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.

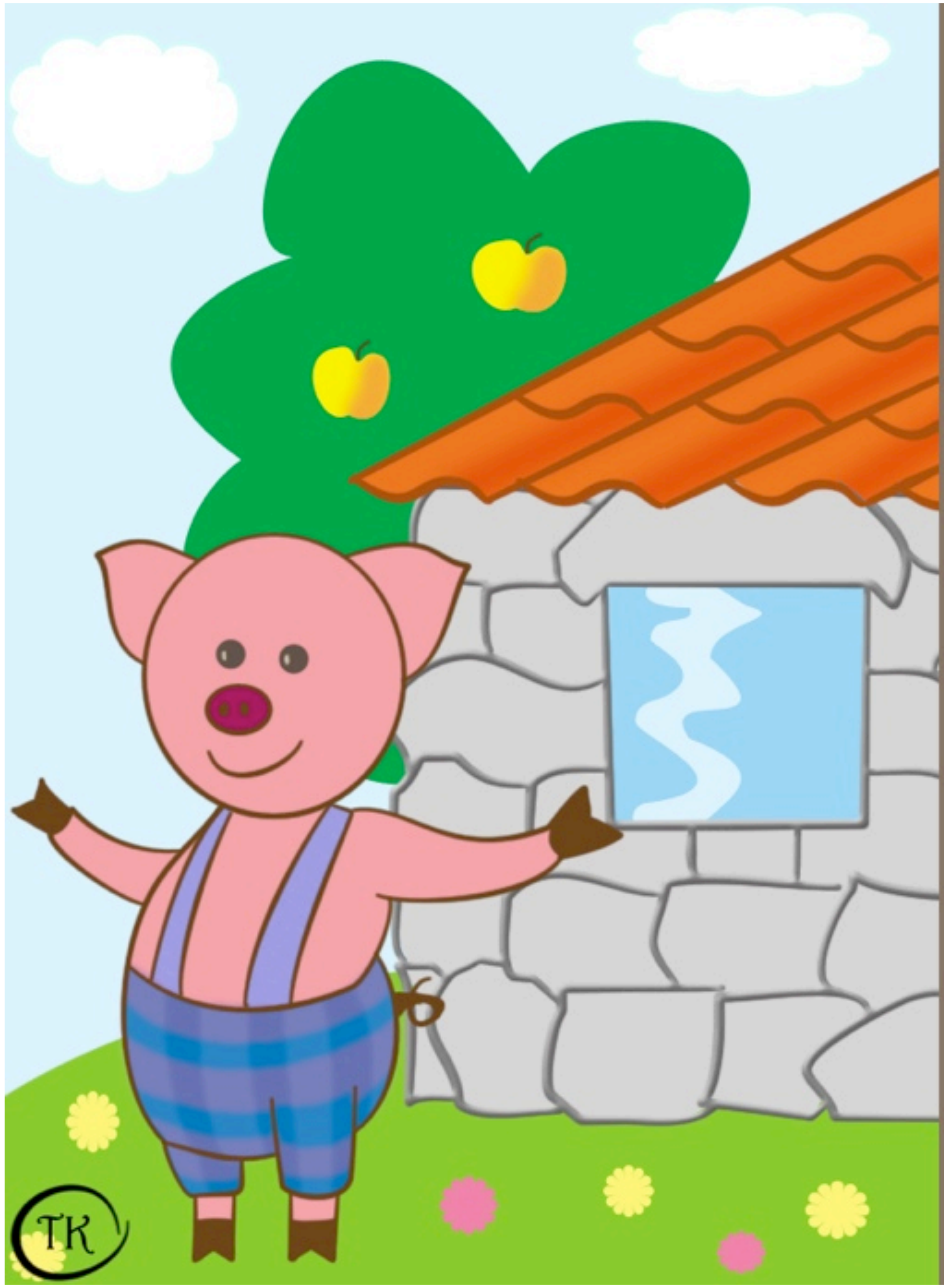
Students cheat. a lot!

1. Differences between clone and plagiarism detection

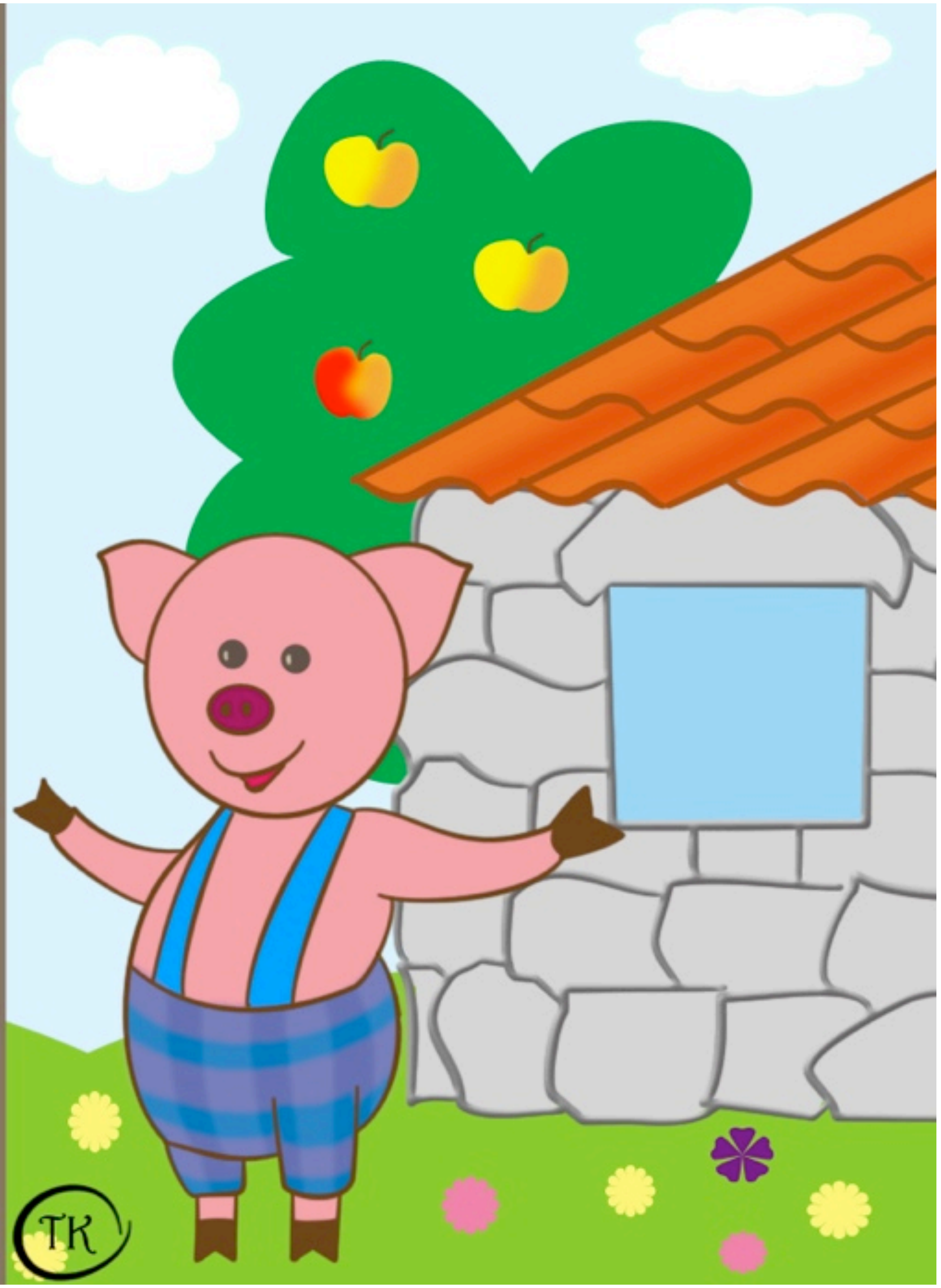
2. A language-independent approach to plagiarism detection

