Software is Data Too: How Should We Deal With It?

Andrian Marcus

SEVERE Group @

DEPARTMENT OF COMPUTER SCIENCE
Michigan = Great Lakes State
Wayne State University
Wayne State University
Computer Science @ WSU
The SEVERE Group

- **SoftwarE Visualization and Evolution REsearch Group**
- Three faculty
  - Václav Rajlich
  - Marwan Abi-Antoun
  - Andi Marcus
- Six Ph.D. students
- Five M.S. students
- Always looking for new Ph.D. students
Main Research Interests

- Software evolution
- Software architecture
- Agile software development
- Program comprehension
- Cognitive models of software development
- Applications of data mining in software engineering
Long Term Research Question

- What is the necessary and sufficient amount of data, information, and knowledge needed to solve a particular software engineering task by the developer?
  - **Necessary** - easier (yet not easy)
  - **Sufficient** - very difficult - higher payoff
  - Ideally, answer this question automatically given a software system and a task
  - The answer does not need to tell how to solve the task. This remains a human decision.
Similar Problem in SE

- How much testing is *necessary* and *sufficient*?
  - Researchers and industry working on this for decades - progress is slow
A View of Software Evolution

- Software evolution = comprehension + learning + other things

- Data and information is converted into knowledge through comprehension and learning

- Not all data (of all format and size) can be comprehended and learned
Data vs. Information vs. Knowledge

- Data = low level - numbers and tokens

- Information = data + semantics (meaning)

- Knowledge = information + learning by relation to existing knowledge

- Learning results in the increase or decrease of knowledge
Information in Software

- **Structural information** - the structural aspects of the source code (e.g., control and data flow)

- **Dynamic information** - behavioral aspects of the program (e.g., execution traces)

- **Lexical information** - captures (part of) the problem domain and developer intentions (e.g., identifiers, comments, documentation, etc.)

- **Process information** - Evolutionary data, history of changes (e.g., CVS logs, bug reports, etc.)
Why Analyze the Textual Information?

- Software = text, structure, behavior
- Text -> *what* is the software doing?
- Structure + behavior -> *how* is the software doing it?
- We need all three for complete code view and comprehension
- Text is the common form of information representation among various software artifacts at different abstraction levels
How to Analyze the Text in Software?

- Natural Language Processing (NLP)
- WordNet and Ontologies
- Information/Text Retrieval (IR/TR)
- Combinations of the above
What Is Information Retrieval?

• The process of actively seeking out information relevant to a topic of interest (van Rijsbergen)
  - Typically it refers to the automatic (rather than manual) retrieval of documents
    • Information Retrieval System (IRS)
  - “Document” is the generic term for an information holder (book, chapter, article, webpage, class body, method, requirement page, etc.)
Information Retrieval System (IRS)

• An IRS is capable of storage, retrieval, and maintenance of information (e.g., text, images, audio, video, and other multi-media objects)

• Difference from DBMS
  - used on unstructured information
  - indexing mechanism used to define “keys”
What Do We Want From an IRS?

- Systemic approach
  - Goal (for a known information need):
    - Return as many relevant documents as possible and as few non-relevant documents as possible

- Cognitive approach
  - Goal (in an interactive information-seeking environment, with a given IRS):
    - Support the user’s exploration of the problem domain and the task completion.
Advantages of TR

- Text Retrieval (TR) = retrieval of textual data - most common applications are the internet search engines
- No predefined grammar and vocabulary
- Some techniques able to infer word relationships without a thesaurus
- Robust with respect to data distribution and type
A Typical TR Application

- Build corpus (i.e., collection of documents)
- Index corpus
1. Formulate a query (Q)
   - Can be done by the user or automatically
2. Compute similarities between Q and the documents in the corpus
3. Rank the documents based on the similarities
4. Return the top N as the result
5. Inspect the results
6. GO TO 1. if needed or STOP
How to Use TR with Software?

1. Formulate the SE task as a retrieval problem
   - Concept location, traceability link recovery, bug triage, etc.

2. Use document-document similarity measures to solve the SE task
   - Coupling and cohesion measurement, impact analysis, defect prediction, clone detection, etc.
Data, Information, Knowledge

- **Data**
  - terms (lexical tokens) from software

- **Information**
  - meaning of terms
  - relationships between terms,
  - relationships between terms and documents
  - relationships between documents

- **Knowledge**
  - using the information to solve an SE task
  - learning which relationships are accidental and which are intentional
Semantic Relationships

• Documents in software
  - class, method, function, e-mail, bug description, test case, UML diagram, requirement, etc.

