Combining Probabilistic Ranking and Latent Semantic Indexing for Feature Identification

Denys Poshyvanyk, Yann-Gaël Guéhéneuc, Andrian Marcus,
Giuliano Antoniol, Václav Rajlich

14th IEEE International Conference on
Program Comprehension (**ICPC'06**)

Athens, Greece

Motivation

- Feature/concept identification (location)
- Concept location – identifying parts of the source code implementing domain concepts
- Reduces search space
- Uses static and/or dynamic analysis

Concept Location in Practice

- Static
  - Dependency based search [Rajlich'00]
  - IR methods [Marcus'04]
- Dynamic
  - Execution traces - Reconnaissance [Wilde'92]
  - Scenario based probabilistic ranking [Antoniol'05]
- Combined
  - Profiling with concept analysis [Eisenbarth'03]
  - Feature dependencies [Salah'05]
  - Feature evolution [Greevy'05]

Shortcomings

- Static analysis:
  - sometimes does not identify all entities implementing a specific concept
  - **recall** is impacted
- Dynamic analysis:
  - sometimes unable to distinguish between overlapping features
  - **precision** is impacted

Our Combination

- Static
  - Dependency based search [Rajlich'00]
  - IR methods [Marcus'04]
- Dynamic
  - Execution traces - Reconnaissance [Wilde'92]
  - Scenario based probabilistic ranking [Antoniol'05]
- Combined
  - Profiling with concept analysis [Eisenbarth'03]
  - Feature dependencies [Salah'05]
  - Feature evolution [Greevy'05]

Novel Hybrid Technique

- Feature identification – decision making problem in presence of uncertainty
- Static (LSI) and dynamic (SBP) experts:
  - LSI queries static documents
  - SBP analyzes dynamic traces of execution scenarios
- Complementary results are combined via affine transformation

Scenario Based Probabilistic Ranking

- Building a model of a program architecture
- Identifying a feature of interest
  - Subset of a program architecture (micro-architectures -> variables, classes, functions, methods)
- Comparing features modeled as micro-architectures

SBP - Feature Identification

- Program model creation
  - static analysis, C++, AOL
- Trace collection
  - (ir) relevant scenarios are executed to collect traces
  - processor emulation (VALGRIND) to improve the precision of data collection
- Knowledge-based filtering
- Probabilistic ranking
  - events are re-weighted (Wilde's equation is renormalized)

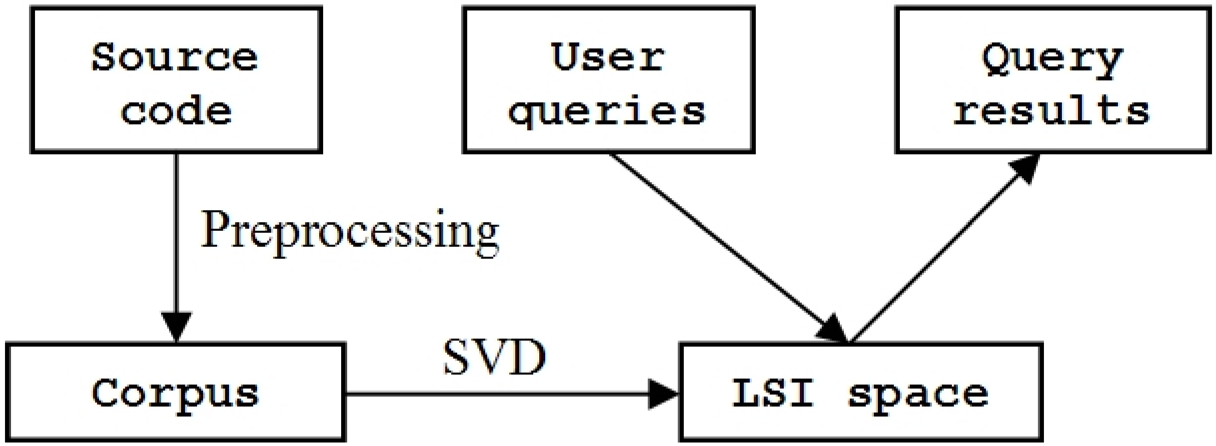
Latent Semantic Indexing

- Vector space model based IR method
  [Dumais'94, Berry'95, Deerwester'90]

