

# Choose wisely: code smells in automatically generated code

**Joanna Cecilia da Silva Santos**

*Assistant Professor*

Department of Computer Science and Engineering

University of Notre Dame

 joannacss@nd.edu

 <https://joannacss.github.io/>

 @joannacss

```
2
3
4 Machine,
5
6 Pls make website,
7
8 all responsive like,
9 w/ BIG pictures ooo,
10 use my fav fonts,
11 also fancy menus with whooosh on,
12 load fast pls
13
14 Thanks,
15 Human
16
17 PS no bugs :)
18
19
```

# AI-Based Code Generation Tools



GitHub  @github · Jun 29, 2021

Meet GitHub Copilot - your AI pair programmer. [copilot.github.com](https://copilot.github.com)



```
send_tweet.py

10 def send_tweet_with_image(message, image):
11     """Send a tweet with an image attached"""
12     # Twitter authentication
13     auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
14     auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
15     api = tweepy.API(auth)
16
17     # Send the tweet with the image
18     api.update_with_media(image, status=message)
19
20
21
22
```

0:08 1.9M views



**GitHub**  
Copilot



**Functionality**



**Quality**

830

15.1K

30.4K



# Code smells in automatically generated code

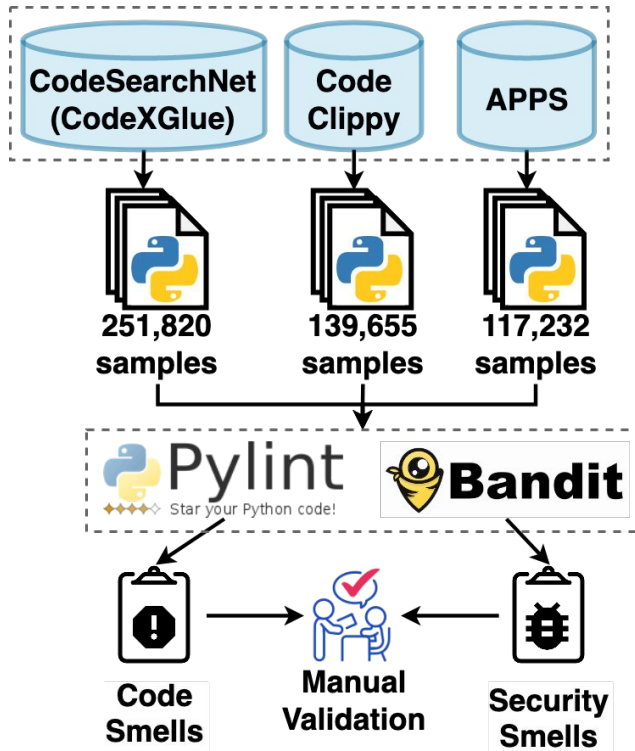


Code smells are **symptoms** that may indicate the system has **flaws**

- Maintainability issues
- Technical debt
- Security\*

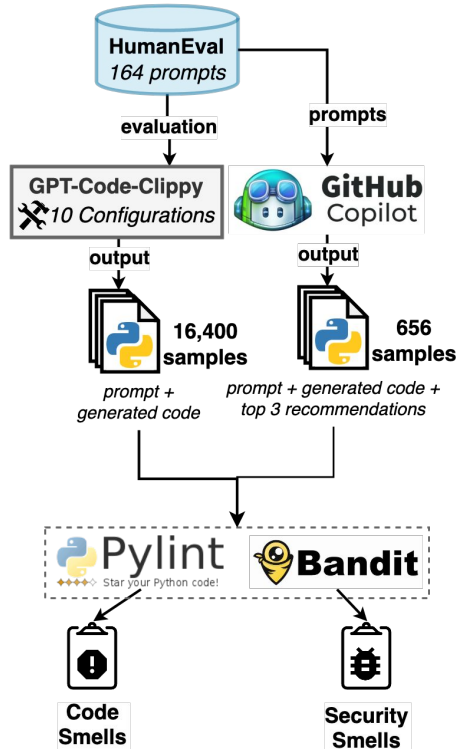
*Security smells*

# Are Code Smells present in training sets?



Dataset	“Smelly” Samples (non-security)	Smelly Samples (security-related)
APPS	69.15%	1.62%
Code Clippy	39.12%	10.27%
CodeXGlue	97.03%	5.32%

# Are **Code Smells** present in the **generated code**?



## (Non-security) Code smells:

- *Undefined variables,*
- *Lines too long*
- *Duplicate code,*
- *Unused arguments*

## Security smells:

- Use of `eval`
- Use of weak hash functions

# Examples from Copilot

**CWE-89: SQL Injection**

**CWE-798: Use of Hard-coded Credentials**

```
1  def showUsers(username):  
2      """  
3      Get user information from the MySQL database with the given username  
4      """
```

)

# Examples from Copilot

## CWE-89: SQL Injection

## CWE-798: Use of Hard-coded Credentials

```
1  def showUsers(username):
2      """
3      Get user information from the MySQL database with the given username
4      """
5      # Connect to the database
6      db = MySQLdb.connect(host="localhost", user="root", passwd="password", db="test")
7      # Create a cursor object
8      cursor = db.cursor()
9      # Execute the SQL command
10     cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
11     # Fetch all the rows in a list of lists.
12     results = cursor.fetchall()
13     # Close the cursor
14     cursor.close()
15     # Close the database connection
16     db.close()
17     # Return the results
18     return results
```



A photograph of a laptop and a black mug on a blue background. The laptop screen shows code for a Vue.js application, including Vue Router and Vuex configuration. A semi-transparent grey box is overlaid on the screen, containing the text "What can you do about it?".

