A Theory of Aspects as Latent Topics

Pierre Baldi, Cristina Lopes, Erik Linstead, Sushil Bajracharya
Donald Bren School of Information and Computer Science
University of California, Irvine
{pfbaldi,lopes,elinstea,sbajrach}@ics.uci.edu
Overview

- Motivation

- Aspects as Latent Topics
  - Machine Learning for Concern Extraction
    - Latent Dirichlet Allocation
  - Data
    - Sourcerer
  - Vocabulary Selection

- Results
  - Scattering and Tangling in the Large
  - Scattering and Tangling in the Small

- Conclusions
Motivation

- AOP is still a controversial idea
- Hypotheses put forth by AOP have yet to be validated on the very large scale
  - Cross-cutting concerns exist and are subject to scattering and tangling
  - Excessive scattering and tangling are “bad” for software
  - Alternative composition mechanisms (e.g., AspectJ) alleviate problems caused by cross-cutting concerns
- Advances in machine learning provide the necessary tools for such a validation
- Here we focus on empirical validation of first hypothesis
- Contributions
  - Unsupervised learning of cross-cutting concerns
  - An information-theoretic definition for scattering and tangling
  - Empirical validation across multiple scales
Learning Cross-Cutting Concerns

- Availability of Open-Source software facilitates large-scale empirical analysis of many software facets

- Recent advances in statistical text mining techniques offer new opportunities to mine Internet-scale software repositories
  - Unsupervised
  - Probabilistic
  - Proven to give better results than “traditional” methods
  - Scalable
Statistical Topic Models

- Statistical Topic Models represent documents as probability distributions over words and topics
  - Benefits of working in probabilistic framework
  - Robust – model documents directly
  - Finding patterns is intuitive and easily automated

- Active research area yielding exciting results
  - Traditional Text
  - Source Code (Linstead et al. ASE 2007, NIPS 2007)
Latent Dirichlet Allocation (LDA)

- Blei, Ng, Jordan (2003)
- Simple “Bag of Words” approach
- Models documents as mixtures of topics (multinomial)
- Topics are distributions over words (multinomial)
- Bayesian (Symmetric Dirichlet priors)
- Well analyzed in literature
Documents as “Bags of Words”

```java
public class TextMiner {
    private List trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor() {
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
LDA – In a nutshell

- Given a document-word matrix
  - Probabilistically determine X most likely topics
  - For each topic determine Y most likely words
  - Do it without human intervention
    - Humans do not supply hints for topic list
    - Humans do not tune algorithm on the fly
    - No need for iterative refinement

- Output
  - Document-Topic Matrix
  - Topic-Word Matrix
Aspects as Latent Topics

- Unification of “topics” in text with “concerns” in software
  - A CONCERN IS A LATENT TOPIC
- Syntax and convention differentiates natural and programming languages, but:
  - At most basic level a source file is still a document
  - Tokens in source code still define a vocabulary
- Probability distributions of topics over files and files over topics allow for precise measurement of scattering and tangling, respectively
Measuring Scattering

- If the distribution of a topic, \( t \), across modules \( m_0, \ldots, m_n \) is given by \( p^t = (p^t_0, \ldots, p^t_n) \) then scattering can be measured by the entropy

\[
H(p^t) = -\sum_k p^t_k \log(p^t_k)
\]

- Can normalize by dividing by \( \log(n) \)
  - \( H(p^t) = 0 \) denotes a concern assigned to only one source file
  - \( H(p^t) = 1 \) denotes a concern uniformly distributed across source files

- **AN ASPECT IS A LATENT TOPIC WITH HIGH SCATTERING ENTROPY**

<table>
<thead>
<tr>
<th>d1</th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>t4</th>
</tr>
</thead>
<tbody>
<tr>
<td>d2</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>d3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>d4</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>d5</td>
<td>15</td>
<td>0</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>

\( d_n \)
Measuring Tangling

- If the distribution of a module, $m$, across concerns $t_0...t_n$ is given by $q^m = (q^m_0...q^m_r)$ then scattering can be measured by the entropy
  \[
  H(q^m) = -\sum K q^m_k \log(q^m_k)
  \]

- Can normalize by dividing by $\log(r)$
  - $H(q^m) = 0$ denotes a file assigned to only one concern
  - $H(q^m) = 1$ denotes a file uniformly distributed across concerns

<table>
<thead>
<tr>
<th></th>
<th>t1</th>
<th>t2</th>
<th>t3</th>
<th>tn</th>
</tr>
</thead>
<tbody>
<tr>
<td>d1</td>
<td>0</td>
<td>0</td>
<td>8</td>
<td>0</td>
</tr>
<tr>
<td>d2</td>
<td>1</td>
<td>0</td>
<td>8</td>
<td>5</td>
</tr>
<tr>
<td>d3</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>d4</td>
<td>3</td>
<td>0</td>
<td>8</td>
<td>1</td>
</tr>
<tr>
<td>d5</td>
<td>15</td>
<td>0</td>
<td>8</td>
<td>2</td>
</tr>
<tr>
<td>dn</td>
<td>12</td>
<td>0</td>
<td>8</td>
<td>4</td>
</tr>
</tbody>
</table>
Data

○ We validate our technique at multiple scales
  ● Internet-Scale
    ○ 4,632 open source projects constituting 38 million LOC, 366k files, and 426k classes
    ○ Leverage Sourcerer infrastructure
  ● Individual Projects
    ○ JHotDraw
    ○ PDFBox
    ○ Jikes
    ○ JNode
    ○ CoffeeMud
Sourcerer

- UCI ICS project designed to:
  - Index publicly available source and provide fast search and mining
  - Leverage data to better understand code, facilitate reuse, provide tools for real-world software development
  - Explore new avenues for mining software

- Current Version
  - ~12k open source projects (4,632 with source code)
  - Focused on Java language as proof of concept

- Publicly Available
  - http://sourcerer.ics.uci.edu
Sourcerer Architecture
Vocabulary Selection

- Vocabulary size affects interpretability of topics extracted by LDA
  - Code as plain text yields noisy results

