Parameterizing and **Assembling** **IR**-based Solutions for **SE Tasks** using Genetic Algorithms

Annibale Panichella, Bogdan Dit, Rocco Oliveto,
Max Di Penta, Denys Poshyvanyk and Andrea De Lucia

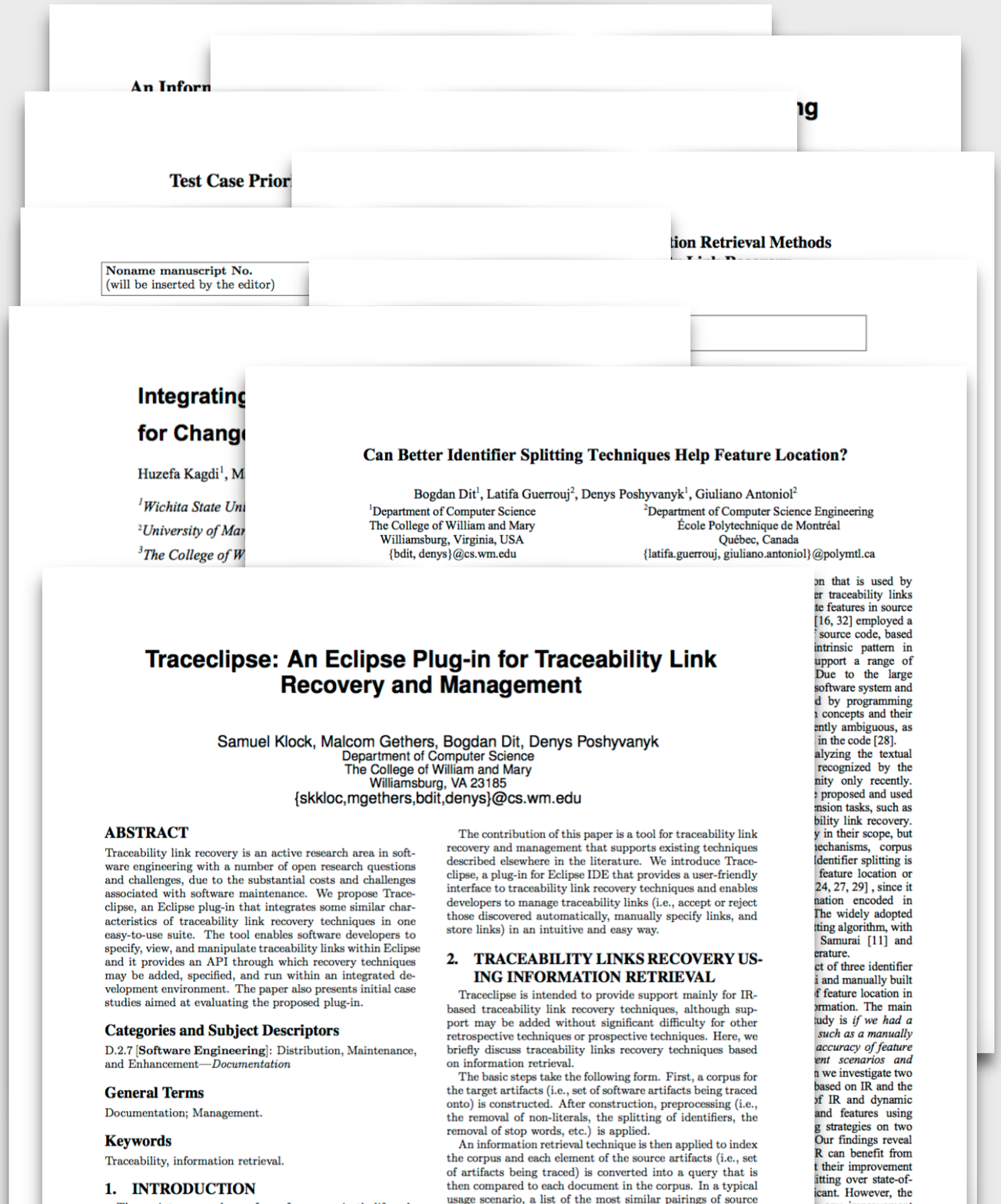
Information Retrieval

600

papers in 10 years

30

applications in S.E.



Information Retrieval

Use Case UC 11.4

Use Case	Insert Laboratory Data
Description	The user inserts the data of a specific laboratory
Events	1. The user opens the Laboratory GUI 2. The user inserts the Laboratory data . . .

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...

public GUILaboratoryData(){
    window = new JFrame();
    insert = new JButton();
    ...
    }

...
}
```

Information Retrieval

Use Case UC 11.4

Use Case	**Insert Laboratory Data**
Description	**The user inserts the data of a specific laboratory**
Events	1. **The user opens the Laboratory GUI** 2. **The user inserts the Laboratory data** . . .

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUILaboratoryData {
private jFrame window;
private jButton insert;
...

public GUILaboratoryData(){
    window = new JFrame();
    insert = new JButton();
    ...
    }

...
}
```

Information Retrieval

Use Case UC 11.4

Use Case	**Insert Laboratory Data**
Description	**The user inserts the data of a specific laboratory**
Events	1. **The user opens the Laboratory GUI** 2. **The user inserts the Laboratory data** . . .

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUI Laboratory Data {
private j Frame window;
private j Button insert;
...

public GUI Laboratory Data(){
    window = new JFrame();
    insert = new JButton();
    ...
    }

...
}
```

Information Retrieval

Use Case UC 11.4

Use Case	**Insert Laboratory Data**
Description	The **user inserts** the **data** of a **specific laboratory**
Events	1. The **user opens** the **Laboratory GUI** 2. The **user inserts** the **Laboratory data** . . .

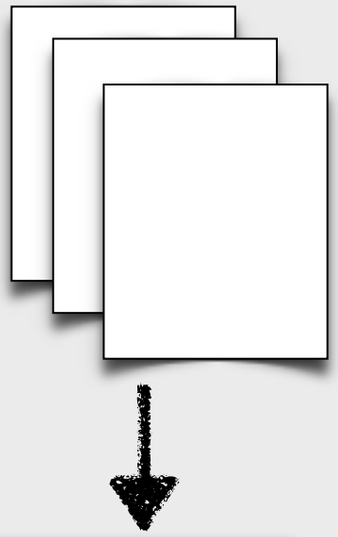
Textual Similarity = 42%

GUILaboratoryData.java

```
/* *This class implements the GUI for
managing laboratories data */

public class GUI Laboratory Data {
private j Frame window;
private j Button insert;
...

public GUI Laboratory Data(){
    window = new JFrame();
    insert = new JButton();
    ...
    }

