];
        return dest;
    }
    return { V: Vector2d }
};
}();
```
Why?

• Programs are (also) written to be read

“Instead of imagining that our main task is to instruct a computer what to do, let us concentrate rather on explaining to human beings what we want a computer to do.” [Don Knuth]
Why?

• Programs are (also) written to be read

• Well-chosen variable names are critical to source code readability, reusability, maintainability

• Example tasks:
  • reverse engineering binaries
  • reverse engineering obfuscated JavaScript
  • consistent styling in large, distributed teams
Why?

• Programs are (also) written to be read

• Well-chosen variable names are critical to source code readability, reusability, maintainability

• Example tasks:
  • reverse engineering binaries
  • reverse engineering obfuscated JavaScript
  • consistent styling in large, distributed teams
Why?

- Programs are (also) written to be read.
- Well-chosen variable names are critical to source code readability, reusability, and maintainability.

Example tasks:
- reverse engineering binaries
- reverse engineering obfuscated JavaScript
- consistent styling in large, distributed teams

Martin Vechev, “Probabilistic Learning From Big Code”. Keynote at ISSTA 2016
Key ingredient

- The “naturalness” of software [Hindle et al, 2011]
Natural languages are complex
Natural languages are complex

Tiger, Tiger burning bright
In the forests of the night
What immortal hand or eye,
Could frame thy fearful symmetry?
..but most utterances are simple & repetitive
English, தமிழ், German

Can be Rich, Powerful, Expressive
English, தமிழ், German
Can be Rich, Powerful, Expressive

..but “in nature” is mostly Simple, Repetitive, Boring
English, தமிழ், German

Can be Rich, Powerful, Expressive

..but “in nature” is mostly Simple, Repetitive, Boring

Statistical Models
English, தமிழ், German
Can be Rich, Powerful, Expressive
..but “in nature” is mostly Simple, Repetitive, Boring

Statistical Models
The “naturalness of software” thesis

Programming Languages are complex...

...but **Natural Programs** are simple & repetitive.

and this, too, CAN BE EXPLOITED!!

[Hindle et al, 2011]
I know what you named your identifiers last summer!

Ik weet wat je je id's genaamd afgelopen zomer!
Variable Name

Guesser

(AUTONYM)

.org

Autonym

AUTONYM
function u(n, r) {
  for (var t in r) n[t] = r[t];
  return n;
}
function u(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}

function mix(dest, src) {
    for (var k in src) dest[k] = src[k];
    return dest;
}
What's the relevance of Machine Translation?
Noisy channel translation model
Noisy channel translation model
Noisy channel translation model

e ➔ distorted message ➔ f
Noisy channel translation model

\[ p(f | e) \]

channel model

\[ e \rightarrow \text{distorted message} \rightarrow f \]
Noisy channel translation model

\[ p(e) \]
language model

\[ p(f \mid e) \]
channel model

\[ e \rightarrow f \]
distorted message
Noisy channel translation model

\[ p(e) \quad p(f | e) \quad \frac{p(e)}{p(f)} \]

language model
channel model
distorted message

Goal: recover \( e \)
Noisy channel translation model

\[ p(e) \]
language model

\[ p(f \mid e) \]
channel model

\[ e \rightarrow \text{distorted message} \]

Goal: recover \( e \)

\[ e_{\text{best}} = \arg\max_e p(e \mid f) \]
Noisy channel translation model

Goals: recover $e$

$p(e)$
language model

$p(f \mid e)$
channel model

$e \rightarrow \rightarrow f$
distorted message

Bayes's theorem

$e_{\text{best}} = \arg\max_e p(e \mid f)$

$= \arg\max_e \frac{p(f \mid e)p(e)}{p(f)}$

$= \arg\max_e p(f \mid e)p(e)$

(for a given $f$)
Noisy channel translation model

Goal: recover $e$

$p(e)$ language model

$p(f | e)$ channel model

$e$ \rightarrow \text{distorted message} \rightarrow f$

$e_{\text{best}} = \arg\max_e \ p(f | e)p(e)$

Translation (channel distortion) model

Language model
Translating French ($f$) to English ($e$)
Translating French \((f)\) to English \((e)\)

Aligned French-English Corpus

\[ p(f | e) \]

Translation model

Language model
Translating French ($f$) to English ($e$)

Aligned French-English Corpus

$p(f \mid e)$

Translation model

$p(e)$

Language model

English Corpus
Translating French \( (f) \) to English \( (e) \)

Aligned French-English Corpus

\[ p(f \mid e) \]

Translation model

English Corpus

\[ p(e) \]

Language model
Translating minified ($f$) to clear JS ($e$)

Aligned Clear-Minified Code Corpus

$p(f \mid e)$

Translation model

$p(e)$

Language model
Translating minified ($f$) to clear JS ($e$)

GitHub + minifier

Aligned Clear-Minified Code Corpus

Clear Code Corpus

$p(f \mid e)$

Translation model

$p(e)$

Language model

MOSES statistical machine translation system
EN: I know what you named your identifiers!

NL: Ik weet wat je je ID's genoemd!
I know what you named your identifiers!

NL: Ik weet wat je je ID's genoemd!
EN: I know what you named your identifiers!

NL: Ik weet wat je je ID's genoemd!

Natural language: non-trivial alignment
- Reordering
- Different length
- Dropped words