- Applied to text retrieval, pattern recognition, natural language understanding

- Known application: Google

Concept Location with LSI



- User defined queries
  - Based on user experience and domain knowledge, little known about querying patterns
- Semi-automated query generation
  - Starts with a user defined query and adds synonyms from the source code, identified by LSI

Combining the Experts

- SBP and LSI – our experts
- SBP – constructing overlapping scenarios
- LSI – formulate a query that captures semantic characteristics of the feature
- Combining judgments of experts :

$$r_{combined}(x) = \lambda r_{sbp}(x) + (1 - \lambda) r_{lsi}(x)$$

Case Study Objectives

- Assess the precision of the novel hybrid technique
- Compare hybrid technique with standalone results for SBP and LSI
- Evaluate the influence of dimensionality reduction factor on the corpus size

Case Study – Mozilla Sizes

- Mozilla v1.6 size related statistics

Item	Count (MLOC)	Item	Count
Header files	8,055 (1.50)	**Classes**	4,853
C files	1,762 (0.90)	**Methods**	53,617
C++ files	4,204 (2.00)	Specializations	5,314
IDL files	2,399 (0.20)	Associations	17,362
XML files	283 (0.12)	Aggregations	6,727
HTML files	2,231 (0.19)		
Java files	56 (0.06)		

First Case Study

- Feature: “Add a bookmark in Mozilla”
- Find the methods and functions, which implement the feature in Mozilla
- Replicated case study to compare with previous results

SBP Results

- Scenario 1: “A user visits an URL, opens Mozilla, clicks on bookmarked URL, loads page and closes Mozilla”
- Scenario 2: “The user acts like in Scenario 1, but once the page is loaded, she saves URL”
- SBP provides 274 methods ranked with probability of 1.0

LSI Results

MLOC	**4.4**
Vocabulary	**85,439**
Number of parsed documents	**68,190**
Number of methods	**48,267**
Number of functions	**19,923**

- LSI Query: *"bookmark newbookmark bookmarkname bookmarkresource bookmarkadddate createbookmark insertbookmarkitem deletebookmark bookmarknode"*

Combined Results

R	300	500	750	1500
1	**CreateB (3)**	**CreateB (6)**	**AddB (1)**	**AddB (1)**
2	**AddB (4)**	**AddB (2)**	**CreateB (14)**	**CreateB (8)**
3	**CreateBC (64)**	Flush	Flush	**CreateBC (19)**
4	InsertResource	**CreateBC (57)**	**CreateBC (36)**	WriteBookmarks
5	ListenToEventQueue	InsertResource	WriteBookmarks	getFolderViaHint
6	Flush	WriteBookmarks	Observe	InsertResource

CreateB – CreateBookmark
AddB – AddBookmarkImmediately
CreateBC - CreateBookmarkInContainer

A Bug / Unwanted Feature

- Bug # 182192 from BugZilla: “quotes (“) are not removed from collected e-mail addresses”
- From: "First Last" <first.last@example.org>
  - First: "First
  - Last: Last“
  - *Difficult to search the address book*

- We use official Bugzilla reports to verify the results: CollectAddress and CollectUnicodeAddress are fixed

SBP Results

- Scenario 1: “A user replies to an e-mail”

- Scenario 2: “A user performs the same action as in Scenario 1 and, using the mouse, the user forces to collect e-mail address of the sender”

- SBP returned 206 methods with score of 1.0

Combined Results

- LSI query: “*collect collected sender recipient email name names address addresses addressbook* ”

Rank	Method name	Rlsi
1	ParseHeadersWithArray	2
2	ParseHeaderAddresses	4
3	***CollectAddress***	***37***
4	OpenInternal	36
5	***CollectUnicodeAddress***	***46***

Discussion

- Combination of SBP and LSI is better than SBP and LSI standalone
- The results tend to improve when increasing the dimensions of LSI space
- The case studies reveal great potential of this combination

Future Work

- Combine with other feature location techniques
  - dependency search
  - clustering
- Determine heuristics to identify the best λ