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
  • Web scraping
  • Using RESTful services
  • Using mirrors of data
Findings from Github

• How have research papers mined GitHub?
• A meta-analysis of 93 research papers

Findings from Github

- How have research papers mined GitHub?
- A meta-analysis of 93 research papers

22.6% of the sample

Findings from Github

• How data was collected?

Findings from Github

• Classification of the limitations reported

39.8% of them did not report any issue on this respect

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).

KD2: data transformation and mining

- **Transformation.**
  It consists of identifying and implementing any data transformations that are required. This may involve the use of various **binning** and **aggregation** methods to be used in transforming the data and **reducing dimensionality of the data**.

- **Data mining.**
  This stage consists of selecting the appropriate **data mining algorithms** to the problem and selected data set. These algorithms then search for patterns that may exist in the data. Appropriate **parameter settings** will need to be determined to ensure the optimal operation of the algorithms.

Applications of Software Analytics

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

Predicting Story Points from User Stories
Agile Context

Themes, Epics, Stories, Tasks

Theme
Increase Website Traffic

Epic
Add new Video Section

Epic
Improve Login Page Usability

User Story
As a User, I would like the validation on
the login page to be very clear so that I
can easily see when/if I make a mistake
when I log in

Task

Task

Task

Task

© 2009 Agile101.net

© Ezequiel Scott 2018

LTAT.05.008 / Software Analytics / © Ezequiel Scott 2018
Release planning based on Story Points
Planning Poker
Predicting Story Points

• Decision Trees
• Naïve Bayes
• Support Vector Machines

Figure 1: Estimation flow chart

Training data

Good and Bad Classifiers

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

**Bad:**
1. insufficient data
2. training error too high
3. classifier too complex
Support Vector Machines

- Given *linearly separable* data
Support Vector Machines

• Given **linearly separable** data
• **Margin** = distance to separating the hyperplane
• Choose hyperplane that maximizes min margin
• Intuitively:
  • Want to separate + from − as much as possible
  • Margin = measure of confidence
What If Not Linearly Separable?

• (1) penalize each point by distance from margin 1
What If Not Linearly Separable?

• (1) penalize each point by distance from margin 1
• (2) map into higher dimensional space in which data becomes linearly separable

\[ \mathbf{x} = (x_1, x_2) \mapsto \Phi(\mathbf{x}) = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2) \]
What If Not Linearly Separable?

\[ \mathbf{x} = (x_1, x_2) \mapsto \Phi(\mathbf{x}) = (1, x_1, x_2, x_1 x_2, x_1^2, x_2^2) \]

- **statistical problem:** amount of data needed often proportional to number of dimensions
- **computational problem:** very expensive in time and memory to work in high dimensions

Solution: Choosing **Kernels**
Recommending issues to developers
How to assign user stories?

- assign issues based on current workload
- assign issues in a round-robin fashion to a group
- copy assignee from a related issue
- copy from another field
- look back in the issue history at previous assignees or users who have commented
- ...

LTAT.05.008 / Software Analytics / © Ezequiel Scott 2018
Recommender Systems
Recommender Systems

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

LTAT.05.008 / Software Analytics / © Ezequiel Scott 2018
How? Problem formulation

Let $I^{m \times q}$ be a matrix that represents all the issues-related info

$$
I^{m \times q} = \begin{pmatrix}
I_{\text{Issue}_1} & 1 & 1 & \ldots & 1 \\
I_{\text{Issue}_2} & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
I_{\text{Issue}_m} & 0 & 0 & \ldots & 0
\end{pmatrix}
$$

Let $D^{n \times z}$ be a matrix that represents the developers

$$
D^{n \times z} = \begin{pmatrix}
D_{\text{Dev}_1} & 1 & 1 & \ldots & 1 \\
D_{\text{Dev}_2} & 0 & 1 & \ldots & 0 \\
\vdots & \vdots & \ddots & \ddots & \vdots \\
D_{\text{Dev}_n} & 0 & 0 & \ldots & 0
\end{pmatrix}
$$
How? Problem formulation (cont.)

