École Polytechnique de Montréal

Baccalauréat en Génie Logiciel

Cours - LOG3430

TP2 Tests unitaires

Auteurs :

Maxime Tremblay-Rheault - 1946878
Maude Desrosiers-C. - 1946801
groupe labo 01

25 octobre 2020

Première partie: tests unitaires

Couverture initiale (en %) des jeux de tests (lignes et branches)

Coverage report: 32%

Module	*statements* ↑	*missing*	*excluded*	*branches*	*partial*	*coverage*
crud.py	179	160	16	72	0	8%
test_crud.py	107	0	0	0	0	100%
email_analyzer.py	87	76	0	18	0	10%
vocabulary_creator.py	58	51	14	28	0	8%
test_email_analyzer.py	31	0	0	0	0	100%
text_cleaner.py	28	18	0	6	0	29%
test_vocabulary_creator.py	17	0	0	0	0	100%
Total	**507**	**305**	**30**	**124**	**0**	**32%**

La couverture initiale pour les modules principaux était faible. En effet elle se situait à 8% pour crud.py, 10% pour email_amalyzer.py et 8% pour vocabulary_creator.py .

Le tableau ci-dessus donne également le détail du nombre de lignes couvertes, exclues et non-couvertes (missions). Il est également possible de constater que crud.py est le programme le plus complexe avec ses 72 branches.

Couverture après l'ajout de tests (en %)

Coverage report: 91%

Module	*statements* ↑	*missing*	*excluded*	*branches*	*partial*	*coverage*
test_crud.py	240	4	0	0	0	98%
crud.py	179	24	16	72	10	85%
email_analyzer.py	87	8	0	18	4	89%
test_email_analyzer.py	60	0	0	0	0	100%
vocabulary_creator.py	58	3	14	28	1	95%
test_flots_donnees.py	25	1	0	6	2	90%
test_vocabulary_creator.py	23	0	0	0	0	100%
renege.py	9	9	101	0	0	0%
text_cleaner.py	4	0	30	0	0	100%
Total	**685**	**49**	**161**	**124**	**17**	**91%**

La couverture se situe à 85% pour crud.py, à 89% pour email_analyzer.py et à 95% pour vocabulary_creator.

Tests ajoutés contribuant à l'amélioration de la couverture

Quatre tests ont étés ajoutés dans test_crud.py afin d'améliorer la couverture. Nous avions ajouté des conditions afin de valider les valeurs d'entrée du courriel et de la date. Ces validations ont été testées avec les tests ci-dessus.