3. Preliminary results

Find the 5 differences



Clones



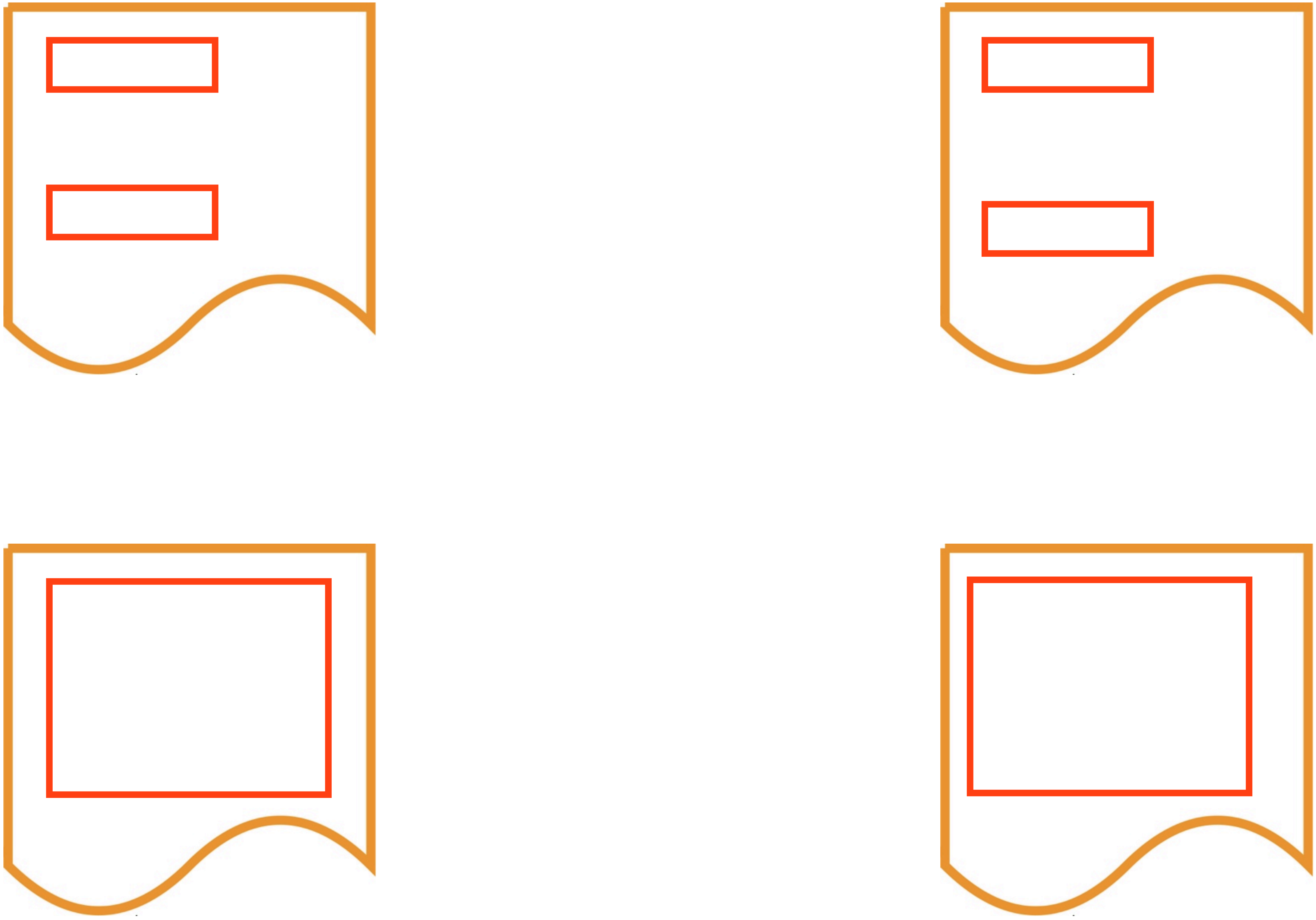
Plagiarism

Extensive transformations

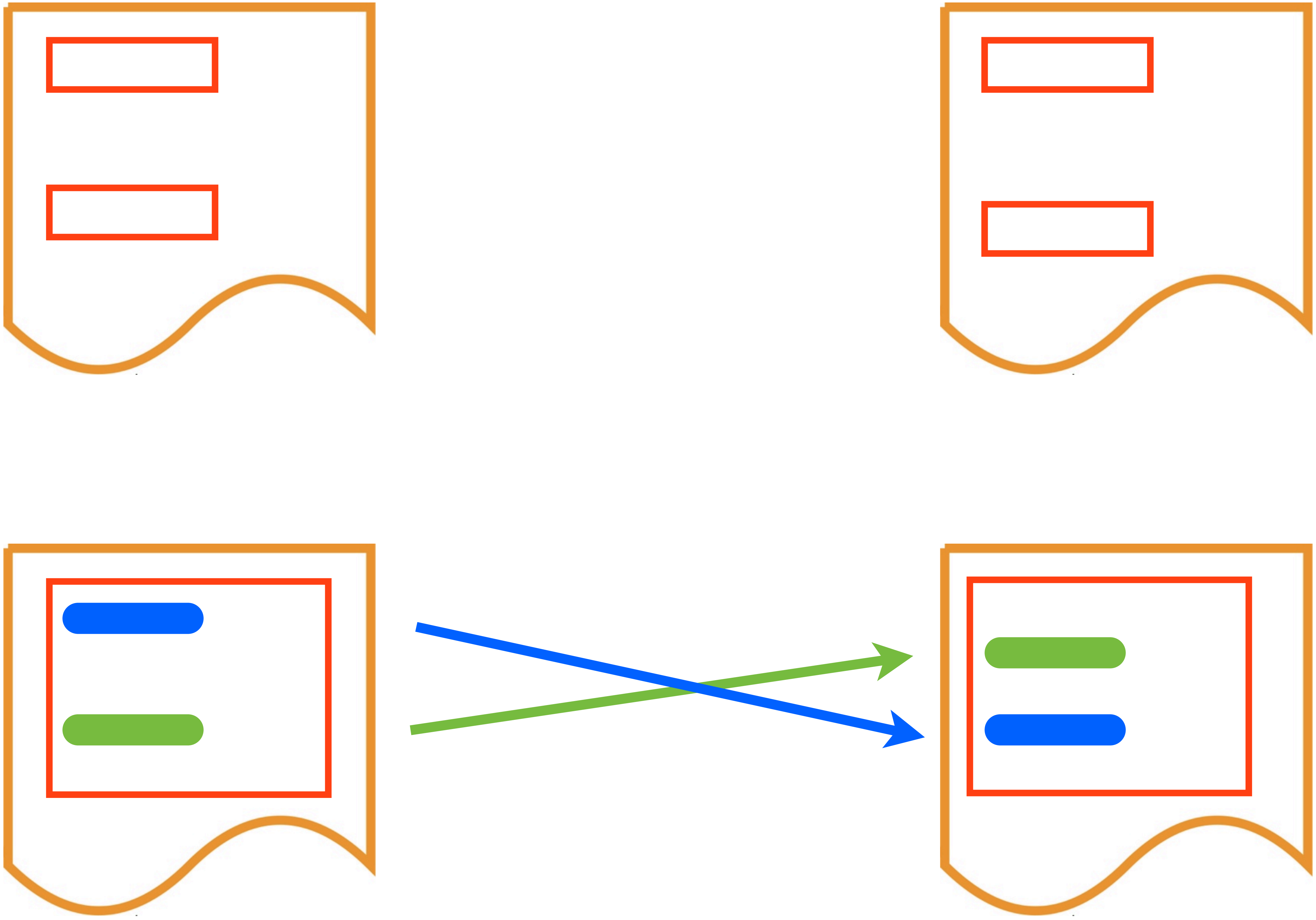
```
data Piece = Vide | Noir | Blanc | Extremite
        deriving (Eq, Show)

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
```