...
}
```

IR Process

Software Artifacts



Term Extraction

Stop-Word List / Function

Morphological Analysis

Term Weighting

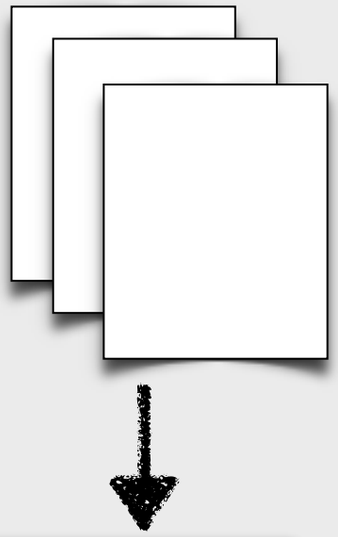
IR Methods

Distance Function

Software Maintenance task:
- Traceability Recovery
- Source code labelling
- Bug duplication
- Feature Location
- …

IR Process

Software Artifacts



Term Extraction

- Special Chars.
- Digits
- White space
- ..

Stop-Word List / Function

- No stop word
- Stop-word function
- Java stop-word list
- English stop-word list
- Italian stop-word list
- …

Morphological Analysis

- No Stemmer
- Porter
- English Snowball
- Italian Snowball
- ….

Term Weighting

- Boolean
- Term Freq
- TF-IDF
- Log(TF+1)
- Entropy

IR Methods

- LSI (k)
- LDA (α, β, n, k)
- VSM
- PLSI
- …

Distance Function

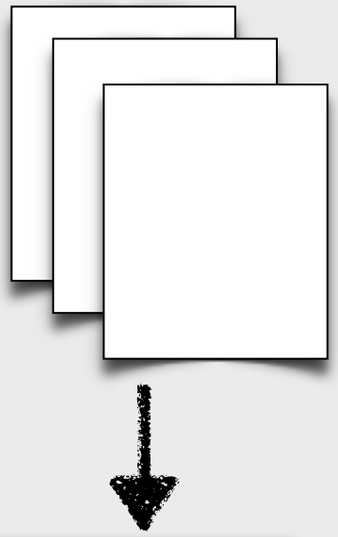
- Cosine Similarity
- Hellinger Distance
- Jaccard Dist.
- Euclidean

Software Maintenance task:

- Traceability Recovery
- Source code labelling
- Bug duplication
- Feature Location
- …

IR Process

Software Artifacts



Term Extraction

- Special Chars.
- Digits
- White space
- ..

Stop-Word List / Function

- No stop word
- Stop-word function
- Java stop-word list
- English stop-word list
- Italian stop-word list
- …

Morphological Analysis

- No Stemmer
- Porter
- English Snowball
- Italian Snowball
- ….

Term Weighting

- Boolean
- Term Freq
- TF-IDF
- Log(TF+1)
- Entropy

IR Methods

- LSI (k)
- LDA (α, β, n, k)
- VSM
- PLSI
- …

Distance Function

- Cosine Similarity
- Hellinger Distance
- Jaccard Dist.
- Euclidean

Which configuration to use?

Software Maintenance task:

- Traceability Recovery
- Source code labelling
- Bug duplication
- Feature Location
- …

What is the “best” IR process?

On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery

Rocco Oliveto*, Malcom Gethers†, Denys Poshyvanyk†, Andrea De Lucia*
*Department of Mathematics and Informatics, University of Salerno, via Ponte don Melillo, Fisciano (SA), Italy
†Computer Science Department, The College of William and Mary, Williamsburg, VA 23185, USA
roliveto@unisa.it, mgethers@cs.wm.edu, denys@cs.wm.edu, adelucia@unisa.it

***Abstract*—Different IR methods have been proposed for recovering traceability links between code and documentation. So far, there is no clear winner among the exploited IR techniques. In this paper we present an empirical study aiming at statistically analyzing the equivalence of several IR-based traceability recovery methods. The analysis is based on Principal Component Analysis and on the analysis of the overlap of the set of candidate links provided by each method. The studied techniques are three widely used IR methods – i.e., the Jensen-Shannon (JS) method, Vector Space Model (VSM), and Latent Semantic Indexing (LSI) – and Latent Dirichlet Allocation (LDA), an IR method previously used to support other software engineering tasks but never used for traceability recovery. The results show that while the three methods previously used for traceability recovery are equivalent, LDA is able to capture a dimension unique to the set of techniques which we considered. Moreover, although the accuracy of LDA is lower than previously used methods, in several cases the combination of LDA with other IR methods improves the traceability recovery accuracy of stand-alone methods.**

***Keywords*-Traceability Recovery; Vector Space Model; Latent Semantic Indexing; Jensen-Shannon method; Latent Dirichlet Allocation; Empirical Studies.**

I. Introduction

