Leveraging Natural Language Analysis of Software: Achievements, Challenges, and Opportunities

Lori L. Pollock
Computer and Information Sciences
University of Delaware
Software is like a car.

It breaks.
Software is like a car.

We want it to go faster.
Software is like a car.

We want more features.
Software is like a car.

It is increasingly complex under the hood.
Software is like a car.

It now requires specialized tools to maintain.
SE community to the rescue
Power Tools

User Interface
- Retrieve User Requests
- Report Results

Automation
- Preprocessing: Build Models
- Gather Information
- Analyze Data

Software Artifacts
- Source Code
- Documentation
- Developer Communication
- Design Documents
- Test Suites
- Code Changes
- ...

Retrie...
Example: Code Search Tool

User Interface:
- Retrieve User Query Words
- Report Ranked List of Related Code Units

Automation:
- Preprocessing: Build/Index Corpus
- Compute Query/Code Similarities
- Rank Code Units

Software Artifacts:
- Method
- Method
- Method
- Method
- ...
Example: Method Summarization Tool

User Interface

Retrieve M’s signature body

Report M’s Summary as phrases

Preprocessing: Build Software Word Usage Model

Select Important statements

Generate Summary phrases

Automation

Software Artifacts

Statement

Statement

Statement

Statement

Statement

...
SE Power Tools Revisited

User Interface
- Retrieve User Requests
- Report Results

Automation
- Preprocessing: Build Models
- Gather Information
- Analyze Data

Software Artifacts
- Source Code
- Documentation
- Developer Communication
- Design Documents
- Test Suites
- Code Changes
...
Power Tools: What Information is used?

**Structural:**
- Control Flow
- Data Flow
- Call Graphs

**Program Dependence**

- **Constants**
- **Types**
- **Inheritance**

**Dynamic:**
- Frequency/order of execution
- Development Process-related:
  - Change logs, bug reports
What else is available?
Consider this code

```java
public static int a(int c, int d) {
    int b;
    b = c * d;
    return b;
}

public static int c(int w, int h) {
    int a;
    a = w * h;
    return a;
}

public static int computeArea(int width, int height) {
    int area;
    area = width * height;
    return area;
}
```

Given a width and height, compute & return the area of a rectangle, OBVIOUSLY.
class Player{

/**
 * Play a specified file with specified time interval
 */

public static boolean play(final File file, final float fPosition, final long length) {

    fCurrent = file;
    try {
        playerImpl = null;
        // make sure to stop non-fading players
        stop(false);
        // Choose the player
        Class cPlayer = file.getTrack().getType().getPlayerImpl();

    }
}
How can we leverage the naming?

class Player{
public static boolean play(final File file, final float fPosition, final long length) {
    fCurrent = file;
    try {
        playerImpl = null;
        stop(false);
        class cPlayer = file.getTrack().getType().getPlayerImpl();
    }
}
Consider Dora the Program Explorer

Natural Language Query
- Maintenance request
- Expert knowledge
- Query expansion

Program Structure
- Representation
- Current: call graph
- Seed starting point

Dora

Relevant Neighborhood
- Subgraph relevant to query

* Dora comes from exploradora, the Spanish word for a female explorer. Hill et al.
Program: JBidWatcher, eBay auction sniping program

Bug: When a user triggers an add auction, nothing happens – there is no effect.

SE Task: Locate code related to ‘add auction’ trigger

Seed: DoAction() method, from prior knowledge
Using only structural information

Query: ‘add auction’

- **DoAction()** has 38 callees, only 2/38 are relevant
  - Locates locally relevant items, but many irrelevant

And what if you wanted to explore more than one edge away?

Irrelevant Methods
Using only lexical information

- 50/1812 methods contain matches to 'add*auction' regular expression query

- Only 2/50 are relevant

Locates globally relevant items, but many irrelevant
Combining structural & lexical

- **Structural**: guide exploration from seed

- **Naming**: prune irrelevant edges

```java
private void DoAdd(Component src) {
    String prompt = "Enter the auction number to add";
    String endResult = prompt.substring(prompt.indexOf("Adding"));
    // ...
    MQFactory.getConcrete("user")
        .enqueue(ADD_AUCTION + endResult);
}
```

```java
private void DoPasteFromClipboard() {
    String auctionId = getClipboardString();
    // ...
    MQFactory.getConcrete("user")
        .enqueue(ADD_AUCTION + auctionId);
}
```
Text Analysis in SE

Challenges

Achievements

Opportunities
So, what is Text Analysis?

**Why?**

To provide important information for building automated and semi-automated recommendation systems and analysis tools to support SE tasks.

