LTAT.05.008:
Software Analytics

Ezequiel Scott
ezequiel.scott@ut.ee
Recap – What have we learnt?

• A notion about **Software Analytics**
• Different **sources of data** worth to analyse
• Different ways to **extract** the data
• Several **common problems** studied within software analytics
• **Data transformation** concepts
• **Data mining** algorithms to address those problems
How to support software analytics?

We rely on the process of knowledge discovery from data (KDD).

How to support software analytics?

We rely on the process of knowledge discovery from data (KDD).

KD3: evaluation and presentation

• **Pattern evaluation**
  To identify the *truly* interesting patterns representing knowledge based on *interestingness measures*

• **Knowledge presentation**
  Where visualization and knowledge representation techniques are used to present mined knowledge to users

Are all patterns interesting?

• *What makes a pattern interesting?*

• *Can a data mining system generate all of the interesting patterns?*

• *Can the system generate only the interesting ones?*
Are all patterns interesting? (I)

What makes a pattern interesting?

• A pattern is **interesting** if it is
  (1) *easily understood* by humans
  (2) *valid* on new or test data with some degree of *certainty*
  (3) potentially *useful*, and
  (4) *novel*

• A pattern is also interesting if it validates a hypothesis that the user *attempt to confirm*.

• An interesting pattern represents **knowledge**

Measures of pattern interestigness

• **Objective measures**
  
  Based on the structure of discovered patterns and the statistics underlying
  
  *e.g. The Support of an Association Rule*

• **Subjective measures**
  
  Based on user beliefs in the data. Allow for finding patterns:
  
  • **Unexpected/Actionable**
    
    Contradicting a user’s belief, or offer strategic information
    
    *e.g. “a large earthquake often follows a cluster of small quakes”*

  • **Expected.** If patterns confirm a hypothesis
    
    *e.g. ?*

Can a data mining system generate all of the interesting patterns?

• Refers to the completeness of a data mining algorithm.
• Generate all possible patterns → It is often unrealistic and inefficient for data mining systems
• Solution: user provided constraints and measures to focus the search

Are all patterns interesting? (III)

Can the system generate only the interesting ones?

- Refers to an optimization problem in data mining.
- Generating only interesting patterns → highly desirable
- Avoid to search through the patterns generated to identify the truly interesting ones
- Progress has been made in this direction; however, such optimization remains a challenging issue in data mining.

Applications of Software Analytics

- combining software product information with apps store data\textsuperscript{1,2};
- using process data to predict overall project effort\textsuperscript{3};
- using software process models to learn effective project changes\textsuperscript{4};
- using operating system logs that predict software power consumption\textsuperscript{5};
- exploring product line models to configure new applications\textsuperscript{6};
- mining natural language requirements to find links between components\textsuperscript{7};
- mining performance data\textsuperscript{8,9};
- using XML descriptions of design patterns to recommend particular designs\textsuperscript{10};
- using email lists to understand the human networks inside software teams\textsuperscript{11};
- linking emails to source code artifacts and classifying their content\textsuperscript{12};
- using execution traces to learn normal interface usage patterns\textsuperscript{13};
- using bug databases to learn defect predictors that guide inspection teams to where the code is most likely to fail\textsuperscript{14–16} and to classify changes as clean or buggy\textsuperscript{17};
- using security data to identify indicators for software vulnerabilities\textsuperscript{18};
- using visualization to support program comprehension\textsuperscript{19};
- using software ontologies to enable natural language queries\textsuperscript{20}; and
- mining code clones to assess the implications of cloning and copy/paste in software.\textsuperscript{21,22}

Predicting Story Points from User Stories
Planning Poker

User Story: As an admin I should be able to create devices.

ESTIMATES

8
5
8
5

MODERATOR

OK

ZENKAI
Predicting Story Points

- Decision Trees
- Naïve Bayes
- Support Vector Machines

Figure 1: Estimation flow chart

Predicting Story Points

Standardize XD logging to align with Spring Boot

Details
- Type: Story
- Priority: Major
- Affects Version/s: 1.2 GA
- Component/s: None
- Labels: None
- Story Points: 8
- Rank (Obsolete): 922372036854775807
- Pull Request URL: https://github.com/spring-projects/spring-xd/pull/1633
- Sprint: Sprint 49

Status: DONE
Resolution: Complete
Fix Version/s: 1.2 RC1

People
- Assignee: David Turanski
- Reporter: David Turanski
- Votes: 0 Vote for this issue
- Watchers: 1 Start watching this issue

Dates
- Created: 20/Apr/15 9:52 AM
- Updated: 21/May/15 1:10 PM
- Resolved: 21/May/15 1:10 PM

Description
In XD today we use commons-logging or slf4j APIs bound to log4j at runtime (configured with log4j.properties).