```javascript
function u(n, r) {

function mix(dest, src){
```
EN: I know what you named your identifiers!
NL: Ik weet wat je je ID's genoemd!

Natural language: non-trivial alignment
• Reordering
• Different length
• Dropped words

Minification: straightforward alignment
Complications

```javascript
function r(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}
```
function r(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}
function \( r(n, r) \) {

for (var \( t \) in \( r \)) \( n[t] = r[t] \);

return \( n \);

}
function r(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}
Complications

(1) Overloading

```javascript
function r(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}

function mix(dest, src) {
}
```
Complications

(1) Overloading

function \( r(n, \ r) \) {
    for (var \( t \) in \( r \)) \( n[t] = r[t] \);
    return \( n \);
}

function \( \text{mix}(\text{dest}, \ src) \) {
}

Scope analysis

Autonym

19x424]function r(n, r){
for (var t in r) n[t] = r[t];
return n;
}
(2) Consistency

```javascript
function r(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}

function mix(dest, src) {
    for (var k in src) dest[k] = src[k];
    return dest;
}
```
Complications

(2) Consistency

```javascript
function r(n, r) {
    for (var t in r) n[t] = r[t];
    return n;
}
```

```javascript
function mix(dest, src) {
    for (var k in list) dest[k] = list[k];
    return dest;
}
```

Idea: try all, let language model decide which is more natural, on average, across ALL lines
Evaluation

• Held-out test set: 2,149 files

• Comparison to JSNice [Raychev et al, 2015]

• Metric: % names recovered
Evaluation

• Held-out test set: 2,149 files

• Comparison to JSNice [Raychev et al, 2015]

• Metric: % names recovered

• Global vs. local names (globals don’t change)

```javascript
var geom2d = function() {
    var t = numeric.sum;
    function r(n, r) {
        this.x = n;
        this.y = r;
    }
    ...
}
```

```javascript
var geom2d = function() {
    var sum = numeric.sum;
    function Vector2d(x, y) {
        this.x = x;
        this.y = y;
    }
    ...
}
```
% names recovered (2,149 test files)
Joining forces

Autonym File Accuracy

JSNice File Accuracy

Frequency

- 60
- 40
- 20
Becoming JSNaughty
Becoming JSNaughty

**Autonym**

- Pre-processing
- Moses SMT
- Post-processing

**JSNice**
% names recovered (2,149 test files)

The chart shows the percentage of names recovered using different renaming techniques:

- **Autonym**
- **JSNice**
- **JSNaughty**

The x-axis represents the names recovered, while the y-axis shows the percentage. Each technique is represented with a box plot, indicating the distribution of the percentage of names recovered across the 2,149 test files.
Examples

### Original: headers
- AUTONYM: headers
- JSNICE: headers
- JSNAUGHTY: headers

### Original: jsonStream
- AUTONYM: i
- JSNICE: s
- JSNAUGHTY: s

### Original: req
- AUTONYM: req
- JSNICE: req
- JSNAUGHTY: req

### Original: res
- AUTONYM: res
- JSNICE: res
- JSNAUGHTY: res

### Original: separator
- AUTONYM: data
- JSNICE: sep
- JSNAUGHTY: sep

### Original: error
- AUTONYM: err
- JSNICE: err
- JSNAUGHTY: err

```javascript
module.exports = http.createServer(function(e, r) {
  var t;
  var i = new stream.Stream();

  var n = "";
  csv().fromStream(e).on("data", function(e, r) {
    if (!t) {...}
    var a = {};
    _.zip(t, e).each(function(e) {...});
    i.emit("data", n + JSON.stringify(a));
    n = ",";
  }).on("end", function(e) {
    i.emit("data", "[]");
    i.emit("end");
  }).on("error", function(e) {
    i.emit("error", e);
    console.log("csv error", e.message);
  });
});
```

### Original: tuple
- AUTONYM: tuple
- JSNICE: key
- JSNAUGHTY: tuple
This material is based upon work supported by the National Science Foundation under Grant No. 1414172.
Machine translation for code

- Oda et al. (ASE ’15): code to pseudocode

```
# Python
if n % 3 == 0:
  Pseudo-code:
  if n is divisible by 3
```

- Karaivanov et al. (Onward! ’14): porting C# to Java

```
// C#
Console.WriteLine( "Hello World!" );

// Java
System.out.println( "Hello World!" );
```
Machine translation for code

- Oda et al. (ASE ’15): code to pseudocode

```python
if n % 3 == 0:
Pseudo-code:
if n is divisible by 3
```

- Karaivanov et al. (Onward! ’14): porting C# to Java

```csharp
// C#
Console.WriteLine ("Hello World!");

// Java
System.out.println("Hello World!");
```

- Nguyen et al. (FSE’ 13, ASE ’15): porting Java to C#

```java
public void findResultEdges() {
    for (Iterator it = dirEdgeList.iterator(); it.hasNext();)
    {
        DirectedEdge de = (DirectedEdge) it.next();...
    }
}

// C#
public void FindResultEdges() {
    foreach (DirectedEdge de in _dirEdgeList){...}
}