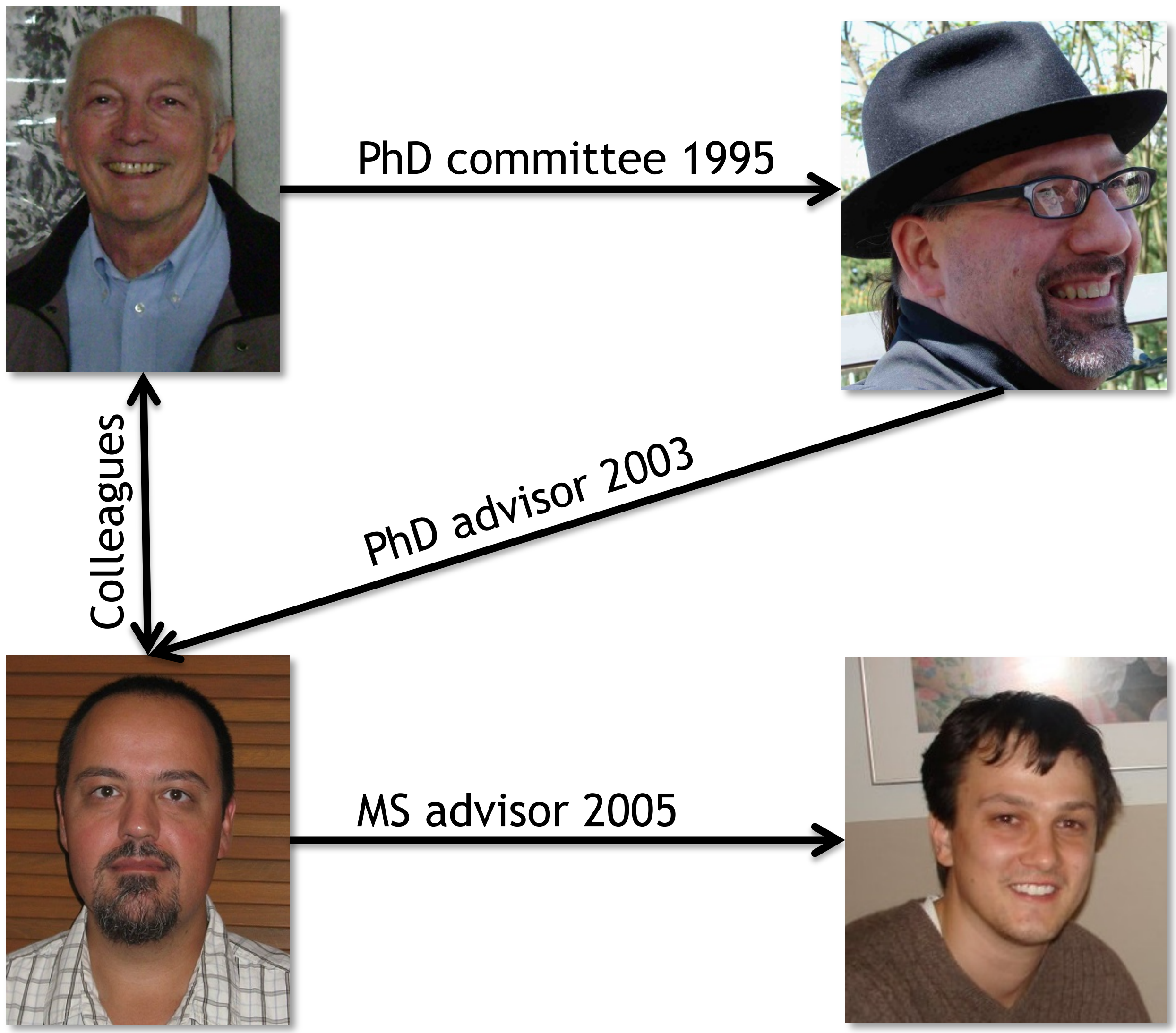
An Information Retrieval Approach to Concept Location in Source Code

WCRE 2004

BEHIND THE PAPER

Andrian Marcus, Andrey Sergeyev, Václav Rajlich, Jonathan Maletic

About the authors



In the beginning …

Used Information Retrieval (LSI) for:

- Reverse engineer ADTs in legacy code [ICSE2001]
- Clone detection [ASE2001]
- Traceability link recovery [ICSE2003]

The birth of an idea

Job interview dinner
February 13, 2003   8:23 pm
Location: Pegasus Taverna, Greektown – Detroit

So what is your research really about?

