Thinking Like a Developer?

Comparing the Attention of Humans with Neural Models of Code

Michael Pradel

Software Lab – University of Stuttgart

Joint work with Matteo Paltenghi

Neural Software Analysis

Learning developer tools from large software corpora

Source code
Execution traces
Documentation
Bug reports
etc.

→ **Machine Learning** → **Predictive tool**

Neural Software Analysis

Learning developer tools from large software corpora

Source code
Execution traces
Documentation
Bug reports
etc.

→ **Machine Learning** →

New code, execution, etc.
↓
Predictive tool
↓
Information useful for developers

What are these models actually learning?

Idea: Compare Humans & Models

Developers

vs.

Machine Learning

Neural models of code

- Same task
- Same code examples
- Measure attention and effectiveness

Methodology

Task 1: Code Summarization

```
{
  if (!prepared(state)) {
    return state.setStatus(MovementStatus.PREPPING);
  } else if (state.getStatus() == MovementStatus.PREPPING) {
    state.setStatus(MovementStatus.WAITING);
  }
  if (state.getStatus() == MovementStatus.WAITING) {
    state.setStatus(MovementStatus.RUNNING);
  }
  return state;
}
```

Input: Method body → **Output: Method name**

`updateState`

Dataset: 250 methods from 10 Java projects *

* *A Convolutional Attention Network for Extreme Summarization of Source Code*, ICML'16

Task 2: Program Repair

```
public double sqrt(double x, double epsilon) {
  double approx = x / 2d;
  while (Math.abs(x - approx) > epsilon) {
    approx = 0.5d * (approx + x / approx);
  }
  return approx;
}
```

Input: Method with a buggy line

↓

Output: Fixed line

```
while (Math.abs(x - approx * approx) > epsilon) {
```

Dataset: 16 bugs from QuixBugs (Java) *

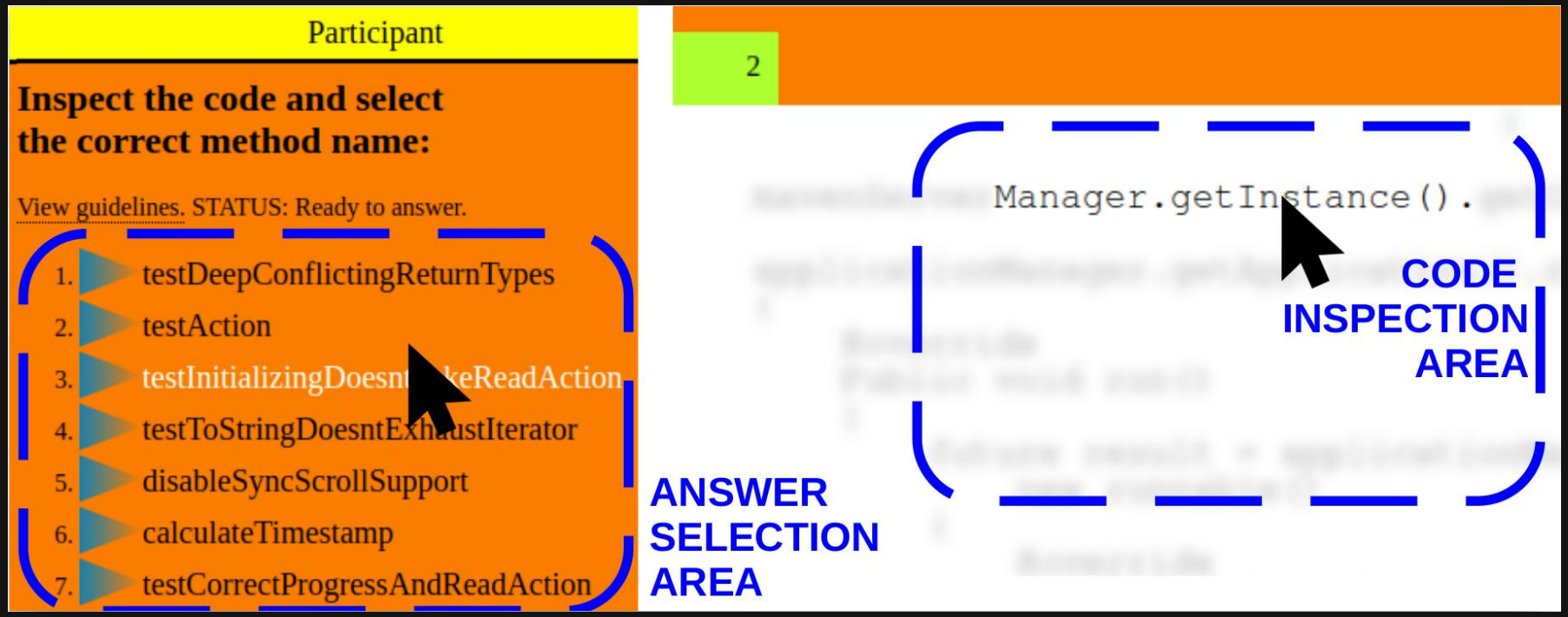
* *QuixBugs: A Multi-Lingual Program Repair Benchmark Set Based on the Quixey Challenge*, SPLASH'17 (Companion)

Capturing Human Attention

- **Goal: Track human attention while performing the task**
- **Approach: Unbluring-based web interface**
  - Initially, all code blurred
  - Moving mouse/cursor temporarily unblurs tokens