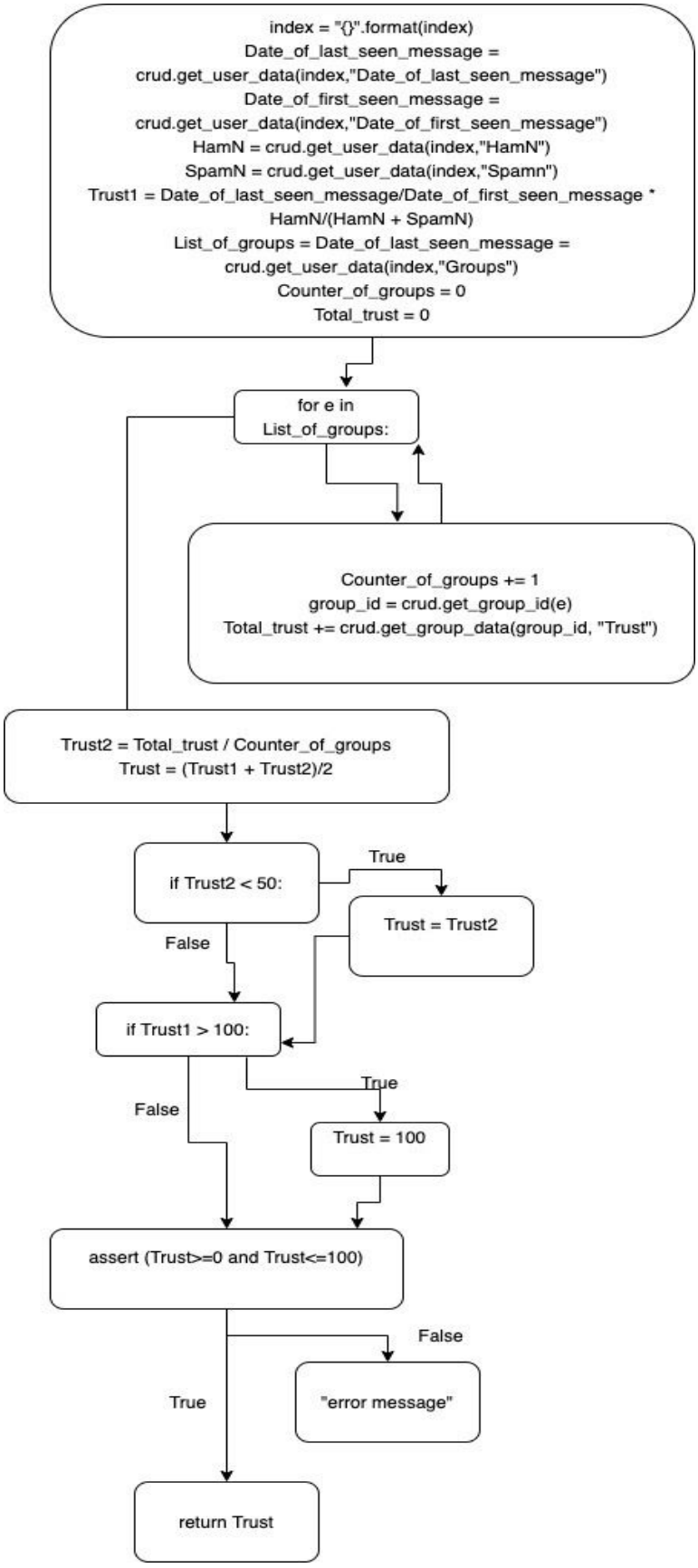
```
47      #Test ajoute: @ is in the email
48      def test_add_new_user_Return_false_if_no_a(self):
49          crud = CRUD()
50          self.assertFalse(crud.add_new_user("newEmailgmail.com","2020-10-10"))
51
52      #Test ajoute:
53      def test_add_new_user_Return_false_if_len_date_not_good(self):
54          crud = CRUD()
55          self.assertFalse(crud.add_new_user("newEmailgmail.com","2020-10"))
```

```
126     #TEST AJOUTE POUR AMELIORER OUVERTURE
127     @patch("crud.CRUD.read_groups_file")
128     def test_add_new_group_Return_false_if_group_exist(
129         self, mock_read_groups_file
130     ):
131         mock_read_groups_file.return_value = self.groups_data
132         crud = CRUD()
133         self.assertFalse(crud.add_new_group("default",50,[]))
```

```
456     #TEST AJOUTE pourrestreindre possibilite d'enlever le groupe par default (group_id : "1")
457     @patch("crud.CRUD.read_groups_file")
458     @patch("crud.CRUD.modify_groups_file")
459     def test_remove_group_Returns_false_for_invalid_id(
460         self, mock_modify_groups_file, mock_read_groups_file
461     ):
462         mock_read_groups_file.return_value = self.groups_data
463         mock_modify_groups_file.return_value = True
464         crud = CRUD()
465         self.assertFalse(crud.remove_group("1"))
466
```

Deuxième partie : tests de flots de données

```python
def calculate_trust_user(index):
    index = "{}".format(index)
    Date_of_last_seen_message = crud.get_user_data(index,"Date_of_last_seen_message")
    Date_of_first_seen_message = crud.get_user_data(index,"Date_of_first_seen_message")
    HamN = crud.get_user_data(index,"HamN")
    SpamN = crud.get_user_data(index,"Spamn")

    Trust1 = Date_of_last_seen_message/Date_of_first_seen_message * HamN/(HamN + SpamN)

    List_of_groups = Date_of_last_seen_message = crud.get_user_data(index,"Groups")
    Counter_of_groups = 0
    Total_trust = 0

    for e in List_of_groups:
        Counter_of_groups += 1
        group_id = crud.get_group_id(e)
        Total_trust += crud.get_group_data(group_id, "Trust")

    Trust2 = Total_trust / Counter_of_groups

    Trust = (Trust1 + Trust2)/2

    if Trust2 < 50:
        Trust = Trust2

    if Trust1 > 100:
        Trust = 100

    assert (Trust>=0 and Trust<=100),"error message"

    return Trust
```


index = "{}".format(index)
Date_of_last_seen_message =
crud.get_user_data(index,"Date_of_last_seen_message")
Date_of_first_seen_message =
crud.get_user_data(index,"Date_of_first_seen_message")
HamN = crud.get_user_data(index,"HamN")
SpamN = crud.get_user_data(index,"Spamn")
Trust1 = Date_of_last_seen_message/Date_of_first_seen_message *
HamN/(HamN + SpamN)
List_of_groups = Date_of_last_seen_message =
crud.get_user_data(index,"Groups")
Counter_of_groups = 0
Total_trust = 0
for e in
List_of_groups:
Counter_of_groups += 1
group_id = crud.get_group_id(e)
Total_trust += crud.get_group_data(group_id, "Trust")
Trust2 = Total_trust / Counter_of_groups
Trust = (Trust1 + Trust2)/2
if Trust2 < 50:
True
Trust = Trust2
False
if Trust1 > 100:
True
False
Trust = 100
assert (Trust>=0 and Trust<=100)
False
True
"error message"
return Trust