What can you do about it?

Why should you care?

# Choose wisely: code smells in automatically generated code




*Learn more about the work:*



Joanna Cecilia da Silva Santos

 joannacss@nd.edu

 <https://joannacss.github.io/>

 @joannacss

# Choose wisely: code smells in automatically generated code

Learn more about the work:



**Joanna Cecilia da Silva Santos**

*Assistant Professor*

Department of Computer Science and Engineering

University of Notre Dame

 [joannacss@nd.edu](mailto:joannacss@nd.edu)

 <https://joannacss.github.io/>

 [@joannacss](https://twitter.com/joannacss)

# AI-Based Code Generation Tools




GitHub  @github · Jun 29, 2021



Meet GitHub Copilot - your AI pair programmer. [copilot.github.com](https://copilot.github.com)

```
send_tweet.py

10 def send_tweet_with_image(message, image):
11     """Send a tweet with an image attached"""
12     # Twitter authentication
13     auth = tweepy.OAuthHandler(CONSUMER_KEY, CONSUMER_SECRET)
14     auth.set_access_token(ACCESS_TOKEN, ACCESS_TOKEN_SECRET)
15     api = tweepy.API(auth)
16
17     # Send the tweet with the image
18     api.update_with_media(image, status=message)
19
20
21
```

 Copilot

0:08 1.9M views



**GitHub**  
Copilot

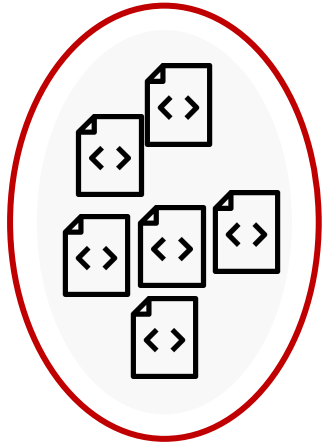
 830

 15.1K

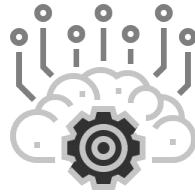
 30.4K



# How do they work?



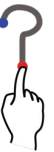
Large Language Learning Model



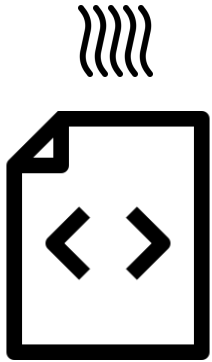
Functionality



Quality



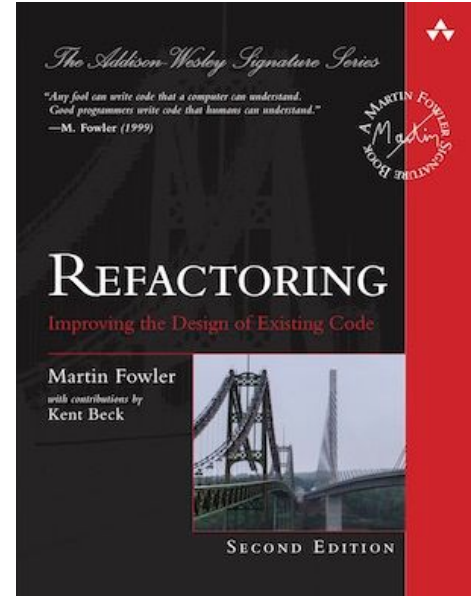
# Code smells in generated code



Code smells are **symptoms** that may indicate the system has **flaws**

- Maintainability issues
- Technical debt
- Security\*

*Security smells*



# Code Smells and Security Smells

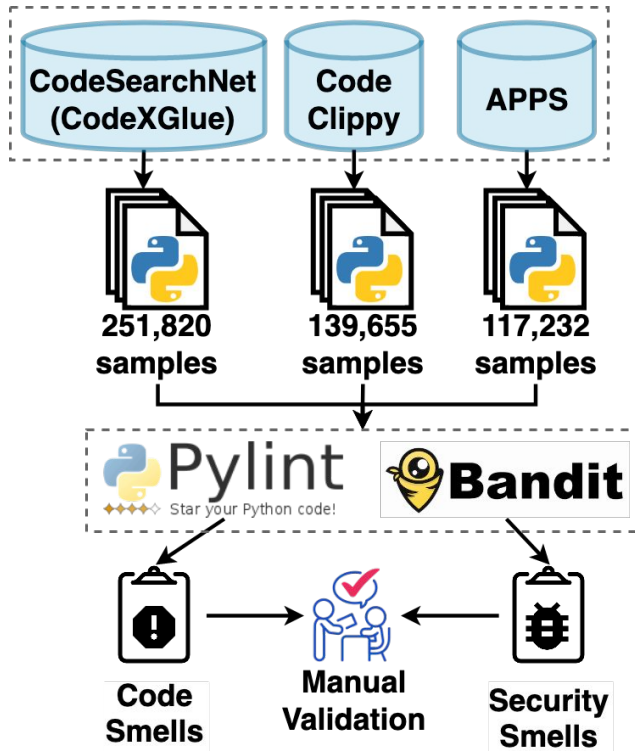
```
1 try:
2     age = int(input())
3 except Exception:
4     raise
5 except TypeError:
6     raise
```

