

Analyse de la présence des stacktraces python sur Stackoverflow.com

Grégory LEFER

Alexandre FRANCOIS

02/12/2014

Table des matières

1	Introduction	1
2	Développement	2
2.1	Reconnaissance de stacktrace python	2
2.2	Parseur de stacktraces	2
2.3	Parseur de questions contenant au moins une stacktrace	3
2.4	Parseur de questions ne contenant pas de stacktraces	3
2.5	Résumé	3
3	Exploitation des datasets	3
3.1	Buckets, baseX, XPath	4
3.2	Stacktrace dataset	6
3.3	Question avec stacktraces	6
3.4	Question sans stacktrace	6
4	Interprétation des données	6
4.1	Stat CrashDataset	6
4.2	Stat Question avec crash	6
4.3	Stat Question sans crash	6
5	Conclusion	6

1 Introduction

Lors du développement de logiciel, il est fréquent de voir apparaître des bugs, due à de nombreux facteurs tel que l'expérience du développeur ou la maturité d'un framework. C'est dans ce but que de nombreux sites proposent leurs services afin de permettrent aux développeurs de pouvoir partager leurs connaissances. Il existe également des sites spécialement conçus afin de pouvoir exposer les problèmes rencontrés lors d'un développement comme par exemple stackoverflow.com.

Une stack trace (ou trace d'appels en français) est un affichage de la pile d'exécution à un instant précis de l'exécution d'un programme. Bien que pouvant être affichée à n'importe quel moment par le développeur, cette dernière est le plus souvent utilisée lors de la phase de débogage. C'est en général ce qui est affiché dans le terminal par l'interpréteur lorsqu'un logiciel rencontre une erreur grave. De nombreux développeurs utilisent celle-ci pour découvrir l'origine de l'erreur.

C'est sur ce constat que nous allons réalisés des statistiques concernant les pratiques des utilisateurs du site stackoverflow.com. Nous allons nous concentrer sur la présence des stack traces dans leurs questions et leur influence sur les réponses apportées (quantité, pertinence, résolution..).

```

1 <Posts
2   Id
3   PostTypeId (1=Question, 2=Answer)
4   AcceptedAnswerId (only present if PostTypeId is 1)
5   ParentID (only present if PostTypeId is 2)
6   Score
7   Body
8   Tags
9 />

```

FIGURE 1 – Structure du fichier Posts.xml minimifié

```

1 (Traceback \(most recent call last\):)
2 (\s+File\s+"[^"]+"\s+,\s+line\s+\d+\s+,\s+in\s+[\n]+\n[\n]+\n)+
3 \s*\w+:[\n]+\n

```

FIGURE 2 – Expression régulière reconnaissant les stacktraces Python

2 Développement

Pour réaliser cette analyse, le contenu des posts du site StackOverFlow (Posts.xml) a été utilisé. StackOverFlow est un site web pour les développeurs rencontrant divers problèmes afin de les exposer et ainsi trouver une aide extérieure. Ce fichier représente 30Go de données contenant l'ensemble des posts depuis la création du site jusqu'au début 2014. La librairie stAx (Streaming Api for XML) qui permet de traiter un document XML de façon simple en consommant peu de mémoire a été utilisé pour lire ce fichier.

2.1 Reconnaissance de stacktrace python

Afin de récupérer les stacktraces python un parser a été créé. Pour savoir si une stacktrace est présente une expression régulière est utilisée.

2.2 Parseur de stacktraces

Le premier dataset créé est un dataset contenant seulement des stacktraces pythons

```

function PARSESTACKTRACEDATASET
  List Posts ← read('Posts.xml')
  for all post ← Posts do
    List Stacktraces ← getStacktraces(post.body)
    for all stacktrace ← Stacktraces do
      writer(stacktrace)
    end for
  end for
end function

```

FIGURE 3 – Recherche des stacktraces python

2.3 Parseur de questions contenant au moins une stacktrace

Le second dataset constitue toutes les questions contenant au moins une stacktraces dans celle-ci.

```
function PARSEQUESTIONWITHSTACKTRACEDATASET
  List Posts  $\leftarrow$  read('Posts.xml')
  List Questions
  for all post  $\leftarrow$  Posts do
    if post.postTypeId is Question then
      List Stacktraces  $\leftarrow$  getStacktraces(post.body)
      if Stacktraces is not empty then
        question = new Question(post, Stacktraces)
        Questions  $\leftarrow$  question
      end if
    else
      if post.postTypeId is Reponse then
        for all question  $\leftarrow$  Questions do
          if question.idPost = post.idParent then
            List Stacktraces  $\leftarrow$  getStacktraces(reponse.body)
            reponse = new Reponse(post, Stacktraces)
            if question.AcceptedAnswerId = reponse.idPost then
              question.AcceptedAnswer  $\leftarrow$  reponse
            else
              question.reponses  $\leftarrow$  reponse
            end if
          end if
        end for
      end if
    end if
  end for
  writer(Questions)
end function
```

FIGURE 4 – Recherche des questions contenant une stackstraces python

2.4 Parseur de questions ne contenant pas de stacktraces

Ce dernier dataset regroupe toutes les questions pythons sans stacktrace afin de d'avoir des données de comparaison avec le deuxieme dataset

2.5 Résumé

3 datasets sont donc disponible, la raison de la création de plusieurs datasets par rapport à un seul est que la génération d'un dataset demande un temps de l'ordre de 25min à 1h, la creation d'un dataset plus important aurait donc été plus long, cette solution a été adapté afin de pouvoir exploiter un dataset pendant la création d'un autre.