Maintaining dependencies (traceability links) between different types of software artifacts is widely recognized as an important support activity both during initial system development and also during the ongoing change management process. In particular, traceability links between the free text documentation associated with the development and maintenance cycle of a software system and its source code is helpful in a number of tasks – such as requirement coverage, program comprehension and impact analysis. Software artifact traceability is also considered as a “best practice” by numerous major software engineering standards (such as CMMI or ISO 15504).

Unfortunately, establishing and maintaining traceability links between software artifacts is a time consuming, error prone, and person-power intensive task. Consequently, despite the advantages that can be gained, effective traceability is rarely established unless there is a regulatory reason for doing so. Extensive effort in the software engineering community (both research and commercial) has been brought forth to improve the explicit connection of documentation and source code. Promising results have been achieved using Information Retrieval (IR) techniques [1], [2] for traceability recovery [3], [4], [5], [6], [7], [8], [9], [10], [11]. IR-based methods propose a list of candidate traceability links on the basis of the similarity between the text contained in the software artifacts. Such methods are based on the conjecture that two artifacts having high textual similarity share several concepts thus they are good candidates to be traced on each other. Several IR methods have been proposed for traceability recovery – e.g., vector space and probabilistic models [1] or Latent Semantic Indexing (LSI) [2]. In general, the retrieval accuracy of IR-based traceability recovery methods is assessed through two measures: recall, measuring the percentage of correct links that were found, and precision, measuring the percentage of found links that were correct. The results achieved are sometimes contrasting and demonstrate no clear winner among IR techniques. Indeed, it seems that all the exploited techniques so far are able to capture the same information when used to calculate the textual similarity between software artifacts.

In this paper we present an empirical study aiming at statistically analyzing the equivalence of different IR-based traceability recovery methods. The comparison is based on Principal Component Analysis (PCA) and on the analysis of the overlap of the set of candidate links provided by each of the IR methods. The studied IR techniques are the Jensen-Shannon (JS) method [3], the Vector Space Model (VSM) [1], LSI [2], and Latent Dirichlet Allocation (LDA) [12]. The first three methods were selected because they are widely used and seem to be the techniques that give the best results [3], [4], [10]. LDA was never used for traceability recovery. However, we also analyze the support given by such a technique during traceability recovery because in a previous study [13] the authors demonstrate that LDA is able to capture some aspects missed by other IR methods, such as LSI, when it is used to calculate the conceptual cohesion of a class.