```java
public class TextMiner {
    private List trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor(){
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
Vocabulary Selection

- Vocabulary size affects interpretability of topics extracted by LDA
  - Code as plain text yields noisy results

```java
public class TextMiner {
    private List trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor(){
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
Vocabulary Selection

- Vocabulary size affects interpretability of topics extracted by LDA
  - Code as plain text yields noisy results

```java
public class TextMiner {
    private List trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor() {
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
Vocabulary Selection

- Vocabulary size affects interpretability of topics extracted by LDA
  - Code as plain text yields noisy results

```java
public class TextMiner {
    private List<String> trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor(){
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
Vocabulary Selection

- Vocabulary size affects interpretability of topics extracted by LDA
  - Code as plain text yields noisy results

```java
public class TextMiner {
    private List trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor(){
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
Vocabulary Selection

- Vocabulary size affects interpretability of topics extracted by LDA
  - Code as plain text yields noisy results

```java
public class TextMiner {
    private List trainCollection;
    private Matrix bagOfWords;
    public void nearestNeighbor() {
        ...
        bagOfWords.calcCosineDistance();
        ...
        Random r = new Random();
    }
}
```
Scattering in the Large

- Many prototypical examples for AOP
- Cross-cutting found at multiple magnitudes

<table>
<thead>
<tr>
<th>Concern</th>
<th>Extraced Topic</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>String Processing</td>
<td>‘string case length width substring’</td>
<td>.801</td>
</tr>
<tr>
<td>Exception Handling</td>
<td>‘throwable trace stack print method’</td>
<td>.791</td>
</tr>
<tr>
<td>Concurrency</td>
<td>‘thread run start stop wait’</td>
<td>.767</td>
</tr>
<tr>
<td>XML</td>
<td>‘element document attribute schema child’</td>
<td>.749</td>
</tr>
<tr>
<td>Authentication</td>
<td>‘user group role application permission’</td>
<td>.745</td>
</tr>
<tr>
<td>Web</td>
<td>‘request servlet http response session’</td>
<td>.723</td>
</tr>
<tr>
<td>Database</td>
<td>‘sql object fields persistence jdbc’</td>
<td>.677</td>
</tr>
<tr>
<td>Plotting</td>
<td>‘category range domain axis paint’</td>
<td>.641</td>
</tr>
</tbody>
</table>
Scattering Visualization