• Relationships are used to rank results in a retrieval task - query-document relationship

• Methods/classes relate to each other semantically -> conceptual coupling

• Conceptual coupling -> cohesion - special case of discourse coherence
SE Tasks

• Concept location in software - formulated as document retrieval problem
• Traceability links recovery between software artifacts - document retrieval/clustering
• Defect prediction - coupling + learning
• Change prediction (i.e., impact analysis) - using cohesion
• Bug triage - document clustering + learning
• Bug severity prediction - learning
TR-based Concept Location [WCRE’04]

1. Creating a corpus of a software system
2. Indexing the corpus with the TR method
   (we used LSI, Lucene, GDS, LDA)
3. Formulating a query
4. Ranking methods
5. Examining results
6. Go to 3 if needed
Tim and I

Tim Menzies cca. 2007
Tim and I
Tim and I

- Basal
- LumA
- LumB
- ERBB2
- Normal
- Unclassified

- Group A
- Group B

ER−
ER+

b) ELMAP2D
c) PCA2D
What Are We Doing?

- **Software artifacts**
- **Raw data**
- **Data features**
- **Analysis data**
- **Information presentation**
- **Visualization**

- Data extraction, cleaning, transformations, etc.
- Data analysis, feature selection
- Pattern detection, clustering, learning, etc.
- Feedback

---

- Data extraction, cleaning, transformations, etc.
- Data analysis, feature selection
- Pattern detection, clustering, learning, etc.
TR Based Concept Location

- **Source code**
  - Data extraction, cleaning, transformations, etc.

- **Corpus**
  - Data analysis, feature selection

- **IR engine index**
  - Pattern detection, clustering, learning, etc.

- **Similarity measures**
  - Ranked list of docs.

- **Feedback**

- **Visualization**
Evrika!

• We are both doing data analysis and mining
• More than that, others are doing the same thing
• This is different than traditional software analysis
• What is it like and how can we make it better?
Who Are the Others?

• Mining Software Repositories (MSR)
• Predictive Models in Software Engineering (PROMISE)
• Search Based Software Engineering (SBSE)
• Recommendation Systems in Software Engineering (RSSE)
• Search-driven development: Users, Infrastructure, Tools and Evaluation (SUITE)
Where Have We Seen This Before?

• **Bioinformatics** = the application of statistics and computer science to the field of molecular biology

• The primary goal of bioinformatics is to increase the understanding of biological processes

• What sets it apart from other approaches is its focus on developing and applying computationally intensive techniques (e.g., pattern recognition, data mining, machine learning algorithms, and visualization) to achieve this goal [source Wikipedia]
Are We Doing Software Informatics?

- **Softinformatics** = the application of statistics and computer science to the field of software engineering
- The primary goal of softinformatics is to increase the understanding of software engineering processes
- What sets it apart from other approaches is its focus on developing and applying computationally intensive techniques (e.g., pattern recognition, data mining, machine learning algorithms, and visualization) to achieve this goal
How Is It Different Than Bioinformatics?

• We have unique challenges:
  - Heterogeneous and correlated data -> difficult data integration problem - no standard representations
  - Data correlations are mostly implicit and sometimes accidental -> hard to explain results and infer causality (see Max’s talk tomorrow)
  - Missing and dirty data -> human in the loop
  - Lack of generality -> learning is limited
  - Lack of specific algorithms -> must borrow from other fields
  - Poor empirical practices -> hard to results and replicate experiments
  - We are analyzing man made data!
What Can We Learn From Bioinformatics?

- It can be done!
- It takes time and huge effort/$$$ - we need to train our researchers in new areas
- We need standard data representations
- We need specific solutions - there are no shortcuts
- We need to improve our empirical practices - share data!
Research Questions 1

• Which software engineering tasks can benefit from data analysis and data mining support?
• What data should we analyze and mine?
• How to integrate heterogeneous data?
• What algorithms should we use for analysis and information extraction?
• How to customize these algorithms?
Research Questions 2

• Can we use these algorithms across systems?
• How to store and share analysis data?
• How can we integrate data analysis and data mining into existing software engineering processes and curriculum?
• How does it fit with software evolution?
Why Is This Hard?

• Example questions:
  - What algorithms should we use for analysis and information extraction?
  - How to customize these algorithms?

• Example SE task:
  - TR based concept location using relevance feedback
TR-based Concept Location [WCRE’04]

1. Creating a corpus of a software system
2. Indexing the corpus with the TR method
   (we used LSI, Lucene, GDS, LDA)
3. Formulating a query
4. Ranking methods
5. Examining results
6. Go to 3 if needed [relevance feedback]
JFace Text Editor Leaves a Black Rectangle on Content Assist text insertion. Inserting a selected completion proposal from the context information popup causes a black rectangle to appear on top of the display.