Let define a matrix of completed issues $C^{m \times n}$, where each element in the matrix $c_{ij}$ represents whether the $i$-issue has been completed by the $j$-developer.

$$C^{m \times n} = \begin{pmatrix}
Issue_1 & Dev_1 & Dev_2 & \ldots & Dev_n \\
Issue_2 & 1 & 1 & \ldots & 1 \\
\vdots & 0 & 1 & \ldots & 0 \\
\vdots & \ldots & \ldots & \ldots & \ldots \\
Issue_m & 0 & 0 & \ldots & 0
\end{pmatrix}$$

$$c_{ij} = \begin{cases}
1 & \text{if the i-task was completed by the j-developer} \\
0 & \text{otherwise}
\end{cases}$$
How? Problem formulation (cont.)

• Get another matrix $A^{n\times q}$ given by the product of the transpose of $C$, $[C^T]^{n\times m}$, and $I^{m\times q}$

• This matrix represents the developer-enhanced transactions regarding the tasks they completed

\[
[C^T]^{n\times m} = \begin{bmatrix}
1 & 0 & 0 & 1 & 0 \\
0 & 1 & 0 & 0 & 1 \\
0 & 0 & 1 & 0 & 0 \\
\end{bmatrix}

I^{m\times q} = \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\end{bmatrix}

A^{n\times q} = \begin{bmatrix}
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
0 & 0 & 1 & 1 & 0 & 0 & 1 & 1 \\
\end{bmatrix}
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.
Clustering

- Clustering is based on **Measure of Similarity**: $d(\ldots, \ldots)$
  - Numerical values $\rightarrow$ Metrics: Euclidean, Manhattan, ...
  - Categorical values $\rightarrow$ Construct your own distance

- Clustering methods:
  - K-means
  - Hierarchical
  - ...

Cluster Analysis – Centroids

• It is common to represent each cluster through their **centroid** or mean vector

• Each centroid dimension value provides a measure of its **significance** in the cluster
Compacteness 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 compacteness ↔ 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 ↔ maximise
K-Means – Choosing k

- **Goal**: to find $k$ that minimises WSS
- **Problem**: WSS keeps decreasing as $k$ increases!
- **Solution**: WSS starts decreasing slowly

$$TSS = WSS + BSS$$

$WSS / TSS < 0.2$

Fix $K$

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

Look for the **elbow** in the plot
Developer profiles

- Issue features in the centroid can be sorted according to these weights and lower weight features can be filtered out.
- The resulting set of features-weight pairs can be viewed as an aggregate developer profile representing the interests or behavior of a significant group of developers.

### Cluster 0
- Dev 1: 0 0 1 1 0 0
- Dev 4: 0 0 1 1 0 0
- Dev 7: 0 0 1 1 0 0

### Cluster 1
- Dev 0: 1 1 0 0 0 1
- Dev 3: 1 1 0 0 0 1
- Dev 6: 1 1 0 0 0 1
- Dev 9: 0 1 1 0 0 1

### Cluster 2
- Dev 2: 1 0 0 1 1 0
- Dev 5: 1 0 0 1 1 0
- Dev 8: 1 0 1 1 1 0

**Aggregated Profile of Cluster 1**

<table>
<thead>
<tr>
<th>Weight</th>
<th>Feature</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.00</td>
<td>B</td>
</tr>
<tr>
<td>1.00</td>
<td>F</td>
</tr>
<tr>
<td>0.75</td>
<td>A</td>
</tr>
<tr>
<td>0.25</td>
<td>C</td>
</tr>
</tbody>
</table>
Exploring the GitHub social network
Social network analysis

Graphs: Basic concepts

- **Graph**: a way of representing the relationships among a collection of objects
  - The objects are called **nodes**
  - The links between objects are called **edges**

- **Degree** of a node is the number of edges ending at the node
- For a directed graph, the **in-degree** and **out-degree** of a node refer to numbers of edges incoming to or outgoing from the node
**Weighted Graphs**

- **Weighted graph**: numeric value is associated with each edge

- Edge **weights** may represent a concept such as similarity, distance, or connection cost

---

**Undirected weighted graph**

- conor@derl.org ← 5 → anne@ucd.ie
- mark@yahoo.ie ← 2 → conor@derl.org
- mark@yahoo.ie ← 4 → maria@gmail.com
- maria@gmail.com ← 3 → mark@yahoo.ie

**Directed weighted graph**

- conor@derl.org ← 2 → anne@ucd.ie
- anne@ucd.ie ← 2 → conor@derl.org
- maria@gmail.com ← 4 → mark@yahoo.ie
- mark@yahoo.ie ← 1 → maria@gmail.com
Graphs: Example

