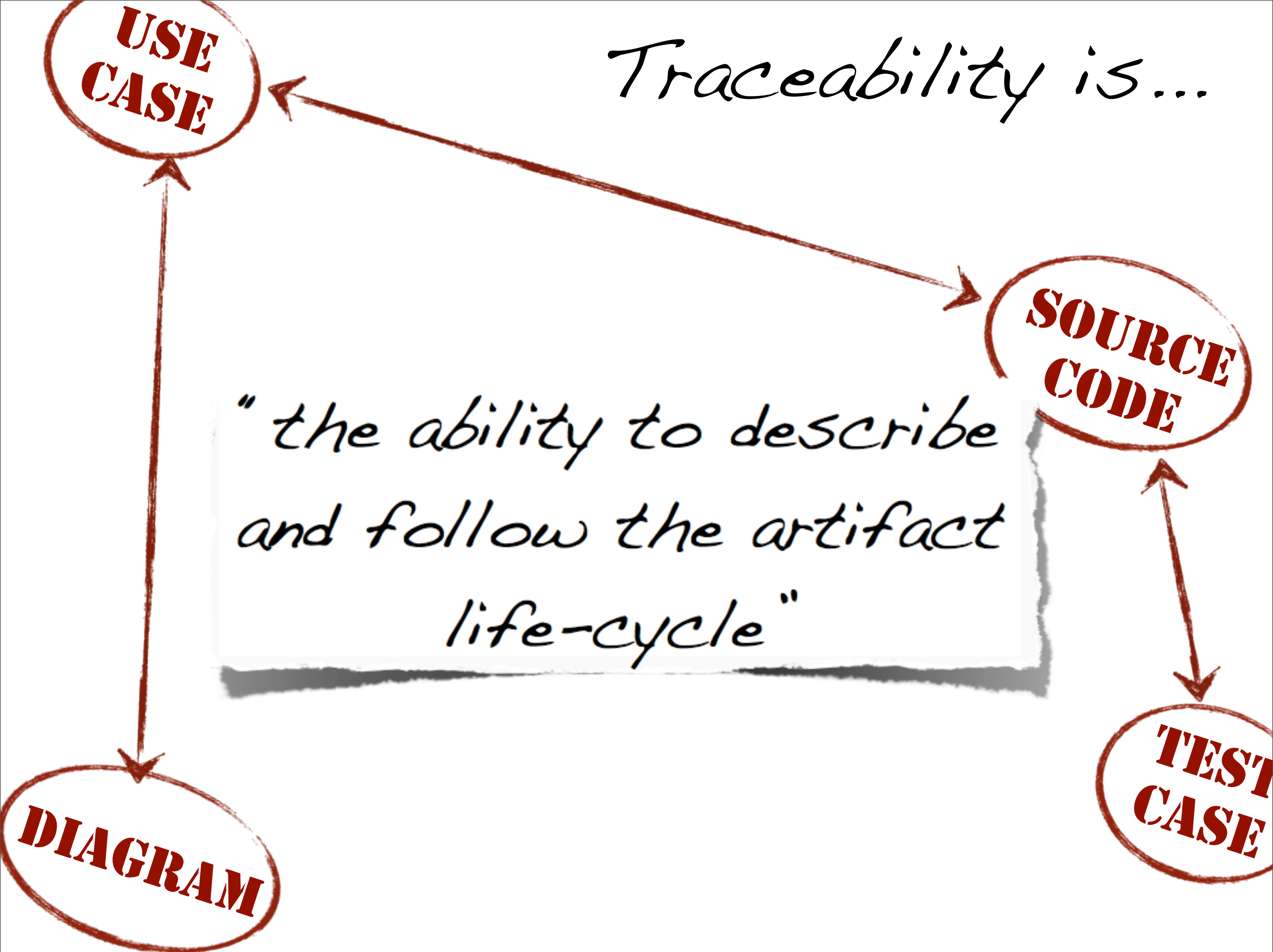


# *On Integrating Orthogonal Information Retrieval Methods to Improve Traceability Recovery*

*Malcom