Ooops … Another ruined interview

Huh?

I do not fully understand LSI. How do you use it?

Well, you could use it to search code you need to change.

Aha! Concept location! Can you do what we did with dependency search?

I think so …

Later that year

Do you have a good project for me?

Yeah. I have the perfect project: LSI + CL.

Guys, I have this brilliant student in my class. He wants to work on the LSI+CL idea. Should we aim to get a paper for IWPC04?

Great. Let's do it!

Of course. Duhhh ...

What should be in the paper?

- Let’s replicate the concept location task done by Vaclav and Kunrong in their IWPC 2000 paper
- They used dependency search
- We will use LSI instead and compare
- We should also use `grep` and show how LSI is different

- We did it and submitted the paper

We regret to inform you …

- "Since LSI has been published in other papers of the authors (ASE 2001, ICSE 2003), I had difficulties in identifying the achievements and impact of the paper."
- "There are far too many references in the paper"
- "The introduction does not clearly state the particular contribution of this paper."
- "The precision and recall results are far from optimal or convincing"

What did we learn?

- “This is a too simple a way of specifying concepts and leads to many false positive and negative matches. A concept is more something like a feature, but at least something with a verb such as *change font to Courier*”

Plan B - FSE 2004

- Better and more substantial reviews
  - Even contained a good idea for future work – to cluster the results

- Distinction between concept location and traceability link recovery

- Description of the empirical part and the results

Plan C - WCRE 2004

How does text retrieval work?

1. Generate a corpus
2. Index the corpus with an IR engine
3. Write a query
4. Rank the documents in the corpus and retrieve the relevant documents
5. Inspect the results
6. If target found then stop

   else go to step 3

How does text retrieval work?

1. Generate a corpus
2. Index the corpus with an IR engine
3. Write a query
4. Rank the documents in the corpus and retrieve the relevant documents
5. Inspect the results
6. If target found then stop

    ~~else go to step 3~~

    else go to step 4

Corpus Generation

- Very simple parsing to extract semantic information (i.e., comments and identifiers)
- split_identifiers & SplitIdentifiers
- Define source code documents with user-defined granularity (e.g., class, methods, functions, declarations, interfaces, etc.)

Query Formulation

- User defined queries
  - Most common, 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

Mosaic Case Study

- Version 2.7 of Mosaic
- Written in C
- Approximately 95,000 LOC
- 269 files
- Generated 2,347 source code documents corresponding to functions and declaration blocks

Comparison with Previous Results

- Assess the LSI based search with other static location methods
- Change request: “Add a new font type in Mosaic - *tiny*”
- Results published in IWPC’00 [Chen’00]
- Concept to locate: *font properties*
- Originally located elements:
  - *mo_set_fonts()*, *mo_get_font_size_from_res()*, *menubar_cb()*, *XtResource resources[]*, *Rdata*, *menuspec*, and *mo_token*

Assessment

- Precision + recall
- Search space reduction coefficient
  - How many elements the user had to visit to locate first (or last) relevant component

User Queries

1. font
2. font size style small regular large
3. font style large small regular family
4. font style bold italics large small regular
5. font size style small regular large family bold italics type
6. font size style small regular large family bold italics
7. font family style bold italics size small regular medium large
8. font size style small regular large family bold italics medium type

Results for User Queries

Q	Investigated documents	Recall	Precision	Last relevant doc. pos.	Step
1	4	0%	0%	89	1
2	54	100%	9.25%	17	4
3	9	60%	33.33%	13	3
4	15	**100%**	**33.33%**	11	4
5	79	100%	6.32%	22	4
6	57	100%	8.77%	14	4
7	49	100%	10.02%	**10**	4
8	72	100%	6.94%	18	4

