LOG3430 - MÉTHODES DE TEST ET DE VALIDATION DU LOGICIEL

LABORATOIRE 1

TESTS UNITAIRES

Département de génie informatique et de génie logiciel
École Polytechnique de Montréal

Amir Cherkaoui Eddeqaqi - 1980843
Mike Folepe - 1894509
Etienne Chevalier Tittley - 2021242

Groupe 2

Hiver 2022

RAPPORT DU TP1:

Partie 1:

- Couverture de crud.py avec les fonctions du fichier test_crud.py :

Name	Stmts	Miss	Cover
/opt/homebrew/lib/python3.9/site-packages/_distutils_hack/__init__.py	96	91	5%
crud.py	226	72	68%
test_crud.py	240	0	100%

- Couverture de crud.py après avoir ajouté des tests supplémentaires :

Name	Stmts	Miss	Cover
/opt/homebrew/lib/python3.9/site-packages/_distutils_hack/__init__.py	96	91	5%
crud.py	226	43	81%
test_crud.py	300	0	100%

Les tests que j'ai ajouté pour améliorer la couverture sont (par ordre d'apparition):
-Un test qui verifie que la fonction get_new_user_id() le bon id (selon la maniere dont les ids sont attribués dans le code).
-Même chose pour l'id de groupe.
-Un test qui verifie si convert_to_unix() convertit correctement en unix timestamp.
-Un test qui verifie que la fonction add_new_user() retourne False si l'email donné n'est pas ecrit correctement.
-Un test qui verifie que la fonction add_new_group() retourne False si un ou plusieurs membres d'un groupe n'existent pas.
-Un test qui verifie que la fonction update_users() retourne False si les parametres donnés ne correspondent pas.
-Un test qui verifie que la fonction update_users() retourne False si la date de dernier message est plus recente que la date de dernier message passé.
-Un test qui verifie que la fonction update_users() retourne False si la date de premier message est plus vieille que la date du dernier message passé.
-Un test qui verifie que la fonction update_users() retourne False si la valeur de SpanM donnée est inferieure à 0.
-Un test qui verifie que la fonction update_users() retourne False si la valeur de Trust donnée est superieure a 100.
-Un test qui verifie que la fonction update_users() retourne False si un groupe donné n'existe pas.
-Un test qui verifie que la fonction update_users() retourne False si la taille de la liste des membres n'est pas correcte.

- Couverture de email_analyzer.py avec les fonctions du fichier test_email_analyzer.py :

```
PS C:\Users\Étienne\Desktop\LOG3430_TP1\LOG3430\TP1> coverage report
Name                      Stmts   Miss  Cover
---------------------------------------------
email_analyzer.py            46      4    91%
test_email_analyzer.py       49      0   100%
text_cleaner.py              29     18    38%
---------------------------------------------
TOTAL                       124     22    82%
```

- Couverture de vocabulary_creator.py avec les fonctions de tests de base :

```
Name                              Stmts   Miss  Cover
-----------------------------------------------------
test_vocabulary_creator.py           32      1    97%
text_cleaner.py                      29     18    38%
vocabulary_creator.py                82     21    74%
-----------------------------------------------------
TOTAL                               143     40    72%
```

- Couverture de vocabulary_creator.py après avoir ajouté des tests supplémentaires :

```
PS C:\Users\Étienne\Desktop\LOG3430_TP1\LOG3430\TP1> coverage report
Name                           Stmts   Miss  Cover
------------------------------------------------
test_vocabulary_creator.py        53      0   100%
text_cleaner.py                   29     18    38%
vocabulary_creator.py             82      6    93%
------------------------------------------------
TOTAL                            164     24    85%
```

Les tests ajoutés pour améliorer la couverture sont:

- Le test test_create_vocab_ham_Returns_vocabulary_with_correct_values qui permet de verifier que le bon dictionnaire avec les bonnes probabilités soit généré par la fonction create_vocab pour les courriels considérés bons

- Le test test_write_data_to_vocab_file_successfully_write_vocabulary_with_correct_valuesqui permet de verifier si la fonction write_data_to_vocab_file écrit le bon dictionnaire généré par la fonction create_vocab dans le fichier vocabulary.json