Gethers, Rocco Oliveto, Denys Poshyvanyk,  
Andrea De Lucia*



*Traceability is...*



# Traceability in practice

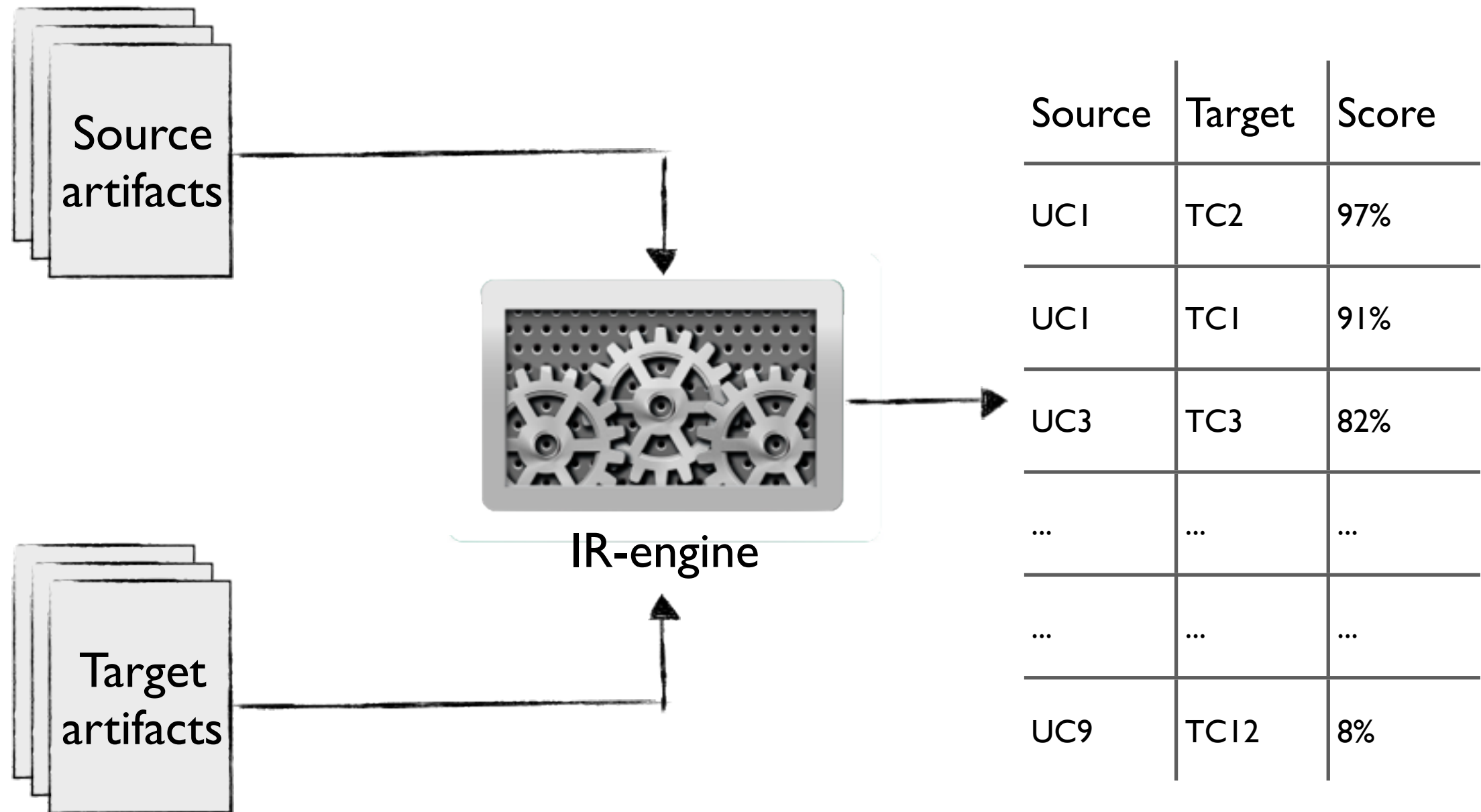