3 Exploitation des datasets

Cette section présente les differentes manières utilisées pour l'exploitation de ces datasets

```

function PARSEQUESTIONWITHOUTSTACKTRACE DATASET
  List Posts  $\leftarrow$  read('Posts.xml')
  Map<Int, Question> Questions
  for all post  $\leftarrow$  Posts do
    if post.postTypeId is Question AND post.Tags contains "python" then
      List Stacktraces  $\leftarrow$  getStacktraces(post.body)
      if Stacktraces is empty then
        question = new Question(post)
        Questions  $\leftarrow$  < post.idPost > < question >
      end if
    else
      if post.postTypeId is Reponse then
        tmp  $\leftarrow$  Questions.get(post.parentId)
        if tmp exist then
          List Stacktraces  $\leftarrow$  getStacktraces(reponse.body)
          reponse = new Reponse(post, Stacktraces)
          if question.AcceptedAnswerId = reponse.idPost then
            question.AcceptedAnswer  $\leftarrow$  reponse
          else
            question.reponses  $\leftarrow$  reponse
          end if
        end if
      end if
    end if
  end for
  writer(Questions)
end function

```

FIGURE 5 – Recherche des questions ne contenant pas de stacktraces python

3.1 Buckets, baseX, XPath

Le dataset contenant l'ensemble des stacktraces à été divisé en bucket en fonction du type d'erreur ou exception survenue. Les datasets contenant les questions et leurs réponses associées ont été exploité grâce au logiciel baseX qui est un système de gestion de base de données XML native et légère et est spécialisé dans le stockage, le requêtage et la visualisation de larges documents et collections de documents XML. Pour effectuer nos différentes requêtes le langage XPath à été utilisé. c'est un langage pour localiser une portion d'un document XML simple d'emploi.

```

1 #Nombre de questions
2   //Question
3   Resultat: 12492
4 #Nombre de questions contenant une reponse acceptee
5   //Question[AcceptedAnswer/Score]
6   Resultat: 7167
7 #Nombre de questions ne contenant pas de reponse acceptee mais ayant des reponse
8   //Question[not(AcceptedAnswer/Score) and Reponses//Score]
9   Resultat: 3796
10 #Nombre de questions ne contenant pas de reponse acceptee mais ayant des reponse avec un
    score positif
11   //Question[not(AcceptedAnswer/Score) and Reponses//Score > 0]
12   Resultat: 2283
13 #Nombre de questions contenant une reponse acceptee mais ayant des reponse avec un score
    plus eleve
14   //Question[Reponses//Score > AcceptedAnswer/Score]
15   Resultat: 713
16 #Nombre de reponse contenant une stacktrace
17   //Question[Reponses//Stack]
18   Resultat: 99

```

FIGURE 6 – Requêtes effectué sur le dataset contenant les questions avec les stacktraces

```

1 #Nombre de questions
2   //Question
3   Resultat: 345217
4 #Nombre de questions contenant une reponse acceptee
5   //Question[AcceptedAnswer/Score]
6   Resultat: 215288
7 #Nombre de questions ne contenant pas de reponse acceptee mais ayant des reponse
8   //Question[not(AcceptedAnswer/Score) and Reponses//Score]
9   Resultat: 98154
10 #Nombre de questions ne contenant pas de reponse acceptee mais ayant des reponse avec un
    score positif
11   //Question[not(AcceptedAnswer/Score) and Reponses//Score > 0]
12   Resultat: 62907
13 #Nombre de questions contenant une reponse acceptee mais ayant des reponse avec un score
    plus eleve
14   //Question[Reponses//Score > AcceptedAnswer/Score]
15   Resultat: 29485
16 #Nombre de reponse contenant une stacktrace
17   //Question[Reponses//Stack]
18   Resultat: 626
19 #Nombre de question ne contenant pas de reponse
20   //Question[not(Reponses/Reponse)]
21   Resultat: 131175

```

FIGURE 7 – Requêtes effectué sur le dataset contenant les questions sans stacktraces

3.2 Stacktrace dataset

3.3 Question avec stacktraces

3.4 Question sans stacktrace

4 Interprétation des données

4.1 Stat CrashDataset

4.2 Stat Question avec crash

4.3 Stat Question sans crash

5 Conclusion

Références