Documentazione Flaky-Tests-MongoDB

ANTONIO TROVATO

ACM Reference Format:
ANTONIO TROVATO. 2022. Documentazione Flaky-Tests-MongoDB. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 AMBITO DEL PROGETTO

Il test di regressione è una pratica molto utilizzata nel campo dell'ingegneria del software, sopratutto durante la fase di manutenzione di sistemi, per determinare se recenti modifiche al codice hanno introdotto errori. Quando una componente software non passa uno di questi test, ci si aspetta di rilevare al suo interno un bug introdotto di recente; purtroppo però quasi sempre ci si ritrova di fronte a particolari ed indesiderabili tipi di test, detti flaky.

1.1 Descrizione del problema

I test flaky sono test non deterministici che possono fallire o avere successo nonostante siano lanciati sulla stessa versione di una componente software. Quando i test hanno un comportamento altalenante tra il successo ed il fallimento senza alcun cambio della versione del codice in esame, gli sviluppatori hanno una vita davvero difficile: è difficile rilevare tali test tramite debug, inoltre i cicli di rilascio possono subire dei grossi rallentamenti. Dunque, determinare se un test sia o meno flaky è un topic molto importante oramai nell'ambito del testing di software. Tecniche attualmente esistenti per rilevare flaky test si affidano alla semplice riesecuzione dei test: se si ottengono diversi risultati dallo stesso test sullo stesso codice, allore il test è sicuramente flaky. Il problema con questa tecnica è che essa è davvero molto dispendiosa; gli sviluppatori dovrebbero eseguire ogni test anche 1000 volte per avere una buona probabilità di rilevare quelli flaky.

Un approccio alternativo consiste nel costruire un classificatore di machine learning che possa distinguere tra test flaky e non flaky. Seguendo questa strada, gli sviluppatori possono usare il classificatore per determinare quali test sono probabilmente flaky, concentrando le riesecuzioni solo su questi.

Il principale problema nello sviluppo di un classificatore consiste nell'ottenimento e nella gestione dei dati che dovrà sfruttare per l'apprendimento durante la sua fase di training. Inoltre, va considerato che anche i modelli di machine learning, come ogni altra componente software, vanno costantemente manutenuti e ritestati, affinché rimangano sempre validi. Dunque è necessario scegliere una strategia valida per il salvataggio di enormi quantità di dati da usare per il training e per il test. Dopo una serie di valutazioni si è visto che la tecnologia di gestione dati più adatta per questo problema è quella NoSql, in quanto il modello di machine learning utilizzato per la predizione di test flaky deve interagire molto di frequente con i dati contenuti nel database; inoltre i campioni contenuti nella base di dati saranno sempre di più, quindi sarà necessario adottare delle strategie efficaci per scalarla, mantenendo però prestazioni molto elevate.

La tecnologia NoSql che è stata valutata come più adatta è MongoDB. MongoDB è infatti del tutto schema-less, facile da scalare, estremamente veloce nell'accesso ai dati, non richiede una conversione dagli oggetti dei linguaggi OO a quelli del database, permette l'implementazione di query complesse ed il tuning.

2 OBIETTIVI DEL PROGETTO

Questo progetto si propone di: convertire un dataset di test (flaky e non) in un database MongoDB, creare un applicazione Java in grado di eseguire le azioni più importanti sul database ottenuto.

3 STEP ESEGUITI PER RAGGIUNGERE L'OBIETTIVO

Partendo dal dataset estratto da "Alshammari's et al. work on flaky tests classification", si vuole ottenere un database MongoDB e successivamente sviluppare un'applicazione Java ingrado di eseguire query sul database.

3.1 Raccolta ed analisi requisiti

Poiché il database che si vuole costruire servirà per testare, allenare e manutenere un modello di machine learning ed inoltre per elaborare alcune informazioni statistiche riguardo i test flaky, sono stati raccolti alcuni requisiti fondamentali per il suo sviluppo. In particolare siamo interessati a conoscere quanti dei test flaky hanno dipendenze con altri casi di test, quanti di loro hanno più di 100 righe di codice, qual è la loro media di righe di codice; inoltre, per andare incontro alle esigenze di mautenzione del modello, è anche necessario avere la possibilità di effettuare ricerche di documenti ed inserire nuovi casi di test (documenti) all'interno del database, in modo da mantenere il modello sempre aggiornato. In fine, si vuole ottenere un sistema che mantenga un buon livello di affidabilità ma che soprattutto sia il più rapido possibile nell'esecuzione delle query.

3.2 Progettazione dello schema concettuale e delle operazioni

Lo schema concettuale è composto da un'unica entità "TestCase" costellata di attributi. Nella tabella seguente ci sono tutte le informazioni riguardanti gli attributi dell'entità.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Nome	Dominio
Id	String
NameProject	String
TestCase	String
tloc	Float
tmcCabe	Float
assertionDensity	Float
assertionRoulette	Float
mysteryGuest	Float
eagerTest	Integer
sensitiveEquality	Float
resourceOptimism	Integer
fireAndForget	Float
loc	Float
cbo	Float
wmc	Float
rfc	Float
halsteadLength	Float
halsteadVolume	Float
classDataShouldBePrivate	Integer
complexClass	Integer
functionalDecomposition	Integer
godClass	Float
spaghettiCode	Integer
isFlaky	Integer

Table 1: Atributi dell'entità TestCase

Di seguito si trova una tabella contenente tutte le operazioni che sono richeste per la realizzazione dei requisiti specificati e che sono state implementate dal client Java.

Operazione	Frequenza
Recupera test flaky	alta
Recupera test non flaky	alta
Recupera test con cbo > 0	bassa
Recupera test con loc > 100	bassa
Proiezione di cbo,loc,isFlaky	alta
Media loc	bassa
Media loc con isFlaky = 1	bassa
Loc massimo per flaky e non flaky	bassa
Inserisci test con nameProject,loc,cbo	alta
Ricerca tramite nameProject	bassa
Ordina per loc crescente	bassa
Rendi indice godClass	bassa
Rimuovi godClass come indice	bassa

Table 2: Operazioni da eseguire

3.3 Scelta del DBMS

Poiché le applicazioni che lavoreranno sul database vi faranno un accesso frequente e poiché la struttura dei dati da salvare al suo interno è schema-less per natura, le tecnologie NoSql sono state vautate come più adatte per il problema in esame. A ciò vanno aggiunte altre motivazioni, come la fondamentale necessità di scalare in maniera efficace i dati salvati e di accedere ad essi in modo estremamente performante, anche a discapito delle classiche proprietà ACID. Tra tutte le possibilità messe a disposizione dal mondo NoSql, la soluzione migliore è apparsa quella dell'utilizzo di un database MongoDB in quanto si adatta alla pefezione a tutti i requisiti appena descritti.

4 CONCLUSIONE

Molto spesso nei progetti software sono presenti test flaky, i quali rendono estremamente difficile la vita degli sviluppatori. Dunque, si cerca di sviluppare tecnologie sempre migliori per il loro rilevamento; in particolare, i classificatori di machine learning sono la soluzione più diffusa. Però, per poter essere manutenuti nel migliore dei modi, è estremamente utile poter salvare in maniera cosìnsistente i dati che vengono utilizzati per il loro training e test. Inoltre, i dati così salvati possono essere sfruttati per estrapolare altre informazioni statistiche sui test in esame. A tale scopo, è stato realizzato un database NoSQL, che grazie alle sue caratteristiche permette un accesso ai dati rapido ed estremamente scalabile. Inoltre, per poter effettuare interrogazioni sul database, è stato sviluppato anche un client Java che implementa tutte le query di maggior interesse.