**USE  
CASE**

**SOURCE  
CODE**

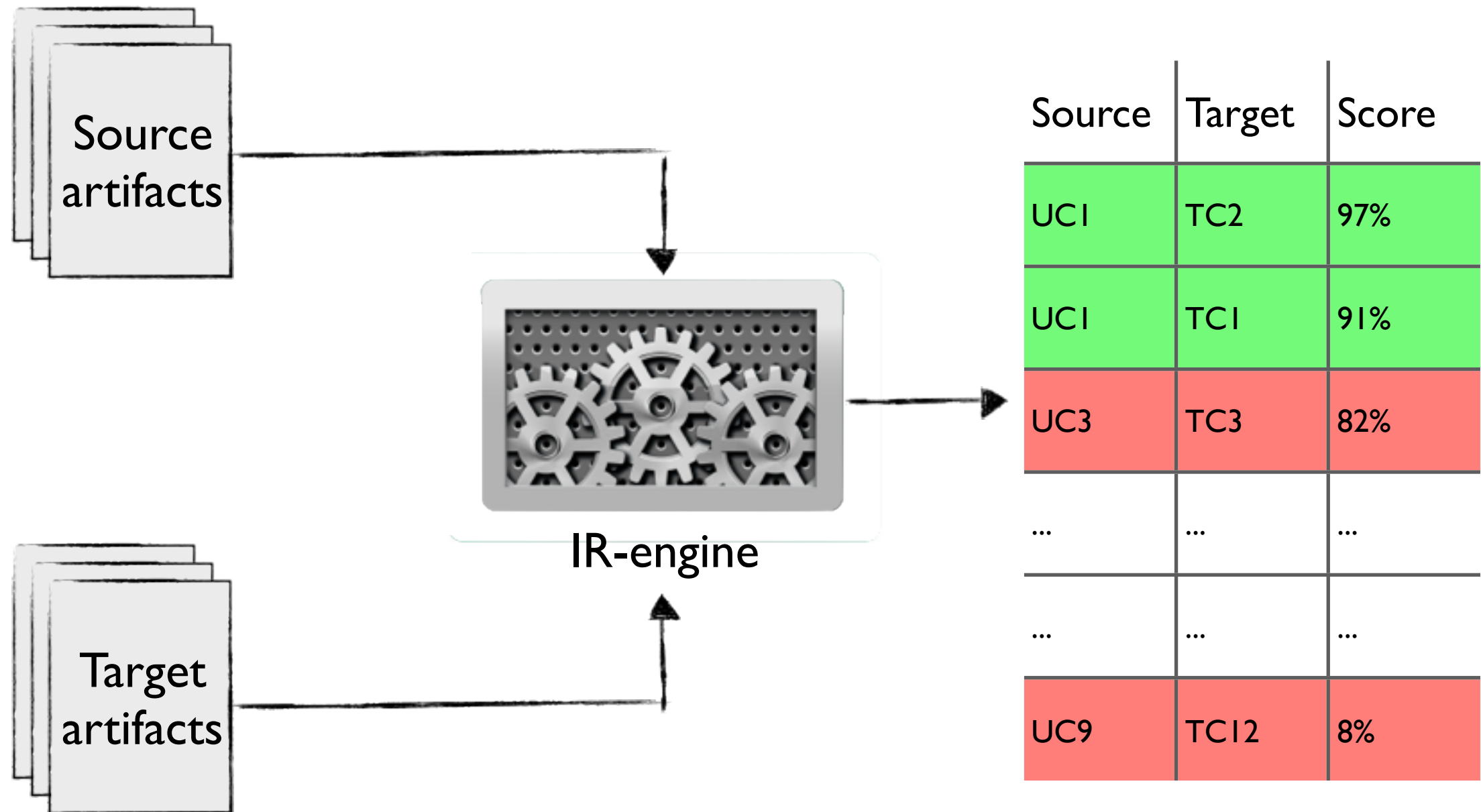
Traceability information  
is still not commonplace  
in software projects!