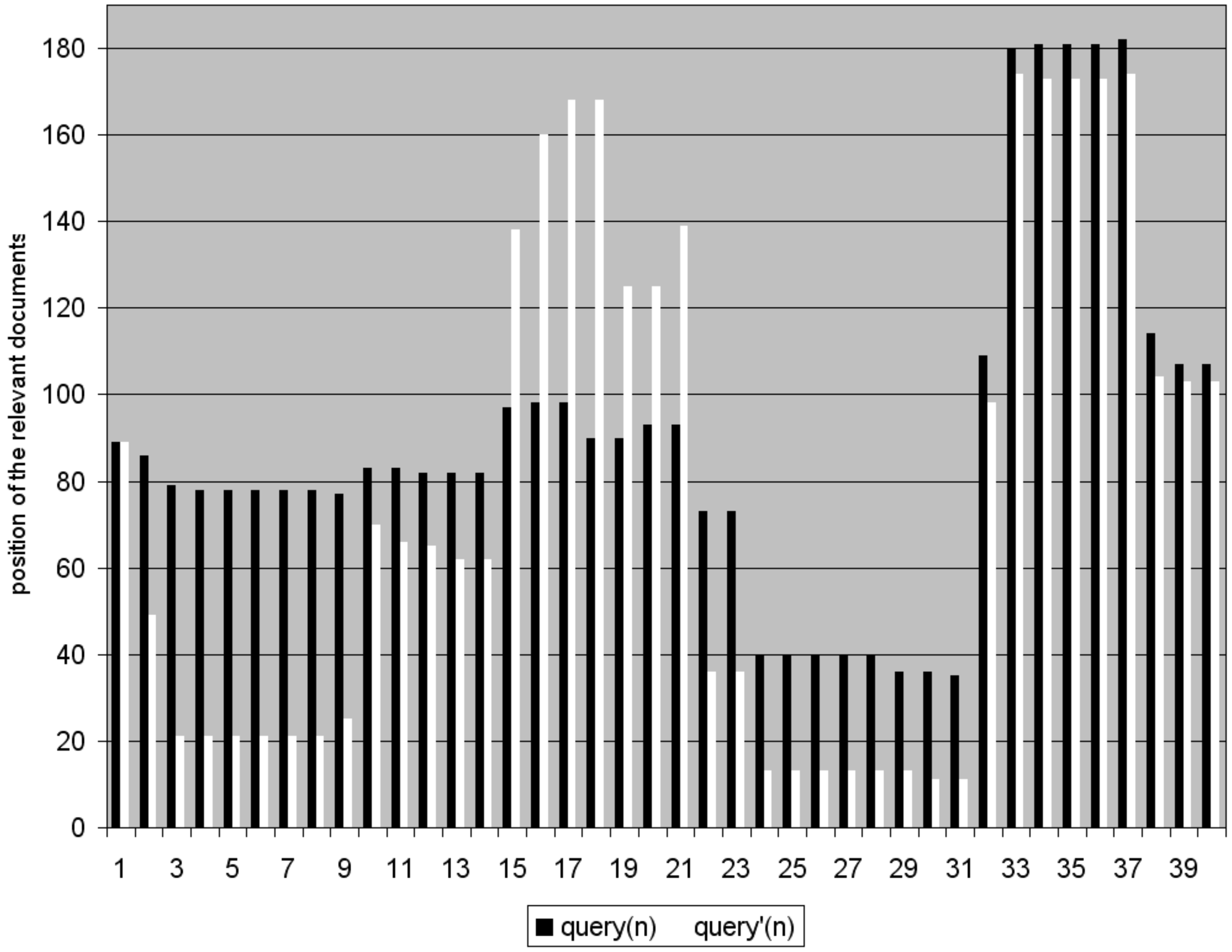
Generated Queries

- Started with the word “font”

- Two types of generated queries
  1. query(n) = “font” + the first n words from the “synonyms list”

  2. query’(n) = the first n words from the “synonyms list”

Results for the Generated Queries



WCRE 2004

Dependency Based Search

- The concept location was used to find the starting point for the impact analysis for the "tiny font" change request
- Due to the granularity level, LSI missed 3 global variables from the impact set
  - *Rdata*, *menuspec*, and *mo_token*
  - These are used in the display feature
- The dependency based search missed *PSFont()* used in postscript generation

Conclusions

- Pros
  - Found elements missed by dependency based search
  - Simple and flexible to use
  - Returns ranked results – advantage over grep
- Cons
  - Missed some data elements (granularity)
  - Depends on the quality of comments and identifiers (grep has the same problem)
- Using LSI (or possibly other IR method) for concept location and source code browsing is promising

Highlights of the paper

- First study on how to formulate queries in IR based code retrieval
  - only in the past 2-3 years this became an active area of research
- Evidence that textual and structural based concept location complement each other
- Compared (to some degree) LSI and `grep`
- Showed that IR-based CL makes sense and revealed where improvements should be made

What did we do after the paper?

- We dropped the way to determine the size of the retrieved set of documents and focused on query reformulation
- Contextualized concept location within the software change process
  - Locate the place in the code where the **first** change will be made. Impact analysis follows.
- Differentiate from traceability link recovery
  - Precision vs. recall
  - Query formulation

What did we do after the paper?