Partie 2:

```python
def utilisater_trust_calcul(self, user_id):
    '''
        Description: fonction pour calculer le niveau de confiance (Trust) de l'utilisateur.
        Sortie: retourne le niveau de confiance, seulement si celui si est entre
                0 et 100. Dans le cas contraire, retourne false
    '''

    time_first_seen_message = self.crud.get_user_data(user_id, "Date_of_first_seen_message")
    time_last_seen_message = self.crud.get_user_data(user_id, "Date_of_last_seen_message")
    NHam = self.crud.get_user_data(user_id, "HamN")
    NSpam = self.crud.get_user_data(user_id, "SpamN")
    groups = self.crud.get_user_data(user_id, "Groups")
    trust2 = 0
    nb_groups = 0

    # Calcul trust1
    trust1 = time_last_seen_message * NHam / time_first_seen_message * (NHam + NSpam)

    # Calcul trust2
    for group in groups:
        trust2 += self.crud.get_groups_data(self.crud.get_group_id(group), "Trust")
        nb_groups += 1
    trust2 = trust2 / nb_groups

    # Calcul trust
    if trust2 < 60:
        trust = trust2
    elif trust1 > 100:
        trust = 100
    else:
        trust = 0.6 * trust1 + 0.4 * trust2 / 2      Amir Cherkaoui, 6 days ago • Rectification

    #verification que trust est bien entre 0 et 100
    if 0 <= trust <= 100:
        return trust
    #return False si trust est en dehors de l'intervalle [0,100]
    return False
```

CFG:

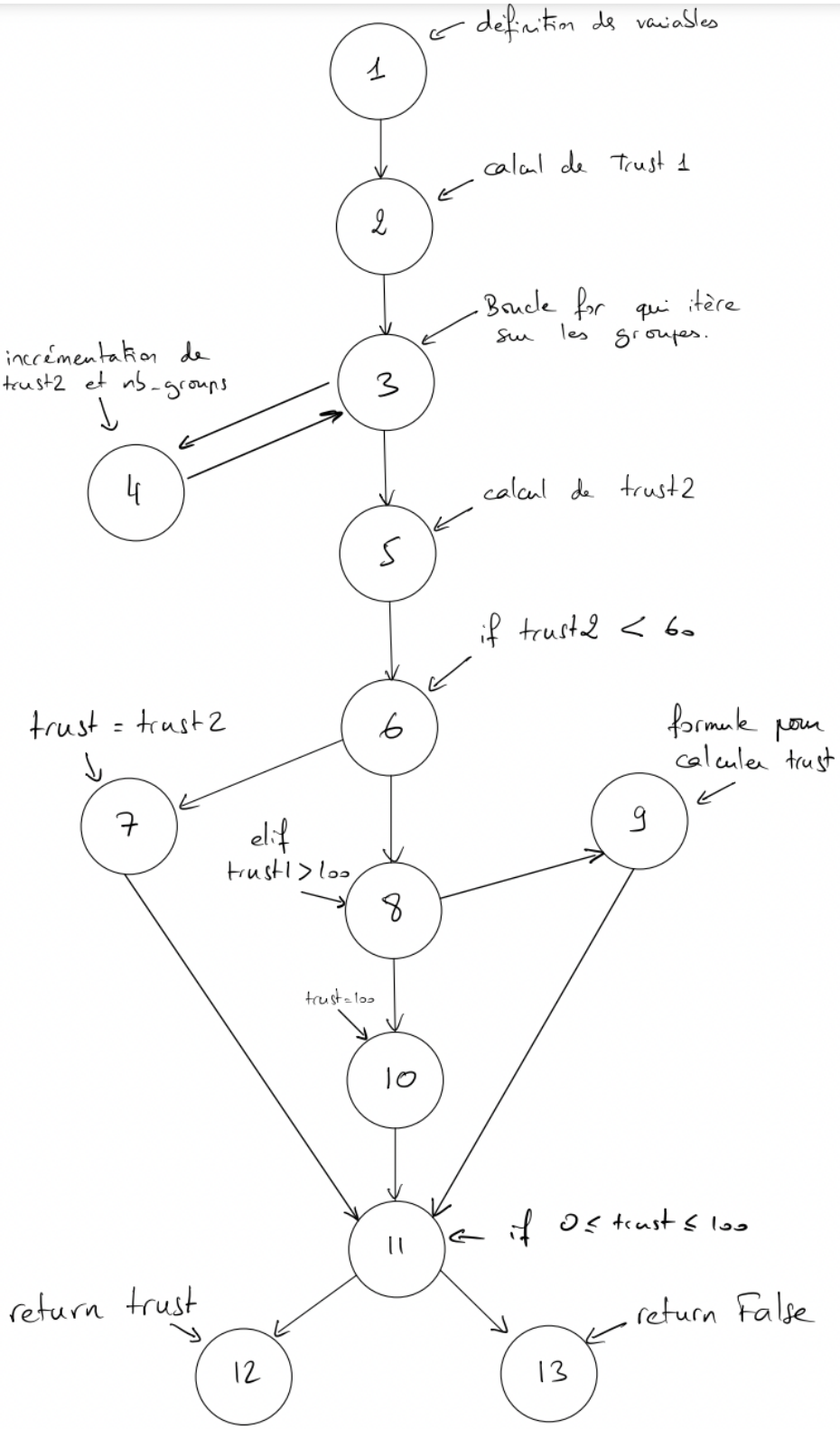
définition de variables
1
calcul de Trust 1
2
Boucle for qui itère sur les groupes.
incrémentation de trust2 et nb_groups
3
4
calcul de trust2
5
if trust2 < 60
6
trust = trust2
7
formule pour calculer trust
9
elif trust1 > 100
8
trust=100
10
if 0 ≤ trust ≤ 100
11
return trust
12
return False
13