Boot uses slf4j APIs backed by logback. This causes some build incompatibilities building a component that depends on spring-xd-dirt and spring-boot, requiring specific dependency exclusions. In order to simplify building and troubleshooting log dependencies, XD should standardize on slf4j APIs (replace any commons-logging Loggers with Slf4j). This is internal only, and would not impact users who are used to seeing log4j properties. An additional step is to replace log4j with logback. This change would be visible to end users but will provide us greater affinity with boot and improve the developer experience. If we make this change it should go into 1.2 GA.
Features

Type
Priority
Components
Watchers
Labels
Sprint
Summary
Description

... Description Length
N-grams

# Type

# watchers

+ An issue report with 3 story points
– An issue report with 1 story point

Common performance measure for classification problems

- **Success**: instance’s class is predicted correctly
- **Error**: instance’s class is predicted incorrectly
- **Classification error rate**: proportion of instances misclassified over the whole set of instances.

\[ e = \frac{FP + FN}{TP + TN + FP + FN} \]

- Classification Error Rate on the entire set can be too optimistic
- Randomly split data into **training** and **test sets**
Training/Validation/Test sets

For large datasets, a single split is usually sufficient.
For smaller datasets, rely on cross validation.
Training/Validation/Test sets

Typically, just enough test points to form a reasonable estimate of generalization error.
Decision boundaries

\[
\text{Score}(x) = w_0 \times \#\text{watchers} + w_1 \times \text{Priority}
\]
Decision boundaries

- For linear classifiers:
  - When 2 weights are non-zero → **Line**
  - When 3 weights are non-zero → **plane**
  - When many weights are non-zero → **hyperplane**
- For more general classifiers
  - → more complicated shapes
Good and Bad Classifiers

Good:
- sufficient data
- low training error
- simple classifier

Bad:
- insufficient data
- training error too high
- classifier too complex
What is good accuracy?

- For **binary** classification:
  
  Half the time, you’ll get it right! (on average)
  
  → classification error = 0.5

- For **multiclass** classification
  
  \[ \text{error} = 1 - \frac{1}{k} \]
  
  error = 0.666 for 3 classes, 0.75 for 4 classes, ...

**Sanity check:**

At the very, very, very least, you should healthily beat random.

Otherwise, it’s (usually) pointless.
Analyze the reported accuracies!

- Is there **class imbalance**?
- How does it compare to a simple, **baseline approach**?
  - Random guessing
  - Majority class
  - ...
- Most importantly:
  
  What accuracy does my application need?
  - What is **good enough** for my user’s experience?
  - What is the **impact of the mistakes** we make?
Sample / true error

Sample Error

The proportion of examples in $S$ that $C$ misclassified

$$\text{error}(C, S) = \frac{1}{|S|} \sum_{(x,y) \in S} \delta(C(x) \neq y)$$

$$\text{acc}(C, S) = \frac{1}{|S|} \sum_{(x,y) \in S} \delta(C(x) = y)$$

True error

The probability to misclassify an instance drawn from $D$ at random

$$\text{error}(C, D) = \sum_{(x,y) \in D} P(x,y) \cdot \delta(C(x) \neq y)$$

We cannot measure the true error.

We can only estimate $\text{error}(C,D)$ by the Sample $\text{error}(C,S)$
Overfitting

If there exists a model with estimated params $w'$, such that:

1) Training error $w^* <$ training error $w'$
2) True error $w^* >$ true error $w'$
K-fold strategy

1. Set aside the test set and split the train set into k folds.

2. For each parameter combination:
   - Train on Fold 1 and compute metric 1.
   - Repeat for other folds and average metrics.