Figure 12: Component-Developer relationship

Figure 13: A snapshot of developer to developer network

Figure 16: A snapshot of component to component network

Ego Networks

• **Ego-centric** methods really focus on the individual, rather than on network as a whole
Community Detection

• A variety of definitions of community/cluster/module exist:
  • A group of nodes which share common properties and/or play a similar role within the graph [Fortunato, 2010].
  • A subset of nodes within which the node-node connections are dense, and the edges to nodes in other communities are less dense [Girvan & Newman, 2002].

[Image: Diagram illustrating community detection with multiple clusters]

[Image: [Girvan & Newman, 2002]]
Community Detection - Example

- The size of a node represents its in-degree.
- Different communities represent different groups of developers who focus on different kinds of projects.
- There is a leader in each community.


Figure 6: Community structures in the follow-network of 2012-08 subset.
LTAT.05.008:
Software Analytics
Practice Session

Ezequiel Scott
ezequiel.scott@ut.ee
Practice Session

The KDD process
Context

Data Mining + Software Engineering

The Scrum Framework:

INPUTS FROM CUSTOMERS, TEAM, MANAGERS & EXECs.

PRODUCT OWNER

THE TEAM

1-4 week SPRINT

SPRINT MASTER

DAILY STAND UP MEETING

SPRINT REVIEW

FINISHED WORK

SPRINT RETROSPECTIVE

SPRINT BACKLOG

SPRINT PLANNING MEETING

TASK BREAKOUT

Sprint end date and team deliverable do not change
## JIRA Dataset

<table>
<thead>
<tr>
<th>Team</th>
<th>User Name</th>
<th>User Story</th>
<th>Story Points</th>
<th>Status</th>
<th>Created</th>
<th>Resolved</th>
</tr>
</thead>
<tbody>
<tr>
<td>SpringXD</td>
<td>User1</td>
<td>As a user, I want...</td>
<td>5</td>
<td>TO DO</td>
<td>2016-03-31T22:35:55.000+0000</td>
<td>2016-03-31T22:35:55.000+0000</td>
</tr>
<tr>
<td>SpringXD</td>
<td>User2</td>
<td>As a manager, I want...</td>
<td>3</td>
<td>DOING</td>
<td>2016-03-14T18:09:51.000+0000</td>
<td>2016-03-14T18:09:51.000+0000</td>
</tr>
<tr>
<td>SpringXD</td>
<td>User1</td>
<td>As an admin, I want...</td>
<td>2</td>
<td>DONE</td>
<td>2016-03-03T13:22:14.000+0000</td>
<td>2016-03-03T18:41:19.000+0000</td>
</tr>
<tr>
<td>SpringXD</td>
<td>User3</td>
<td>As a user, I want...</td>
<td>1</td>
<td>DONE</td>
<td>2016-02-29T10:00:18.000+0000</td>
<td>2016-03-13T10:24:15.000+0000</td>
</tr>
</tbody>
</table>

...
Interesting Problems

I. Estimating story points from User Stories
II. Recommending task to developers
III. Exploring the GitHub social network
IV. Sentiment analysis from commits
Problem (I)

Estimating **Story Points** from User Stories

\[ \langle \text{User, Description, Type, ... , Estimates} \rangle \]

Predictive models

Given some user story \( \rightarrow \) Estimate (Story Points)

“Recommend an estimate to junior (?) developers”
Problem (II)

Recommending task to developers

<table>
<thead>
<tr>
<th>Task 1</th>
<th>Task 2</th>
<th>Task 3</th>
<th>Task 4</th>
<th>Task 5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev 1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dev 2</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dev 3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

Tasks’ features

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Task 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Task 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Task 3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Task 4</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Task 5</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

=}

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dev 1</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dev 2</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>Dev 3</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>
Problem (III)

Exploring the GitHub social network
Problem (IV)

Sentiment analysis from commits

Sentiment Analysis of Commit Comments in GitHub: An Empirical Study

Emitz Guzman, David Azócar, Yang Li
Technische Universität München
Faculty of Informatics
Garching, Germany
emitza.guzman@mytum.de, dazocar@gmail.com, liya@in.tum.de