1. `createContextInfoPopup()` in `org.eclipse.jface.text.contentassist.ContextInformationPopup`

2. `configure()` in `org.eclipse.jdt.internal.debug.ui.JDIContentAssistPreference`

3. `showContextProposals()` in `org.eclipse.jface.text.contentassist.ContextInformationPopup`

New Query
Time to Make Decisions - building the corpus -

- What granularity should the corpus have? That is, what should be a document in the corpus: a class, a file, or a method? → at least 3 choices.
- Should we include comments or not in the corpus? → 2 choices (yes/no).
- Should we stem the terms in the corpus or not? → 2 choices (yes/no).
- If so, which stemmer to use? → at least 2 choices.
- Should we filter out commonly used terms (i.e., stop-words) → 2 choices (yes/no).
- Should we split identifiers or not? → 2 choices (yes/no).
- Should we keep the original form of the identifier? → 2 choices (yes/no)
Time to Make Decisions
- indexing -

• Which IR method to use for indexing: a vector space model (VSM), Lucene, Latent Semantic Indexing (LSI), Latent Dirichlet Allocation (LDA), etc. → at least 5 choices.

• These IR methods have their own parameters. For example, in LSI is important to establish the dimensionality reduction factor, LDA has a user defined number of topics, and so on. → in average each IR method has at least 3 parameters with many possible values for each.

• What similarity measure should we use between two documents? → at least 2 choices.
Time to Make Decisions  
- relevance feedback -

• Should we include relevance feedback in the process? Relevance feedback allows for the automatic reformulation of queries. → 2 choices (yes/no).

• If so, which relevance feedback algorithm to use? → at least 3 choices.

• Relevance feedback methods have several parameters each. → at least 4 parameters with many possible values each.
Rocchio Equation

\[ Q_1 = \alpha \ Q_0 + \frac{\beta}{n_1} \sum_{i=1}^{n_1} R_i - \frac{\gamma}{n_2} \sum_{i=1}^{n_2} S_i \]

where

\( Q_0 \) = the vector for the initial query
\( R_i \) = the vector for the relevant document \( i \)
\( S_i \) = the vector for the non-relevant document \( i \)
\( n_1 \) = the number of relevant documents chosen
\( n_2 \) = the number of non-relevant documents chosen
\( \alpha, \beta \) and \( \gamma \) tune the importance of relevant and non-relevant terms
(in some studies best to set \( \beta \) to 0.75 and \( \gamma \) to 0.25)
Time to Make Decisions
- clustering -

- Should we cluster the corpus or the results? → 3 choices.
- If so, what clustering method to use? → at least 3 choices.
- Most clustering methods use some parameters (e.g., think of the k from k-Means). → at least 1 parameter with several possible values.
How to Make These Decisions?

• Imagine trying all the possible combinations - impossible!
• Imagine trying 10% of the combinations - impractical for a single group

• We need shared data, replication, community effort, active learning from repeated experiments
A Paradigm Shift

• A new law of software evolution: *the size of software data increases with time*

• Software engineering is no longer just development and process management - it must includes information and knowledge management
How to Do It?

- Tom Zimmermann proposes a new position of software project analyst - management oriented view [ICSM 2010 keynote panel]

- We need a developer oriented view as well
  - New position does not really work
  - Should developers learn analytics?
  - We need new research that produces tools and methodologies, then train developers to use them
News Alert

• All ICSM 2010 Best papers for each track used some data analysis, data mining, or information retrieval to solve a particular task (see Max Di Penta here)

• One of the keynote panelists talked about Software Analytics (Tom Zimmermann)

• The Future of Software Engineering Research workshop at FSE 2010 will feature at least two papers on the topic
Acknowledgments

• People:
  - **WSU**: Denys Poshyvanyk, Vaclav Rajlich, Andrey Sergeyev, Dapeng Liu, Maksym Petrenko, Yubo Dong, Sonia Haiduc
  - **Other places**: Giuliano Antoniol, Jonathan Maletic, Yann Gael-Gueheneuc, Tim Menzies, Greg Gay, SEMERU Group, Rocco Oliveto, Gabriele Bavota, Andrea De Lucia, Paolo Tonella, Surafel Lemma

• Funding:
  - CCF-0438970
  - CCF-0820133
  - CCF-1017263
  - 2005 Eclipse Innovation Award
  - 1R01HG003491-01A1
Summary

• Too much data in software - we need specialized data analysis and mining techniques to deal with it
• Potentially rich research area - so far more questions than answers
• We need to integrate data analysis and mining into software evolution practices, research, and training
• We need to change how we do empirical work
Why Is All This Important?

- Think how much data was generated in a large active project in the time this talk was prepared and delivered
- Could any one of us deal with it as is?
- We need the assistance of automated data extraction and analysis tools
- Data mining, information retrieval, machine learning, and statistical analysis applied to software engineering data should become a mainstream research area in our field
Your Turn for Questions