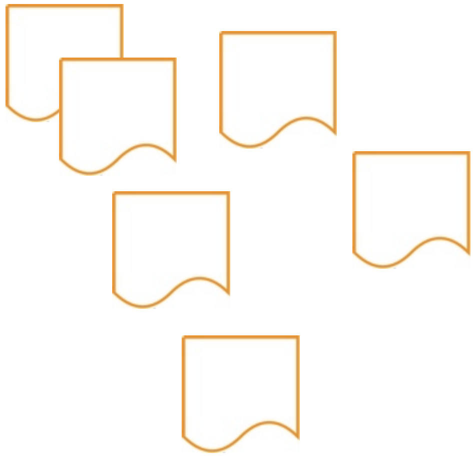
Larger clones



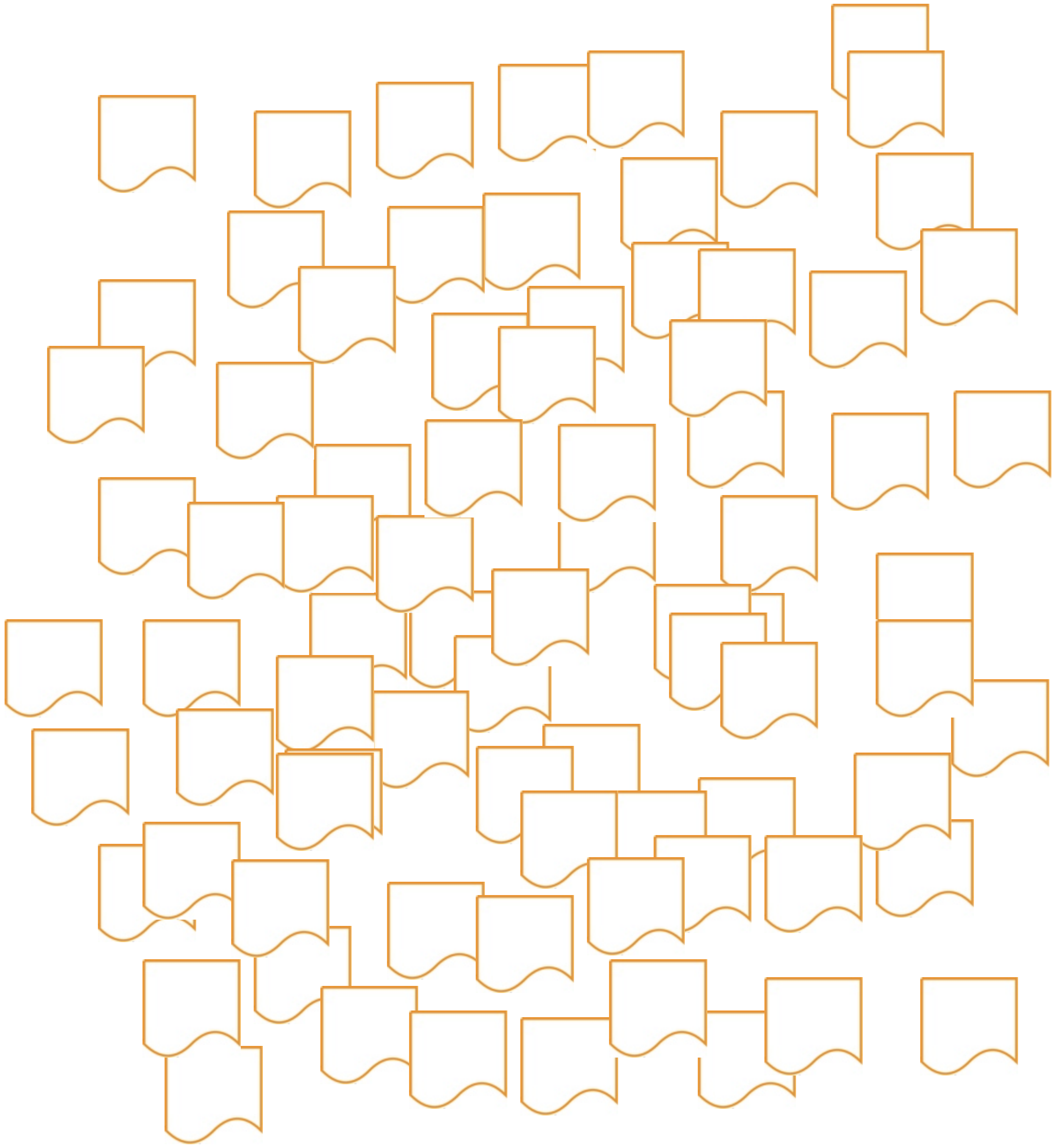
Larger clones



Less documents

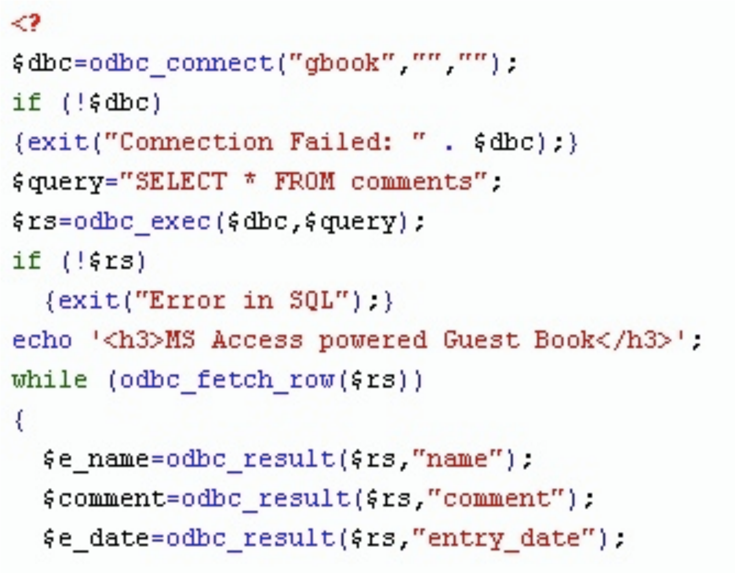


Assignment 4

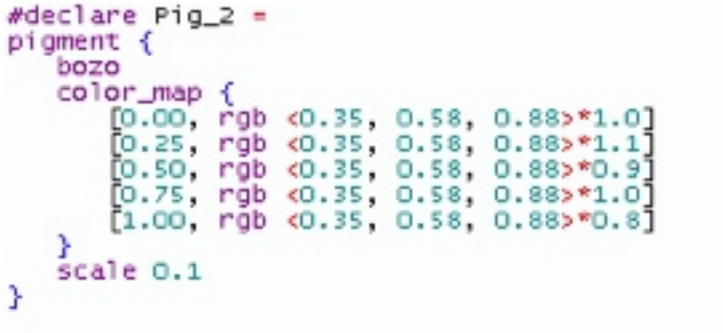


Linux

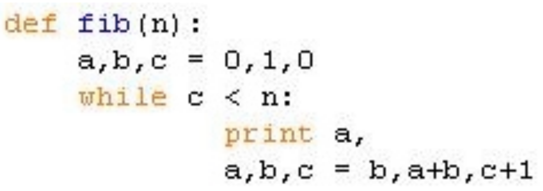
Several languages



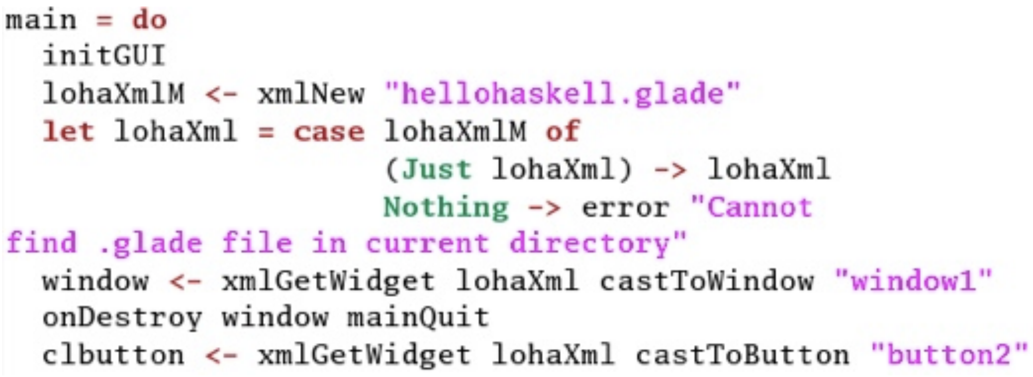
```
<?
$dbc=odbc_connect("gbook","","");
if (!$dbc)
{exit("Connection Failed: " . $dbc);}
$query="SELECT * FROM comments";
$rs=odbc_exec($dbc,$query);
if (!$rs)
  {exit("Error in SQL");}
echo '<h3>MS Access powered Guest Book</h3>';
while (odbc_fetch_row($rs))
{
  $e_name=odbc_result($rs,"name");
  $comment=odbc_result($rs,"comment");
  $e_date=odbc_result($rs,"entry_date");
```



```
#declare Pig_2 =
pigment {
    bozo
    color_map {
        [0.00, rgb <0.35, 0.58, 0.88>*1.0]
        [0.25, rgb <0.35, 0.58, 0.88>*1.1]
        [0.50, rgb <0.35, 0.58, 0.88>*0.9]
        [0.75, rgb <0.35, 0.58, 0.88>*1.0]
        [1.00, rgb <0.35, 0.58, 0.88>*0.8]
    }
    scale 0.1
}
```

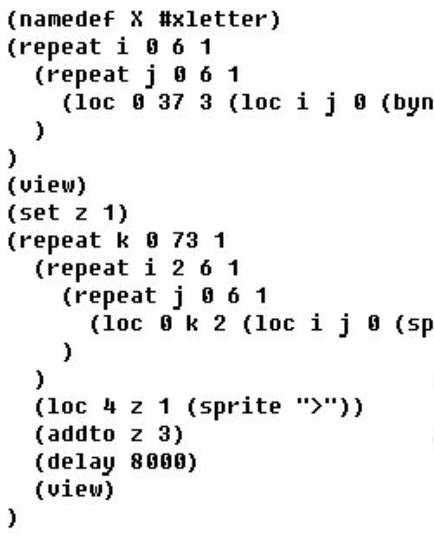


```
def fib(n):
    a,b,c = 0,1,0
    while c < n:
            print a,
            a,b,c = b,a+b,c+1
```