Tableau des Def, C-Use, P-Use selon les nœuds définis dans le CFG

Noeud	Def	C-Use	P-Use
1	Time_first_seen_message, Time_last_seen_message, NHam, NSpam, Trust2, nb_groups, Groups		
2	Trust1	Time_first_seen_message, Time_last_seen_message, NHam, NSpam	
3			Groups
4	Trust2, nb_groups	Trust2, nb_groups	
5	Trust2	Trust2, nb_groups	
6			Trust2
7	Trust	Trust2	
8			Trust1
9	Trust	Trust1, Trust2	
10	Trust		
11			Trust
12		Trust	
13			

<u>All definition coverage</u>

DC-Paths de chaque definition :

- Time_first_seen_message : nœud 1 -> Path1 = {1, 2}
- Time_last_seen_message : noeud 1 -> Path2 = {1, 2}
- NHam : noeud 1 -> Path3 = {1, 2}
- NSpam : noeud 1 -> Path4 = {1, 2}
- Nb_groups : noeud 1, 5 -> Path5 = {1, 2, 3, 4} et Path6 = {4, 3, 5}
- Trust2 : nœud 1, 5, 6 -> Path7 = {1, 2, 3, 4}, Path 8 = {4, 3, 5} et Path9 = {5, 6, 8, 9}
- Groups : noeud 3 -> Path10 = {1, 2, 3}
- Trust1 : noeud 2 -> Path11 = {2, 3, 4, 3, 5, 6, 8, 9}
- Trust : noeud 8, 10, 11 -> Path12 = {7, 11, 12} et Path13 = {9, 11, 12} et Path14 = {10, 11 ,12}

Jeu de tests

On prend les Paths suivants :

- PathA = {1, 2, 3, 4, 3, 5, 6, 7, 11, 12} couvre Path1, Path2, Path3, Path4, Path10, Path12
- PathB = {1, 2, 3, 4, 3, 5, 6, 8, 9, 11, 12} couvre Path5, Path6, Path7, Path8, Path14
- PathC = {1, 2, 3, 4, 3, 5, 6, 8, 10, 11, 12} couvre Path9, Path 11, Path13

Et on obtient les tests :

- D1 = <{user_id faisant partie d'un groupe = [Trust = 50] }, return = {50}>

- D2 = <{user_id = [Date_of_last_seen_message = 1 896 844 800.0, Date_of_first_seen_message = 5 034 560, NHam = 15, NSpam = 5], faisant partie d'un groupe = [Trust = 100] }, return = {100}>

- D3 = <{user_id = [Date_of_last_seen_message = 1000.0, Date_of_first_seen_message = 1000, NHam = 20, NSpam = 40], faisant partie d'un groupe = [Trust = 100] }, return = {10.2}>

All C-Use coverage

DC-Paths partant de chaque noeud de définition des variables à chaque C-Use des variables :

- Time_first_seen_message : noeud 1 -> Path1 = {1, 2}
- Time_last_seen_message : noeud 1 -> Path2 = {1, 2}
- NHam : noeud 1 -> Path3 = {1, 2}
- NSpam : noeud 1 -> Path4 = {1, 2}
- Nb_groups : noeud 1, 5 -> Path5 = {1, 2, 3, 4} et Path6 = {4, 3, 4} et Path7 = {4, 3, 5}
- Trust2 : nœud 1, 5 -> Path8 = {1, 2, 3, 4} et Path9 = {4, 3, 4} et Path10 = {4, 3, 5} et Path11 = {5, 6, 8, 9}
- Trust1 : nœud 2 -> Path12 = {2, 3, 4, 3, 5, 6, 8, 9}
- Trust : nœud 8, 10, 11 -> Path13 = {7, 11, 12} et Path14 = {9, 11, 12} et Path15 = {10, 11, 12}

Jeu de tests

On prend les Paths suivants :

- PathA = {1, 2, 3, 4, 3, 5, 6, 7, 11, 12} couvre Path1, Path2, Path3, Path4, Path5, Path7, Path8, Path10, Path13
- PathB = {1, 2, 3, 4, 3, 5, 6, 8, 10, 11, 12} couvre Path15
- PathC = {1, 2, 3, 4, 3, 4, 3, 5, 6, 8, 9, 11, 12} couvre Path6, Path9, Path11, Path12, Path14

Et on obtient les tests :