Variable	Def	C-Use	P-Use
index	4,5	5, 6, 7, 8, 9, 13	
Chemins partiels pour index CAS ALL-DEFS: {4, 5}, {5,6} CAS ALL-C-USES: {4, 5}, {5, 6, 7, 8, 9, 11, 13} CAS ALL-P-USES: {} CAS ALL-USES: {4, 5}, {5, 6, 7, 8, 9, 11, 13}			
Date_of_last_seen_message	6,	11	
Chemins partiels pour Date_of_last_seen_message CAS ALL-DEFS : {6, 7, 8, 9, 11} CAS ALL-C-USES: {6, 7, 8, 9, 11} CAS ALL-P-USES: {} CAS ALL-USES: {6, 7, 8, 9, 11}			
Date_of_first_seen_message	7	11	
Chemins partiels pour Date_of_first_seen_message CAS ALL-DEFS : {7, 8, 9, 11} CAS ALL-C-USES: {7, 8, 9, 11} CAS ALL-P-USES: {} CAS ALL-USES: { 7, 8, 9, 11}			
HamN	8	11	
Chemins partiels pour HamN CAS ALL-DEFS : {8, 9, 11} CAS ALL-C-USES: {8, 9, 11} CAS ALL-P-USES: {} CAS ALL-USES: { 8, 9, 11}			
SpamN	9	11	
Chemins partiels pour SpamN			

CAS ALL-DEFS : {9, 11} CAS ALL-C-USES: {9, 11} CAS ALL-P-USES: {} CAS ALL-USES: { 9, 11}			
Trust1	11	24	29
Chemins partiels pour Trust1 CAS ALL-DEFS : {11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24} CAS ALL-C-USES: {11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24} CAS ALL-P-USES: {11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29} // Trust2 > 50 CAS ALL-USES: { 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29}			
List_of_groups	13		17
Chemins partiels pour List_of_groups CAS ALL-DEFS : {13, 14, 15, 17} CAS ALL-C-USES: {} CAS ALL-P-USES: {13, 14, 15, 17} CAS ALL-USES: { 13, 14, 15, 17}			
Counter_of_groups	14, 18	18, 22	
Chemins partiels pour Counter_of_groups CAS ALL-DEFS : {14, 15, 17, 18}, {18, 19, 20, 17, 22} CAS ALL-C-USES: {14, 15, 17, 18}, {18, 19, 20, 17, 22} CAS ALL-P-USES: {} CAS ALL-USES: { 14, 15, 17, 18}, {18, 19, 20, 17, 22}			
Total_trust	15, 20	20, 22	
Chemins partiels pour Total_trust CAS ALL-DEFS : {15, 17, 18, 19, 20}, {20, 17, 22} CAS ALL-C-USES: {15, 17, 18, 19, 20}, {20, 17, 22} CAS ALL-P-USES: {} CAS ALL-USES: { 15, 17, 18, 19, 20}, {20, 17, 22}			
e	17	19	
Chemins partiels pour e CAS ALL-DEFS : {17, 18, 19}			

CAS ALL-C-USES: {17, 18, 19}

CAS ALL-P-USES: {}

CAS ALL-USES: { 17, 18, 19}

group_id	19	20	

Chemins partiels pour group_id

CAS ALL-DEFS : {19, 20}

CAS ALL-C-USES: {19, 20}

CAS ALL-P-USES: {}

CAS ALL-USES: { 19, 20}

Trust2	22	24, 27	26

Chemins partiels pour Trust2

CAS ALL-DEFS : {22, 24}

CAS ALL-C-USES: {22, 24, 26, 27}

CAS ALL-P-USES: {22, 24, 26}

CAS ALL-USES: { 22, 24, 26, 27}

Trust	24, 27, 30	34	32

Chemins partiels pour Trust

CAS ALL-DEFS :
{24, 26, 29, 32, 34} // Trust2 > 50 && Trust1 < 100
{27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50
{30, 32, 34} // Trust1 > 100 && Trust2 > 50

CAS ALL-C-USES:
{24, 26, 29, 32, 34} // Trust2 > 50 && Trust1 < 100
{27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50
{30, 32, 34} // Trust1 > 100 && Trust2 > 50

CAS ALL-P-USES:
{24, 26, 29, 32} // Trust2 > 50 && Trust1 < 100
{27, 29, 32} // Trust1 < 100 && Trust2 < 50
{30, 32} // Trust1 > 100 && Trust2 > 50