Capturing Human Attention

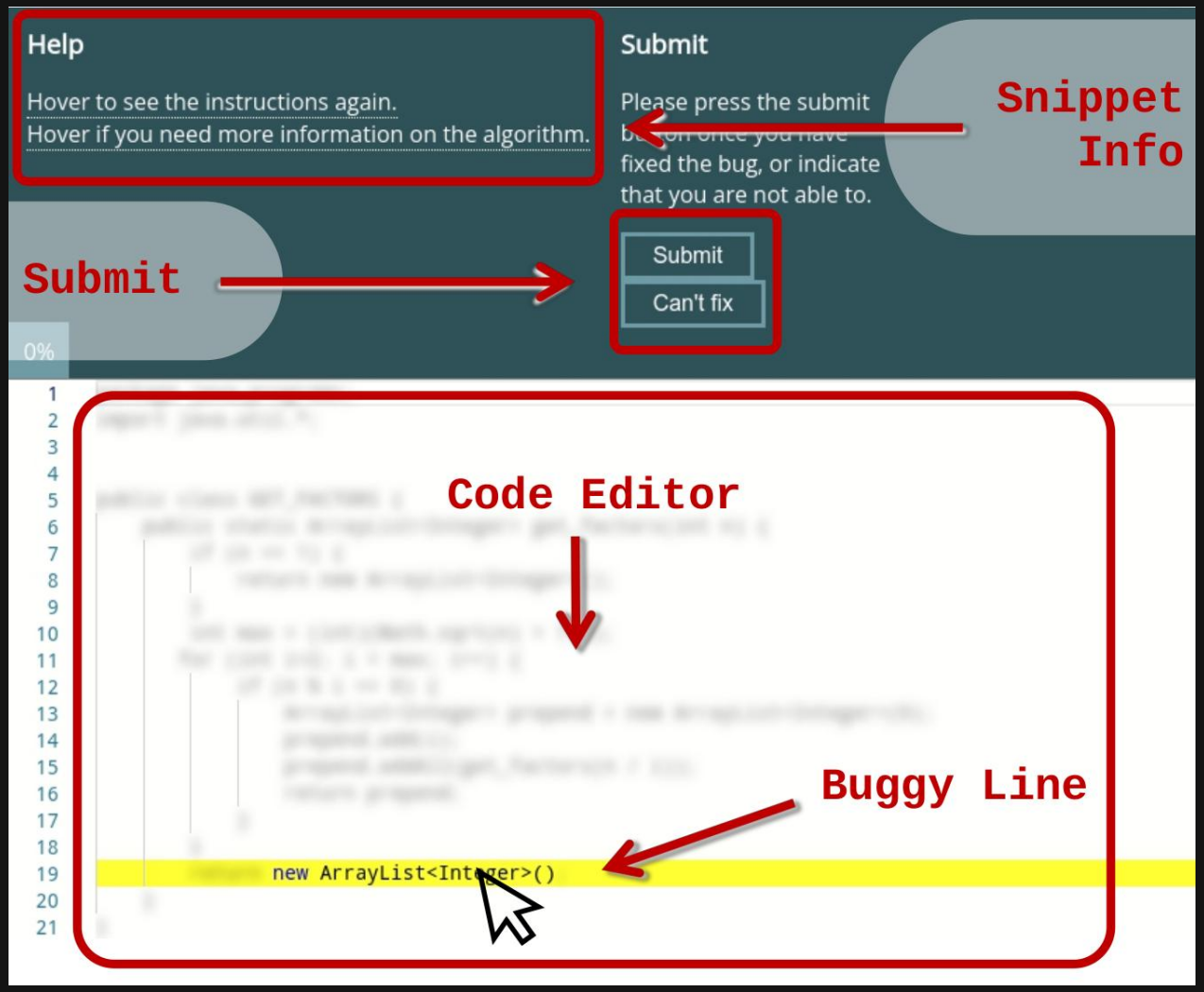
Task 1: Code Summarization



91 participants; 1,508 human attention records

Capturing Human Attention

Task 2: Program Repair



27 participants; 98 bug fixing records

Capturing Human Attention

Summarize fine-grained attention record into attention map:

```
public class SQRT {
    public static double sqrt(double x, double epsilon
        double approx = x / 2d;
        while (Math.abs(x-approx) > epsilon) {
            approx = 0.5d * (approx + x / approx);
        }
        return approx;
    }
}
```

Model Attention

	Attention	
	Regular	Copy
Code summarization		
CNN, ICML'16	✔	✔
Transformer, ACL'20	✔	✔
Program repair		
SequenceR, TSE'21	✔	✔
Recoder, FSE'21	✔	✘

Results

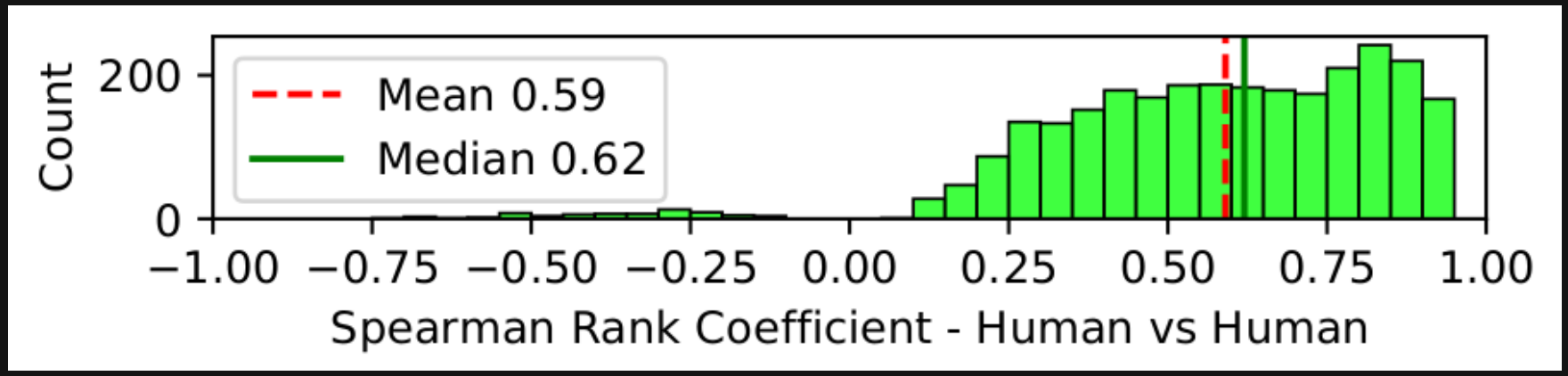
Human-Model Agreement

Do developers and models focus on the same tokens?