- D1 = <{user_id faisant partie d'un groupe [Trust = 50] }, return = {50}>

- D2 = <{user_id = [Date_of_last_seen_message = 1 896 844 800.0, Date_of_first_seen_message = 5 034 560, NHam = 15, NSpam = 5], faisant partie d'un groupe = [Trust = 100] }, return = {100}>

- D3 = <{user_id = [Date_of_last_seen_message = 1000.0, Date_of_first_seen_message = 1000, NHam = 20, NSpam = 40], faisant partie de 2 groupes = [Trust = 100] et [Trust = 50] }, return = {15.2}>

All P-Use coverage

DC-Paths partant de chaque noeud de définition des variables à chaque P-Use des variables :

- Trust2 : nœud 6 -> Path1 = {5, 6}
- Trust1 : nœud 2 -> Path2 = {2, 3, 4, 3, 5, 6, 8}
- Trust : nœud 8, 10, 11 -> Path3 = {7, 11} et Path4 = {9, 11} et Path5 = {10, 11}
- Groups : nœud 1 -> Path6 = {1, 2, 3}

Jeu de tests

On prend les Paths suivants :

- PathA = {1, 2, 3, 4, 3, 5, 6, 7, 11, 12} couvre Path1, Path3, Path6
- PathB = {1, 2, 3, 4, 3, 5, 6, 8, 10, 11, 12} couvre Path2, Path5
- PathC = {1, 2, 3, 4, 3, 4, 3, 5, 6, 8, 9, 11, 12} couvre Path4

Et on obtient les tests :

- D1 = <{user_id faisant partie d'un groupe [Trust = 50] }, return = {50}>

- D2 = <{user_id = [Date_of_last_seen_message = 1 896 844 800.0, Date_of_first_seen_message = 5 034 560, NHam = 15, NSpam = 5], faisant partie d'un groupe = [Trust = 100] }, return = {100}>

- D3 = <{user_id = [Date_of_last_seen_message = 1000.0, Date_of_first_seen_message = 1000, NHam = 20, NSpam = 40], faisant partie de 2 groupes = [Trust = 100] et [Trust = 50] }, return = {15.2}>

All-Use coverage

DC-Paths partant de chaque noeud de définition des variables à chaque C-Use et P-Use des variables :

- Time_first_seen_message : noeud 1 -> Path1 = C-Use{1, 2}
- Time_last_seen_message : noeud 1 -> Path2 = C-Use{1, 2}
- NHam : noeud 1 -> Path3 = C-Use{1, 2}
- NSpam : noeud 1 -> Path4 = C-Use{1, 2}
- Nb_groups : noeud 1, 5 -> Path5 = C-Use{1, 2, 3, 4} et Path6 = C-Use{4, 3, 4} et Path7 = C-Use{4, 3, 5}
- Trust2 : nœud 1, 5, 6 -> Path8 = C-Use{1, 2, 3, 4} et Path9 = C-Use{4, 3, 4} et Path10 = C-Use{4, 3, 5} et Path11 = C-Use{5, 6, 8, 9} et Path12 = P-Use{5, 6}
- Trust1 : nœud 2 -> Path13 = C-Use{2, 3, 4, 3, 5, 6, 8, 9} et Path14 = P-Use{2, 3, 4, 3, 5, 6, 8}
- Trust : nœud 8, 10, 11 -> Path15 = C-Use{7, 11, 12} et Path16 = C-Use{9, 11, 12} et Path17 = C-Use{10, 11, 12} et Path18 = P-Use{7, 11} et Path19 = P-Use{9, 11} et Path20 = P-Use{10, 11}
- Groups : nœud 1 -> Path21 = {1, 2, 3}

Jeu de tests

On prend les Paths suivants :

- PathA = {1, 2, 3, 4, 3, 5, 6, 7, 11, 12} couvre Path1, Path2, Path3, Path4, Path5, Path7, Path8, Path10, Path12, Path15, Path18, Path21
- PathB = {1, 2, 3, 4, 3, 5, 6, 8, 10, 11, 12} couvre Path14, Path17, Path20
- PathC = {1, 2, 3, 4, 3, 4, 3, 5, 6, 8, 9, 11, 12} couvre Path6, Path9, Path11, Path13, Path16, Path19

Et on obtient les tests :

- D1 = <{user_id faisant partie d'un groupe [Trust = 50] }, return = {50}>

- D2 = <{user_id = [Date_of_last_seen_message = 1 896 844 800.0, Date_of_first_seen_message = 5 034 560, NHam = 15, NSpam = 5], faisant partie d'un groupe = [Trust = 100] }, return = {100}>

- D3 = <{user_id = [Date_of_last_seen_message = 1000.0, Date_of_first_seen_message = 1000, NHam = 20, NSpam = 40], faisant partie de 2 groupes = [Trust = 100] et [Trust = 50] }, return = {15.2}>