Assunzioni progetto ISW2

Filippo Muscherà

0338276

1 Assunzioni

Qui sono elencate le assunzioni fatte durante lo sviluppo di questo progetto.

1.1 Jira

- Se una release ricavata da Jira non ha nessun commit associato, viene scartata.
- Per associare ogni commit a una release si confronta la data del commit con quella di inizio e fine delle release, trovando l'unica possibile release tale che:
  data inizio release $\leq$ data commit $\leq$ data fine release.
- Le release sono state numerate ordinalmente secondo la loro data di inizio. Le release "vuote" sono state rimosse prima di questa operazione, in modo da non poter avere "buchi" nella numerazione delle release.
- La OV e la IV di un ticket sono state sempre ricavate dalla data di apertura e chiusura del ticket, associando queste date con la release presente in quel momento.

1.2 Ticket e Proportion

- Se un ticket possedeva delle AV, la prima di queste è stata scelta come IV.
- Un ticket è stato considerato *valido* per proportion se:
  - La lista di AV non è vuota.
  - $OV < FV$.
  - Tra le affected version è compresa la OV.
  - Tra le affected version **non** è compresa la FV.
- Eventuali buchi nelle AV sono poi stati riempiti dopo aver controllato la valdiità del ticket per proportion.
- Sulla base di quanto scritto nel Paper su Proportion, la treshold per scegliere se usare proportion increment o cold start è stata settata a 5. (< 5 ticket cold start, $\geq$ increment).
- Per semplicità, quando si usano altri progetti per fare cold start, si usano tutti i ticket disponibili, senza controllare che fossero disponibili nella release relativa all'iterazione di walk-forward che si sta considerando.
- Sempre per semplicità, il labelling della buggyness realistica viene rieseguito interamente con il nuovo valore di proportion ad ogni iterazione. Comunque va anche detto che, ad ogni iterazione, si hanno a disposizione un numero $\geq$ di ticket, e quindi la stima del proportion sarà più precisa, e il dataset risultante potrebbe essere potenzialmente migliore. In ogni caso, realizzare il codice in questo modo è stato più semplice e veloce.

1.3 Git e Commit

- I commit sono stati associati ai ticket controllando che il commit message contenesse il codice del ticket. Per semplicità non sono stati effettuati controlli per vedere se la data del commit fosse:
  ticket opening date $\leq$ commit date $\leq$ ticket closing date.
  Si è fatta questa scelta sia per questioni di semplicità di progettazione,

sia perché si è ritenuto che fosse un'evenienza poco probabile. Inoltre si sarebbe comunque persa l'associazione di un ticket a un commit, quando questa era comunque abbastanza evidente, dal momento che il messaggio del commit conteneva l'ID del ticket. A risentirne sarebbe stato il labelling della buggyness. Alcune classi sarebbero risultate non buggy quando è abbastanza chiaro che lo fossero, per una discrepanza tra le date di ticket e commit. Si è dunque scelto di procedere in questa maniera.

1.4 Weka e Walk-Forward

- In Weka, oltra ai tre classificatori "standard", è stato considerato anche il classificatore "Multilayer Perceptron", della categoria "function" di Weka. Questo è stato fatto per avere un classificatore per ognuna delle quattro principali categorie disponibili su Weka, e provare a dare quindi una maggior validità alle comparazioni tra i vari modelli.

- La formula per il sampleSizePercent dell'oversampling è stata ricavata direttamente dalla documentazione di Weka, ed è la seguente:
$sampleSizePercent = 2 \cdot 100 \cdot \frac{|classeMaggioranza|}{|istanzeTotali|}$

- Dal training set, per semplicità non è stata tolta la release più recente. Questo è stato fatto esclusivamente per velocizzare la realizzazione del progetto. Infatti rimuovere la release più recente ad ogni iterazione avrebbe contribuito a ridurre lo snoring e creare un training set migliore per i modelli di machine learning.

- Per il walk-forward non sono state considerate le prime due iterazioni:

  - La prima iterazione sarebbe stata quella in cui il modello, con un training set nullo avrebbe dovuto predire la buggyness della prima release. Questo scenario è stato considerato privo di interesse e poco sensato.
  - La seconda sarebbe stata quella in cui il modello, addestrato sulla prima release, avrebbe dovuto predire la seconda. Questo step è stato saltato poiché il calcolo della buggyness realistica in questa fase non avrebbe prodotto risultati. Non ci sarebbero potuti essere bug post-release, avendo una sola release. Il dataset quindi avrebbe contenuto solo istanze con buggyness "no", e le predizioni

sul testing set sarebbero quindi state tutte "no", di conseguenza. Per questo motivo, anche il secondo step del walk-forward è stato saltato, e si è partiti direttamente dal terzo.
{*Nota: Come verifica si è anche provato a costruire i dataset di BookKeeper e OpenJPA per questo step, e l'assunzione è stata confermata: nei due training set non era presente nessuna istanza buggy, dato che qui questa veniva calcolata realisticamente.*}