Language-Independent Clone Detection Applied to Plagiarism Detection

Romain Brixtel, Mathieu Fontaine, Boris Lesner, Cyril Bazin
University of Caen

Romain Robbes
University of Chile
Students cheat.
Students cheat. a lot!
1. Differences between clone and plagiarism detection

2. A language-independent approach to plagiarism detection

3. Preliminary results
Find the 5 differences

Clones

Plagiarism
Extensive transformations

data Piece = Vide | Noir | Blanc | Extremite
    deriving (Eq, Show)

type Plateau = [[Piece]]
type Points = (Int,Int)

taillePlateau :: Int
taillePlateau = 8

positions :: [Points]
positions = [(0, 1), (1, 1), (1, 0), (1, -1), (0, -1), (1, -1), (-1, 0), (-1, 1)]

initialisePlateau :: Plateau
initialisePlateau = [[ (f x y) | x <- [0..9] ] | y <- [0..9]]
    where f x y
        | x == 0 || y == 0 || x == 9 || y == 9  = Extremite
        | x == 4 && y == 4 = Blanc
        | x == 5 && y == 5 = Blanc
        | x == 4 && y == 5 = Noir
        | x == 5 && y == 4 = Noir
        | otherwise = Vide

data Piece = Empty | Black | White | Wall deriving (Eq, Show)
type Board = [[Piece]]
type Pt = (Int,Int)

boardSize :: Int
boardSize = 8

directions :: [Pt]
directions = [(0, 1), (-1, 1), (-1, 0), (-1, -1), (0, -1), (1, -1), (1, 0), (1, 1)]

initBoard:: Board
initBoard = [[ (f x y) | x <- [0..9] ] | y <- [0..9]]
    where f x y
        | x == 0 || y == 0 || x == 9 || y == 9  = Wall
        | x == 4 && y == 4 = White
        | x == 5 && y == 5 = White
        | x == 4 && y == 5 = Black
        | x == 5 && y == 4 = Black
        | otherwise = Empty
Larger clones
Larger clones
Less documents

Assignment 4  Linux
Several languages

<?
$dbc=odbc_connect("gbook","","");
if (!$dbc)
    exit("Connection Failed: ". $dbc);;
$qary="SELECT * FROM comments";
$rs=odbc_exec($dbc,$qary);
if (!$rs)
    exit("Error in SQL");
for ($i=0; $i<count($rs); $i++)
    echo '<h3>RS Access powered Guest Book</h3>

while (odbc_fetch_row($rs))
{
    $name=odbc_result($rs,"name");
    $comment=odbc_result($rs,"comment");
    $date=odbc_result($rs,"entry_date");

declare Pig_2 =
pigment {  
  color_map {  
    0.00, rgb <0.35, 0.58, 0.88>*1.0  
    0.25, rgb <0.35, 0.58, 0.88>*1.0  
    0.50, rgb <0.35, 0.58, 0.88>*0.9  
    0.75, rgb <0.35, 0.58, 0.88>*0.8  
    1.00, rgb <0.35, 0.58, 0.88>*0.7  
  }
  scale 0.1
}  

}  

if (ldb(n))
    a,b,c = 0,1,0
while c < n:
    print a,
    a,b,c = b,a+b,c+1

main = do
    initGUI
    lohaXmlM <- xmlNew "helloworld.glade"
    let lohaXml = case lohaXmlM of
        (Just lohaXml) -> lohaXml
        Nothing -> error "Cannot find .glade file in current directory"
    window <- xmlGetWindow lohaXml castToWindow "window1"
    onDestory window mainQuit
    cbutton <- xmlGetWidget lohaXml castToButton "button2"

Bash
Prolog
Lisp
Scheme
Haskell
Smalltalk
C
C++
Python
PHP
Java
Perl
CLIPS
XML & XSLT
Ruby
Pov-ray
SQL
HTML & CSS
Javascript