- Given for each code example
  - Human attention vector $\vec{h}$
  - Model attention vector $\vec{m}$
- Measure agreement between them
  - Spearman rank correlation: $\frac{cov(rg_{\vec{h}}, rg_{\vec{m}})}{\sigma_{rg_{\vec{h}}}, \sigma_{rg_{\vec{m}}}}$

Results: Summarization

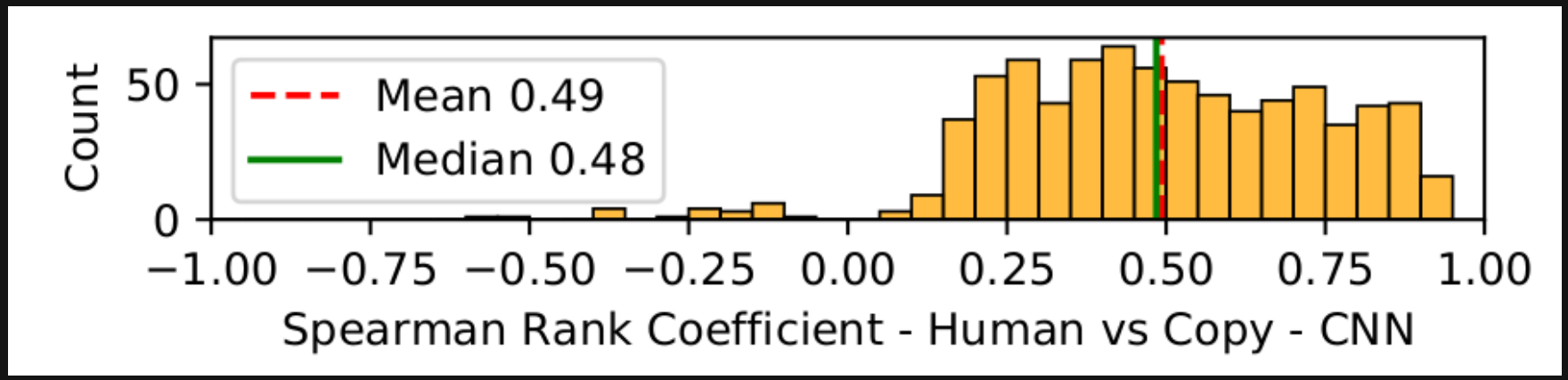
Human-human agreement:

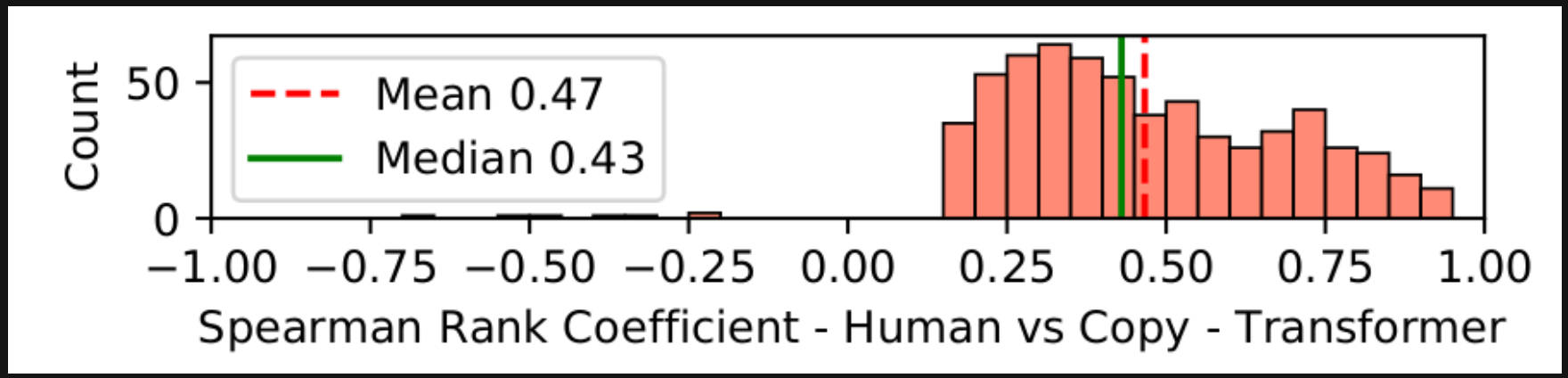


Developers mostly agree on what code matters most

Results: Summarization

Human vs. copy attention:

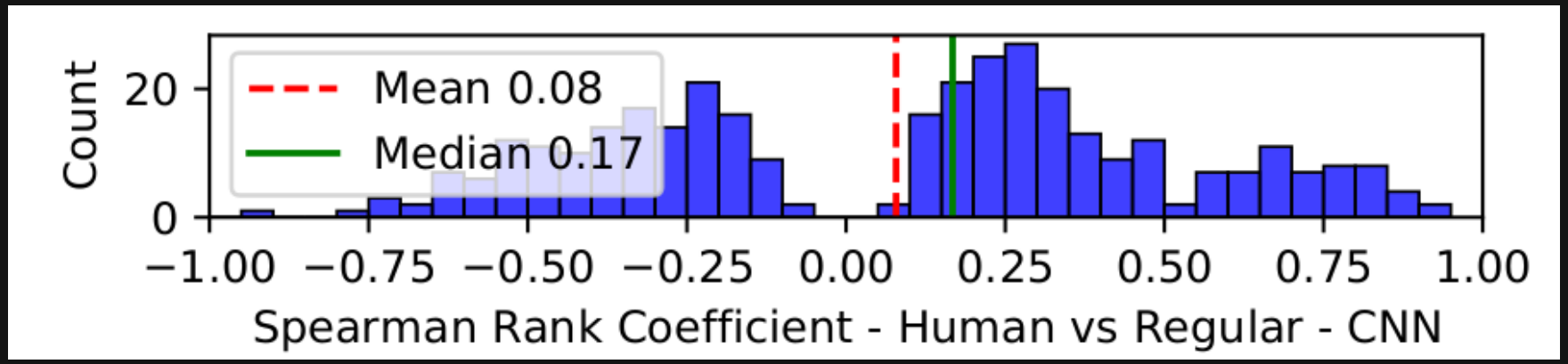


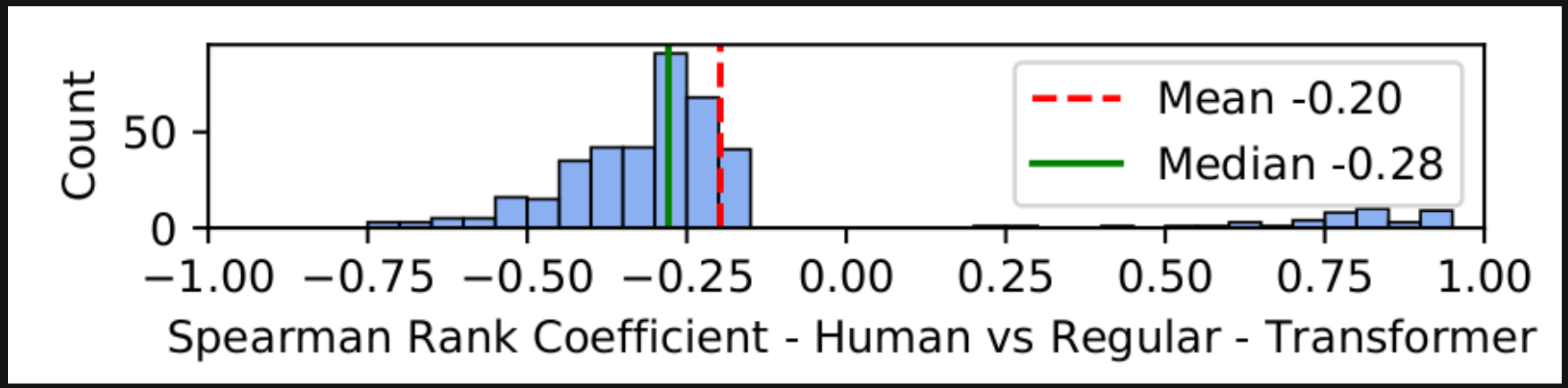


Empirical justification for copy attention

Results: Summarization

Humans vs. regular attention:

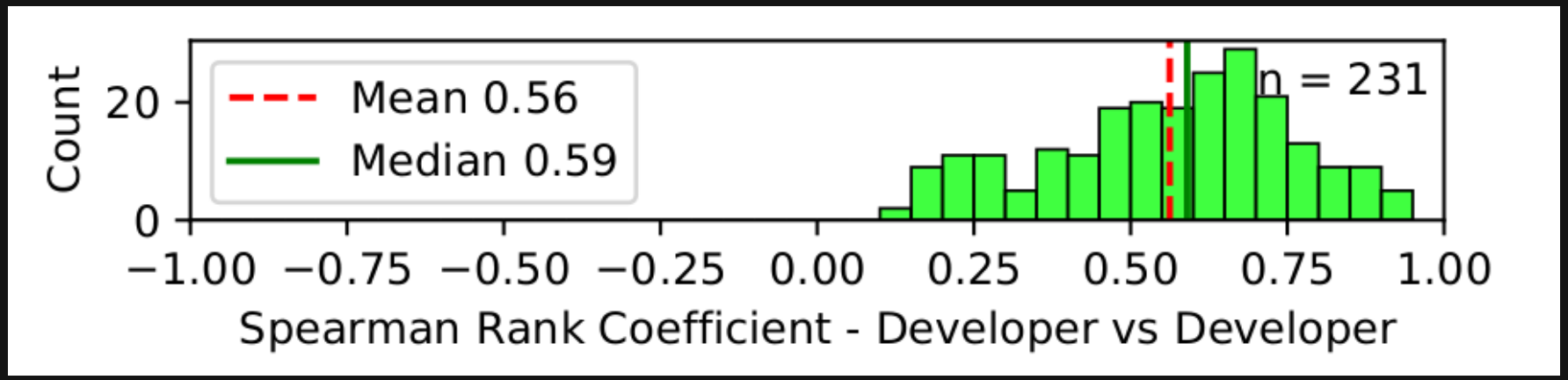




Lots of room for improvement!

Results: Program Repair

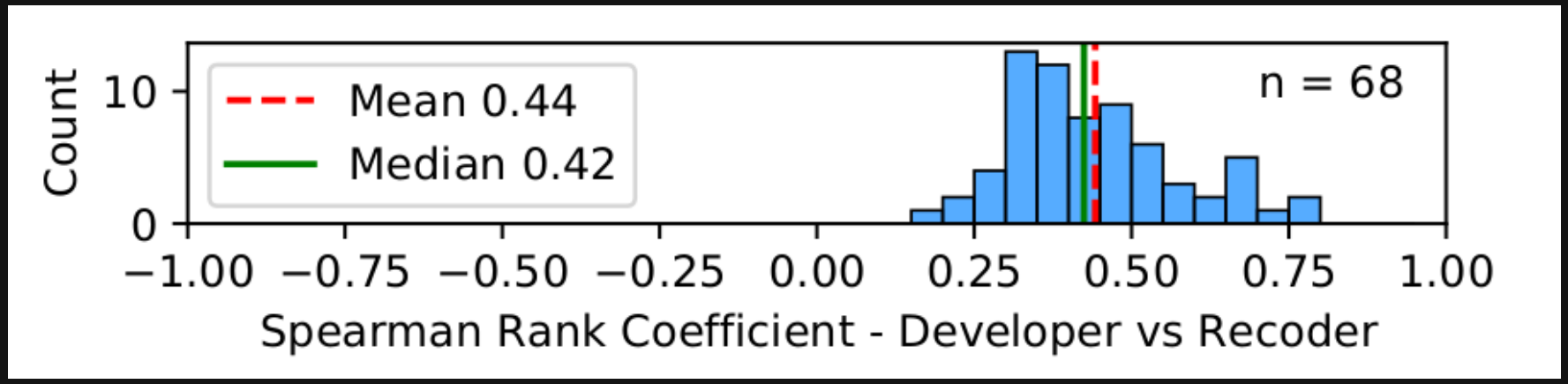
Human-human agreement:

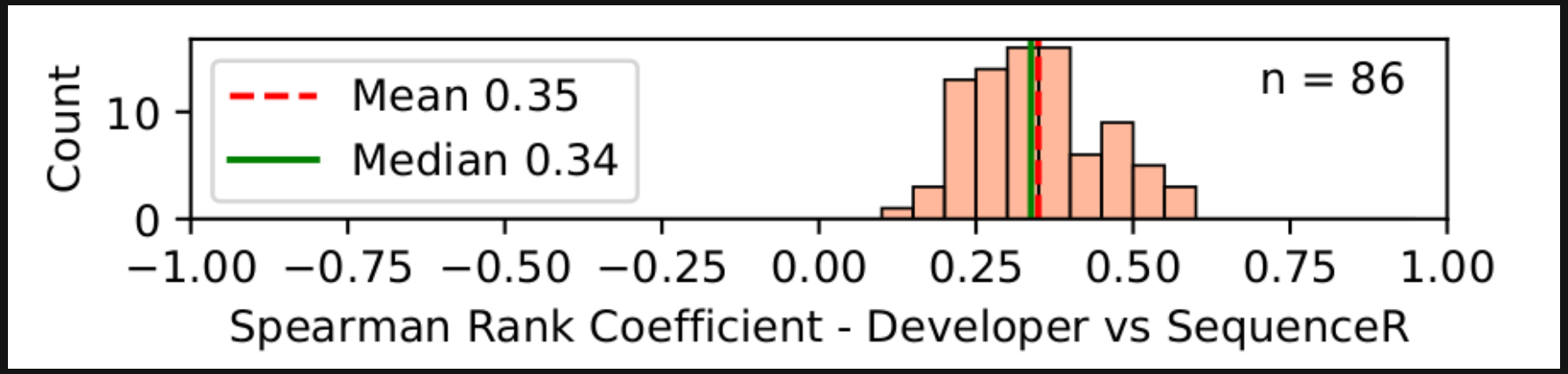


Developers mostly agree on what code matters most

Results: Program Repair

Human-model agreement:





Some room for improvement

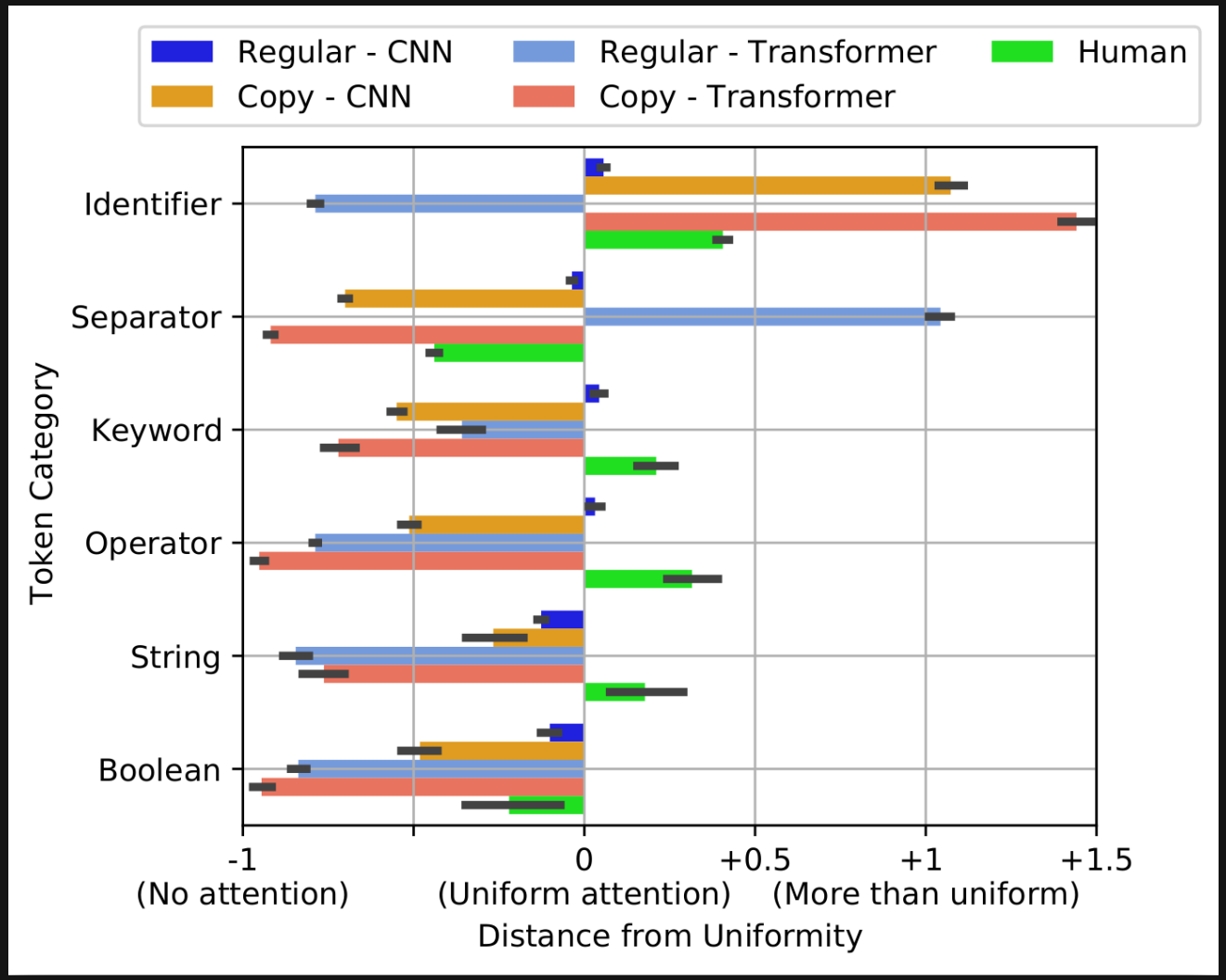
Tokens to Focus On

What kind of tokens to focus on?