The empirical analysis has been conducted on a relatively small software repository, i.e., EasyClinic, and on a larger repository, i.e., eTour. The studied IR methods have been used to recover traceability links between the use cases and

It is **not possible** to build a set of guidelines for assembling IR-based solutions for a given data set

What is the “best” IR process?

On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery

Noname manuscript No.
(will be inserted by the editor)

Large-scale Information Retrieval in Software Engineering - An Experience Report from Industrial Application

Michael Unterkalmsteiner, Tony Gorschek, Robert Feldt and Niklas Lavesson

the date of receipt and acceptance should be inserted later

Abstract *Background:* Software Engineering activities are information intensive. Research proposes Information Retrieval (IR) techniques to support engineers in their daily tasks, such as establishing and maintaining traceability links, fault identification, and software maintenance. *Objective:* We describe an engineering task, test case selection, and illustrate our problem analysis and solution discovery process. The objective of the study is to gain an understanding of to what extent IR techniques (one potential solution) can be applied to test case selection and provide decision support in a large-scale, industrial setting. *Method:* We analyze, in the context of the studied company, how test case selection is performed and design a series of experiments evaluating the performance of different IR techniques. Each experiment provides lessons learned from implementation, execution, and results, feeding to its successor. *Results:* The three experiments led to the following observations: 1) there is a lack of research on scalable parameter optimization of IR techniques for software engineering problems; 2) scaling IR techniques to industry data is challenging, in particular for latent semantic analysis; 3) the IR context poses constraints on the empirical evaluation of IR techniques, requiring more research on developing valid statistical approaches. *Conclusions:* We believe that our experiences in conducting a series of IR experiments with industry grade data are valuable for peer researchers so that they can avoid the pitfalls that we have encountered. Furthermore, we identified challenges that need to be addressed in order to bridge the gap between laboratory IR experiments and real applications of IR in the industry.

Keywords Test Case Selection · Information Retrieval · Data Mining · Experiment

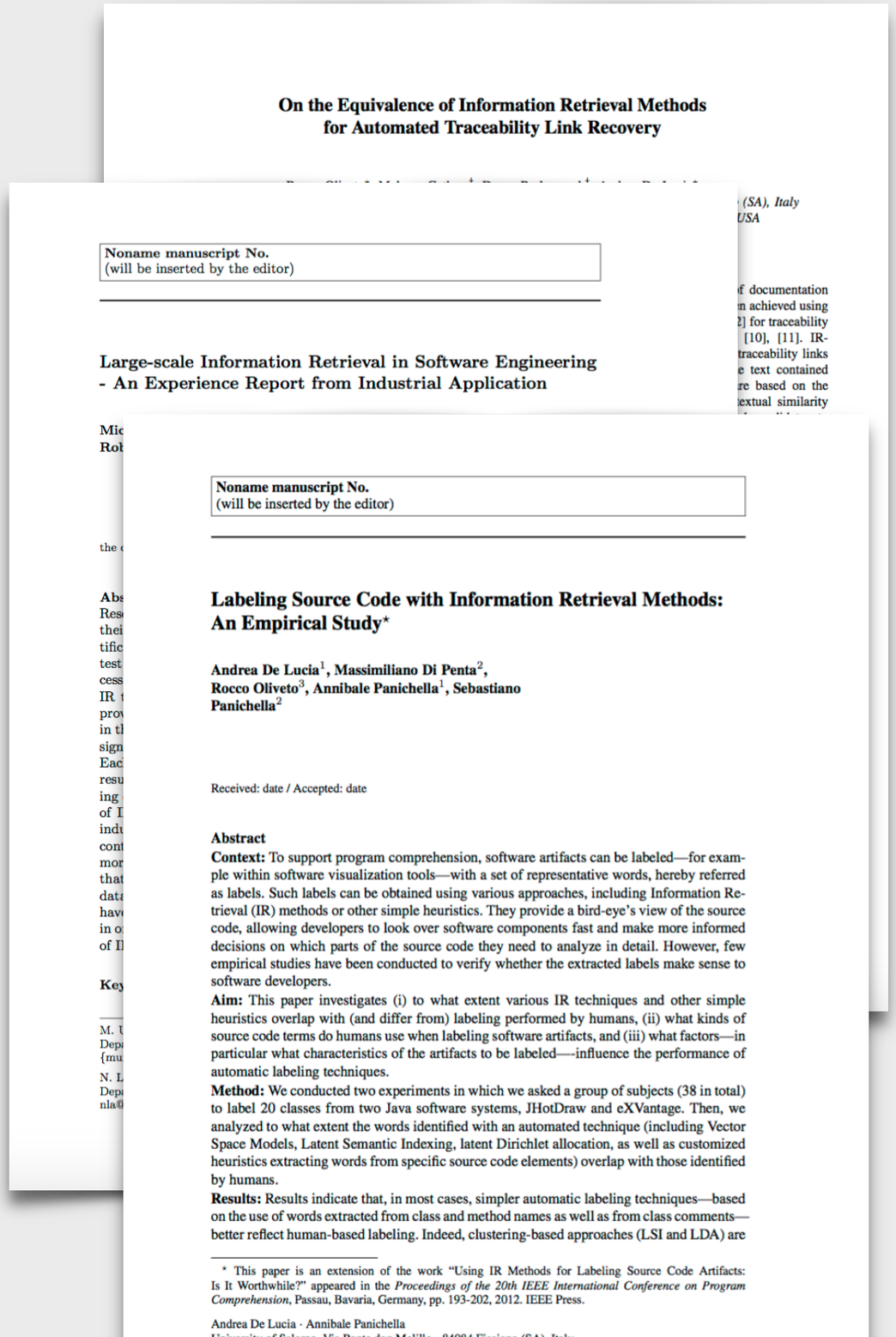
M. Unterkalmsteiner, T. Gorschek, R. Feldt
Department of Software Engineering, Blekinge Institute of Technology E-mail: {mun,tgo,rfd}@bth.se

N. Lavesson
Department of Computer Science and Engineering, Blekinge Institute of Technology E-mail: nla@bth.se

It is **not possible** to build a set of guidelines for assembling IR-based solutions for a given data set

Different datasets and different SE tasks require different IR parameters

What is the “best” IR process?



On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery

Noname manuscript No.
(will be inserted by the editor)

Large-scale Information Retrieval in Software Engineering - An Experience Report from Industrial Application

Noname manuscript No.
(will be inserted by the editor)

Labeling Source Code with Information Retrieval Methods: An Empirical Study*