CAS ALL-USES:
{24, 26, 29, 32, 34} // Trust2 > 50 && Trust1 < 100
{27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50
{30, 32, 34} // Trust1 > 100 && Trust2 > 50

Chemins Minimaux

CAS ALL-USES:
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29, 32, 34} // Trust2 > 50 && Trust1 < 100
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29, 30, 32, 34} // Trust1 > 100 && Trust2 > 50
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20,22, 24, 26, 27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50

CAS ALL-C-USES:
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29, 32, 34} // Trust2 > 50 && Trust1 < 100
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29, 30, 32, 34} // Trust1 > 100 && Trust2 > 50
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50

CAS ALL-P-USES:
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29, 32, 34} // Trust2 > 50 && Trust1 < 100
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 17, 22, 24, 26, 29, 30, 32, 34} // Trust1 > 100 && Trust2 > 50
{4, 5, 6, 7, 8, 9, 11, 13, 14, 15, 17, 18, 19, 20, 22, 24, 26, 27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50

All-Def
{4, 5, 6, 7-8-9-11-13-14-15-17-18-19-20-17-22-24, 26, 29, 32, 34}   // Trust2 > 50 && Trust1 < 100
{4-5-6-7-8-9-11-13-14-15-17-18-19-20-17-22-24, 26, 29, 30, 32, 34} // Trust1 > 100 && Trust2 > 50
{4-5-6-7-8-9-11-13-14-15-17-18-19-20-17-22-24, 26, 27, 29, 32, 34} // Trust1 < 100 && Trust2 < 50

N.B Les chemins minimaux les même

Proposer des jeux de test pour chaque critère

Données en entrée du jeu de test


```
{
  "1": {
     "name": "FriendlyGuy@mail.com",
     "Trust": 1.0,
     "Spamn": 0,
     "HamN": 2
     "Date_of_first_seen_message": 1603591810.0,
     "Date_of_last_seen_message": 1097971200.0,
     "Groups": [
        "family",
        "friends"
     ]
  },
 "2": {
     "name": "MediumGuy@mail.com",
     "Trust": 0.75,
     "Spamn": 1,
     "HamN": 3,
```

```
    "Date_of_first_seen_message": 968976000.0,
    "Date_of_last_seen_message": 968976000.0,
    "Groups": [
      "default",
      "friends"
    ]
  }
  "3": {
    "name": "SuperVilain@doom.com",
    "Trust": 0.0,
    "Spamn": 2,
    "HamN": 0,
    "Date_of_first_seen_message": 968976000.0,
    "Date_of_last_seen_message": 968976000.0,
    "Groups": [
      "garbage",
      "junk"
    ]
  }
}
```


Ici, on pose que FriendlyGuy ainsi que chacun de ses groupes ont une Trust level de 100%.
Avec un quotient de Date_of_last_seen_message / Date_of_first_seen_message supérieur à 1, on aura:
Trust1 : (quotient > 1) fois (100%) est supérieur à 1 Trust1 > 100%
Trust2 : (family trust + friends trust) / 2 = 100% Trust2 > 50%
où:
family (trust=100%)
friends(trust=100%)

On pose que MediumGuy ainsi que tous ses groupes associés ont un Trust level d'environ 75%.
Trust1 : (quotient ~= 1) x (75%) est inférieur à 1 Trust1 < 100%
Trust2 : (default trust + friends trust) / 2 = 75% Trust2 > 50%
où
default (trust=50%)
friends(trust=100%)

On pose finalement que SuperVilain ainsi que tous ses groupes ont un Trust level de 0%.
Avec une un quotient de Date_of_last_seen_message / Date_of_first_seen_message proche de 1, on aura:
Trust1 : (quotient ~= 1) x (0%) Trust1 < 100%
Trust2 : moyenne de 0% Trust2 < 50%
car:
grarbage (trust ~= 0%)
junk (trust ~= 0%)

	FriendlyGuy	MediumGuy	SuperVilain
Trust2 < 50	F	F	V
Trust1 > 100	V	F	F

Résultats attendus du jeu de test

Avec ceci, on couvre les conditions imposées par les contraintes.
Et on obtient les résultats suivant:
FriendlyGuy Trust = 100
MediumGUy Trust = 75
SuperVilain Trust = 0

Quel critère est le plus strict?

Le critère le plus strict est All-Uses puisqu'il doit absolument passer par chaque DEF et par chaque USE . Ainsi, le nombre de chemins partiels pour ce critère était plus élevé.

- All Def est moins strict que All-Uses car il suffit d'y trouver un DC path pour chaque DEF.
- ALL-C-USES est moins strict que All-Uses car il n'est pas requis de passer par les P-USES.
- ALL-P-USES est moins strict que All-Uses car il n'est pas requis de passer par les C-USES.