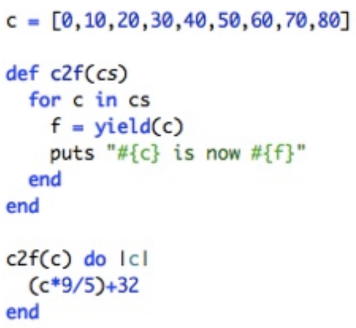


```
main = do
  initGUI
  lohaXmlM <- xmlNew "hellohaskell.glade"
  let lohaXml = case lohaXmlM of
                (Just lohaXml) -> lohaXml
                Nothing -> error "Cannot
find .glade file in current directory"
  window <- xmlGetWidget lohaXml castToWindow "window1"
  onDestroy window mainQuit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

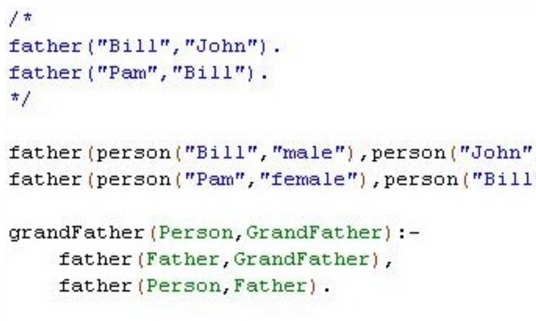
Bash
Prolog
Lisp
Scheme
Haskell
Smalltalk
C
C++
Python
PHP
Java
Perl
CLIPS
XML & XSLT
Ruby
Pov-ray
SQL
HTML & CSS
Javascript



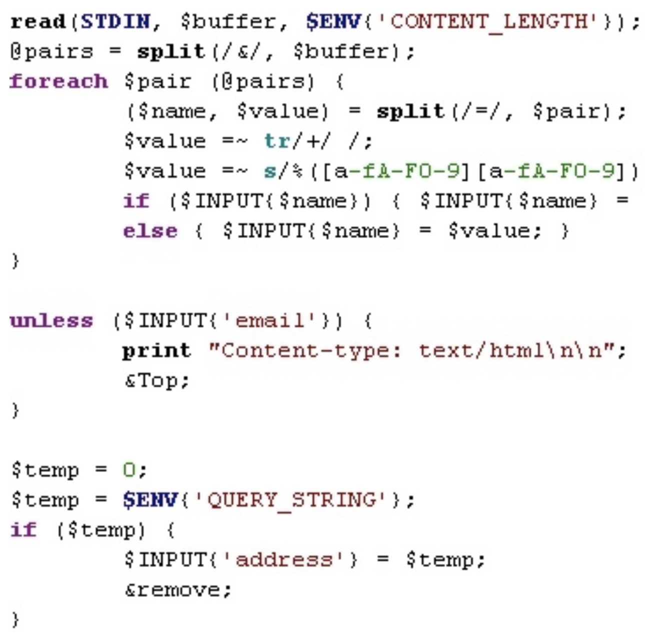
```
(namedef X #xletter)
(repeat i 0 6 1
  (repeat j 0 6 1
    (loc 0 37 3 (loc i j 0 (byname X))
  )
)
(view)
(set z 1)
(repeat k 0 73 1
  (repeat i 2 6 1
    (repeat j 0 6 1
      (loc 0 k 2 (loc i j 0 (sprite ".'
    )
  )
  (loc 4 z 1 (sprite ">"))
  (addto z 3)
  (delay 8000)
  (view)
)
```



```
c = [0,10,20,30,40,50,60,70,80]

def c2f(cs)
  for c in cs
    f = yield(c)
    puts "#{c} is now #{f}"
  end
end

c2f(c) do |c|
  (c*9/5)+32
end
```



```
/*
father("Bill","John").
father("Pam","Bill").
*/

father(person("Bill","male"),person("John","male")).
father(person("Pam","female"),person("Bill","male")).

grandFather(Person,GrandFather):-
    father(Father,GrandFather),
    father(Person,Father).
```

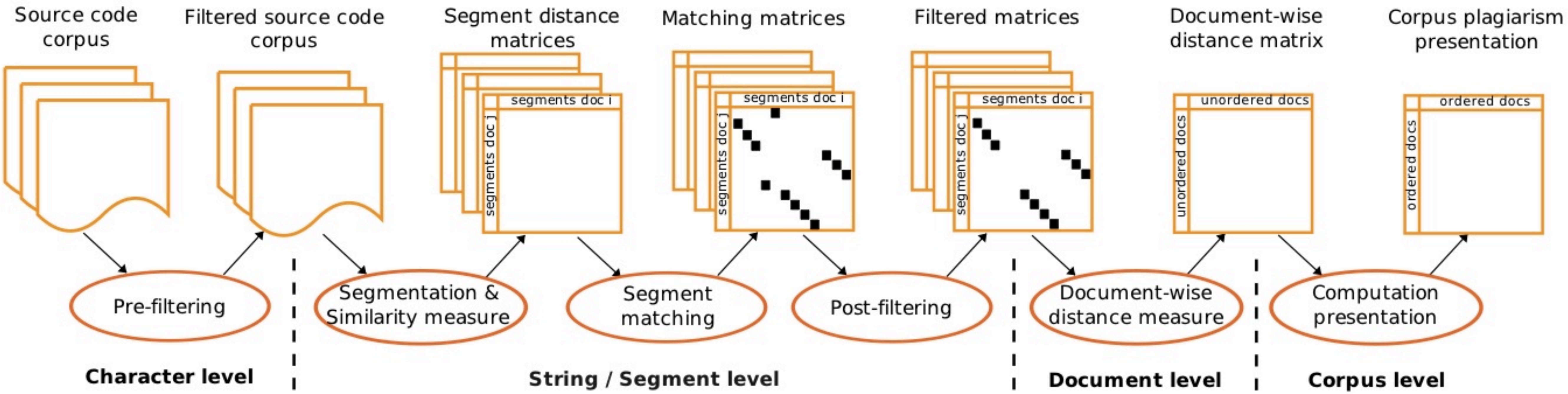


```
read(STDIN, $buffer, $ENV{'CONTENT_LENGTH'});
@pairs = split(/&/, $buffer);
foreach $pair (@pairs) {
        ($name, $value) = split(/=/, $pair);
        $value =~ tr/+/ /;
        $value =~ s/%([a-fA-F0-9][a-fA-F0-9])
        if ($INPUT{$name}) { $INPUT{$name} =
        else { $INPUT{$name} = $value; }
}

unless ($INPUT{'email'}) {
        print "Content-type: text/html\n\n";
        &Top;
}

