The University Of Sheffield.

Attack of the Code Clones

Software Reengineering
(COM3523 / COM6523)

The University of Sheffield

What is a code clone?

Repeated fragments of code that do the same thing.

Remember the "cross-cutting concerns" discussed wrt. Aspect Oriented Programming.

- The same sequences of commands required to interact with an API / write to a file, etc.

Why do we not like code clones?

- Contribute to **"code bloat"** - more code to manage.
- **Harder to maintain** - changes to a cloned piece of code need to be replicated across all clones.
- Can lead to **inconsistent updates**.
- Can lead to **copyright or code usage license violations** (especially when code is copied without attribution).

A Symptom of Design Deterioration

Studies suggest that **7-23% of a typical software system is cloned**.

Some obvious mechanisms to avoid duplication.

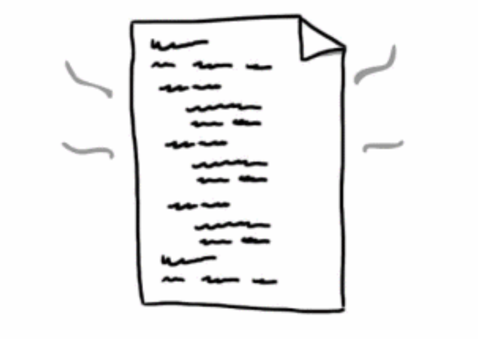
- The whole point of functions or methods!
- In an Object-Oriented System, there is also inheritance.

Not necessarily the result of "bad design".

- System may have been well-designed to begin with.

Tend to arise out of day-to-day code evolution.

- Can be difficult to create links (i.e. function calls) across a code-base.



By Bokanko - CC BY-SA 3.0

Kapser, Cory J., and Michael W. Godfrey. ""Cloning considered harmful" considered harmful: patterns of cloning in software." Empirical Software Engineering 13.6 (2008): 645.

The Dangers of Copying and Pasting from StackOverflow

Ragkhitwetsagul, C., Krinke, J., Paixao, M., Bianco, G., & Oliveto, R. (2019). Toxic code snippets on stack overflow. *IEEE Transactions on Software Engineering*, *47*(3), 560-581.

Solutions on StackOverflow are often pasted from existing projects.

- Often outdated - e.g. contain vulnerabilities that have been since fixed in the host projects.
- Ragkhitwetsagul et al. found this in 66% (101) of the StackOverflow clones they examined.

Often violate license agreements.

- StackOverflow applies CC Attribution-Sharealike 3.0 license to content in posts.
- Cannot host code that is, for example, published under GPL license.
- Ragkhitwetsagul et al. found 7,112 clones of 214 license-incompatible code snippets in 2,427 GitHub projects.

The Challenge of Detecting Clones

Consider these 16 lines of code.

- Can you spot the clone?

Code does the same thing, but isn't *identical*.

- Just one variable name the difference here.
- Could be worse…

These fragments are right next to each other.

Consider a system of ~675,000 Lines of Code (Weka).

- Thousands of methods.
- Duplicate code could be in completely different packages.

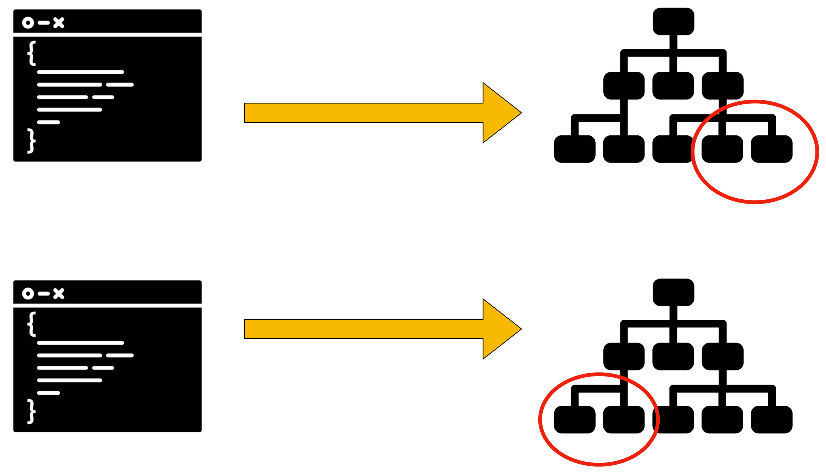
```
extern int array_a[];
extern int array_b[];

int sum_a = 0;

for (int k = 0; k < 4; k++){
   sum_a += array_a[k];
}

int average_a = sum_a / 4;

int sum_b = 0;
// going to sum it all up...
for (int i = 0; i < 4; i++)
   sum_b += array_b[i];

int average_b = sum_b / 4;
```

Four Types of Code Duplication

Type	Meaning
1: Exact	Exact copy. Perhaps some differences in white space usage, but the code is identical.
2: Parameterised	Type 1, **but** variables in the code might have different names.
3: Near miss	Type 2, **but** some statements might be changed, added, or removed.
4: Semantic	The syntax may be completely different, but the behaviour (the semantics) of the two pieces of code is identical.