*analysis of the natural language used by programmers in writing software (source code + other software artifacts)*
Flavors of Text Analysis

Information/Text Retrieval (IR/TR)

Given query words, retrieve documents containing unstructured data related to those topics:
* For a known information need, return as many relevant docs as possible
* To enable the user to explore a problem domain

Natural Language Processing (NLP)

Software that will automatically analyze, understand, and generate languages that humans use naturally (e.g., English)
* To know what concepts a word or phrase represents
* To know how to link those concepts together in a meaningful way
Natural Language in Comments: Different Types (by content)

- Descriptive /* show save dialog and get file name */
- Notes /* TODO: fix this! */
- Cross-reference /* @see setData */
- Explanatory /* we clone the vector to avoid deadlock */
- And other types ....
Natural Language in Descriptive Comments: Conventions

// Play a specified file with specified time interval

/* Registers the text to display in a tool tip. The text displays when the
* cursor lingers over the component.
* @param text the string to display. If the text is null, the tool tip is
* turned off for this component. */

• Not a full sentence
• Multiline -> later, full sentences with period
• 1st line: Often starts with a verb and then the direct object
• Contain Java doc components
Natural Language in Identifiers: Significance & Studies

Carla, the compiler writer

I don’t care about identifier names.

Pete, the programmer

So, I can use a, b, c since I hate to tpye.

Molly, the maintainer

I guess if you never change projects, get sick, or retire and become a sheep farmer.

Identifiers play a key role in program comprehension and follow conventions.

- Useful for software tools: metrics, traceability, program understanding
- Metaphors, morphology, scope, part of speech hints

• [Caprile & Tonella] [Liblit et al.] [Deissenboeck & Pizka], Lawrie, Binkley et al.] [Host & Oestvold]
Natural Language in Identifiers
Conventions

month average_score medianScore cWord2Num
hostname sizeof SIMPLETYPENAME
jLabel PHP_id cmp
ASTVisitorTree ConvertASCIItoUTF
sortList sortedList

• Single and multiple words (multi-words)
• Camel case and underscores for visible split, but not always
• Abbreviations, sometimes different semantics in different code units
• Conventions based on entity being named
Text Retrieval: Overview

Preprocessing

- Prepare Corpus: Word stemming, ...
- Index Corpus: Vector Space, LSI, ...

User Interface

- Retrieve User Query Words
- Report Ranked List of Related Code Units
- Compute Doc & Query Semantic Similarities
- Rank Code Units

Query Words

... Doc = File, Class, or Method
class Player{
public static boolean play(final File file, final float fPosition, final long length) {
    fCurrent = file;
    try {
        playerImpl = null;
        stop(false);
        class cPlayer = file.getTrack().getType().getPlayerImpl();
        ...
    }
    Prepare Corpus: Remove non-literals/stop words; Split ids; Stem
    Play    play    file    f Position    length
    f Current    file
    play Impl    stop
    c Play    file    get Track    get Type    get Play Impl
**Text Retrieval in SE: Example**

### Index Corpus

<table>
<thead>
<tr>
<th>play</th>
<th>file</th>
<th>f</th>
<th>position</th>
<th>length</th>
<th>current</th>
<th>impl</th>
<th>stop</th>
<th>c</th>
<th>get</th>
<th>track</th>
<th>type</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>3</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>C2</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Process query against indexed corpus -> ranked list of relevant docs
NLP: Overview

High Level NLP Task
- Text Analysis
- Text Summarization
- Text Translation
- Text Generation

Basic NLP Analysis
- Word Stemming
- Part-of-speech Tagging
- Parsing/Chunking
- Word Relation Identification
- Word Disambiguation

SE Task
- SE Tool

Doc = File, Class, or Method
public UserList getUserListFromFile(String path) throws IOException {
    try {
        File tmpFile = new File(path);
        return parseFile(tmpFile);
    }
    catch (java.io.IOException e) {
        throw new IOrException("UserList format issue" + path + " file " + e);
    }
}
NLP in SE:
Generating Phrases by Lexicalization

print(current); /* print current */

But what is current?
/* print current document */

Context implies what ‘current’ is;
→ Type information can provide context

<table>
<thead>
<tr>
<th>Type Name</th>
<th>Variable Name</th>
<th>Generated Phrase</th>
</tr>
</thead>
<tbody>
<tr>
<td>CallFrame</td>
<td>parentFrame</td>
<td>parent call frame</td>
</tr>
</tbody>
</table>

Selectable is an adjective

Frame is not repeated

• **Phrase Generation leverages:**
  – Part of speech of words in type and variable names
  – Overlap between type & variable names
For both TR and NLP in SE: Lexical-level Analysis

- Split Multi-word Identifiers
- Tag words with Part-of-speech
- Expand Abbreviations/Acronyms
- Identify Word Relations (Synonyms, ...)
- Extract & Preprocess Words from Identifiers
- Mine Usage of Words
- Stem Words