$temp = 0;
$temp = $ENV{'QUERY_STRING'};
if ($temp) {
        $INPUT{'address'} = $temp;
        &remove;
}
```

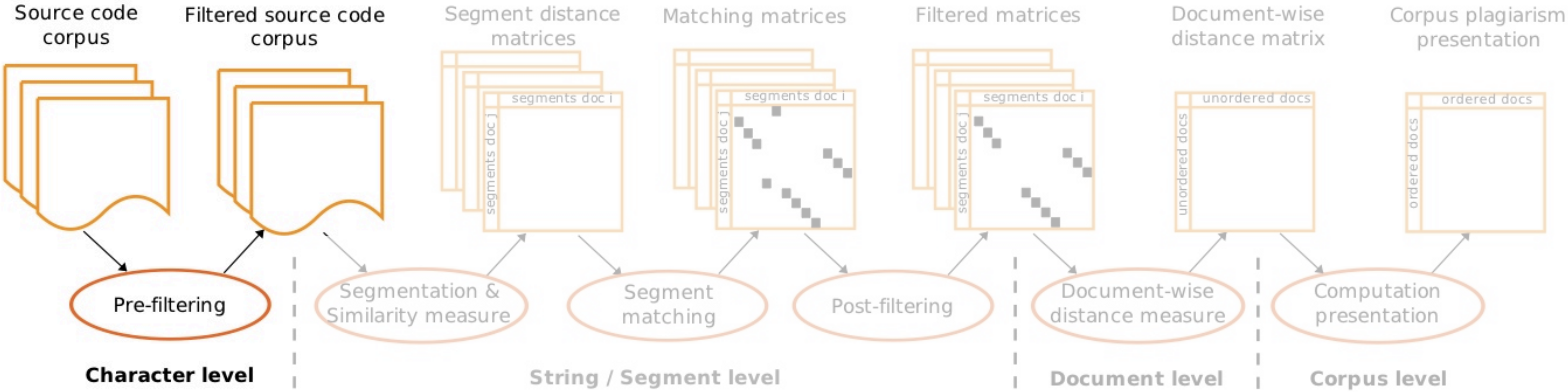
We want to catch everyone

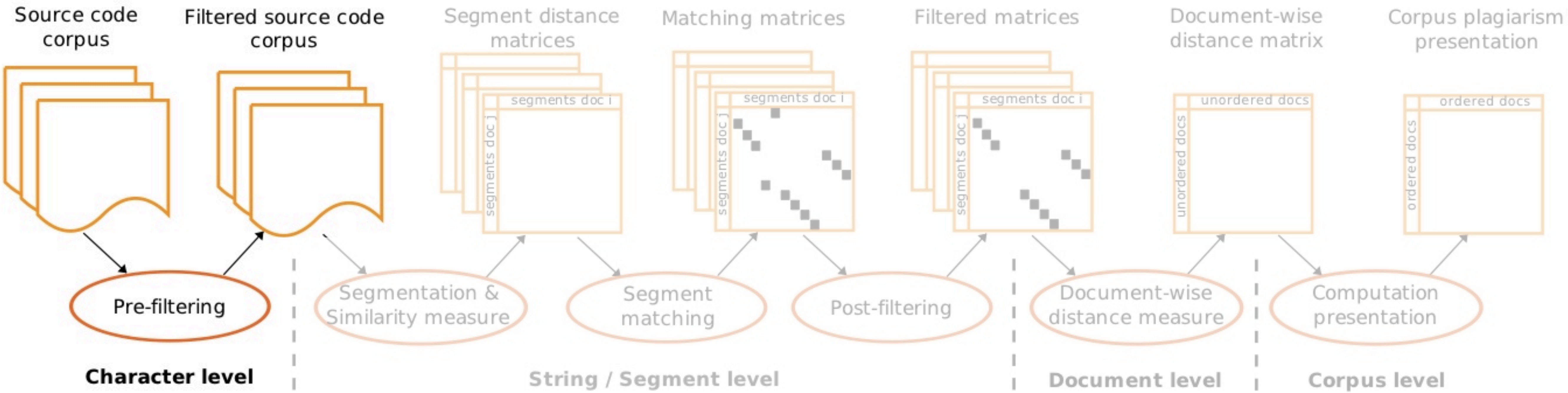
Differences and consequences

1. Extensive transformations	Extensive normalization
2. Larger clones with reordering	Line matching algorithm
3. Less documents	Less performance need
4. Several languages	Language independent
5. We want to catch everyone	Recall > precision

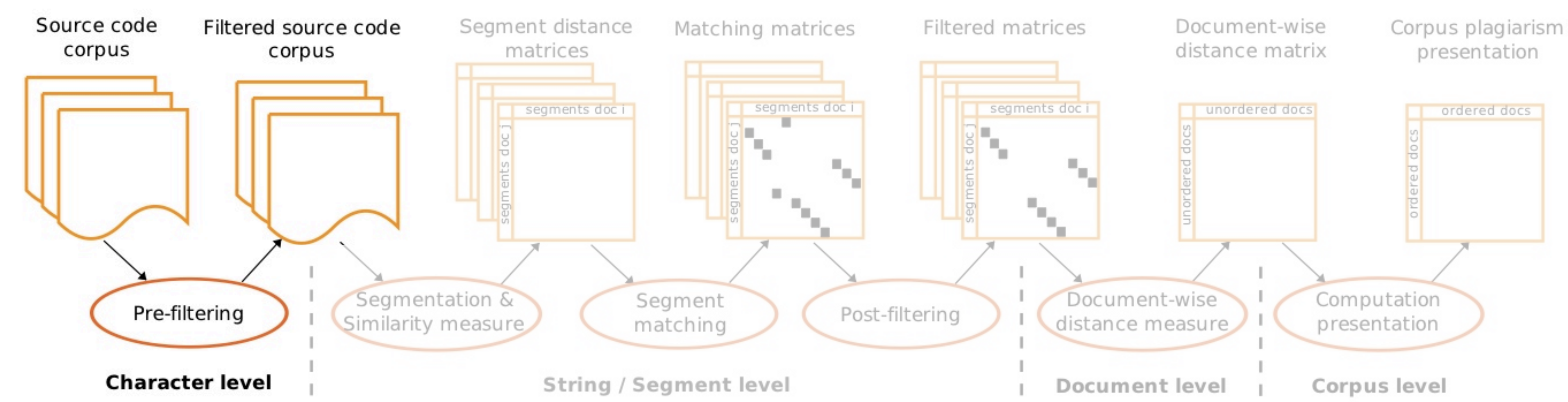


Introducing ... the "Pomp-o-mètre"


Source code corpus
Filtered source code corpus
Segment distance matrices
Matching matrices
Filtered matrices
Document-wise distance matrix
Corpus plagiarism presentation
segments doc i
segments doc j
segments doc i
segments doc j
segments doc i
segments doc j
unordered docs
unordered docs
ordered docs
ordered docs
Pre-filtering
Segmentation & Similarity measure
Segment matching
Post-filtering
Document-wise distance measure
Computation presentation
Character level
String / Segment level
Document level
Corpus level


Source code corpus
Filtered source code corpus
Segment distance matrices
Matching matrices
Filtered matrices
Document-wise distance matrix
Corpus plagiarism presentation
segments doc i
segments doc j
unordered docs
ordered docs
Pre-filtering
Segmentation & Similarity measure
Segment matching
Post-filtering
Document-wise distance measure
Computation presentation
Character level
String / Segment level
Document level
Corpus level

[A-Za-z0-9]+ → ’t’



Source code corpus

Filtered source code corpus

Segment distance matrices

Matching matrices

Filtered matrices

Document-wise distance matrix

Corpus plagiarism presentation

Pre-filtering

Segmentation & Similarity measure

Segment matching

Post-filtering

Document-wise distance measure

Computation presentation

Character level

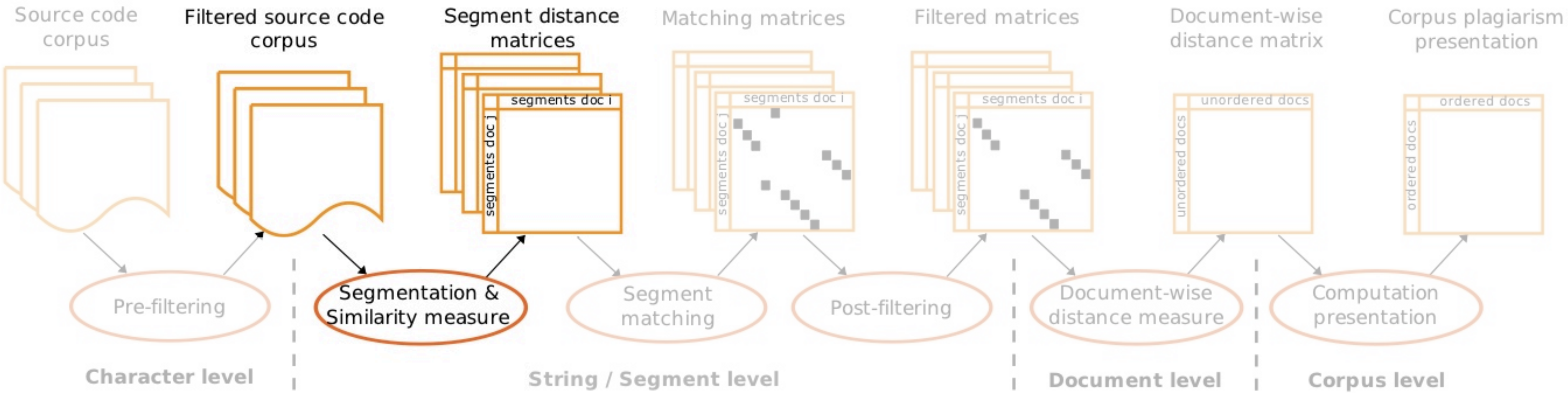
String / Segment level

Document level

Corpus level

[A-Za-z0-9]+ → 't'

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

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

taillePlateau :: Int                              t :: t
taillePlateau = 8                                 t = t
```