Identifying clones

Comparing Parse-Trees



Can be expensive.

- Graph isomorphism is NP-complete.

Requires a parser.

- Language specific.

High-fidelity

- Straightforward to establish what the variables are, etc.

Text Processing to the Rescue

Possible to process large volumes of source code without parsers.

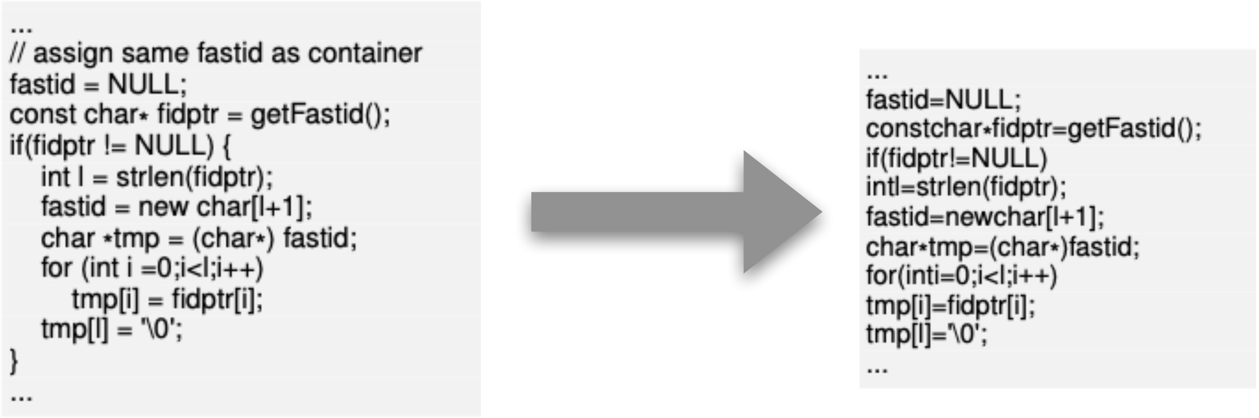
- Treat them as normal "text".
- Turn text into "data" that can be analysed.
- Lots of distance measures, encoding techniques, etc.

Files are commonly compared on a pair-wise basis.

Particularly good for type-1 clone detection.

Pre-process code to "normalise" it.

Eliminate brackets, white space, comments.



```
...
// assign same fastid as container
fastid = NULL;
const char* fidptr = getFastid();
if(fidptr != NULL) {
   int l = strlen(fidptr);
   fastid = new char[l+1];
   char *tmp = (char*) fastid;
   for (int i =0;i<l;i++)
      tmp[i] = fidptr[i];
   tmp[l] = '\0';
}
...
```

```
...
fastid=NULL;
constchar*fidptr=getFastid();
if(fidptr!=NULL)
intl=strlen(fidptr);
fastid=newchar[l+1];
char*tmp=(char*)fastid;
for(inti=0;i<l;i++)
tmp[i]=fidptr[i];
tmp[l]='\0';
...
```

How similar are two text documents?

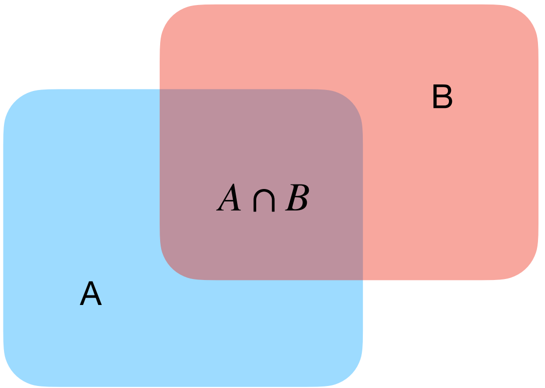
Jaccard Similarity

Your document is a set of elements.

- Each element might be a line of text for example.

A measure for how similar two sets are.