Diagram:
- Central node: Extract & Preprocess Words from Identifiers
  - Connections:
    - Split Multi-word Identifiers
    - Tag words with Part-of-speech
    - Expand Abbreviations/Acronyms
    - Identify Word Relations (Synonyms, ...)
    - Mine Usage of Words
    - Stem Words
Splitting Multi-words

**Challenges**

- **Mixed case:** medianScore
- **Same case:** sortedList, notype, textbox
- **Abbreviations:** ASTVisitorNode, cmp

**Current Strategies**

- Standard & customized dictionaries
- Word frequencies in code
- Abbreviation expansion during id splitting

None have conquered the same case problem to high accuracy.
Expanding Abbreviations

Challenges (of nondictionary words)

Prefix (attr, obj, param, i);
Acronyms (ftp)
Combination (println)

Dropped Letter (msg)
Misspelling (instanciation)
No boundary (filesize)

And, the same abbreviation can have different expansions depending on domain or context

Current Strategies

- Manually create table of common short forms in code
- Mine expansions from the code, look nearby first

Instance
Instruction
Instantiate
Install

Control Flow Graph
Context-Free Grammar
Configuration
Configure
Tagging Part of Speech

**Challenges**

void copyMenuItems(Menu)

(noun, base verb)  (noun)  (plural noun)

Boolean copiedItem()

**Current Strategies**

Develop rules based on naming conventions,
entity being named,
context of entity
Solving Vocabulary Mismatch: Identifying Word Relations

What words are similar to “remove”?

Humans: Refine query by adding related words
- Error prone and time consuming

Strategies
Some IR techniques can automatically expand query:
- Digital thesaurus with semantic similarity
- Latent Semantic Indexing and related approaches
Synonyms are not always enough for searching

Query: “money transaction”  Not successful
Query: “bank transaction”  successful
But <money, bank> not synonymous

Other Semantic Similarity Types:
- Hypernyms and Hyponyms
- Meronyms and Holonyms
- Topically related

All these types can be identified by current semantic similarity techniques. (WordNet)
But not always adequate for software.
Lexical-level Analysis Opportunities

- Split Multi-word Identifiers
- Expand Abbreviations/Acronyms
- Stem Words
- Identify Word Relations (Synonyms, ...)
- Tag words with Part-of-Speech
- Extract & Preprocess Words from Identifiers
- Increase Accuracy
Corpus-level Analysis

Perform analysis and generate results

Text generators

Query (re) formulation mechanisms

TR Similarity Models

Software Word Usage Models

Search Engines

Software property analyzers

Corpus analyzers
<table>
<thead>
<tr>
<th>Topic</th>
<th>Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traceability links recovered and maintenance among software engineering artifacts</td>
<td>66</td>
</tr>
<tr>
<td>Concept, feature or concern location and aspect mining in source code</td>
<td>50</td>
</tr>
<tr>
<td>Change impact analysis in source code</td>
<td>8</td>
</tr>
<tr>
<td>Restructuring and refactoring</td>
<td>13</td>
</tr>
<tr>
<td>Software reuse</td>
<td>19</td>
</tr>
<tr>
<td>Architecture/design recovery</td>
<td>4</td>
</tr>
<tr>
<td>Quality assessment and software measurement</td>
<td>21</td>
</tr>
<tr>
<td>Defect Prediction</td>
<td>2</td>
</tr>
<tr>
<td>Recommending developers</td>
<td>4</td>
</tr>
<tr>
<td>Discovery of web services</td>
<td>3</td>
</tr>
<tr>
<td>Licensing</td>
<td>4</td>
</tr>
<tr>
<td>Requirement Analysis/Engineering</td>
<td>9</td>
</tr>
<tr>
<td>Clone detection</td>
<td>1</td>
</tr>
<tr>
<td>Program comprehension general</td>
<td>8</td>
</tr>
<tr>
<td>Bug triage</td>
<td>8</td>
</tr>
<tr>
<td>Software Evolution Analysis</td>
<td>3</td>
</tr>
<tr>
<td>Software Categorization</td>
<td>4</td>
</tr>
<tr>
<td>Domain Analysis/Software Product Lines</td>
<td>1</td>
</tr>
<tr>
<td>Other tasks</td>
<td>3</td>
</tr>
<tr>
<td>Software miniaturization</td>
<td>1</td>
</tr>
</tbody>
</table>
Growth of Text Analysis

Marcus et al.
"Going Forward with Text Analysis"

* **Apply** text analysis to
  * develop new tools and improve tools

* **Combine** information
  * Structure + Text + Dynamic

* **Explore configurations** of analyses

* **Improve Evaluations**
  * Lack of common infrastructure
The Next Five Years of Text Analysis in Software Maintenance
TODAY: 15:35 – 17:35
Belvedere