Source code corpus

Filtered source code corpus

Segment distance matrices

Matching matrices

Filtered matrices

Document-wise distance matrix

Corpus plagiarism presentation

Pre-filtering

Segmentation & Similarity measure

Segment matching

Post-filtering

Document-wise distance measure

Computation presentation

Character level

String / Segment level

Document level

Corpus level

```
t t = t | t | t | t
```

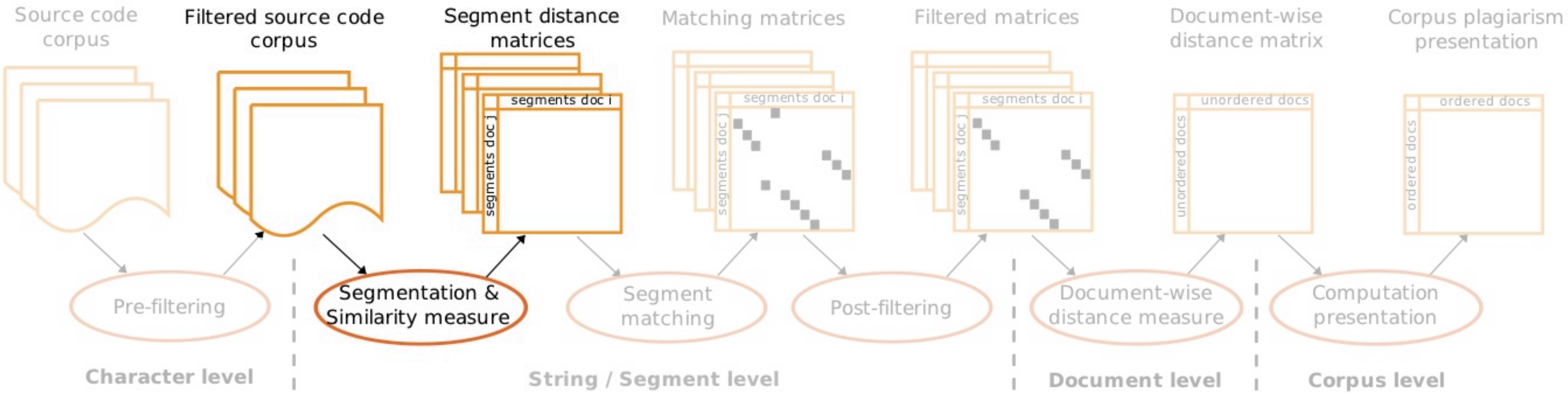
```
    t (t, t)
```

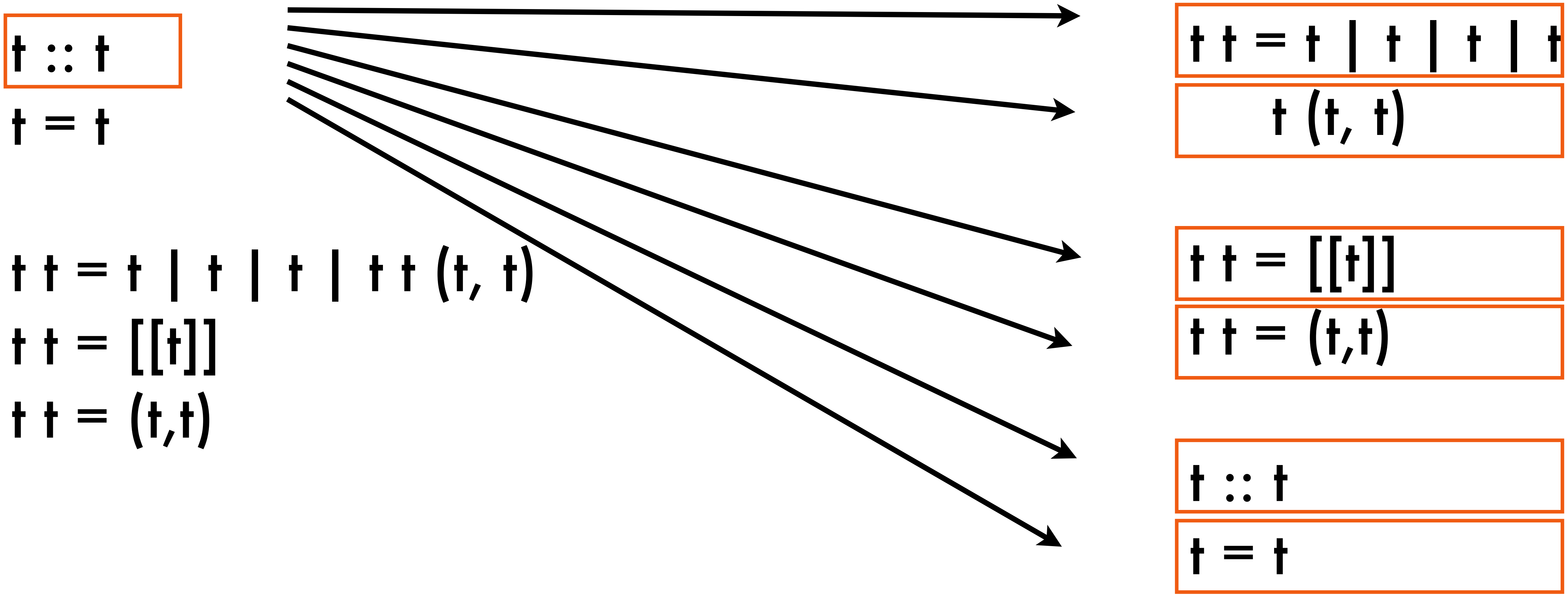
```
t t = [[t]]
```

```
t t = (t,t)
```

```
t :: t
```

```
t = t
```





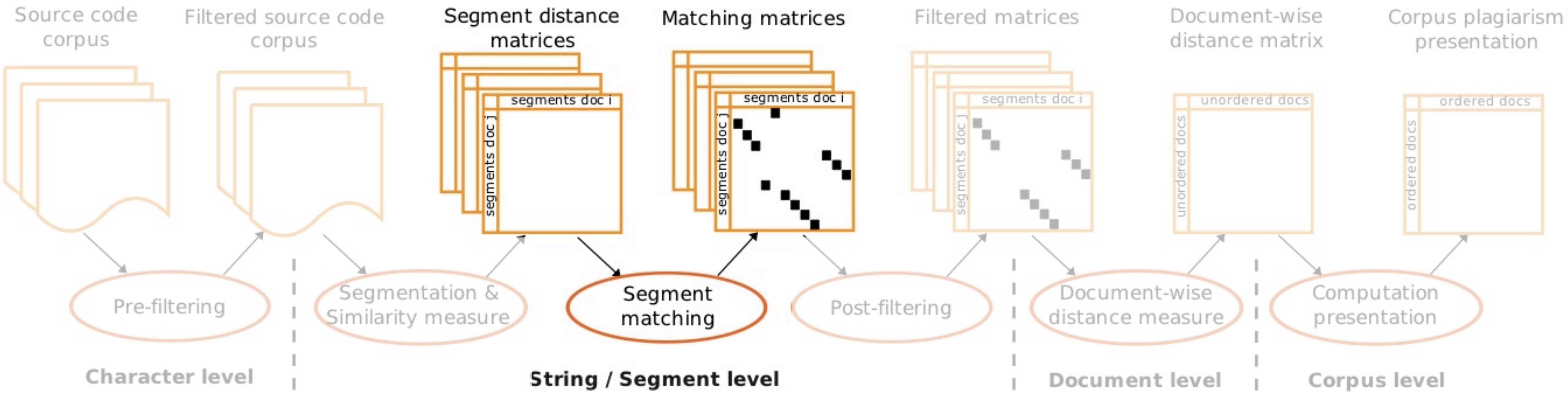
t :: t
t = t

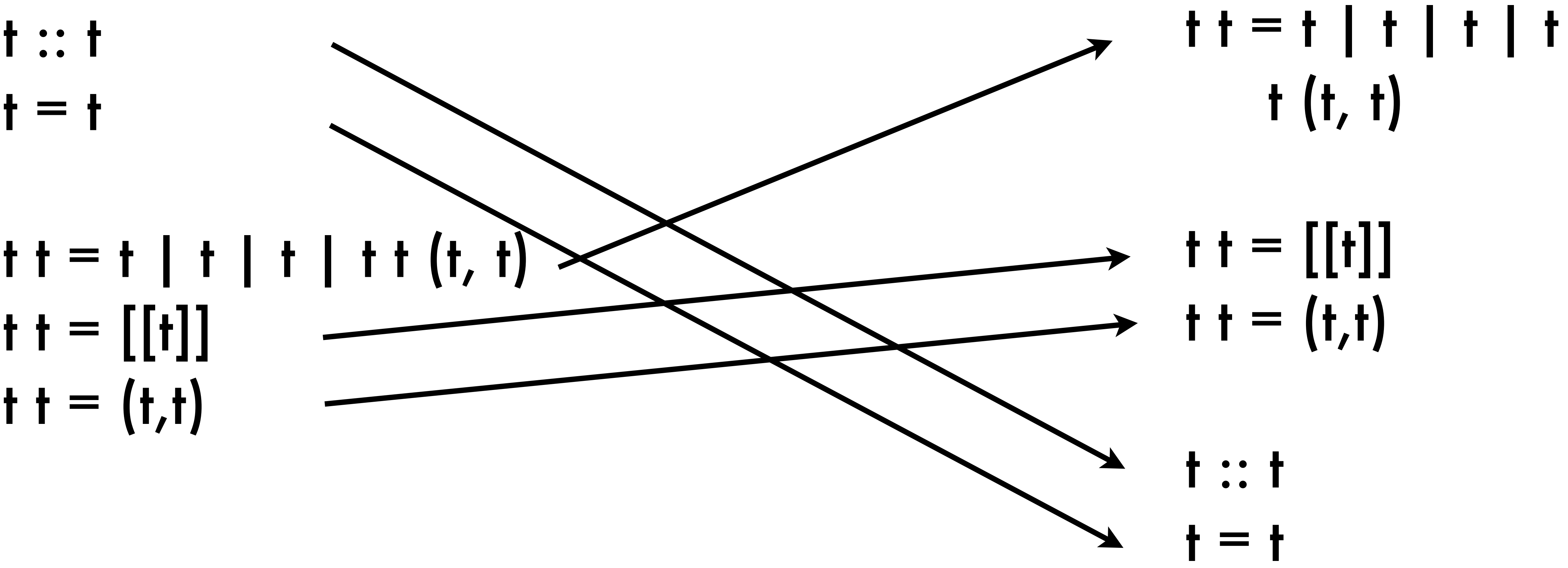
t t = t | t | t | t t (t, t)
t t = [[t]]
t t = (t,t)