- Different kinds: Identifiers, separators, etc.
- For each kind, compute distance from uniformity
  - $= 0$ means uniform attention
  - -1 means no attention at all
  - > 0 means more than uniform attention

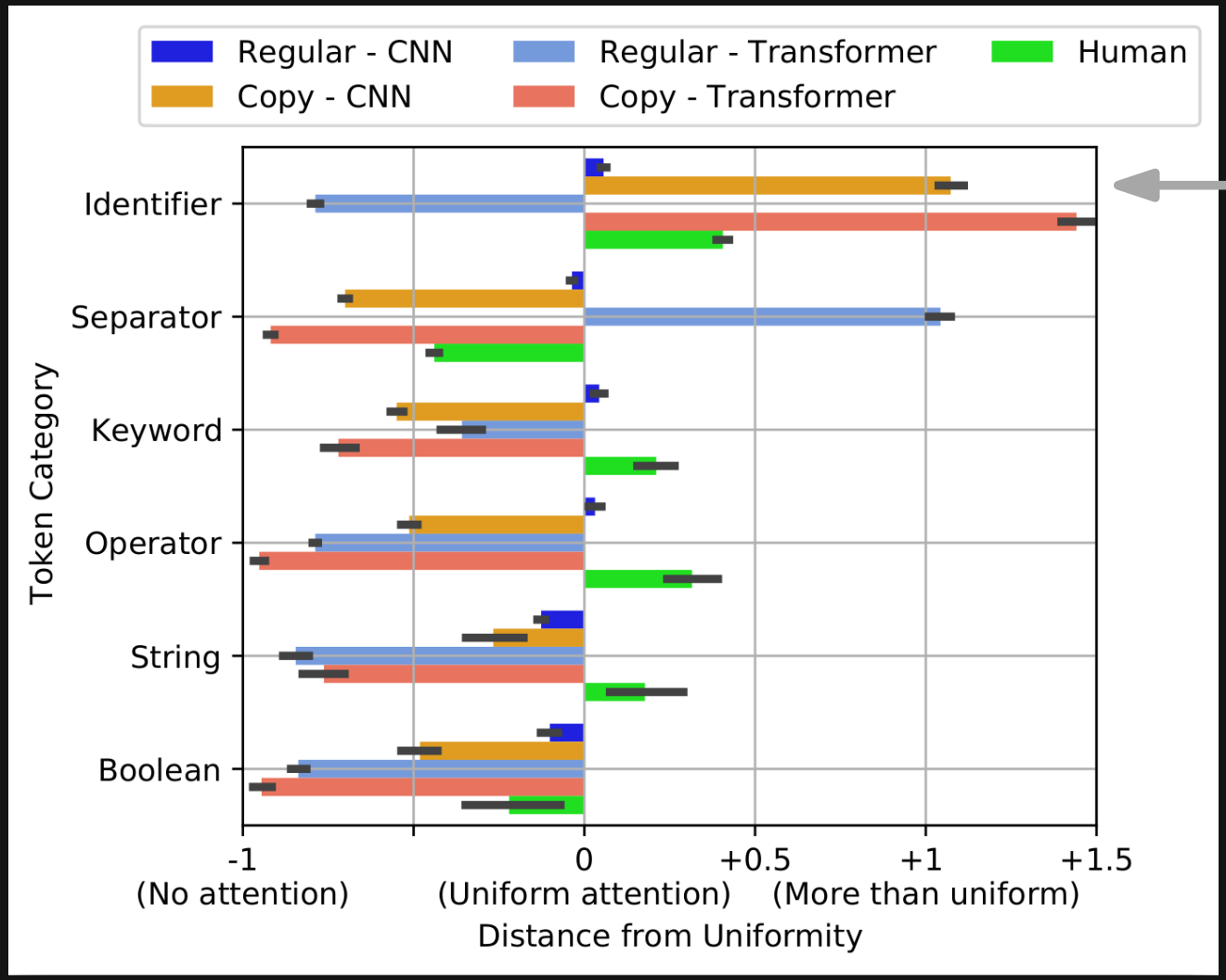
Results: Summarization

Distance from uniformity:



Results: Summarization

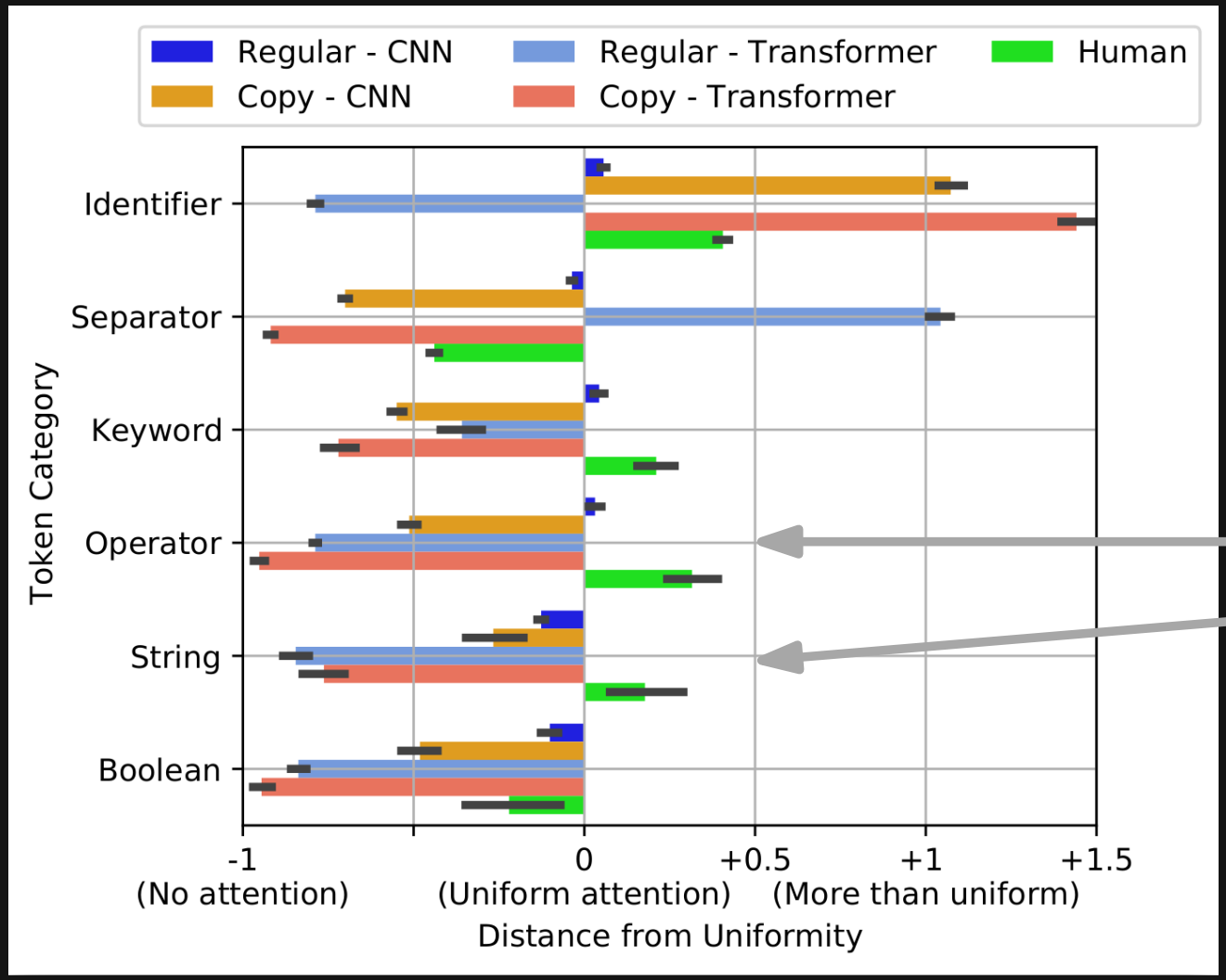
Distance from uniformity:



Identifiers are deemed important

Results: Summarization

Distance from uniformity:



Models mostly ignore some kinds of tokens

Results: Summarization

Example from Transformer model:

```
{
    log.debug("Requesting new token");
    int status = getHttpClient().executeMethod(method);
    if (status != 200)
    {
        throw new exception("Error logging in: " + method.getStatusLine());
    }
    document document = new saxBuilder(false).build(method.getResponseBodyAsStream()).getDocument();
    xPath path = xPath.newInstance("/response/token");
    element result = (element)path.selectSingleNode(document);
    if (result == null)
    {
        element error = (element)xPath.newInstance("/response/error").selectSingleNode(
            document);
        throw new exception(error == null ? "Error logging in" : error.getText());
    }
    myToken = result.getTextTrim();
}
```

Regular attention of neural model

```
{
    log.debug("Requesting new token");
    int status = getHttpClient().executeMethod(method);
    if (status != 200)
    {
        throw new exception("Error logging in: " + method.getStatusLine());
    }
    document document = new saxBuilder(false).build(method.getResponseBodyAsStream()).getDocument();
    xPath path = xPath.newInstance("/response/token");
    element result = (element)path.selectSingleNode(document);
    if (result == null)
    {
        element error = (element)xPath.newInstance("/response/error").selectSingleNode(
            document);
        throw new exception(error == null ? "Error logging in" : error.getText());
    }
    myToken = result.getTextTrim();
}
```

Human attention

Results: Summarization

Example from Transformer model:

```
{
    log.debug("Requesting new token");
    int status = getHttpClient().executeMethod(method);
    if (status != 200)
    {
        throw new exception("Error logging in: " + method.getStatusLine());
    }
    document document = new saxBuilder(false).build(method.getResponseBodyAsStream()).getDocument();
    xPath path = xPath.newInstance("/response/token");
    element result = (element)path.selectSingleNode(document);
    if (result == null)
    {
        element error = (element)xPath.newInstance("/response/error").selectSingleNode(
            document);
        throw new exception(error == null ? "Error logging in" : error.getText());
    }
    myToken = result.getTextTrim();
}
```

Regular attention of neural model

```
{
    log.debug("Requesting new token");
    int status = getHttpClient().executeMethod(method);
    if (status != 200)
    {
        throw new exception("Error logging in: " +
    }
    document document = new saxBuilder(false).build
    xPath path = xPath.newInstance("/response/token
    element result = (element)path.selectSingleNode
    if (result == null)
    {
        element error = (element)xPath.newInstance("/response/error").selectSingleNode(
            document);
        throw new exception(error == null ? "Error logging in" : error.getText());
    }
    myToken = result.getTextTrim();
}
```

Human attention

Model "wastes" attention on understanding syntax

Results: Summarization

Example from Transformer model:

```
{
    log.debug("Requesting new token");
    int status = getHttpClient().executeMethod(method);
    if (status != 200)
    {
        throw new exception("Error logging in: " + method.getStatusLine());
    }
    document document = new saxBuilder(false).build(method.getResponseBodyAsStream()).getDocument();
    xPath path = xPath.newInstance("/response/token");
    element result = (element)path.selectSingleNode(document);
    if (result == null)
    {
        element error = (element)xPath.newInstance("/response/error").selectSingleNode(
            document);
        throw new exception(error == null ? "Error
    }
    myToken = result.getTextTrim();
}
```