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} = \frac{|A \cap B|}{|A| + |B| - |A \cap B|}$$



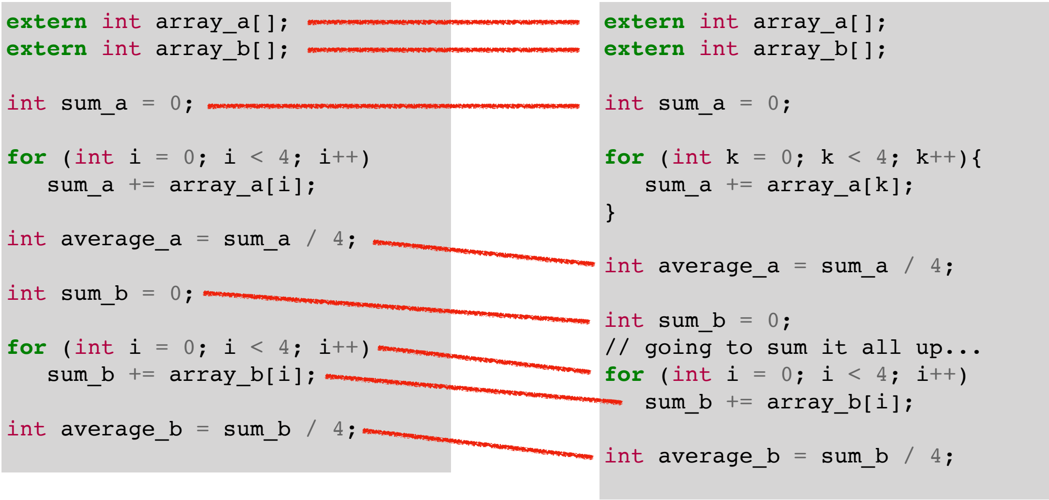
The encoding of these sets is flexible.

- When comparing a pair of files, the sets could contain strings corresponding to each line in the respective files.
- When comparing statements, the sets could contain characters.

There are alternatives!

- Sets can be problematic - do not consider frequency.

Example for comparing two documents



```
extern int array_a[];
extern int array_b[];

int sum_a = 0;

for (int i = 0; i < 4; i++)
   sum_a += array_a[i];

int average_a = sum_a / 4;

int sum_b = 0;

for (int i = 0; i < 4; i++)
   sum_b += array_b[i];

int average_b = sum_b / 4;
```

```
extern int array_a[];
extern int array_b[];

int sum_a = 0;

for (int k = 0; k < 4; k++){
   sum_a += array_a[k];
}

int average_a = sum_a / 4;

int sum_b = 0;
// going to sum it all up...
for (int i = 0; i < 4; i++)
   sum_b += array_b[i];

int average_b = sum_b / 4;
```

Numbers assume that we ignore white-space (empty lines), and lines with single brackets, etc.

$$|A \cap B| = 8$$

$$|A| = 10 \quad |B| = 12$$

$$\frac{|A \cap B|}{|A| + |B| - |A \cap B|} = \frac{8}{10 + 12 - 8} = \frac{8}{14} = 0.57$$

Dot Plots

We can learn lessons from Bioinformatics

Similar to the problem of finding binding sites in protein sequences.

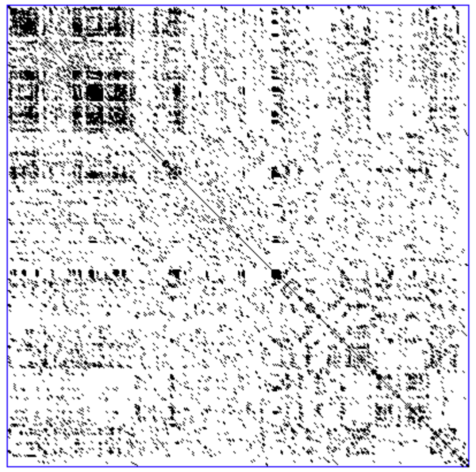
Sequences of characters, need to find the commonalities.

Common to use “dot plots”.

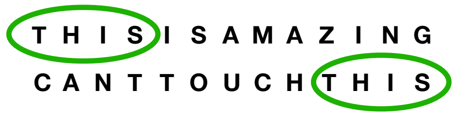
One sequence is represented by the x-axis, and one by the y-axis.

A “dot” happens if an element from the x-axis matches corresponding element in the y-axis.

At scale: Can summarise code comparison in a picture.