t t = t | t | t | t
t (t, t)

t t = [[t]]
t t = (t,t)

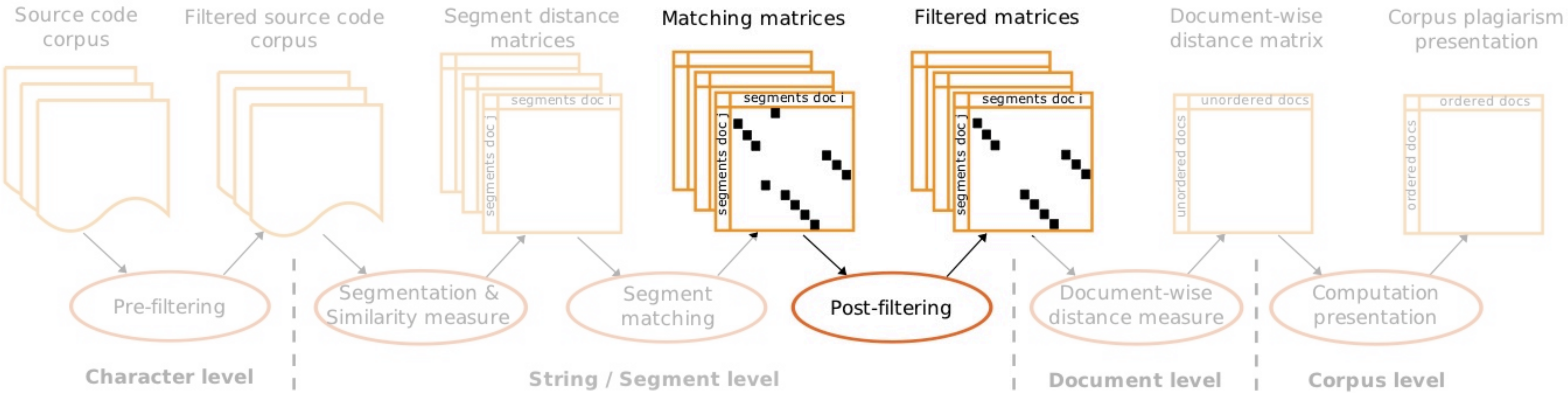
t :: t
t = t





```
t :: t
t = t

t t = t | t | t | t t (t, t)
t t = [[t]]
t t = (t,t)
```

```
t t = t | t | t | t
  t (t, t)

t t = [[t]]
t t = (t,t)

t :: t
t = t
```