- Changed the way to perform empirical evaluations
- Employed change reenactment and user simulation with repository and bug data
  - Allowed large empirical studies
  - Many papers now are on “bug localization”

Interesting fact about WCRE 2004

- The top three most cited WCRE papers from in the last 10 years are all from WCRE 2004 and are on related topics:
  - An IR approach to concept location in source code (Marcus, Sergeyev, Rajlich, Maletic)
  - Aspect mining through FCA of execution traces (Tonella, Ceccato)
  - Identifying aspects using fan-in analysis (Marin, Van Deursen, Moonen)
- We organized a panel at ICSM 2005 with Tonella, Rajlich, Van Deursen, Koschke to clarify the differences between concepts, features, aspects, and concerns

What else did we do after the paper?

- Used other IR engines
- Re-ranking of results using FCA
- Combine IR with dependency search
- Combine IR with execution traces
- Presentation of results
  - summarization, visualization
- Query reformulation
  - Via relevance feedback
  - Automatic (machine learning)
  - Knowledge representation (partial ontologies)

What did others do after the paper?

On Concept Location:

- Corpus preparation
  - Identifier splitting, etc.
- Combined IR+Dependency+Traces
- Query reformulation
  - Query expansion with synonyms, etc.
- IR engines/indexing
  - LDA, VSM, BM25, term boosting, incremental indexing, etc.
- Extending code retrieval to large repositories

What did others do after the paper?

On other topics using IR:

- Traceability link recovery
- Change impact analysis
- Restructuring and refactoring
- Retrieval for reuse
- Architecture/design recovery
- Bug triage
- Defect prediction

What did others do after the paper?

On other topics using IR:

- Quality measurement (coupling, cohesion)
- Requirements analysis
- Clone detection
- Reverse engineering
- Discovery of web services
- Etc.

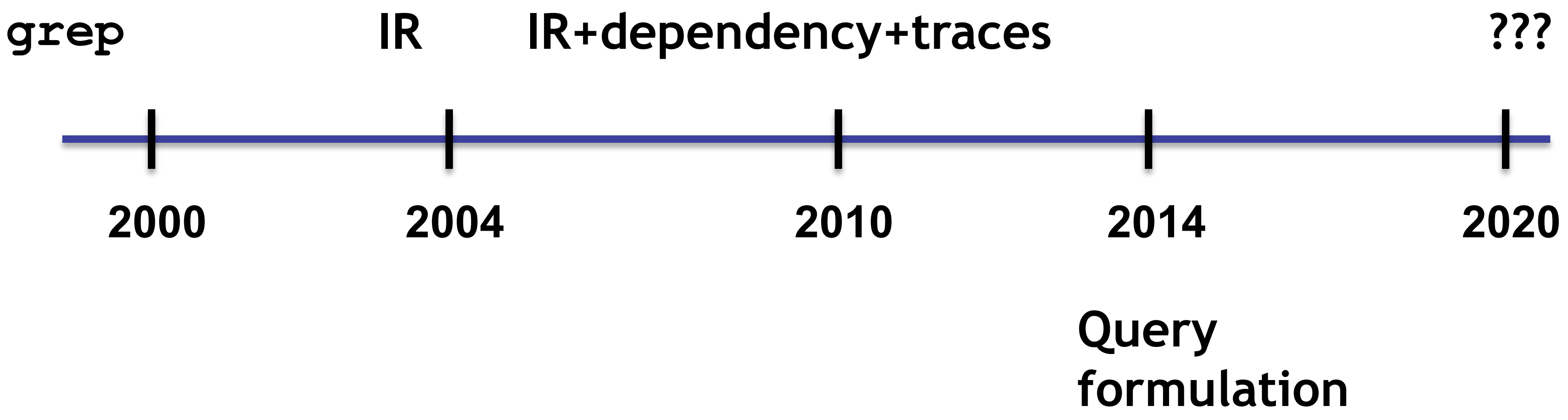
What is left to do?

- Corpus building – mature
  - Work left on comments
- Query (re)formulation – maturing
- (Re)ranking of results – maturing
- Presentation of results – maturing
- Search + navigation – maturing
- IR engine – maturing
- Benchmarks – infancy
- Cognitive issues (e.g., learning) – infancy

What are we doing next?

- Query (re)formulation
- IR engine configuration
- Benchmark
- Cognitive issues

Text-based concept location


grep
IR
IR+dependency+traces
???
2000
2004
2010
2014
2020
Query formulation