Andrea De Lucia[1], Massimiliano Di Penta[2], Rocco Oliveto[3], Annibale Panichella[1], Sebastiano Panichella[2]

Received: date / Accepted: date

Abstract
Context: To support program comprehension, software artifacts can be labeled—for example within software visualization tools—with a set of representative words, hereby referred as labels. Such labels can be obtained using various approaches, including Information Retrieval (IR) methods or other simple heuristics. They provide a bird-eye's view of the source code, allowing developers to look over software components fast and make more informed decisions on which parts of the source code they need to analyze in detail. However, few empirical studies have been conducted to verify whether the extracted labels make sense to software developers.
Aim: This paper investigates (i) to what extent various IR techniques and other simple heuristics overlap with (and differ from) labeling performed by humans, (ii) what kinds of source code terms do humans use when labeling software artifacts, and (iii) what factors—in particular what characteristics of the artifacts to be labeled—influence the performance of automatic labeling techniques.
Method: We conducted two experiments in which we asked a group of subjects (38 in total) to label 20 classes from two Java software systems, JHotDraw and eXVantage. Then, we analyzed to what extent the words identified with an automated technique (including Vector Space Models, Latent Semantic Indexing, latent Dirichlet allocation, as well as customized heuristics extracting words from specific source code elements) overlap with those identified by humans.
Results: Results indicate that, in most cases, simpler automatic labeling techniques—based on the use of words extracted from class and method names as well as from class comments—better reflect human-based labeling. Indeed, clustering-based approaches (LSI and LDA) are

* This paper is an extension of the work “Using IR Methods for Labeling Source Code Artifacts: Is It Worthwhile?” appeared in the *Proceedings of the 20th IEEE International Conference on Program Comprehension*, Passau, Bavaria, Germany, pp. 193-202, 2012. IEEE Press.

Andrea De Lucia · Annibale Panichella

It is **not possible** to build a set of guidelines for assembling IR-based solutions for a given data set

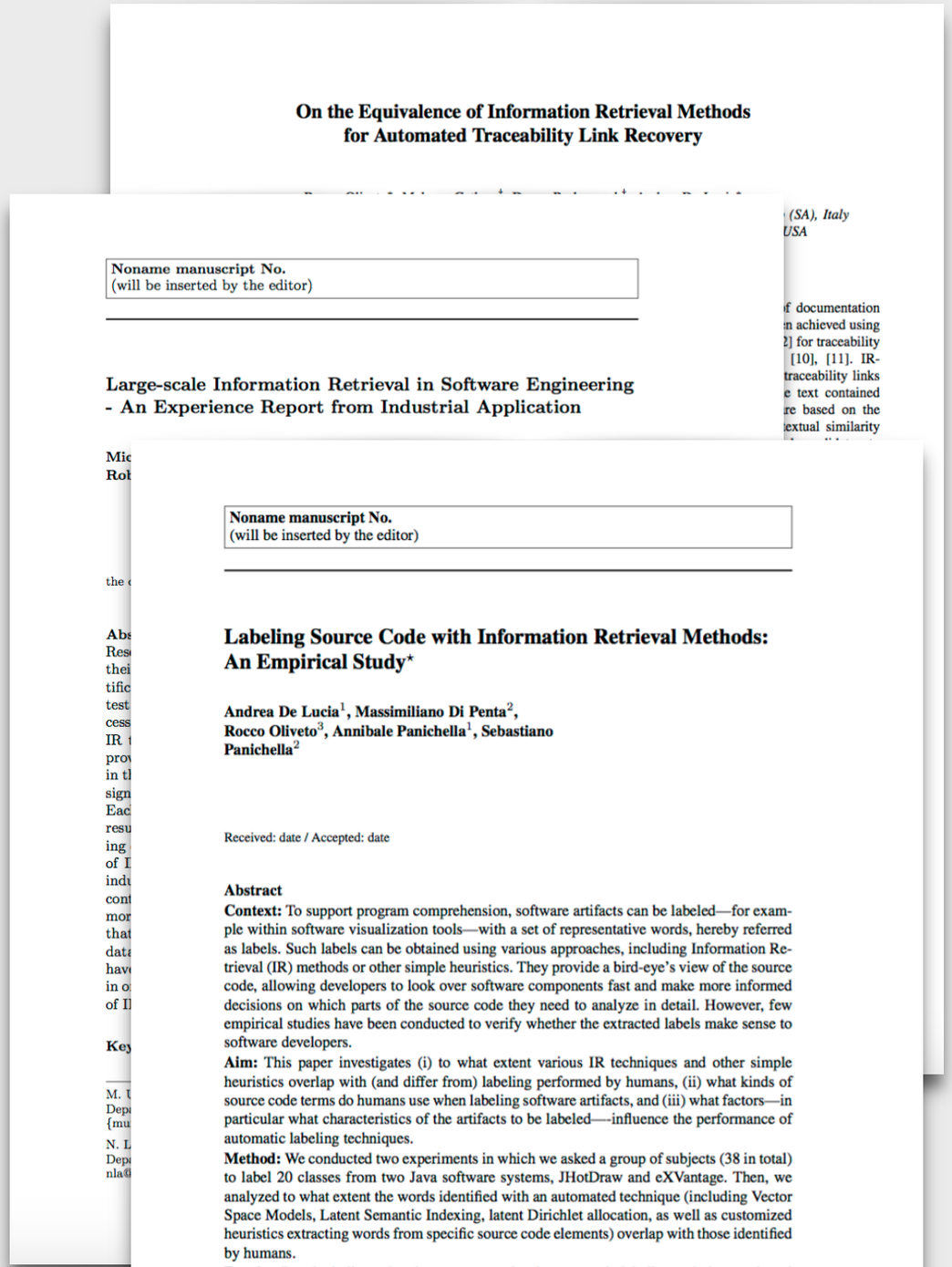
Different datasets and different SE tasks require different IR parameters

If not well calibrated, IR techniques perform worst than simple heuristics.

A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella. “**Labeling Source Code with Information Retrieval Methods: An Empirical Study**”. EMSE 2014

What is the “best” IR process?

for the project and
for the SE task



On the Equivalence of Information Retrieval Methods for Automated Traceability Link Recovery

Noname manuscript No.
(will be inserted by the editor)

Large-scale Information Retrieval in Software Engineering - An Experience Report from Industrial Application

Noname manuscript No.
(will be inserted by the editor)

Labeling Source Code with Information Retrieval Methods: An Empirical Study*

Andrea De Lucia[1], Massimiliano Di Penta[2], Rocco Oliveto[3], Annibale Panichella[1], Sebastiano Panichella[2]

Received: date / Accepted: date

Abstract