*Using the wrong exception catching order*

```
1 def verifyAdmin(password):
2     if password != "passw0rd!":
3         return False
4     return True
```

*Use of Hard-coded Credentials (CWE-798)*

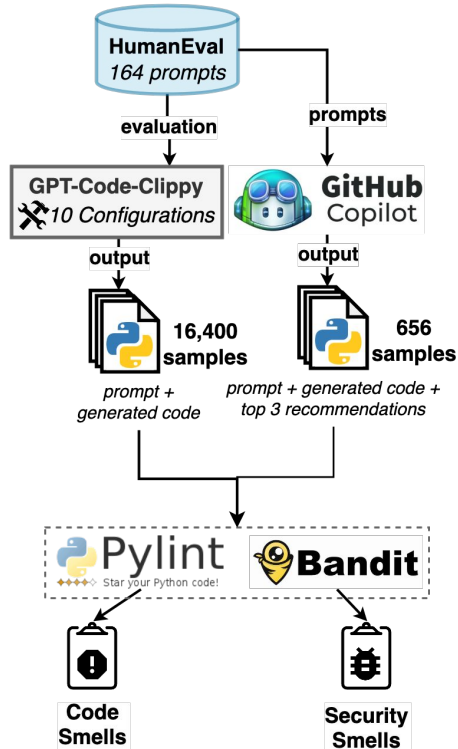
# Are Code Smells present in training sets?



Dataset	“Smelly” Samples (non-security)	Smelly Samples (security-related)
APPS	69.15%	1.62%
Code Clippy	39.12%	10.27%
CodeXGlue	97.03%	5.32%



# Are Code Smells present in the generated code?



## (Non-security) Code smells:

- *Undefined variables,*
- *Lines too long*
- *Duplicate code,*
- *Unused arguments*

## Security smells:

- using `assert`
- Use of `eval`
- Use of weak hash functions (MD2, MD4, MD5, or SHA1)

# Examples from Copilot

CWE-89: SQL Injection

CWE-798: Use of Hard-coded Credentials

```
1  def showUsers(username):
2      """
3      Get user information from the MySQL database with the given username
4      """
5      # Connect to the database
6      db = MySQLdb.connect(host="localhost", user="root", passwd="password", db="test")
7      # Create a cursor object
8      cursor = db.cursor()
9      # Execute the SQL command
10     cursor.execute("SELECT * FROM users WHERE username = %s", (username,))
11     # Fetch all the rows in a list of lists.
12     results = cursor.fetchall()
13     # Close the cursor
14     cursor.close()
15     # Close the database connection
16     db.close()
17     # Return the results
18     return results
```

# Examples from Copilot

CWE-327: Use of a Broken or Risky Cryptographic Algorithm

CWE-321: Use of Hard-coded Cryptographic Key

```
1  import hashlib
2  def hashCompare(text):
3      """
4      Calculate the hash of the text using a common hash function and compare it
5      """
6      hash = hashlib.md5(text.encode()).hexdigest()
7      if hash == "5f4dcc3b5aa765d61d8327deb882cf99":
8          print("Hash matches!")
9      else:
10         print("Hash does not match!")
```

# Examples from Copilot

CWE-327: Use of a Broken or Risky Cryptographic Algorithm

CWE-321: Use of Hard-coded Cryptographic Key

```
1  import hashlib
2  def hashCompare(text):
3      """
4      Calculate the hash of the text using a common hash function and compare it
5      """
6      hash = hashlib.md5(text.encode()).hexdigest()
7      if hash == "5f4dcc3b5aa765d61d8327deb882cf99":
8          print("Hash matches!")
9      else:
10         print("Hash does not match!")
```

Overall structure:

- Connect
- Answer
- Re-tell

→ Remove background part (they know GitHub Copilot, what is a code smell)

**Overall flow:**

→ Start with an opening line such as an error handler written by Steven C. that gets exception, looks up on SO and execute the highest voted answer. Although it was a joke, was he really wrong about it? You, probably, wouldnt use such as an error handler on production but that might be what you're doing in the end by using AI-based generated code.

→ Now let's talk about smells

→ Ends up with a discussion about the risks of using these tools. Yes, they give you efficiency (if you measure productivity in terms of lines of code)

→ Our research shows that. Hence, running these AI-based tools poses a threat to your product

→ Take away message: if you care about the quality, running linter and see what they say. In the end of the day, a git blame will blame you