Автоматическое распознавание плагиата в домашних работах по программированию

Марина Мазурова, 2 курс

Аннотация

Несмотря на обилие программ для распознавания плагиата в текстах как на естественном, так и на искусственном языке, существует весьма ограниченное количество программ, способных обрабатывать небольшие фрагменты программного кода. Тем не менее, потребность в таких программах существует: например, программы такого рода могут использоваться для поиска плагиата в студенческих домашних работах по программированию. Настоящая работа посвящена описанию программы, распознающей плагиат в домашних работах студентов-лингвистов, начавших изучать язык программирования Python 2.

Как скрывают плагиат?

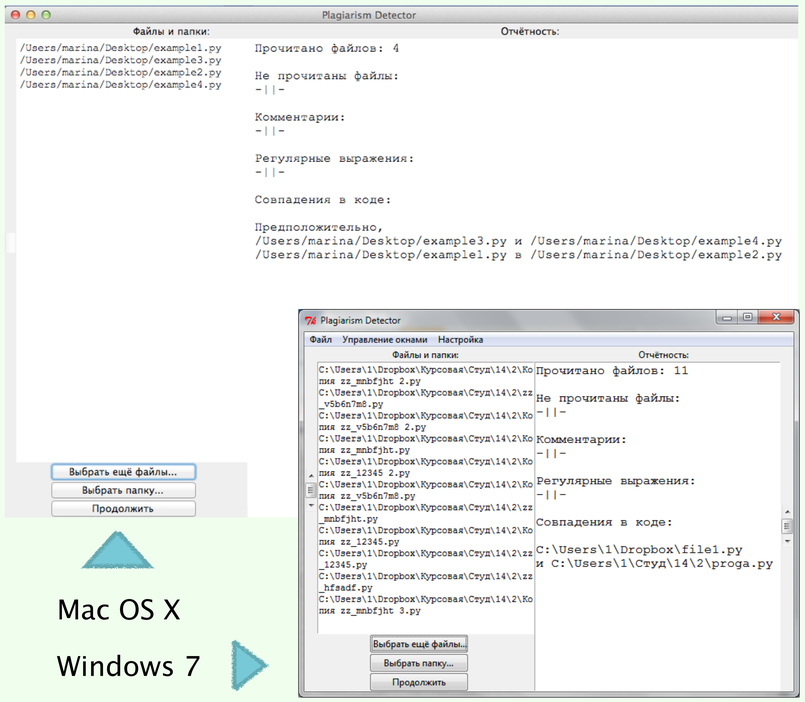
В исходном коде:

- Переименовывают переменные
- Меняют местами строки, функции, условия в условных операторах
- Добавляют различия в оформлении — лишние пробелы, другие кавычки и т.п.
- Комбинируют указанные выше способы

В регулярных выражениях:

- Никак
- Совершают простые синтаксические преобразования
- Меняют смысловое содержание, оставляя старую структуру

Интерфейс



Mac OS X

Windows 7

Литература

L. Moussiades, A. Vakali. *PDetect: A Clustering Approach for Detecting Plagiarism in Source Code Datasets* // *The Computer Journal*. Vol. 48, No. 6. Oxford: Oxford University Press, 2005. P. 652-661.

C. Roy, J. Cordy, R. Koschke. *Comparison and Evaluation of Code Clone Detection Techniques and Tools: A Qualitative Approach* // *Science of Computer Programming*. Vol.74, Issue 7. Amsterdam: Elsevier North-Holland, Inc., 2009. P. 470-495.

S. Schleimer, D. S. Wilkerson, A. Aiken. *Winnowing: local algorithms for document fingerprinting* // *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*. ACM, 2003. P. 76-85.

Как это работает?

```
slova=[]
slovo=raw_input ('Input a word:')
while slovo!='':
    slova.append (slovo)
    slovo =raw_input ('Input a word:')
def count (x):
    b=0
    vow= 'aeyuio'
    for y in x:
        if y in vow: break
        b+=1
    return b
for i in range(len (slova)):
  for j in range (1,len (slova)-i):
    if count(slova[j])<count(slova[j-1]):
            c=slova[j-1]
            slova[j-1]=slova[j]
            slova[j]=c
for t in slova: print t
```

```
def vowels(word): # this func counts
#vowels in a word
    v = u"aeyuio"
    counter = 0
    for letter in word:
        if letter in v:
            break
        counter += 1
    return counter

word = raw_input(u"Введите слово: ")
a = [] # array for words inserted
while word != "":
    a.append(word)
    word = raw_input(u"Введите слово: ")
for i in range(len(a)):
    for j in range(1, len(a) - i):
        if vowels(a[j-1]) > vowels(a[j]):
            temp = a[j - 1]
            a[j - 1] = a[j]
            a[j] = temp
for i in a:
    print i
```