3. Choose the parameter combination with the best metrics.
Holdout strategy

1. Split your data into train/validation/test.
2. For each parameter combination:
   - Train a model.
   - Compute metric on validation set.
3. Choose the parameter combination with the best metric.
   - Retrain model on all training data.
   - Compute metric on test set.
   - Test metric (can compare with other models).
Classification measures

- More insight over a classifier’s behavior
  - PRECISION = TP / (TP + FP)
  - RECALL = TP / (TP + FN)

- Comparing different approaches is difficult when using two evaluation measures

$$f_B = \left(1 + \beta^2\right) \frac{P \cdot R}{\left(\beta^2 \cdot P\right) + R}$$

$\beta \rightarrow$ relative importance of P and R

Popular setting $\beta = 1$
Accuracy of effort estimation model

- Mean Magnitude of Relative Error (MRE)

\[ MRE = \frac{|y - \hat{y}|}{y} \]

- Mean Absolute Error (MAE)

\[ MAE = \frac{1}{N} \sum_{i} |y_i - \hat{y}_i| \]

- Standardized Accuracy (SA)

\[ SA = \left( 1 - \frac{MAE}{MAE_{RandomGuess}} \right) \times 100 \]
Recommending issues to developers
Recommender Systems
Recommender Systems

- Issue database
- Selected issues
- New issue
- Feature extraction and selection
- Clustering identification
Cluster Analysis

- **Clustering** is a data mining technique that groups together a set of items having similar characteristics.

- Applying clustering algorithms on matrix $A$ allow us to discover developer clusters.
Compactness and Separation

• Within Cluster Sums of Squares (WSS)

\[
WSS = \sum_{i=1}^{N_C} \sum_{x \in C_i} d(X, \bar{X}_{C_i})^2
\]

Measure of compactness \(\leftarrow\) minimise

• Between Cluster Sums of Squares (BSS)

\[
BSS = \sum_{i=1}^{N_C} |C_i| \cdot d(\bar{X}, \bar{X}_{C_i})^2
\]

Measure of separation \(\leftarrow\) maximise

\(\bar{X}_{C_i}\) Cluster centroid
\(X\) Object
\(C_i\) Cluster
\(N_C\) #Clusters
\(\bar{X}\) Sample mean
\(|C_i|\) Objects in cluster
K-Means – Choosing k

• **Goal**: to find k that minimises WSS

• **Problem**: WSS keeps decreasing as k increases!

• **Solution**: WSS starts decreasing slowly

\[ TSS = WSS + BSS \]

\[ \frac{WSS}{TSS} < 0.2 \]}

The elbow point will tell you when further increasing k has no significant influence on your compactness and separation

---

**Rule of Thumb**

0.2

Look for the **elbow** in the plot

---

**TSS = WSS + BSS**

**WSS / TSS < 0.2**

**Fix K**

---

The elbow point will tell you when further increasing k has no significant influence on your compactness and separation

---

**K-Means – Choosing k**

**Goal**: to find k that minimises WSS

**Problem**: WSS keeps decreasing as k increases!

**Solution**: WSS starts decreasing slowly

\[ TSS = WSS + BSS \]

\[ \frac{WSS}{TSS} < 0.2 \]}

The elbow point will tell you when further increasing k has no significant influence on your compactness and separation

---

**Rule of Thumb**

0.2

Look for the **elbow** in the plot

---

**TSS = WSS + BSS**

**WSS / TSS < 0.2**

**Fix K**

---

The elbow point will tell you when further increasing k has no significant influence on your compactness and separation

---

**K-Means – Choosing k**

**Goal**: to find k that minimises WSS

**Problem**: WSS keeps decreasing as k increases!

**Solution**: WSS starts decreasing slowly

\[ TSS = WSS + BSS \]

\[ \frac{WSS}{TSS} < 0.2 \]}

The elbow point will tell you when further increasing k has no significant influence on your compactness and separation

---

**Rule of Thumb**

0.2

Look for the **elbow** in the plot

---

**TSS = WSS + BSS**

**WSS / TSS < 0.2**

**Fix K**

---

The elbow point will tell you when further increasing k has no significant influence on your compactness and separation
Why not use classification accuracy?

- **accuracy** = fraction of items correctly classified
- Here, not interested in what a person does not like
- Rather, how quickly can we discover the relatively few liked items?
A performance metric for recommender systems

• How many liked items were recommended?
  \( \text{RECALL} = \frac{\text{#liked and shown}}{\text{#liked}} \)

• How many recommended items were liked?
  \( \text{PRECISION} = \frac{\text{#liked and shown}}{\text{#shown}} \)

In the context of recommending issues to developers:

liked \( \rightarrow \) the set of issues already done by a developer

shown \( \rightarrow \) the set of recommended issues

Optimal recommender

\( \text{PRECISION} = 1 \)

\( \text{RECALL} = 1 \)
**Precision recall curve**

**Input:** A specific recommender system

**Output:** Algorithm-specific precision-recall curve

To draw curve, vary threshold on # items recommended

For each setting, calculate the precision and recall

Which algorithm is best?