{def x $letter}
(repeat i 0 6 1)
(repeat j 0 6 1)
(loc 0 37 3 (loc i j 0 (byname X))
}
{view}
(set z 1)
(repeat k 0 73 1)
(repeat i 2 6 1)
(repeat j 0 6 1)
(loc 0 k 2 (loc i j 0 (sprite ".")
(loc 4 z 1 (sprite ":")
(addto z 3))
{def z(c)
  for c in cs
      f = yield(c)
      puts "$c is now #f"
end}
def z(c)

/=
father["Bill","John"].
father["Fred","Bill"].
/= 
c2f(c) do lcl
  (c=9/5)+32 
end 

grandfather(Person,Grandfather) :-
    father[Parent,Grandfather],
    father[Father,Parent].

read(STDIN, $buffer, SHEV('CONTENT_LENGTH'));
$pairs = split('/', $buffer);
foreach $pair @pairs |
    ($name, $value) = split('/', $pair);
    $value = eval/$value/;
    $value = $value[a-zA-Z][a-Z][a-Za-Z] 
if (input($name)) { $INPUT$=name; 
else { $INPUT$=name, $value; 

unless ($INPUT$=email)) { 
    print "Content-type: text/html\n\n";
    ctrl; 

    $temp = 0;
    $temp = prop('QUERY_STRING');
    s = ($temp)
    foreach ( 
        $INPUT$=address = $temp;
        $return; 
    )
We want to catch everyone
Differences and consequences

1. Extensive transformations

2. Larger clones with reordering

3. Less documents

4. Several languages

5. We want to catch everyone

Extensive normalization

Line matching algorithm

Less performance need

Language independent

Recall > precision
Introducing ... the "Pomp-o-mètre"
[A-Za-z0-9]+ → 't'
data Piece = Vide | Noir | Blanc | Extremite
deriving (Eq, Show)

type Plateau = [[Piece]]
type Points = (Int,Int)
taillePlateau :: Int
taillePlateau = 8

[A-Za-z0-9]+ → 't'
t t = t  |  t  |  t  |  t
   t (t, t)
t t = [[[t]]]
t t = (t, t)
t :: t
t = t
<table>
<thead>
<tr>
<th></th>
<th>9</th>
<th>21</th>
<th>28</th>
<th>23</th>
<th>1</th>
<th>4</th>
<th>31</th>
<th>12</th>
<th>5</th>
<th>25</th>
<th>13</th>
<th>22</th>
</tr>
</thead>
<tbody>
<tr>
<td>9</td>
<td>0.00</td>
<td>0.77</td>
<td>0.80</td>
<td>0.89</td>
<td>0.87</td>
<td>0.80</td>
<td>0.90</td>
<td>0.87</td>
<td>0.81</td>
<td>0.86</td>
<td>0.81</td>
<td>0.84</td>
</tr>
<tr>
<td>21</td>
<td>0.77</td>
<td>0.00</td>
<td>0.80</td>
<td>0.92</td>
<td>0.89</td>
<td>0.81</td>
<td>0.84</td>
<td>0.86</td>
<td>0.81</td>
<td>0.80</td>
<td>0.82</td>
<td>0.89</td>
</tr>
<tr>
<td>28</td>
<td>0.80</td>
<td>0.80</td>
<td>0.00</td>
<td>0.92</td>
<td>0.92</td>
<td>0.88</td>
<td>0.92</td>
<td>0.88</td>
<td>0.89</td>
<td>0.91</td>
<td>0.92</td>
<td>0.94</td>
</tr>
<tr>
<td>23</td>
<td>0.89</td>
<td>0.92</td>
<td>0.00</td>
<td>0.78</td>
<td>0.80</td>
<td>0.88</td>
<td>0.85</td>
<td>0.88</td>
<td>0.81</td>
<td>0.90</td>
<td>0.85</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>0.87</td>
<td>0.89</td>
<td>0.92</td>
<td>0.78</td>
<td>0.00</td>
<td>0.81</td>
<td>0.87</td>
<td>0.88</td>
<td>0.89</td>
<td>0.88</td>
<td>0.88</td>
<td>0.90</td>
</tr>
<tr>
<td>4</td>
<td>0.80</td>
<td>0.81</td>
<td>0.88</td>
<td>0.80</td>
<td>0.81</td>
<td>0.00</td>
<td>0.55</td>
<td>0.69</td>
<td>0.76</td>
<td>0.80</td>
<td>0.88</td>
<td>0.84</td>
</tr>
<tr>
<td>31</td>
<td>0.90</td>
<td>0.84</td>
<td>0.92</td>
<td>0.88</td>
<td>0.87</td>
<td>0.55</td>
<td>0.00</td>
<td>0.79</td>
<td>0.83</td>
<td>0.88</td>
<td>0.91</td>
<td>0.91</td>
</tr>
<tr>
<td>12</td>
<td>0.87</td>
<td>0.86</td>
<td>0.88</td>
<td>0.85</td>
<td>0.88</td>
<td>0.69</td>
<td>0.79</td>
<td>0.00</td>
<td>0.85</td>
<td>0.87</td>
<td>0.91</td>
<td>0.85</td>
</tr>
<tr>
<td>5</td>
<td>0.81</td>
<td>0.81</td>
<td>0.89</td>
<td>0.89</td>
<td>0.76</td>
<td>0.83</td>
<td>0.85</td>
<td>0.00</td>
<td>0.81</td>
<td>0.89</td>
<td>0.92</td>
<td></td>
</tr>
<tr>
<td>25</td>
<td>0.86</td>
<td>0.80</td>
<td>0.91</td>
<td>0.81</td>
<td>0.88</td>
<td>0.80</td>
<td>0.88</td>
<td>0.87</td>
<td>0.81</td>
<td>0.00</td>
<td>0.89</td>
<td>0.90</td>
</tr>
<tr>
<td>13</td>
<td>0.81</td>
<td>0.82</td>
<td>0.92</td>
<td>0.90</td>
<td>0.88</td>
<td>0.88</td>
<td>0.91</td>
<td>0.91</td>
<td>0.89</td>
<td>0.89</td>
<td>0.00</td>
<td>0.90</td>
</tr>
<tr>
<td>22</td>
<td>0.84</td>
<td>0.89</td>
<td>0.94</td>
<td>0.85</td>
<td>0.90</td>
<td>0.84</td>
<td>0.91</td>
<td>0.85</td>
<td>0.92</td>
<td>0.90</td>
<td>0.90</td>
<td>0.00</td>
</tr>
</tbody>
</table>