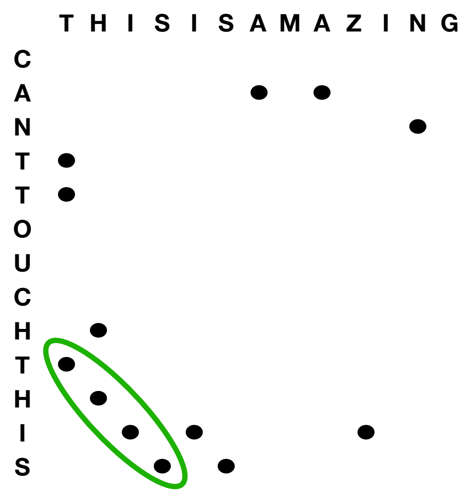


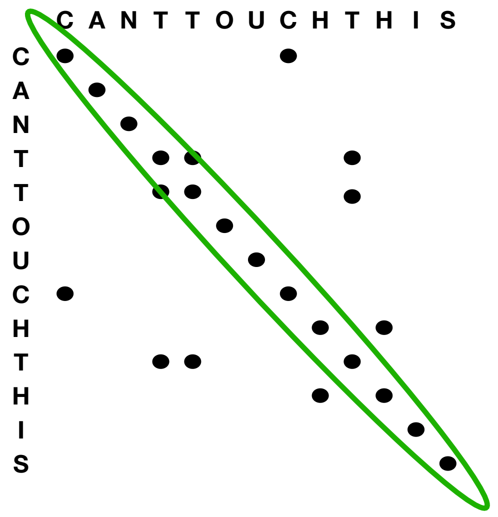
A DNA dot plot of a human zinc finger transcription factor (GenBank ID NM_002383), showing regional self-similarity.

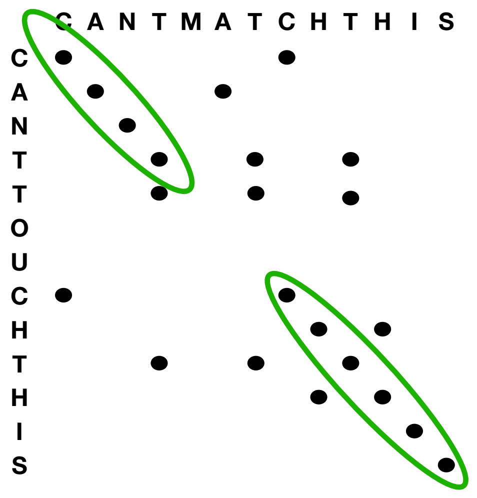


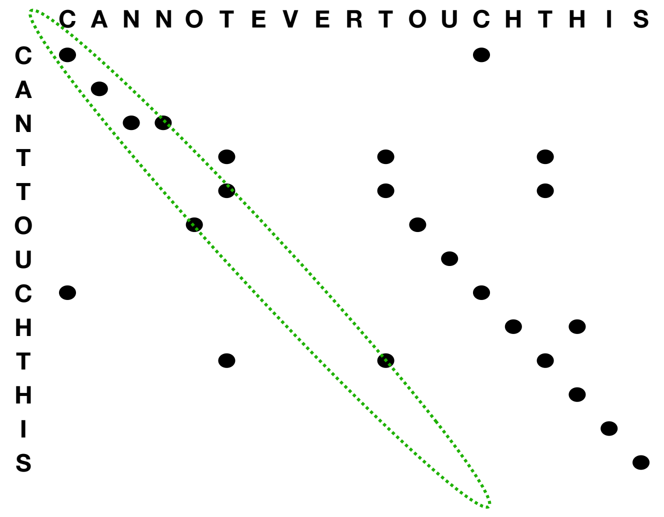
THISISAMAZING

CANTTOUCHTHIS


THISISAMAZING
CANTTOUCHTHIS


CANTTOUCHTHIS
CANTTOUCHTHIS


CANTMATCHTHIS
CANTTOUCHTHIS


CANNOTEVERTOUCHTHIS
CANTTOUCHTHIS

Key Take-aways

- Code clones are problematic for a number of reasons
  - Indicate design problems.
  - Can also indicate potential vulnerabilities or license violations from other sources, e.g. StackOverflow.
- Four types of code clone.
  - Type 1 to Type 4 - remember what they are?
- Several detection approaches.
  - Parsing text and comparing parse-trees.
  - Text-processing - e.g. Jaccard Index between text files.
- Can be visualised by Dot plots.