- Largest AUC
- For a given precision, want recall as large as possible
Recommender system example with content-based filtering
Recommender system example with content-based filtering

Incorrect message bus is used at runtime

<table>
<thead>
<tr>
<th>Details</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Type</td>
<td>Bug</td>
</tr>
<tr>
<td>Status</td>
<td>TO DO</td>
</tr>
<tr>
<td>Priority</td>
<td>Major</td>
</tr>
<tr>
<td>Resolution</td>
<td>Unresolved</td>
</tr>
<tr>
<td>Affects Version/s</td>
<td>None</td>
</tr>
<tr>
<td>Fix Version/s</td>
<td>None</td>
</tr>
<tr>
<td>Component/s</td>
<td>None</td>
</tr>
<tr>
<td>Labels</td>
<td>None</td>
</tr>
<tr>
<td>Story Points</td>
<td>3</td>
</tr>
<tr>
<td>Rank (Obsolete)</td>
<td>9223372036954775807</td>
</tr>
</tbody>
</table>

**Description**

For some reason, the message bus is bound to incorrect transport (different from what is set as XD_TRANSPORT) at runtime.

This is from the container log:
```
...
2015-04-21 13:42:35,144 1.2.0.SNAP INFO RedisMessageListenerContainer-4 sink.a2 - test
```

**Type**
- Bug

**Status**
- TO DO

**Priority**
- Major

**Resolution**
- Unresolved

**Affects Version(s)**
- None

**Fix Version/s**
- None

**Component/s**
- None

**Labels**
- None

**Story Points**
- 3

**Assignee**
- Unassigned

**Reporter**
- ilayaperumal gopinathan

**Votes**
- 0

**Watchers**
- 2

**Created**
- 21/Apr/15 1:50 PM

**Updated**
- 31/Aug/15 5:47 PM

**Resolved**
# How much Similarity?

## Issue report XD-3762

**Description**
According to the documentation we can load jars dynamically at module creation time by exploiting the attribute module.classloader in the properties file:

http://docs.spring.io/spring-td/docs/1.3.1.RELEASE/reference/html/module-class-loading

**Type:** Bug  
**Status:** TO DO  
**Priority:** Critical  
**Story Points:** 1

**People**

- **Assignee:** Unassigned  
- **Reporter:** issam

**Dates**

- **Created:** 07/Nov/16 9:10 AM  
- **Updated:** 07/Nov/16 9:12 AM

## Issue report XD-3756

**Description**
I download the spring XD example projects, and run through the steps according the README file for the project. I tried to change the hadoop-site.xml, server.xml and wordcount.xml files but I failed to get it. I am blocked by this issue. Thank you very much in advance for help. Best Regards.

**Type:** Story  
**Status:** TO DO  
**Priority:** Blocker  
**Story Points:** 3

**People**

- **Assignee:** Unassigned  
- **Reporter:** haihua liang

**Dates**

- **Created:** 23/May/16 9:11 AM  
- **Updated:** 24/May/16 8:32 AM

---

Sim(Issue$_i$, Issue$_j$)
### Similarity Matrix

<table>
<thead>
<tr>
<th></th>
<th>XD-3762</th>
<th>XD-3756</th>
<th>XD-2473</th>
<th>XD-2477</th>
<th>XD-2344</th>
<th>XD-2211</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XD-3756</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XD-2473</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XD-2477</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XD-2344</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XD-2211</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Similarity Measure

Issue report **XD-3762**
Type: bug
Status: TO DO
Priority: Critical
Story Points: 1
...

Issue report **XD-3756**
Type: Story
Status: TO DO
Priority: Blocker
Story Points: 3
...

\[ Sim(Issue_i, Issue_j) = \omega_1 f(A_{1i}, A_{1j}) + \omega_2 f(A_{2i}, A_{2j}) + \ldots + \omega_n f(A_{ni}, A_{nj}) \]

- \( n \): total number of issues
- \( \omega_k \): the weight for the k-feature
- \( A_k \): the k-feature
- \( f \): a function to compare two feature values.
  