Context: To support program comprehension, software artifacts can be labeled—for example within software visualization tools—with a set of representative words, hereby referred as labels. Such labels can be obtained using various approaches, including Information Retrieval (IR) methods or other simple heuristics. They provide a bird-eye's view of the source code, allowing developers to look over software components fast and make more informed decisions on which parts of the source code they need to analyze in detail. However, few empirical studies have been conducted to verify whether the extracted labels make sense to software developers.
Aim: This paper investigates (i) to what extent various IR techniques and other simple heuristics overlap with (and differ from) labeling performed by humans, (ii) what kinds of source code terms do humans use when labeling software artifacts, and (iii) what factors—in particular what characteristics of the artifacts to be labeled—influence the performance of automatic labeling techniques.
Method: We conducted two experiments in which we asked a group of subjects (38 in total) to label 20 classes from two Java software systems, JHotDraw and eXVantage. Then, we analyzed to what extent the words identified with an automated technique (including Vector Space Models, Latent Semantic Indexing, latent Dirichlet allocation, as well as customized heuristics extracting words from specific source code elements) overlap with those identified by humans.
Results: Results indicate that, in most cases, simpler automatic labeling techniques—based on the use of words extracted from class and method names as well as from class comments—better reflect human-based labeling. Indeed, clustering-based approaches (LSI and LDA) are

* This paper is an extension of the work “Using IR Methods for Labeling Source Code Artifacts: Is It Worthwhile?” appeared in the *Proceedings of the 20th IEEE International Conference on Program Comprehension*, Passau, Bavaria, Germany, pp. 193-202, 2012. IEEE Press.

Andrea De Lucia · Annibale Panichella

It is **not possible** to build a set of guidelines for assembling IR-based solutions for a given data set

Different datasets and different SE tasks require different IR parameters

If not well calibrated, IR techniques perform worst than simple heuristics.

A. De Lucia, M. Di Penta, R. Oliveto, A. Panichella, S. Panichella. “**Labeling Source Code with Information Retrieval Methods: An Empirical Study**”. EMSE 2014

Strategy to find the best configuration…

Strategy to find the best configuration…

It must be unsupervised..

Strategy to find the best configuration…

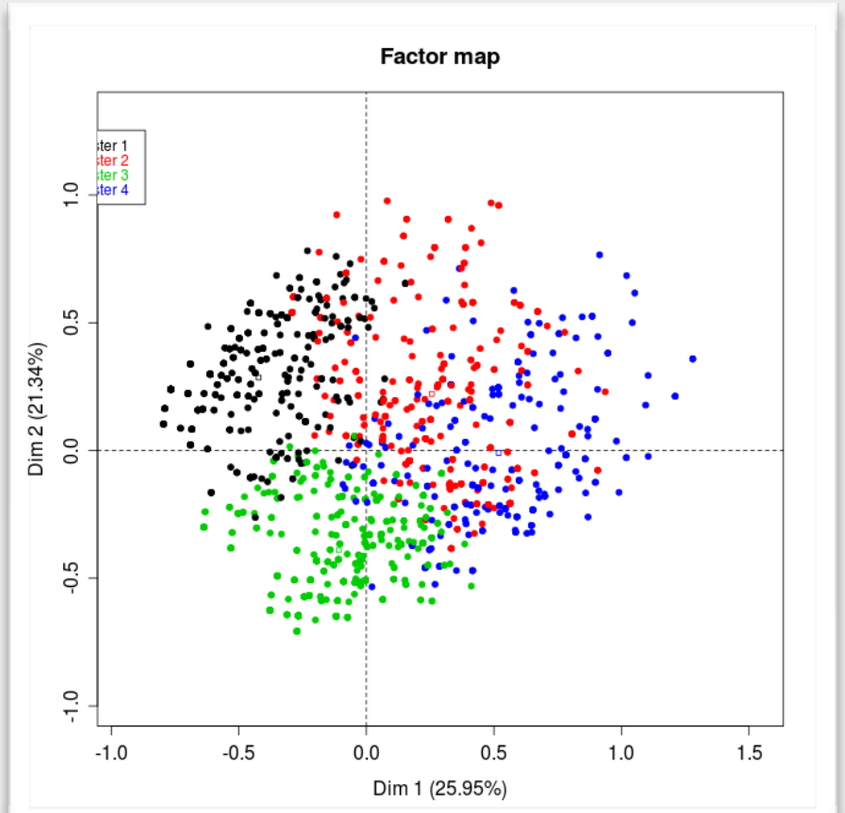
It must be unsupervised..

and general…

Our Idea

Information Retrieval

Clustering Analysis



Conjecture: there is a ***relationship*** between **quality of clusters** and I**R process performances**

Predicting the performances?

Term Extraction

- Special Chars.
- Digits
- White space
- ..

Stop-Word List / Function

- No stop word
- Stop-word function
- Java stop-word list
- English stop-word list
- Italian stop-word list
- …

Morphological Analysis

- No Stemmer
- Porter
- English Snowball
- Italian Snowball
- ….

Term Weighting

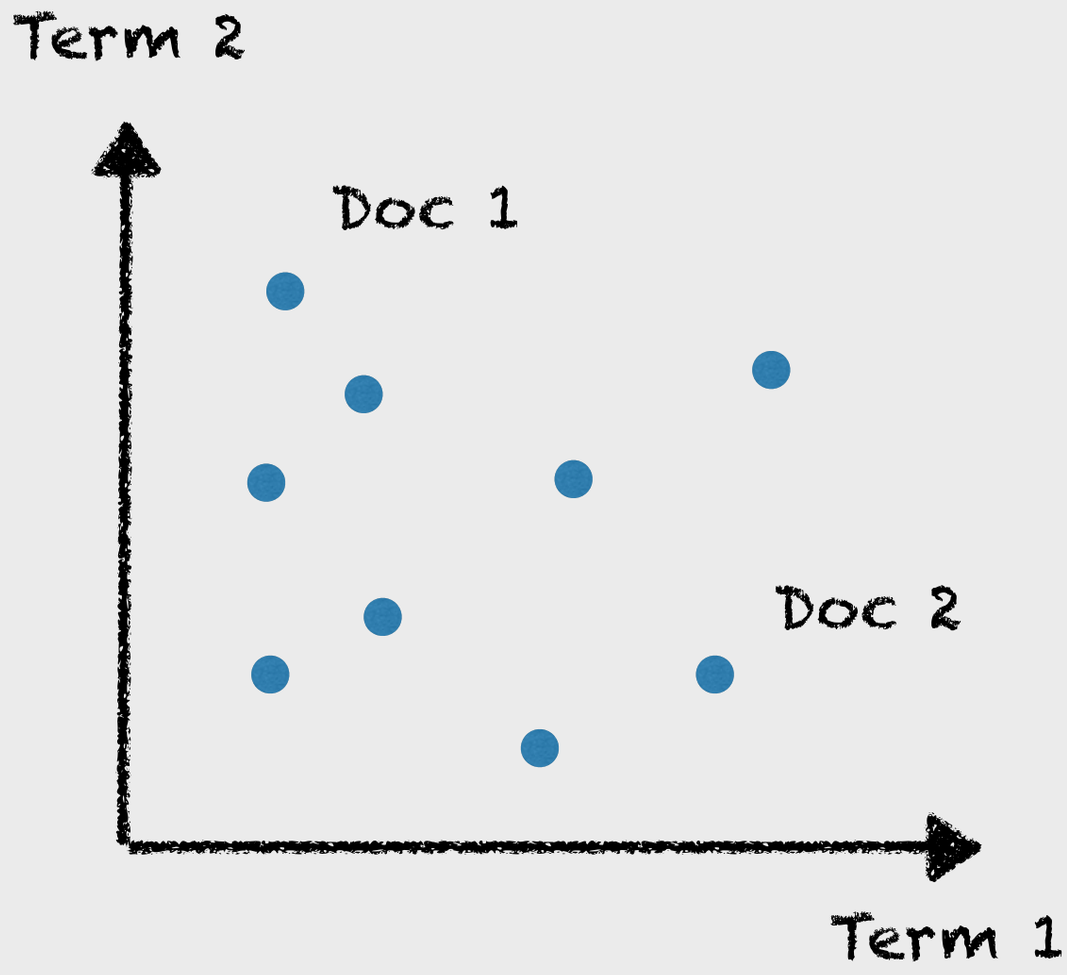
- Boolean
- Term Frequency
- TF-IDF
- Log(TF+1)
- Entropy

Distance Function

- LSI (k)
- LDA (**α**, β, n, k)
- VSM

IR Methods

- Cosine Similarity
- Hellinger Distance
- Jaccard Dist.
- Euclidean



Predicting the performances?