Получаем файлы с исходным кодом

```
def FUNC(V1):
    V2 = 0
    V4 = LINE
    for V3 in V1:
        if V3 in V4:
            break
        V2 += 1
    return V2

V1 = []
V2 = raw_input(LINE)
while V2 != LINE:
    V1.append(V2)
    V2 = raw_input(LINE)
for V3 in range(len(V1)):
  for V4 in range (1, len(V1) - V3):
    if FUNC(V1[V4 - 1]) > FUNC(V1[V4]):
            V5 = V1[V4 - 1]
            V1[V4 - 1] = V1[V4]
            V1[V4] = V5
for V3 in V1:
    print V3
```

```
def FUNC(V1):
    V4 = LINE
    V2 = 0
    for V3 in V1:
        if V3 in V4:
            break
        V2 += 1
    return V2

V1 = raw_input(LINE)
V2 = []
while V1 != LINE:
    V2.append(V1)
    V1 = raw_input(LINE)
for V3 in range(len(V2)):
    for V4 in range(1, len(V2) - V3):
        if FUNC(V2[V4-1]) > FUNC(V2[V4]):
            V5 = V2[V4 - 1]
            V2[V4 - 1] = V2[V4]
            V2[V4] = V5
for V3 in V2:
    print V3
```

Сравниваем части кода программ

```
V1 = []
V2 = raw_input(LINE)
while V2 != LINE:
    V1.append(V2)
    V2 = raw_input(LINE)
def vow(V1):
    V4 = LINE
    V2 = 0
    for V3 in V1:
        if V3 in V4:
            break
        V2 += 1
    return V2
def count(V1):
    V2 = 0
    V4 = LINE
    for V3 in V1:
        if V3 in V4: break
        V2 += 1
    return V2
for V3 in range(len(V1)):
  for V4 in range (1, len(V1) - V3):
    if count(V1[V4]) < count(V1[V4 - 1]):
            V5 = V1[V4 - 1]
            V1[V4 - 1] = V1[V4]
            V1[V4] = V5
for V3 in V1: print V3
V1 = raw_input(LINE)
V2 = []
while V1 != LINE:
    V2.append(V1)
    V1 = raw_input(LINE)
for V3 in range(len(V2)):
    for V4 in range(1, len(V2) - V3):
        if vow(V2[V4-1]) > vow(V2[V4]):
            V5 = V2[V4 - 1]
            V2[V4 - 1] = V2[V4]
            V2[V4] = V5
for V3 in V2:
    print V3
```

Разделяем код на смысловые части, удаляем незначащие фрагменты — комментарии, пустые строки и т.п.

```
V1 = []
V2 = raw_input(LINE)
while V2 != LINE:
    V1.append(V2)
    V2 = raw_input(LINE)
for V3 in range(len(V1)):
  for V4 in range (1, len(V1) - V3):
    if FUNC(V1[V4 - 1]) > FUNC(V1[V4]):
            V5 = V1[V4 - 1]
            V1[V4 - 1] = V1[V4]
            V1[V4] = V5
for V3 in V1:
    print V3
def FUNC(V1):
    V2 = 0
    V4 = LINE
    for V3 in V1:
        if V3 in V4:
            break
        V2 += 1
    return V2
```

```
def FUNC(V1):
    V4 = LINE
    V2 = 0
    for V3 in V1:
        if V3 in V4:
            break
        V2 += 1
    return V2
V1 = raw_input(LINE)
V2 = []
while V1 != LINE:
    V2.append(V1)
    V1 = raw_input(LINE)
for V3 in range(len(V2)):
    for V4 in range(1, len(V2) - V3):
        if FUNC(V2[V4 - 1]) > FUNC(V2[V4]):
            V5 = V2[V4 - 1]
            V2[V4 - 1] = V2[V4]
            V2[V4] = V5
for V3 in V2:
    print V3
```

Устраняем прочие различия в оформлении

Регулярные выражения

```
(не)?регулярн((ы|о)(й|е)?|ая?|42)
(TEXT)?TEXT((L|L)(L|L)?|TEXT?|NUMS)
(без)?грамот((ы|о)(й|е)?|ая?|42)
```

Все лексические части замещаются метками, а синтаксические остаются нетронутыми

Перспективы

- Переработка системы подсчетов результатов, введение ранжирования случаев частичного плагиата по релевантности
- Устранение уязвимостей, позволяющих скрыть факт списывания от программы
- Добавление возможности работы с программами, использующими функции из любых подключаемых модулей — введение автоматической генерации контекстов
- Создание версий для обработки программ на Python 3 и других языках программирования

2013г.
Научный руководитель — Т.А. Архангельский