  This must consider the type of the data (nominal, ratio, ordinal, etc.)
# Similarity Matrix

<table>
<thead>
<tr>
<th></th>
<th>XD-3762</th>
<th>XD-3756</th>
<th>XD-2473</th>
<th>XD-2477</th>
<th>XD-2344</th>
<th>XD-2211</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td></td>
<td>0.25</td>
<td>0.577</td>
<td>0.866</td>
<td>0.289</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-3756</td>
<td>0.25</td>
<td></td>
<td>0.577</td>
<td>0</td>
<td>0.577</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-2473</td>
<td>0.577</td>
<td>0.577</td>
<td></td>
<td>0.333</td>
<td>0</td>
<td>0.333</td>
</tr>
<tr>
<td>XD-2477</td>
<td>0.866</td>
<td>0</td>
<td>0.333</td>
<td></td>
<td>0.333</td>
<td>0.333</td>
</tr>
<tr>
<td>XD-2344</td>
<td>0.289</td>
<td>0.577</td>
<td>0</td>
<td>0.333</td>
<td></td>
<td>0.667</td>
</tr>
<tr>
<td>XD-2211</td>
<td>0.577</td>
<td>0.577</td>
<td>0.333</td>
<td>0.333</td>
<td>0.667</td>
<td></td>
</tr>
</tbody>
</table>
## Similar issues

<table>
<thead>
<tr>
<th></th>
<th>XD-3762</th>
<th>XD-3756</th>
<th>XD-2473</th>
<th>XD-2477</th>
<th>XD-2344</th>
<th>XD-2211</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td>0.25</td>
<td></td>
<td>0.577</td>
<td>0.866</td>
<td>0.289</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-3756</td>
<td>0.25</td>
<td>0.577</td>
<td>0.866</td>
<td>0.289</td>
<td>0.577</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-2473</td>
<td>0.577</td>
<td>0.577</td>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
</tr>
<tr>
<td>XD-2477</td>
<td>0.866</td>
<td>0</td>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
<td>0.333</td>
</tr>
<tr>
<td>XD-2344</td>
<td>0.289</td>
<td>0.577</td>
<td>0</td>
<td>0.333</td>
<td>0.333</td>
<td>0.667</td>
</tr>
<tr>
<td>XD-2211</td>
<td>0.577</td>
<td>0.577</td>
<td>0.333</td>
<td>0.333</td>
<td>0.667</td>
<td></td>
</tr>
</tbody>
</table>
Similarity Sorting

Most similar issues to XD-3762 (one row of the matrix)

<table>
<thead>
<tr>
<th></th>
<th>XD-3762</th>
<th>XD-3756</th>
<th>XD-2473</th>
<th>XD-2477</th>
<th>XD-2344</th>
<th>XD-2211</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td></td>
<td>0.25</td>
<td>0.577</td>
<td>0.866</td>
<td>0.289</td>
<td>0.577</td>
</tr>
</tbody>
</table>

Most similar issues to XD-3762

<table>
<thead>
<tr>
<th></th>
<th>XD-3762</th>
<th>XD-2477</th>
<th>XD-2473</th>
<th>XD-2211</th>
<th>XD-2344</th>
<th>XD-3756</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td></td>
<td>0.866</td>
<td>0.577</td>
<td>0.577</td>
<td>0.289</td>
<td>0.25</td>
</tr>
</tbody>
</table>

Top-3 similar issues to XD-3762 = \{ XD-2477, XD-2473, XD-2211 \}

Top-4 similar issues to XD-3762 = \{ XD-2477, XD-2473, XD-2211, XD-2344 \}

...
Top-N recommendations

For a particular developer:

For each issue done by a developer

Add the top-n most similar issues to a list $L$

Sort the list $L$ according to their similarity value

Recommend the top-N issues on $L$
### Top-N recommendations example

#### Issues done by developer “D1”

<table>
<thead>
<tr>
<th>Key</th>
<th>Assignee</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td>D1</td>
</tr>
<tr>
<td>XD-2211</td>
<td>D1</td>
</tr>
</tbody>
</table>

#### Issues done by developer “D2”

<table>
<thead>
<tr>
<th>Key</th>
<th>Assignee</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-2473</td>
<td>D2</td>
</tr>
</tbody>
</table>