Term Extraction

- Special Chars.
- **Digits**
- **White space**
- ..

Stop-Word List / Function

- No stop word
- Stop-word function
- **Java stop-word list**
- English stop-word list
- Italian stop-word list
- …

Morphological Analysis

- No Stemmer
- **Porter**
- English Snowball
- Italian Snowball
- ….

Term Weighting

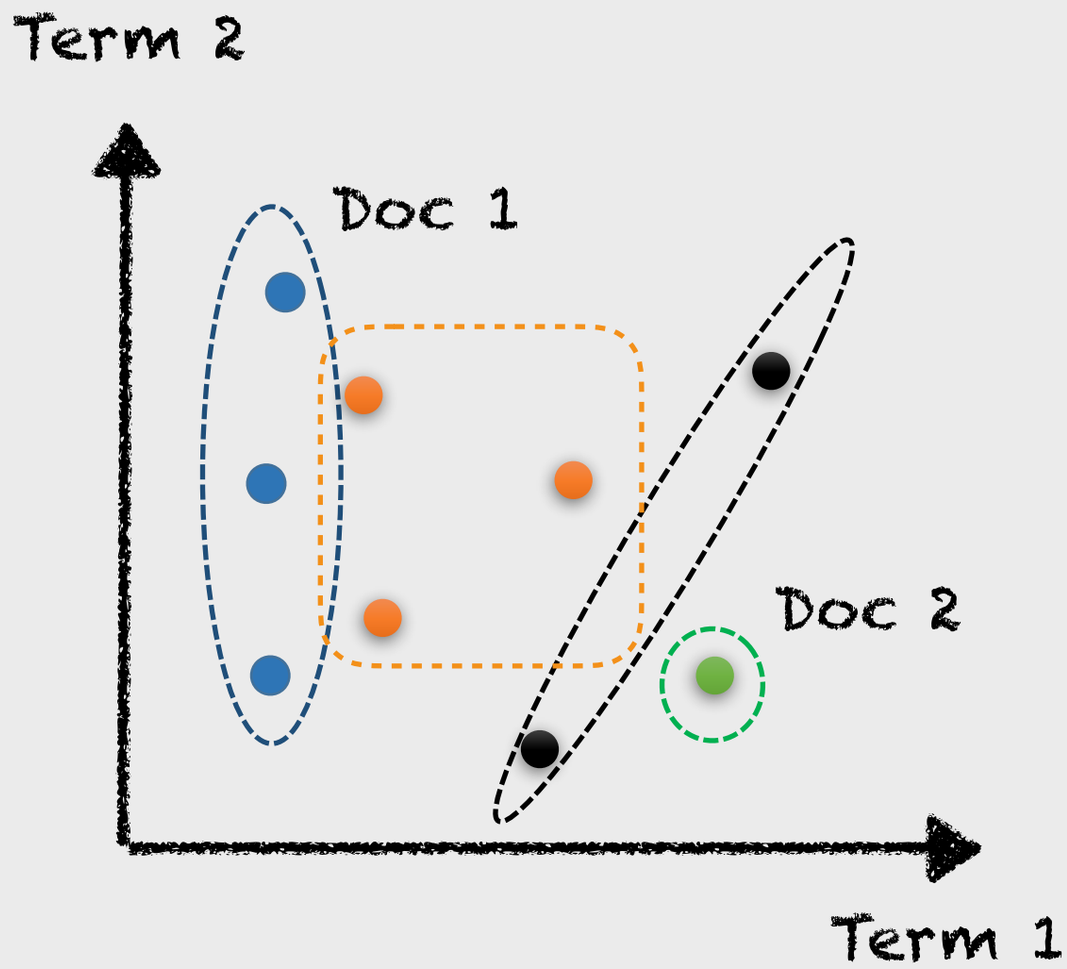
- Boolean
- **Term Frequency**
- TF-IDF
- Log(TF+1)
- Entropy

Distance Function

- **LSI (k = 10)**
- LDA (α, β, n, k)
- VSM

IR Methods

- **Cosine Similarity**
- Hellinger Distance
- Jaccard Dist.
- Euclidean



Predicting the performances?

Term Extraction

- **Special Chars.**
- **Digits**
- White space
- ..

Stop-Word List / Function

- No stop word
- Stop-word function
- **Java stop-word list**
- English stop-word list
- Italian stop-word list
- …

Morphological Analysis

- **No Stemmer**
- Porter
- English Snowball
- Italian Snowball
- ….

Term Weighting

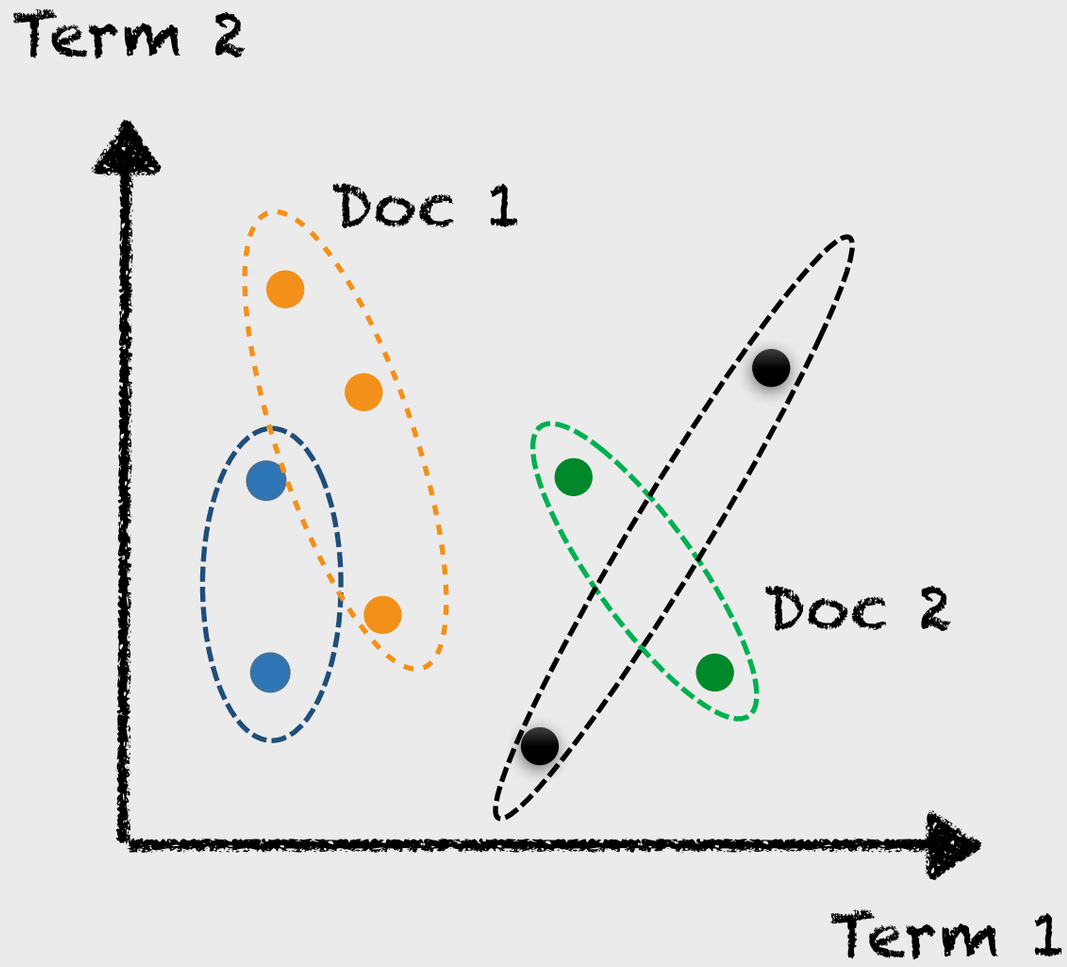
- Boolean
- Term Frequency
- TF-IDF
- **Log(TF+1)**
- Entropy

Distance Function

- **LSI (k = 5)**
- LDA (**α**, β, n, k)
- VSM

IR Methods

- Cosine Similarity
- Hellinger Distance
- **Jaccard Dist.**
- Euclidean



Predicting the performances?

Term Extraction

Special Chars.
Digits
White space
..

Stop-Word List / Function

No stop word
Stop-word function
Java stop-word list
English stop-word list
Italian stop-word list
…

Morphological Analysis

No Stemmer
Porter
English Snowball
Italian Snowball
….

Term Weighting

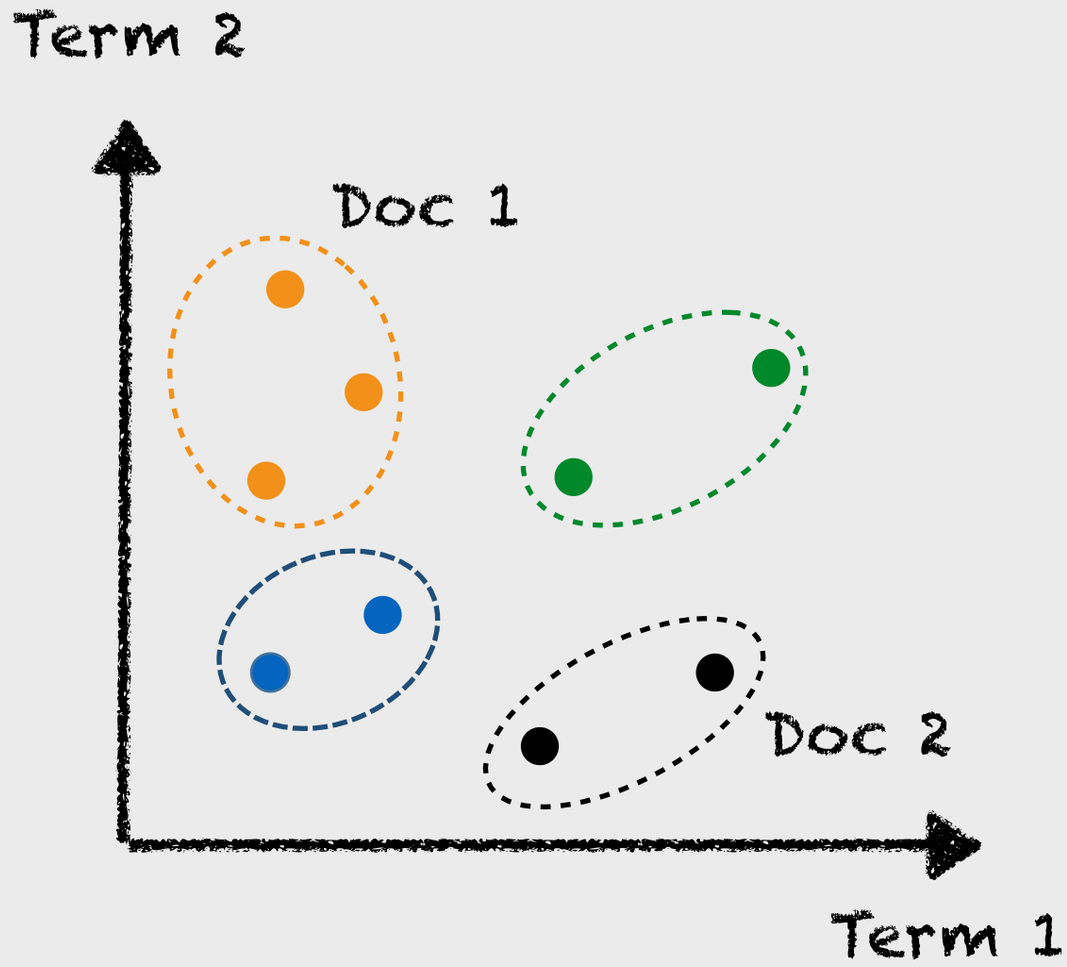
Boolean
Term Frequency
TF-IDF
Log(TF+1)
Entropy

Distance Function

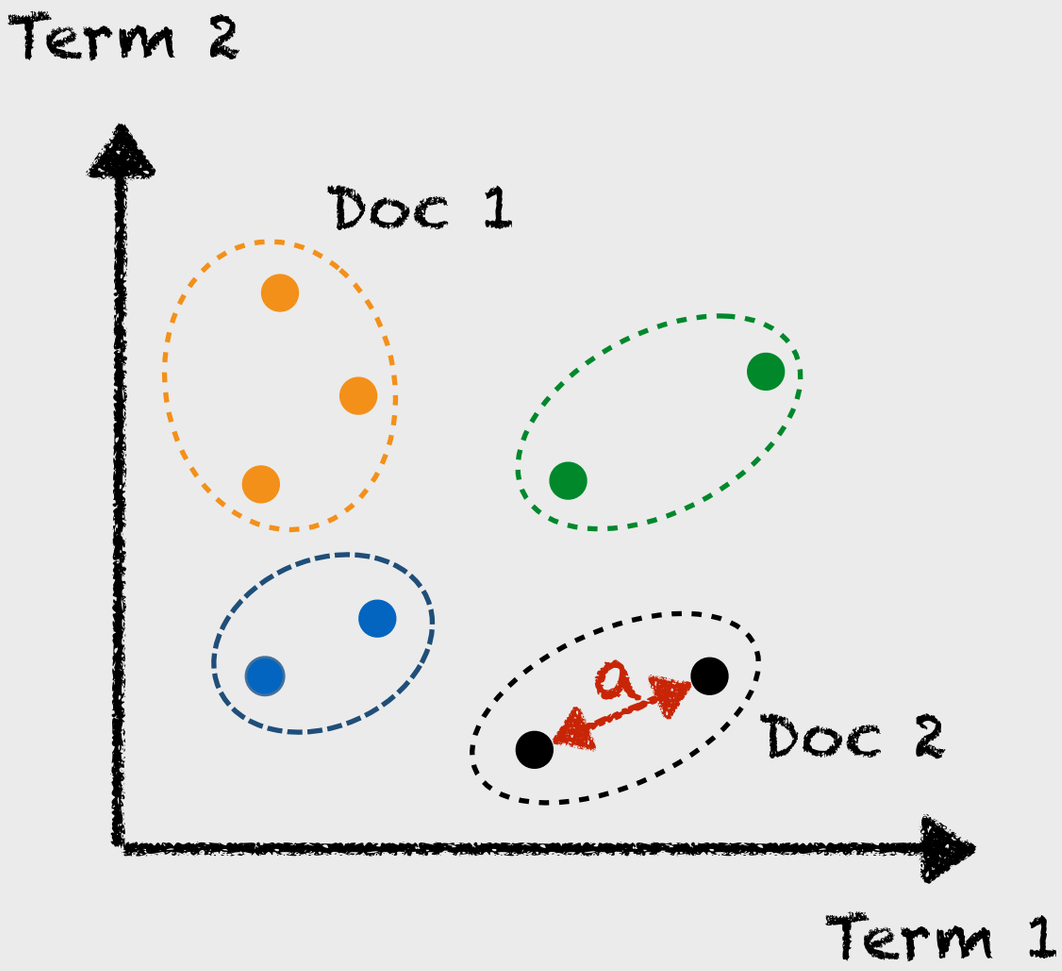
LSI (k = 3)
LDA (**α**, β, n, k)
VSM

IR Methods

Cosine Similarity
Hellinger Distance
Jaccard Dist.
Euclidean Dist.

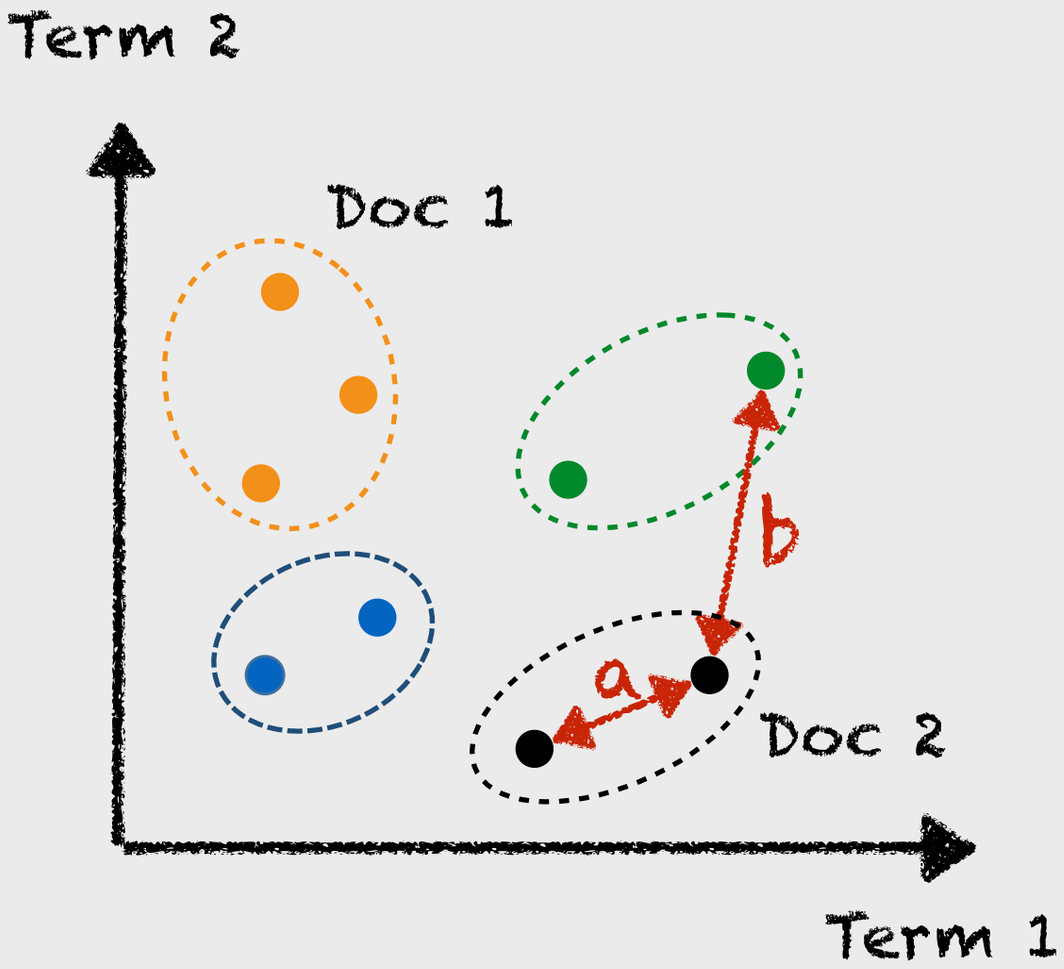


Silhouette Coefficient



a = Cluster Cohesion

Silhouette Coefficient



a = Cluster Cohesion

b = Cluster Separation

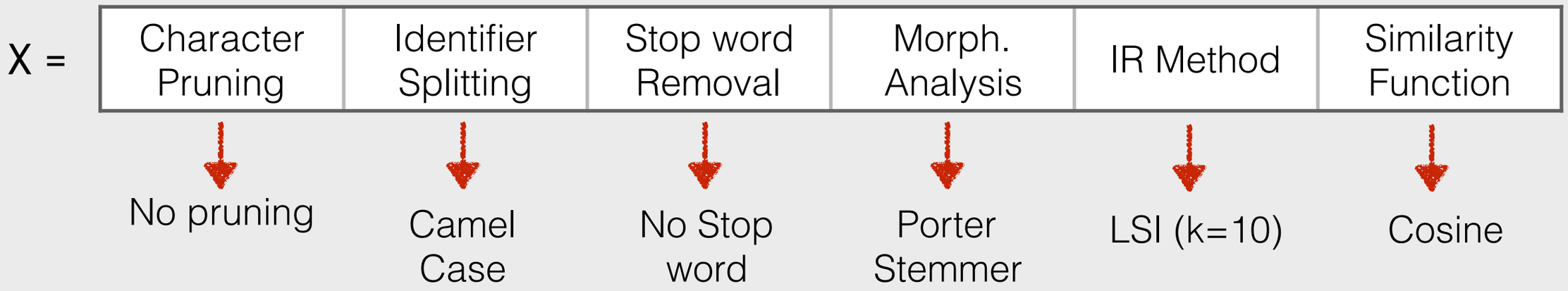
Good Cluster:
Separation >> Cohesion

$$\text{Silhouette} = \frac{b - a}{\max\{a, b\}} \quad \in [-1; 1]$$

Search-Based Solution (GA-IR)

1) **Problem Reformulation:** *Finding the IR process which maximises the quality of clusters*

2) **Solution Encoding:**



3) **Fitness Function:** Silhouette Coefficient $\frac{b - a}{\max\{a, b\}}$

4) **Solver:** Genetic Algorithms
   - Uniform mutation with probability of 1/n
   - Population size = 50
   - Single -point crossover with probability 0.80
   - N. generations = 100

Empirical Evaluation

Task1: Traceability Recovery

Project	LOC	Source	Target
EasyClinic	20k	Use Case	Java Classes
eTour	40K	Use Case	Java Classes
i-Trus	10k	Use Case	JSP

Performance metrics:

- Precision
- Recall
- Average Precision

Task2: Duplicate Bug Report Identification

Project	Version	# Reports	# Dupl.
Eclipse	3.0	224	44

Performance metrics:

- Recall Rate
- List size

Empirical Evaluation

Task1: Traceability Recovery

Project	LOC	Source	Target
EasyClinic	20k	Use Case	Java Classes
eTour	40K	Use Case	Java Classes
i-Trus	10k	Use Case	JSP

Performance metrics:

- Precision
- Recall
- Average Precision

Task2: Duplicate Bug Report Identification

Project	Version	# Reports	# Dupl.
Eclipse	3.0	224	44

Performance metrics:

- Recall Rate
- List size

Task 1: Traceability Recovery

System = eTour

Size = 40K LOC

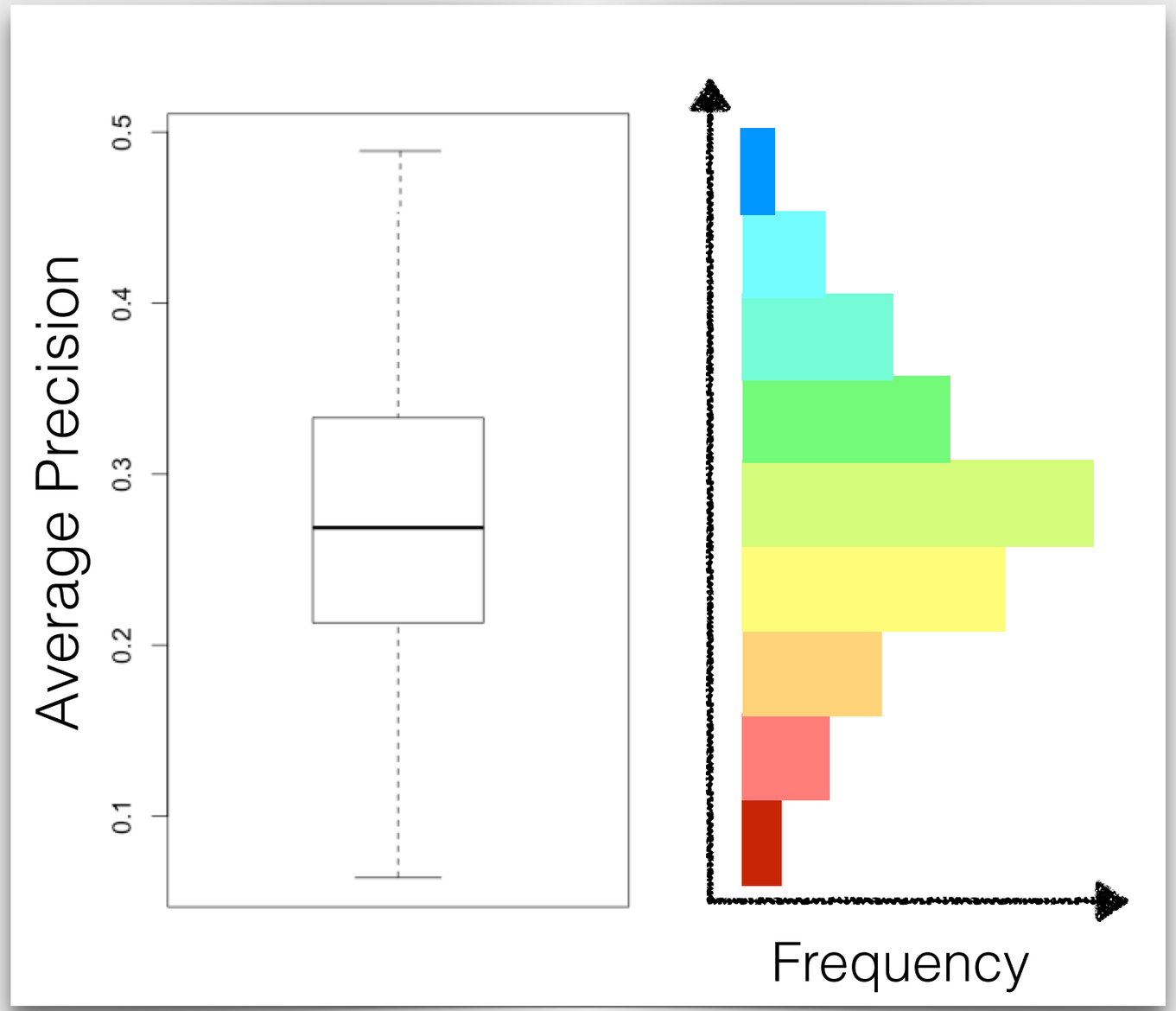
IR Config. = 89,640