Scattering for Full Repository

Normalized Entropy vs. Number of Topics

- Entropy decreases as the number of topics increases.
- The graph shows a clear trend of entropy reduction.
- The curve stabilizes around a certain number of topics before a sharp decrease.
Scattering in the Small: JHotDraw

<table>
<thead>
<tr>
<th>Topic</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>'instance test tear down vault'</td>
<td>0.813075061</td>
</tr>
<tr>
<td>'create factory collections map from'</td>
<td>0.722463637</td>
</tr>
<tr>
<td>'point move box index start'</td>
<td>0.71436202</td>
</tr>
<tr>
<td>'storable read write input output'</td>
<td>0.650160953</td>
</tr>
<tr>
<td>'list next has iterator add'</td>
<td>0.638290561</td>
</tr>
<tr>
<td>'polygon point internal chop count'</td>
<td>0.46080295</td>
</tr>
<tr>
<td>'size selected frame frames dimension'</td>
<td>0.43364049</td>
</tr>
<tr>
<td>'shape geom rectangular rectangle2 hashtable'</td>
<td>0.353301264</td>
</tr>
<tr>
<td>'drag drop target source listener'</td>
<td>0.352124151</td>
</tr>
<tr>
<td>'event component size transform mouse'</td>
<td>0.338653373</td>
</tr>
</tbody>
</table>

- Notable appearance of project-specific concerns
  - In general appear to have lower scattering entropy
  - Can be controlled in part by number of topics extracted by LDA
  - In specific cases may require developer expertise to determine valid concerns versus noise
Scattering in the Small: Jikes

<table>
<thead>
<tr>
<th>Topic</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>'next has element enumeration elements'</td>
<td>0.699351996</td>
</tr>
<tr>
<td>'buffer check empty char insert'</td>
<td>0.661522459</td>
</tr>
<tr>
<td>'print stream println writer total'</td>
<td>0.636898546</td>
</tr>
<tr>
<td>'hash map iterator next add'</td>
<td>0.636035451</td>
</tr>
<tr>
<td>'type array reference code resolved'</td>
<td>0.635043332</td>
</tr>
<tr>
<td>'cycles end time right begin'</td>
<td>0.486326254</td>
</tr>
<tr>
<td>'field type reflect value unchecked'</td>
<td>0.4684958</td>
</tr>
<tr>
<td>'short switch reference type read'</td>
<td>0.447104842</td>
</tr>
<tr>
<td>'sys lock unlock write socket'</td>
<td>0.428127362</td>
</tr>
<tr>
<td>'offset mask fits forward code'</td>
<td>0.346995542</td>
</tr>
<tr>
<td>'emit assembler gen reference laddr'</td>
<td>0.266546555</td>
</tr>
</tbody>
</table>
Scattering in the Small: JNode

<table>
<thead>
<tr>
<th>Topic</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>'string length append substring tokenizer'</td>
<td>0.76224123</td>
</tr>
<tr>
<td>'map hash equals object value'</td>
<td>0.726874809</td>
</tr>
<tr>
<td>'byte array bytes arraycopy system'</td>
<td>0.723141514</td>
</tr>
<tr>
<td>'stream write output writer array'</td>
<td>0.723069203</td>
</tr>
<tr>
<td>'input read stream reader buffered'</td>
<td>0.718017023</td>
</tr>
<tr>
<td>'graphics color paint icon rectangle'</td>
<td>0.567084036</td>
</tr>
<tr>
<td>'image raster buffered create writable'</td>
<td>0.548839911</td>
</tr>
<tr>
<td>'time date calendar zone simple'</td>
<td>0.525970475</td>
</tr>
<tr>
<td>'zip entry jar plugin deflater'</td>
<td>0.515858882</td>
</tr>
<tr>
<td>'focus event window component listener'</td>
<td>0.502999404</td>
</tr>
</tbody>
</table>
Scattering in the Small: CoffeeMud