Source code corpus

Filtered source code corpus

Segment distance matrices

Matching matrices

Filtered matrices

Document-wise distance matrix

Corpus plagiarism presentation

segments doc i

segments doc j

unordered docs

ordered docs

Pre-filtering

Segmentation & Similarity measure

Segment matching

Post-filtering

Document-wise distance measure

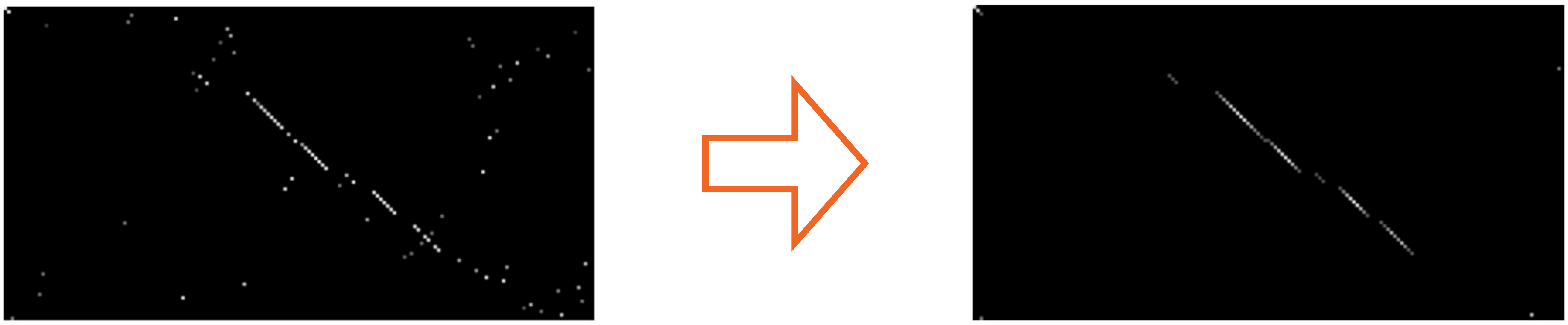
Computation presentation

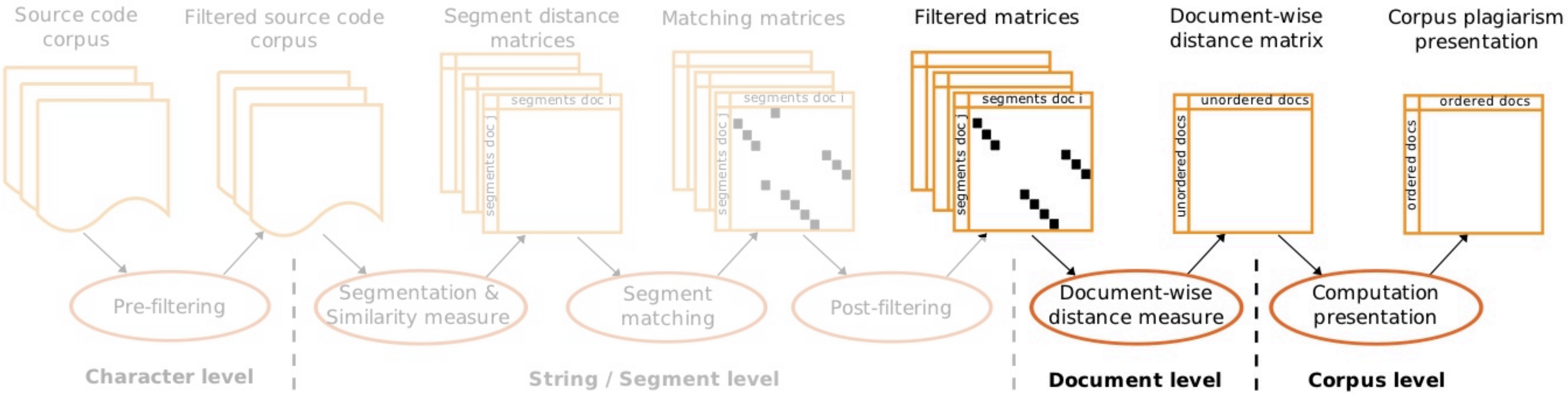
Character level

String / Segment level

Document level

Corpus level





Source code corpus

Filtered source code corpus

Segment distance matrices

Matching matrices

Filtered matrices

Document-wise distance matrix

Corpus plagiarism presentation

Pre-filtering

Segmentation & Similarity measure

Segment matching

Post-filtering

Document-wise distance measure

Computation presentation

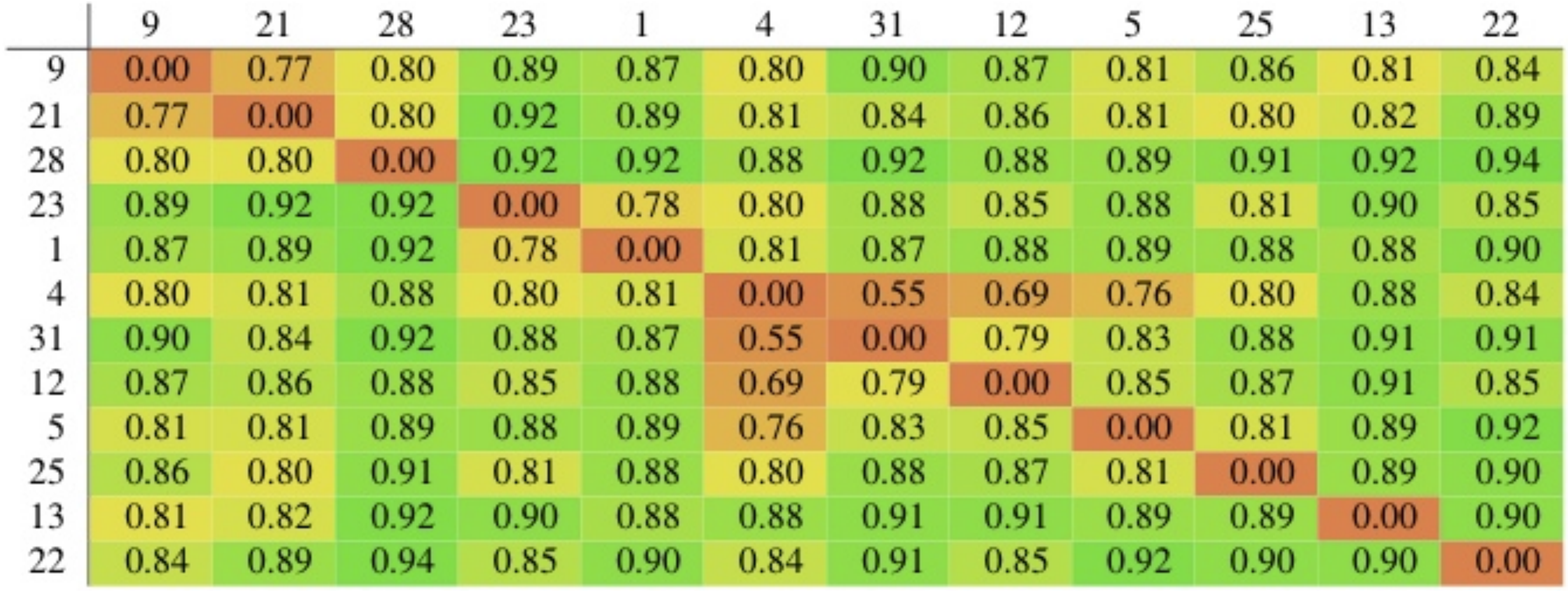
Character level

String / Segment level

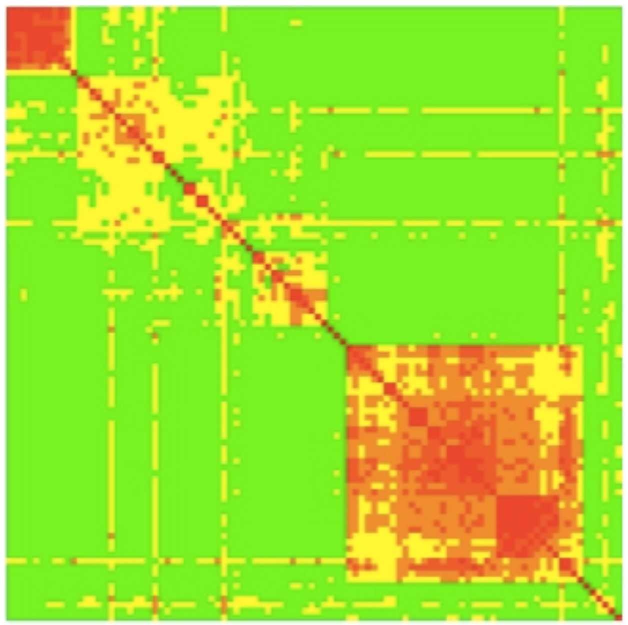
Document level

Corpus level

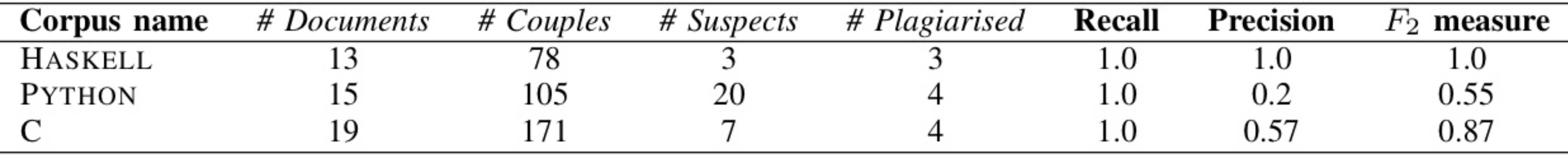
segments doc i
segments doc j
unordered docs
unordered docs
ordered docs
ordered docs



	9	21	28	23	1	4	31	12	5	25	13	22
9	0.00	0.77	0.80	0.89	0.87	0.80	0.90	0.87	0.81	0.86	0.81	0.84
21	0.77	0.00	0.80	0.92	0.89	0.81	0.84	0.86	0.81	0.80	0.82	0.89
28	0.80	0.80	0.00	0.92	0.92	0.88	0.92	0.88	0.89	0.91	0.92	0.94
23	0.89	0.92	0.92	0.00	0.78	0.80	0.88	0.85	0.88	0.81	0.90	0.85
1	0.87	0.89	0.92	0.78	0.00	0.81	0.87	0.88	0.89	0.88	0.88	0.90
4	0.80	0.81	0.88	0.80	0.81	0.00	0.55	0.69	0.76	0.80	0.88	0.84
31	0.90	0.84	0.92	0.88	0.87	0.55	0.00	0.79	0.83	0.88	0.91	0.91
12	0.87	0.86	0.88	0.85	0.88	0.69	0.79	0.00	0.85	0.87	0.91	0.85
5	0.81	0.81	0.89	0.88	0.89	0.76	0.83	0.85	0.00	0.81	0.89	0.92
25	0.86	0.80	0.91	0.81	0.88	0.80	0.88	0.87	0.81	0.00	0.89	0.90
13	0.81	0.82	0.92	0.90	0.88	0.88	0.91	0.91	0.89	0.89	0.00	0.90
22	0.84	0.89	0.94	0.85	0.90	0.84	0.91	0.85	0.92	0.90	0.90	0.00



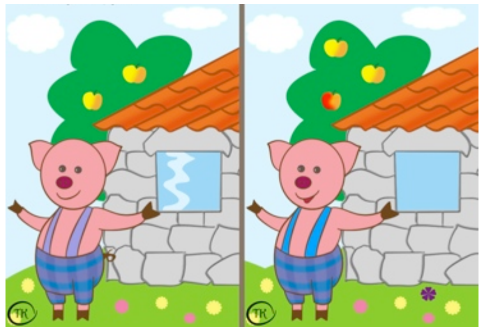
Empirical validation on 3 corpuses



Corpus name	*# Documents*	*# Couples*	*# Suspects*	*# Plagiarised*	**Recall**	**Precision**	F_2 **measure**
Haskell	13	78	3	3	1.0	1.0	1.0
Python	15	105	20	4	1.0	0.2	0.55
C	19	171	7	4	1.0	0.57	0.87

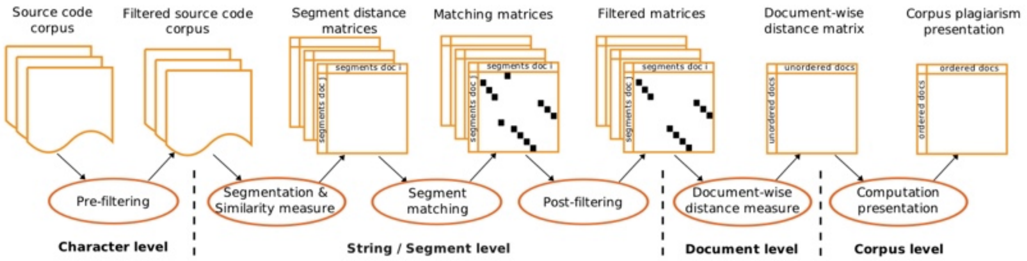
We consider that we detect plagiarism when the distance between a pair of documents is less than the mean distances of the matrix

1.



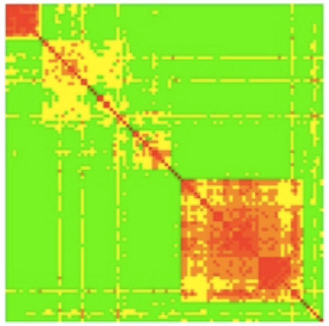
clone and plagiarism detection are similar, but distinct problems

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]+ → ’t’

Related work

Ducasse et.al.
Wettel & Marinescu

Baldr
Anti-Copias