Task 1: Traceability Recovery

System = eTour

Size = 40K LOC

IR Config. = 89,640



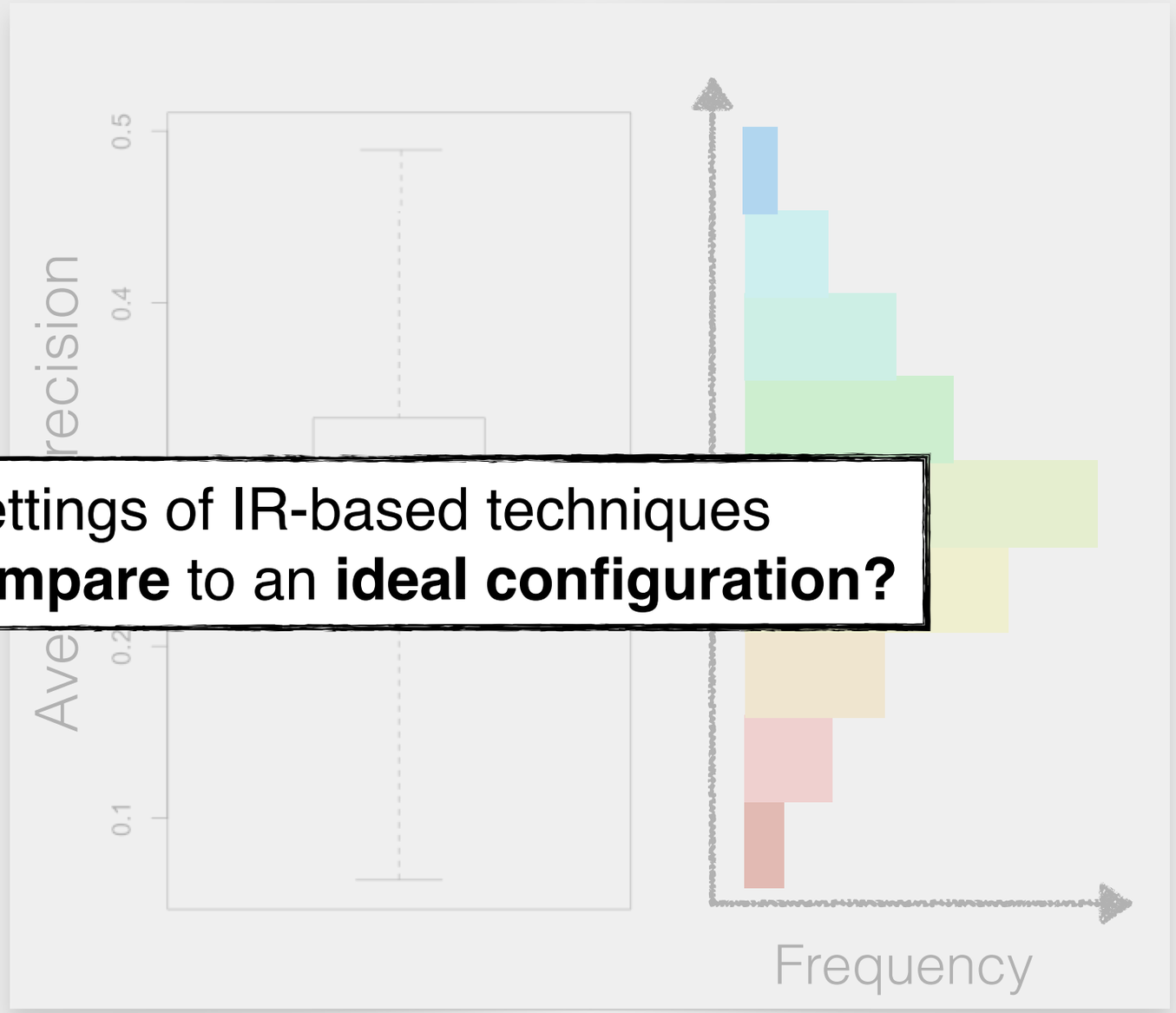
High Results Variability

Task 1: Traceability Recovery

System = eTour

Size = 40K LOC

IR Config. = 89,640



RQ1: How do the settings of IR-based techniques instantiated by GA-IR **compare** to an **ideal configuration?**

Task 1: Traceability Recovery

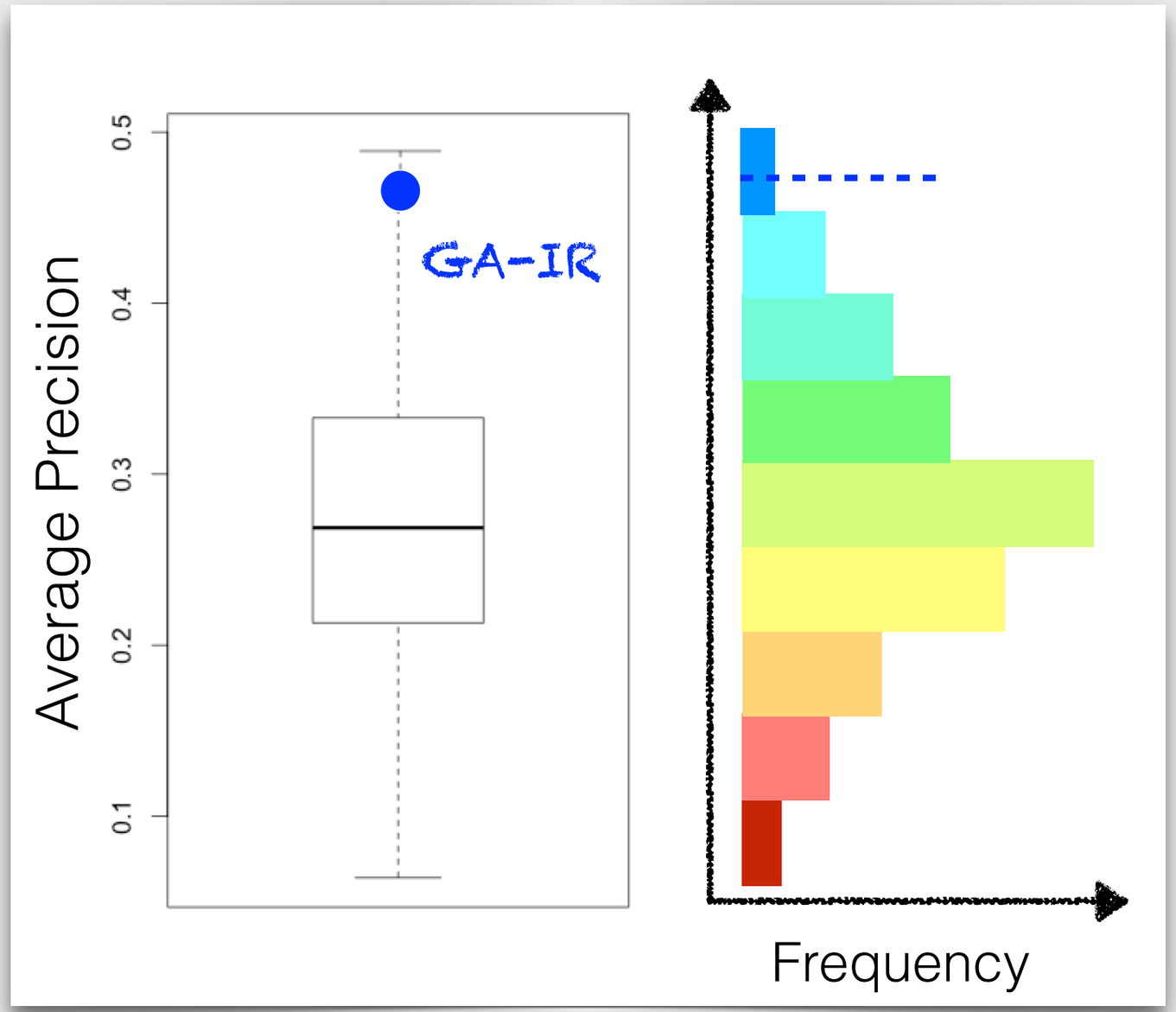
System = eTour

Size = 40K LOC

IR Config. = 89,640

Average Precision:
- Ideal Config. = 47.01%
- GA-IR = 46.94%

Mean over 30 independent runs



RQ1: How do the settings of IR-based techniques instantiated by GA-IR **compare** to an **ideal configuration?**

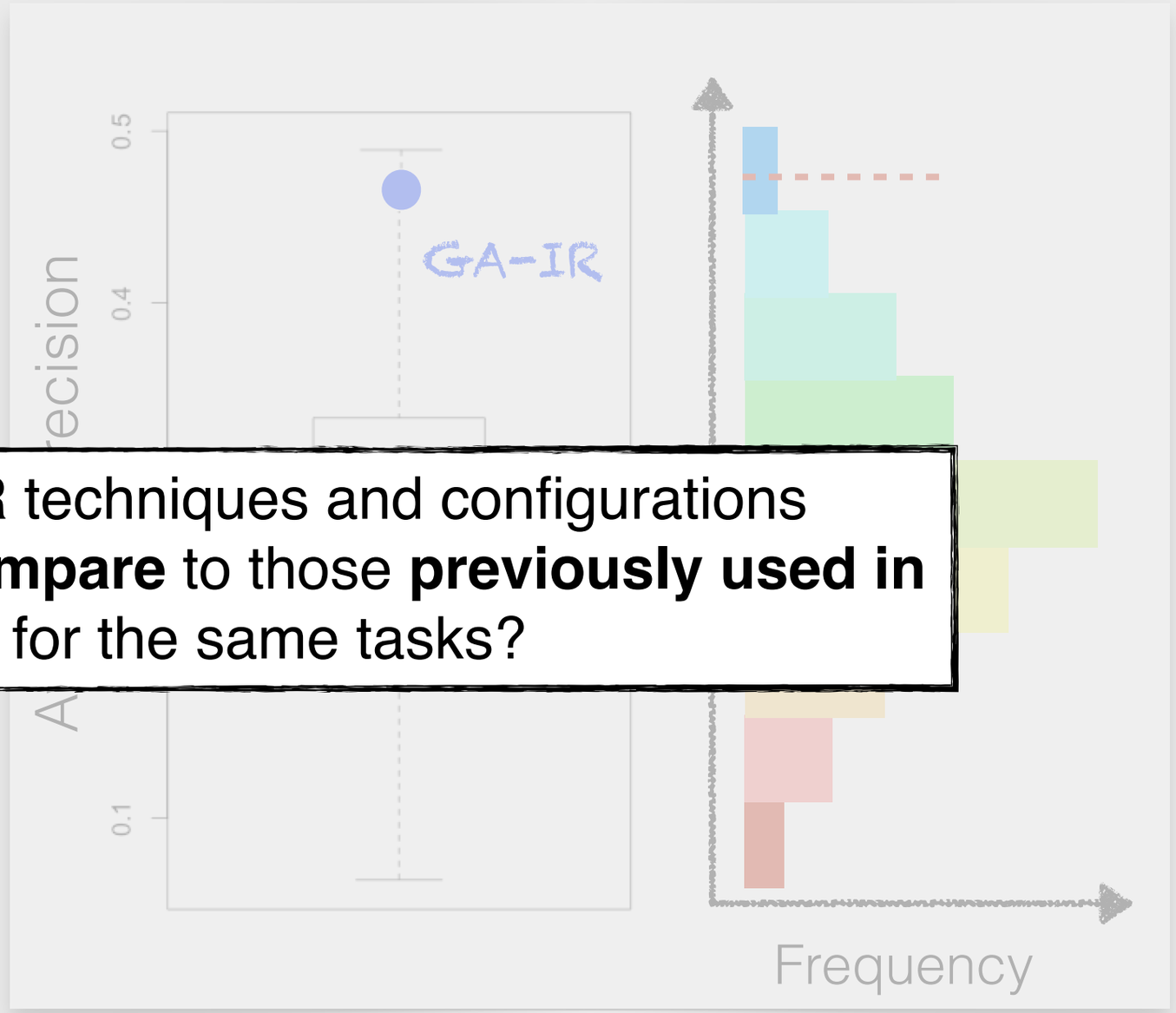
Task 1: Traceability Recovery

System = eTour

Size = 40K LOC

IR Config. = 89,640

Avera
- Ideal Config. = 47.01%
- GA-IR = 46.94%



RQ2: How do the IR techniques and configurations instantiated by GA-IR **compare** to those **previously used in literature** for the same tasks?

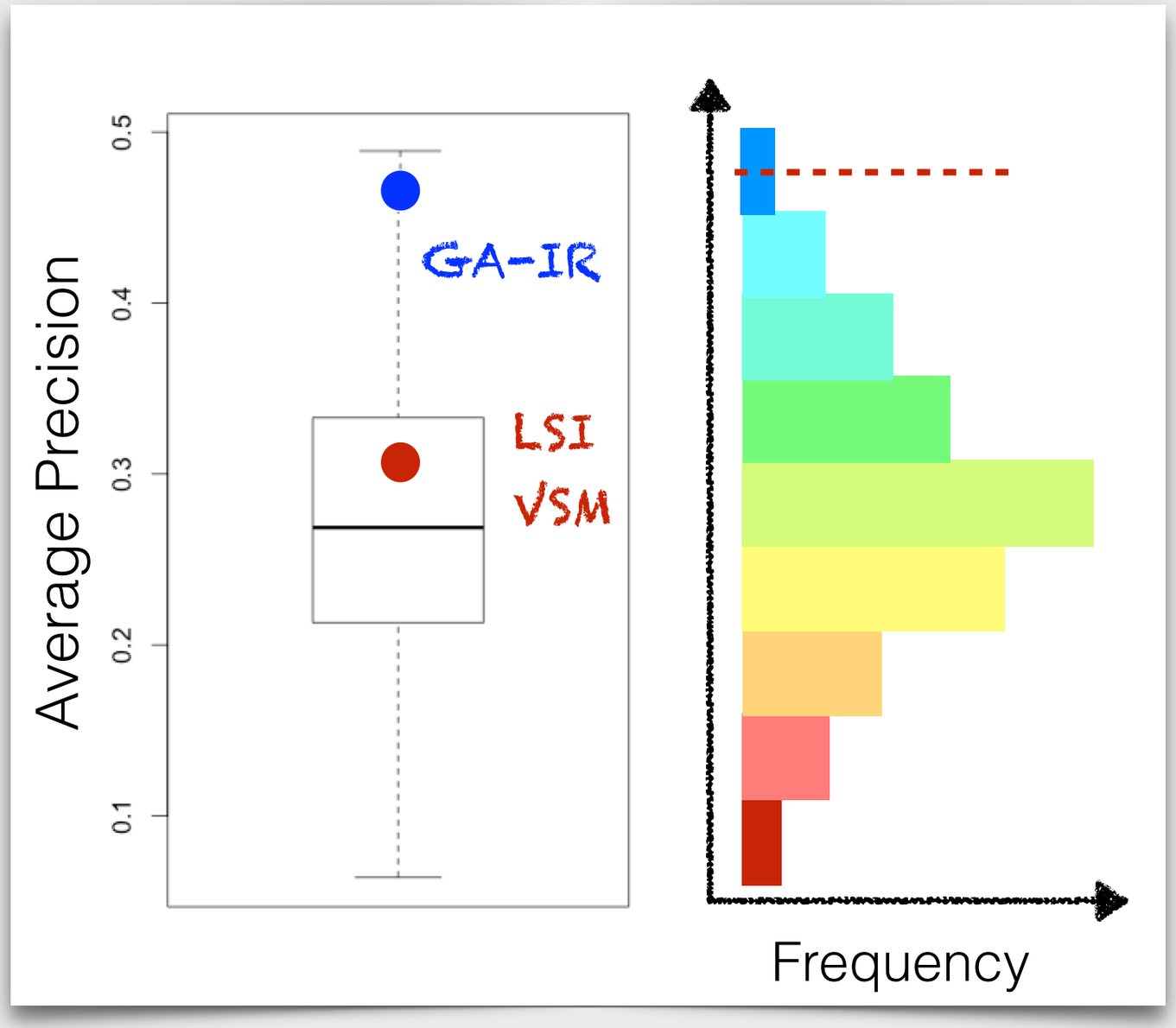
Task 1: Traceability Recovery

System = eTour

Size = 40K LOC

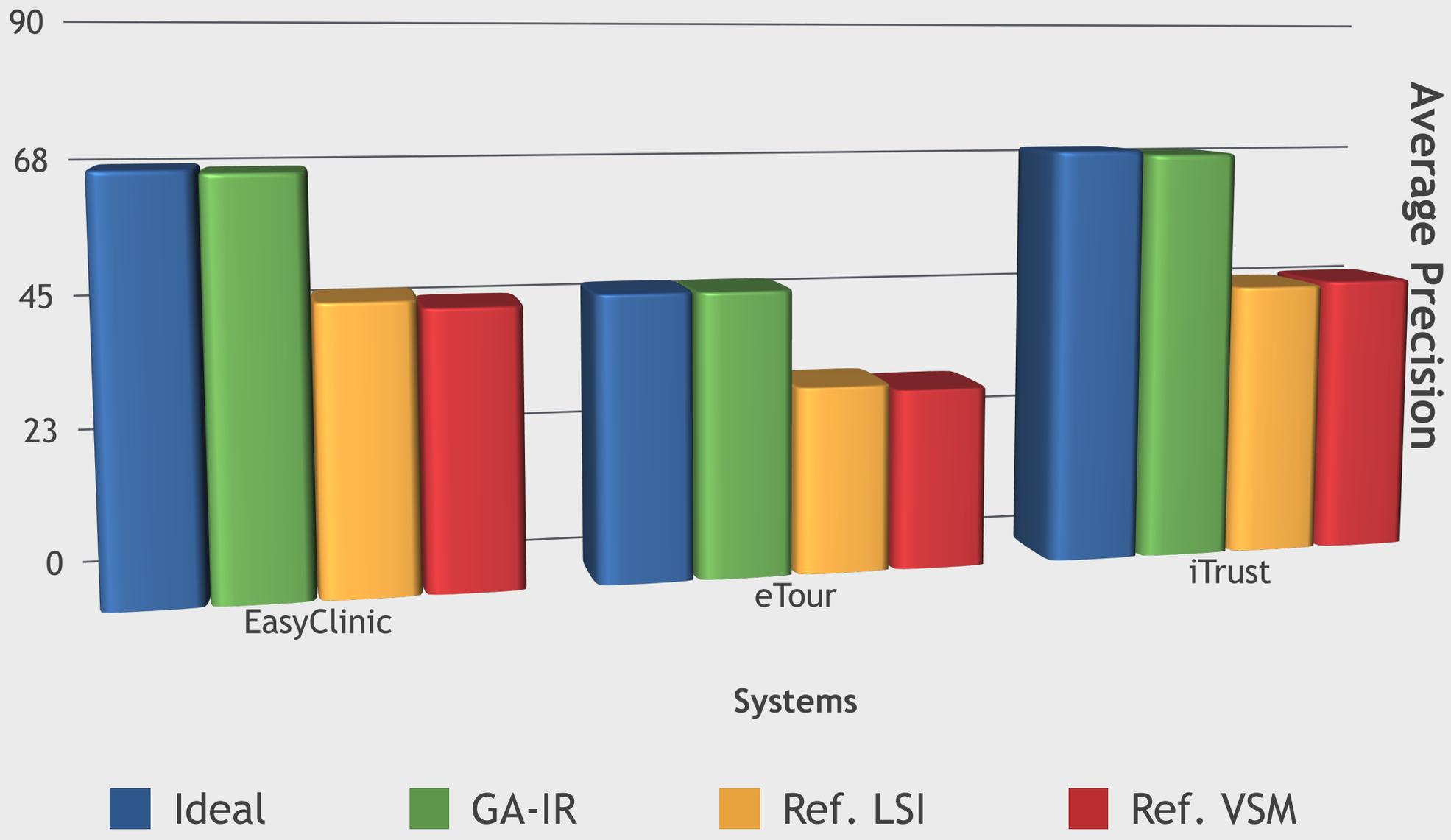
IR Config. = 89,640

Average Precision:
- Ideal Config. = 47.01%
- GA-IR = 46.94%
- Ref. LSI = 30.93%
- Ref. VSM = 29.94%

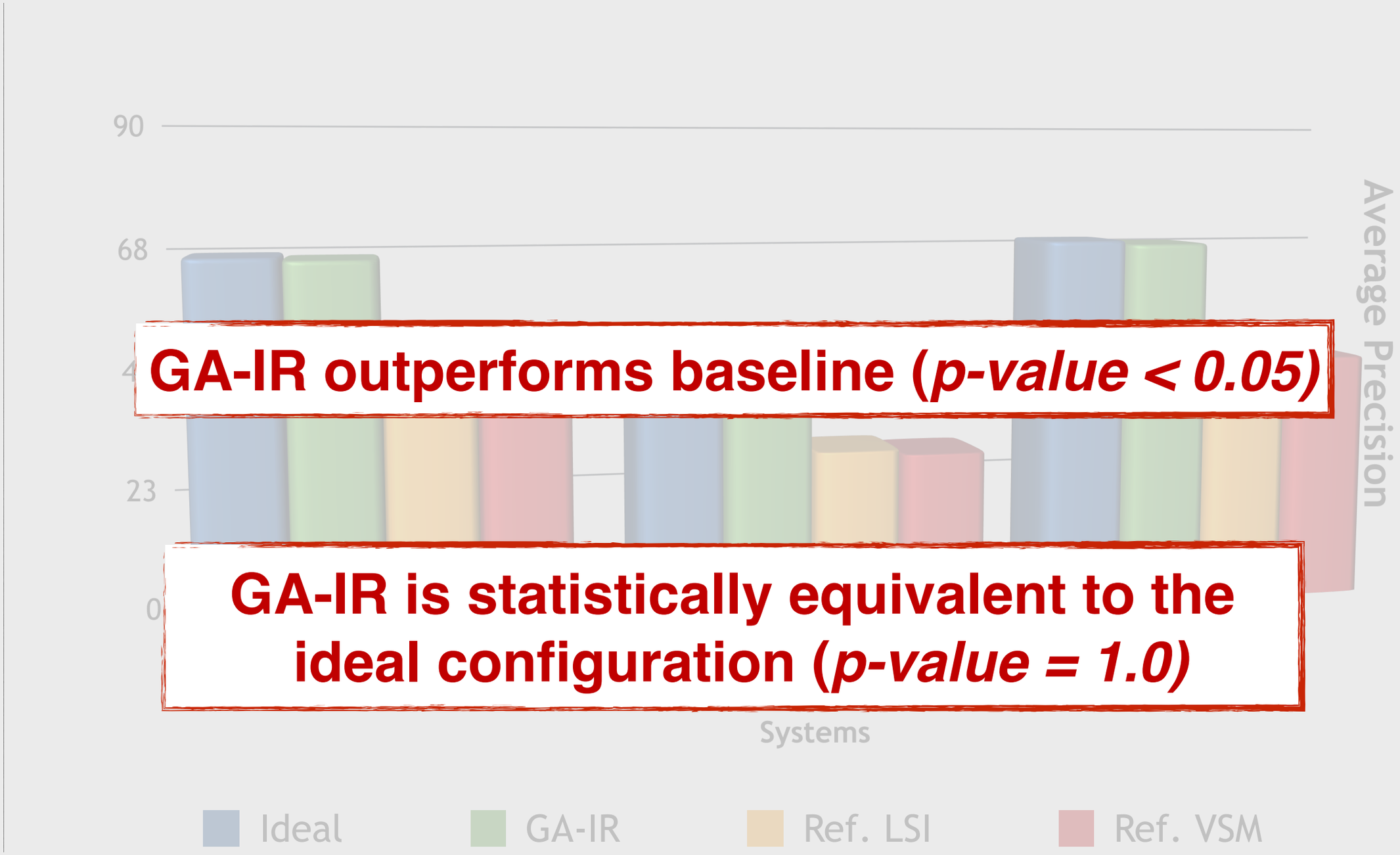


RQ2: How do the IR techniques and configurations instantiated by GA-IR **compare** to those **previously used in literature** for the same tasks?

Task 1: Traceability Recovery



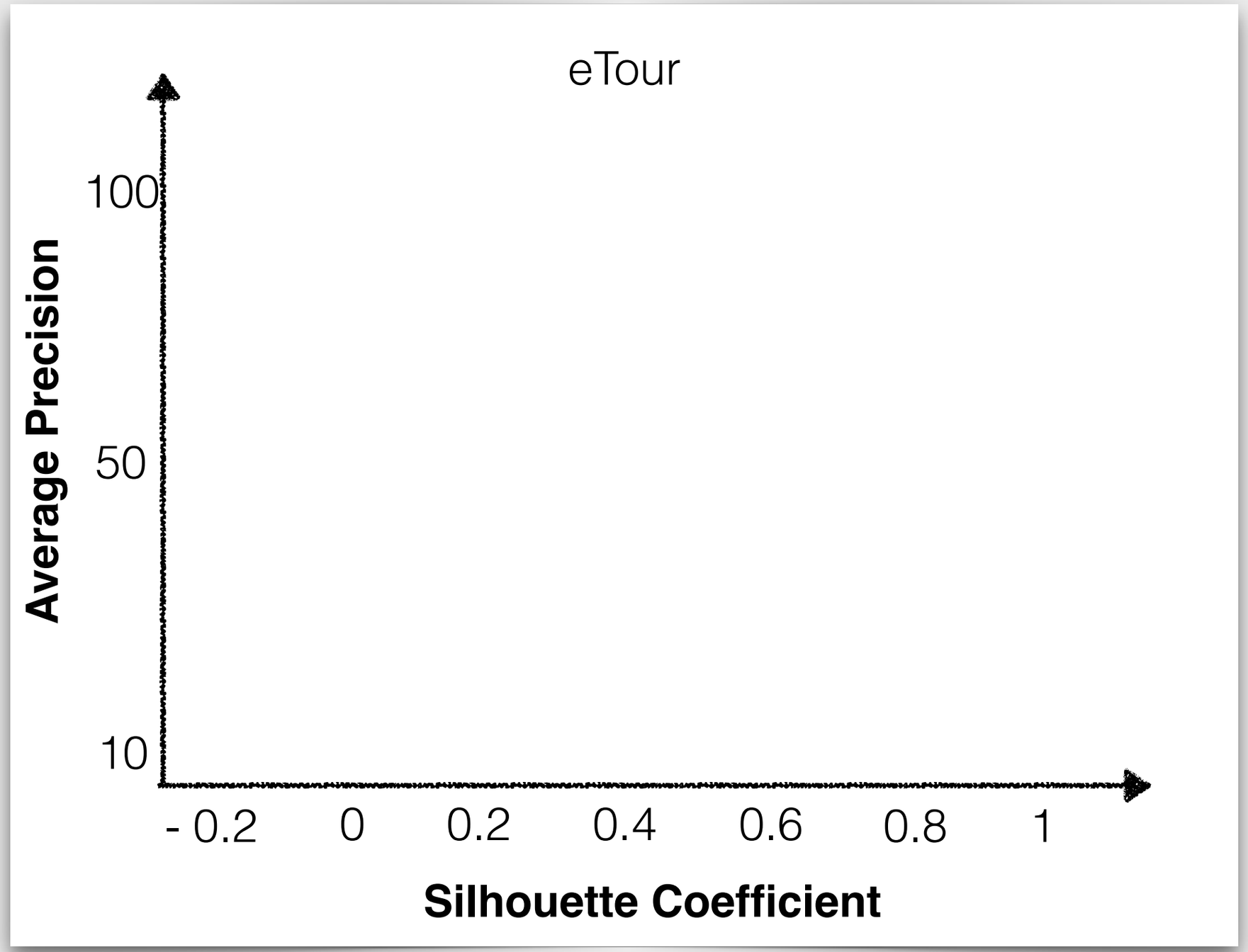
Task 1: Traceability Recovery



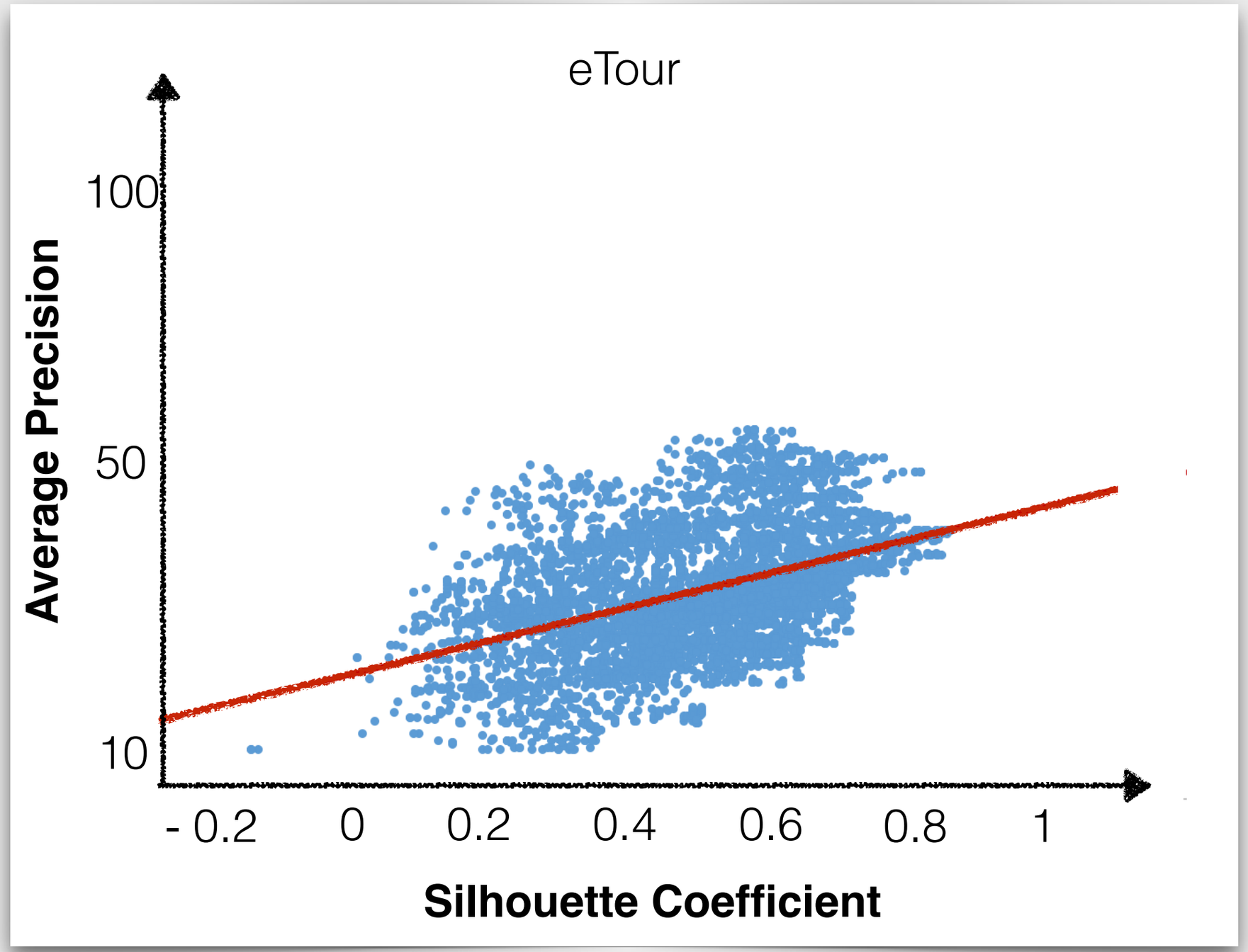
GA-IR outperforms baseline (*p-value < 0.05*)

GA-IR is statistically equivalent to the ideal configuration (*p-value = 1.0*)

Clustering Hypothesis



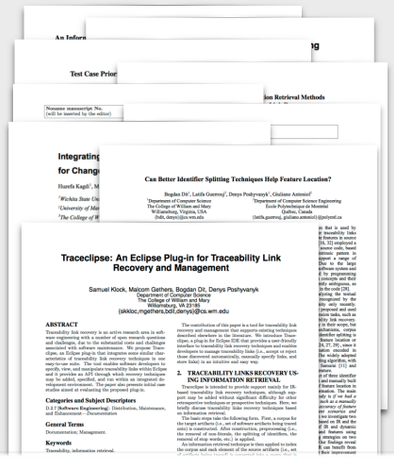
Clustering Hypothesis



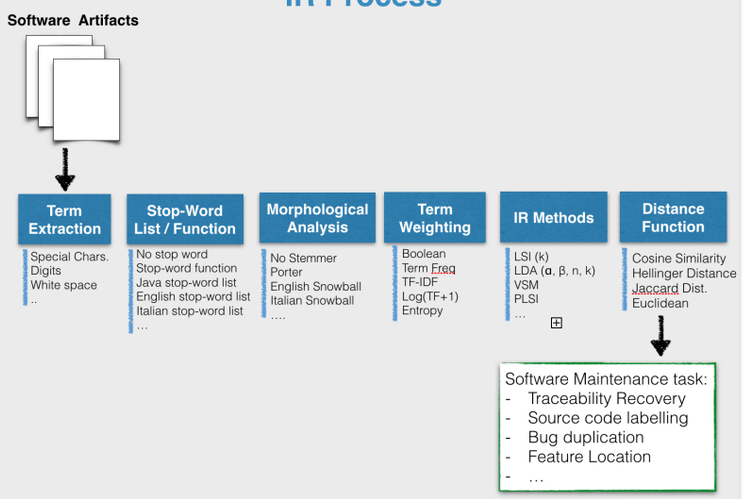
Information Retrieval

600
papers in 10 years

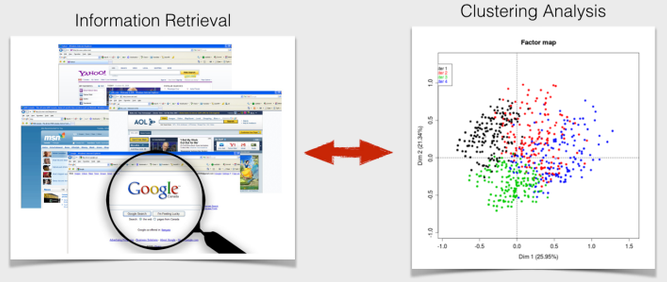
30
applications in S.E.



IR Process



Our Idea



Conjecture: there is a ***relationship*** between **quality of clusters** and I**R process performances**

Search-Based Solution (IR-GA)

1) **Problem Reformulation:** *Finding the IR process which maximises the quality of clusters*

2) **Solution Encoding:**

X =	Character Pruning	Identifier Splitting	Stop word Removal	Morph. Analysis	IR Method	Similarity Function
	↓	↓	↓	↓	↓	↓
	No pruning	Camel Case	No Stop word	Porter Stemmer	LSI (k=10)	Cosine

3) **Fitness Function:** Silhouette Coefficient $\frac{b - a}{\max\{a, b\}}$

4) **Solver:** Genetic Algorithms

Empirical Evaluation

Task1: Traceability Recovery

Project	LOC	Source	Target
EasyClinic	20k	Use Case	Java Classes
eTour	40K	Use Case	Java Classes
i-Trus	10k	Use Case	JSP

Performance metrics:
- Precision
- Recall
- Average Precision

Task2: Duplicate Bug Report Identification

Project	Version	# Reports	# Dupl.
Eclipse	3.0	224	44

Performance metrics:
- Recall Rate
- List size

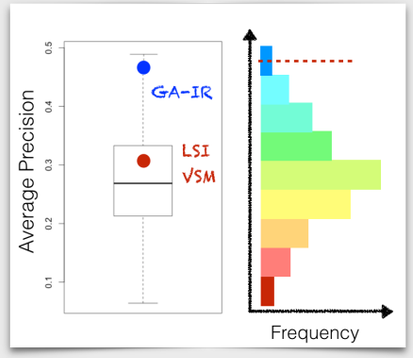
Task 1: Traceability Recovery

System = eTour

Size = 40K LOC

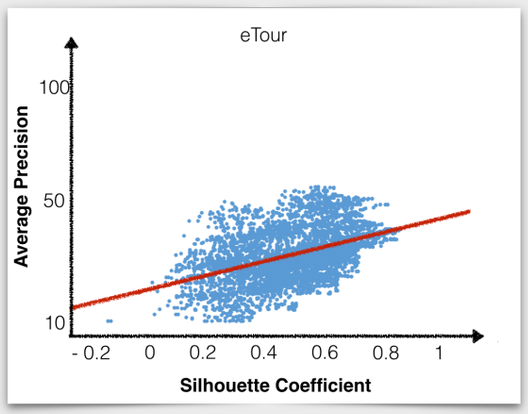
 # IR Config. = 89,640

Average Precision:
- Ideal Config. = 47.01%
- GA-IR = 46.94%
- Ref. LSI = 30.93%
- Ref. VSM = 29.94%



RQ2: How do the IR techniques and configurations instantiated by GA-IR **compare** to those **previously used in literature** for the same tasks?

Clustering Hypothesis



Thanks!

Question?