#### Similar issues

<table>
<thead>
<tr>
<th>Key</th>
<th>Similar to ...</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td>XD-2477</td>
<td>0.866</td>
</tr>
<tr>
<td>XD-2473</td>
<td>XD-2473</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-2211</td>
<td>XD-2344</td>
<td>0.667</td>
</tr>
<tr>
<td>XD-2211</td>
<td>XD-3762</td>
<td>0.577</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Key</th>
<th>Similar to ...</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3756</td>
<td></td>
<td>0.577</td>
</tr>
<tr>
<td>XD-3762</td>
<td></td>
<td>0.577</td>
</tr>
</tbody>
</table>
Top-N recommendations

Issues done by developer “D1”

<table>
<thead>
<tr>
<th>Key</th>
<th>Assignee</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td>D1</td>
</tr>
<tr>
<td>XD-2211</td>
<td>D1</td>
</tr>
</tbody>
</table>

Issues done by developer “D2”

<table>
<thead>
<tr>
<th>Key</th>
<th>Assignee</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-2473</td>
<td>D2</td>
</tr>
</tbody>
</table>

Similar issues

<table>
<thead>
<tr>
<th>Key</th>
<th>Similar to ...</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td>XD-2477</td>
<td>0.866</td>
</tr>
<tr>
<td>XD-3762</td>
<td>XD-2473</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-2211</td>
<td>XD-2344</td>
<td>0.667</td>
</tr>
<tr>
<td>XD-2211</td>
<td>XD-3762</td>
<td>0.577</td>
</tr>
</tbody>
</table>

Get the top-N recommendations

Key Similar to ... Similarity
XD-3762 XD-2477 0.866
XD-3762 XD-2473 0.577
XD-2211 XD-2344 0.667
XD-2211 XD-3762 0.577

Similar issues

<table>
<thead>
<tr>
<th>Key</th>
<th>Similar to ...</th>
<th>Similarity</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-2473</td>
<td>XD-3762</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-2473</td>
<td>XD-3756</td>
<td>0.577</td>
</tr>
</tbody>
</table>

Get the top-N recommendations
## Top-N recommendations

The result of a similarity measure

<table>
<thead>
<tr>
<th>Similar to ...</th>
<th>Rank value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-2477</td>
<td>0.866</td>
</tr>
<tr>
<td>XD-2473</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-2344</td>
<td>0.667</td>
</tr>
<tr>
<td>XD-3762</td>
<td>0.577</td>
</tr>
</tbody>
</table>

Top-2 recommendations for developer “D1”

<table>
<thead>
<tr>
<th>Similar to ...</th>
<th>Rank value</th>
</tr>
</thead>
<tbody>
<tr>
<td>XD-3762</td>
<td>0.577</td>
</tr>
<tr>
<td>XD-3756</td>
<td>0.577</td>
</tr>
</tbody>
</table>

Top-2 recommendations for developer “D2”
Precision and Recall – An example

In the context of recommending issues to developers:

**liked** → the set of issues already done by a developer

**shown** → the set of recommended issues

**Example:**

Suppose that developer D1 has already done \{ I1, I2, I3 \}

The top-2 recommendations for D1 are: \{ I8, I3 \}

liked = \{ I1, I2, I3 \}

#liked = 3

shown = \{ I8, I3 \}

#shown = 2
Precision and Recall – An example

In the context of recommending issues to developers:

\( \#( \text{liked and shown} ) \rightarrow \) the number of elements in the intersection between the "liked" and the "shown" set

\[
\text{liked} = \{ I_1, I_2, I_3 \}
\]

\(\#\text{liked} = 3\)

\[
\text{shown} = \{ I_8, I_3 \}
\]

\(\#\text{shown} = 2\)

\[
( \text{liked and shown} ) = \{ I_1, I_2, I_3 \} \cap \{ I_8, I_3 \} = \{ I_3 \}
\]

\(\#( \text{liked and shown} ) = 1\)
Precision and Recall – An example

liked = { I1, I2, I3 }

#liked = 3

shown = { I8, I3 }

#shown = 2

( liked and shown ) = { I1, I2, I3 } ∩ { I8, I3 } = { I3 }

#( liked and shown ) = 1

recall = #( liked and shown ) / #( liked ) = 1 / 3 = 0.33

precision = #( liked and shown ) / #( shown ) = 1 / 2 = 0.5

The overall precision/recall of the system is given by the average of those metrics along all the developers in the system.