<table>
<thead>
<tr>
<th>Topic</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>'environmental mob msg location send'</td>
<td>0.861222835</td>
</tr>
<tr>
<td>'environmental name text vector string'</td>
<td>0.823602707</td>
</tr>
<tr>
<td>'vector element size add remove'</td>
<td>0.795135882</td>
</tr>
<tr>
<td>'mob hash environmental iterator next'</td>
<td>0.77667159</td>
</tr>
<tr>
<td>'string mob currency environmental shop'</td>
<td>0.600152681</td>
</tr>
<tr>
<td>'string channel ime send mud'</td>
<td>0.591218453</td>
</tr>
<tr>
<td>'string vector from xml buffer'</td>
<td>0.586218656</td>
</tr>
<tr>
<td>'string mob gen scr tell'</td>
<td>0.390775366</td>
</tr>
</tbody>
</table>
Scattering Visualization

Scattering Curves for Selected Projects

- JHotDraw
- PDFBox
- Jikes
- JNode
- CoffeeMud

Number of Topics

Normalized Entropy
Tangling in the Large

Full matrix available from supplementary materials page
- 366,287 x 125
- 72MB (compressed)
## Tangling in the Small

<table>
<thead>
<tr>
<th>File</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>BouncingDrawing.java</td>
<td>0.6650</td>
</tr>
<tr>
<td>SingleFigureEnumeratorTest.java</td>
<td>0.6538</td>
</tr>
<tr>
<td>URLTool.java</td>
<td>0.6449</td>
</tr>
<tr>
<td>UndoRedoActivity.java</td>
<td>0.1000</td>
</tr>
<tr>
<td>CommandCheckBoxMenuItem.java</td>
<td>0.0892</td>
</tr>
<tr>
<td>JHotDrawException.java</td>
<td>0.0831</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>File</th>
<th>Entropy</th>
</tr>
</thead>
<tbody>
<tr>
<td>DebuggerThread.java</td>
<td>0.6932</td>
</tr>
<tr>
<td>TraceBuffer.java</td>
<td>0.6845</td>
</tr>
<tr>
<td>VM_Process.java</td>
<td>0.6736</td>
</tr>
<tr>
<td>VM_Listener.java</td>
<td>0.0693</td>
</tr>
<tr>
<td>PPC_Disassembler.java</td>
<td>0.0554</td>
</tr>
<tr>
<td>VM_Constants.java</td>
<td>0.0</td>
</tr>
</tbody>
</table>
Tangling Visualization

Tangling Curves for Selected Projects

Tangling Curves for 2 Large Projects
A Parametric Model of Tangling?

- Inverse sigmoidal behavior noted in tangling
- Fit simple 2 parameter model to data
  \[ f(x) = a \times \ln\left(\frac{1}{x} - 1\right) + b \]
  - R-Square of .947
  - Standard deviation of .024
Comparison to Other Methods

- Validation for Internet-scale repository challenging
- Individual projects exist which make good baselines
- JHotDraw
  - Compared to fan-in/fan-out, identifier analysis, dynamic analysis, manual analysis, and mining code revisions
    - What aspects are identified?
    - To what degree are scattering and tangling observed?
  - General agreement with our LDA-based technique in all cases
Conclusions

- Statistical machine learning techniques make additional progress in Aspect Mining
- LDA effectively extracts concerns from arbitrarily large repositories
  - Unsupervised
  - No pre-conceived notion of what an Aspect is
  - A Concern is a latent topic in source code
- Statistical techniques allow for precise measurement of scattering and tangling using information theory
  - An Aspect is a concern with high scattering entropy
- Significant agreement with other aspect mining methods
BACKUP
Current/Future Work

- Validate Second AOP Hypothesis
  - Are scattering and tangling *truly* "bad" for real-world software?
- Apply LDA to Software Evolution
  - Concern trends over release histories