The diagram illustrates the process of plagiarism detection, starting with a source code corpus and filtered source code corpus, followed by segmentation and similarity measures, and then segment matching. The filtering and matching processes are repeated, leading to document matching and finally presentation of plagiarism results at the document and corpus levels. The table provides a heatmap representation of the matching scores, with values ranging from 0.00 to 1.00.
Empirical validation on 3 corpuses

<table>
<thead>
<tr>
<th>Corpus name</th>
<th># Documents</th>
<th># Couples</th>
<th># Suspects</th>
<th># Plagiarised</th>
<th>Recall</th>
<th>Precision</th>
<th>$F_2$ measure</th>
</tr>
</thead>
<tbody>
<tr>
<td>HASKELL</td>
<td>13</td>
<td>78</td>
<td>3</td>
<td>3</td>
<td>1.0</td>
<td>1.0</td>
<td>1.0</td>
</tr>
<tr>
<td>PYTHON</td>
<td>15</td>
<td>105</td>
<td>20</td>
<td>4</td>
<td>1.0</td>
<td>0.2</td>
<td>0.55</td>
</tr>
<tr>
<td>C</td>
<td>19</td>
<td>171</td>
<td>7</td>
<td>4</td>
<td>1.0</td>
<td>0.57</td>
<td>0.87</td>
</tr>
</tbody>
</table>

We consider that we detect plagiarism when the distance between a pair of documents is less than the mean distances of the matrix.
1. Clone and plagiarism detection are similar, but distinct problems.

2. The "pomp-o-mètre" is language-independent and features extensive normalization.

3. Larger empirical validations are needed, but no large plagiarism benchmark exists.
My controversial statement

\[[A-Za-z0-9]+ \rightarrow 't'\]
Related work

Ducasse et.al.
Wettel & Marinescu

Baldr
Anti-Copias