Model ignores tokens important to developers

```
    log.debug("Requesting new token");
    int status = getHttpClient().executeMethod(metho
    if (status != 200)
    {
        throw new exception("Error logging in: " + method.getStatusLine());
    }
    document document = new saxBuilder(false).build(method.getResponseBodyAsStream()).getDocument();
    xPath path = xPath.newInstance("/response/token");
    element result = (element)path.selectSingleNode(document);
    if (result == null)
    {
        element error = (element)xPath.newInstance("/response/error").selectSingleNode(
            document);
        throw new exception(error == null ? "Error logging in" : error.getText());
    }
    myToken = result.getTextTrim();
}
```

Human attention

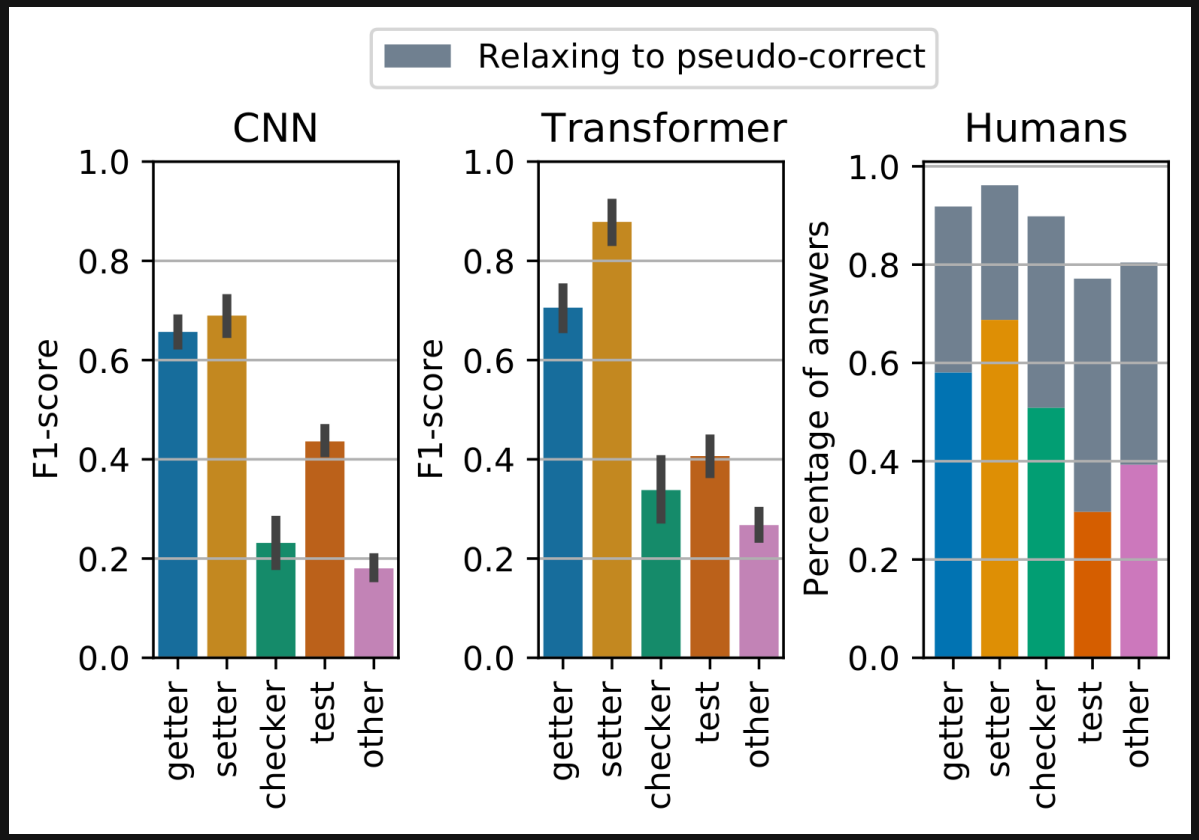
Effectiveness

Comparing developers and models w.r.t. their effectiveness at solving the task

- Strengths and weaknesses?
- Can current models compete with developers?

Results: Summarization

Comparing different kinds of methods:



Models underperform on non-trivial methods

Results: Program Repair

Success rate during program repair:

	Plausible patch ratio	
	Top-5	Top-100
SequenceR	2/80 (2.5%)	17/1395 (1.2%)
Recoder	2/80 (2.5%)	10/908 (1.1%)

Results: Program Repair

Success rate during program repair:

	Plausible patch ratio	
	Top-5	Top-100
SequenceR	2/80 (2.5%)	17/1395 (1.2%)
Recoder	2/80 (2.5%)	10/908 (1.1%)
	5-7 developers/bug	
Developers	68/98 (69.4%)	

Models are far from human effectiveness

Effectiveness vs. Agreement

Are models more effective when they agree more with developers?

Results: Summarization

Human-model agreement for all vs. accurate predictions:

	Spearman rank correl.	
	All methods	Methods with F1 $\geq$ 0.5
CNN (regular)	0.08	0.24
CNN (copy)	0.49	0.55
Transformer (reg.)	-0.20	0.02
Transformer (copy)	0.47	0.55

More human-like predictions are more accurate

Implications

- **Direct human-model comparison**
  - Helps understand why models (do not) work
- **Should create models that mimic humans**
  - Use human attention during training
  - Design models that address current weaknesses
    - E.g., understanding string literals

Conclusions

- **Available for future research:**
  - Interface for capturing human attention
  - Datasets of human attention records
- **More details:**

  Thinking Like a Developer? Comparing the Attention of Humans with Neural Models of Code, ASE'21