**DIAGRAM**

**TEST  
CASE**



*IR-based traceability recovery process*



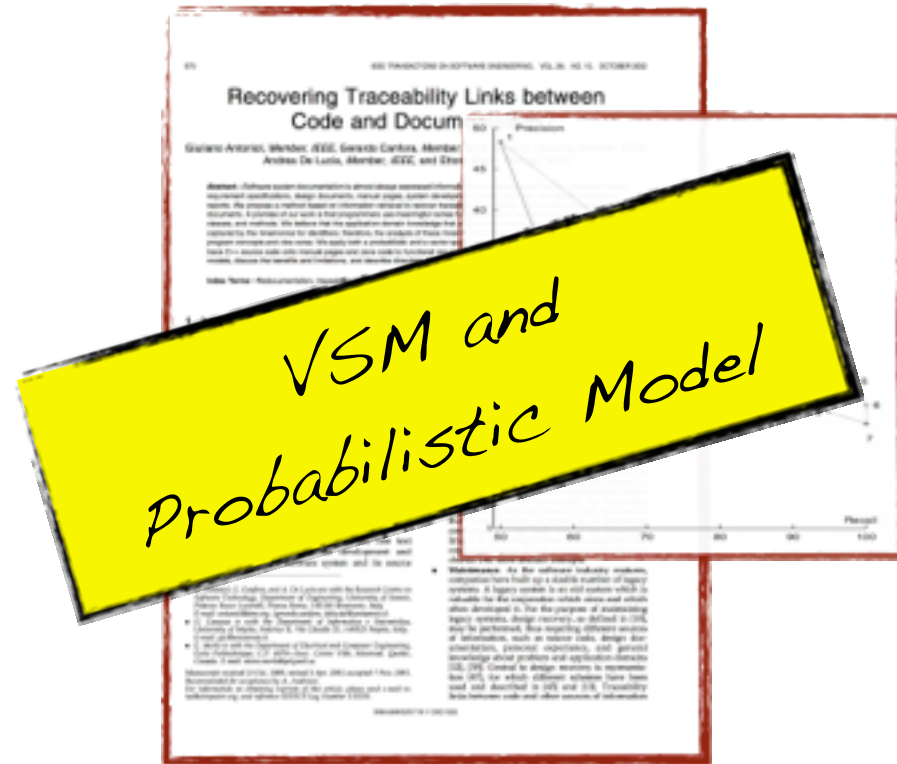
*IR-based traceability recovery process*

# State of the art

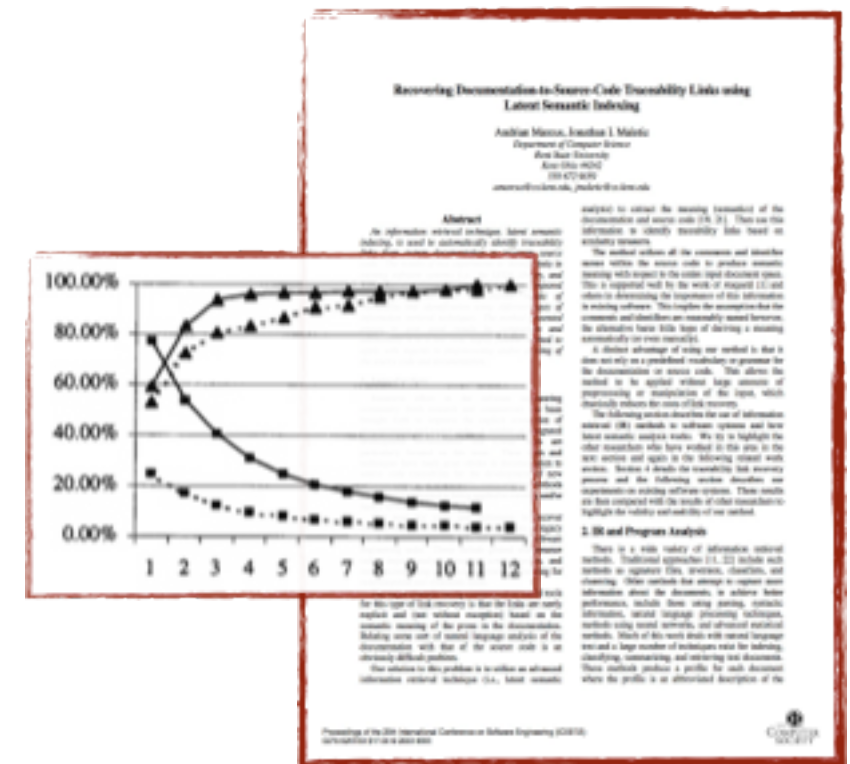
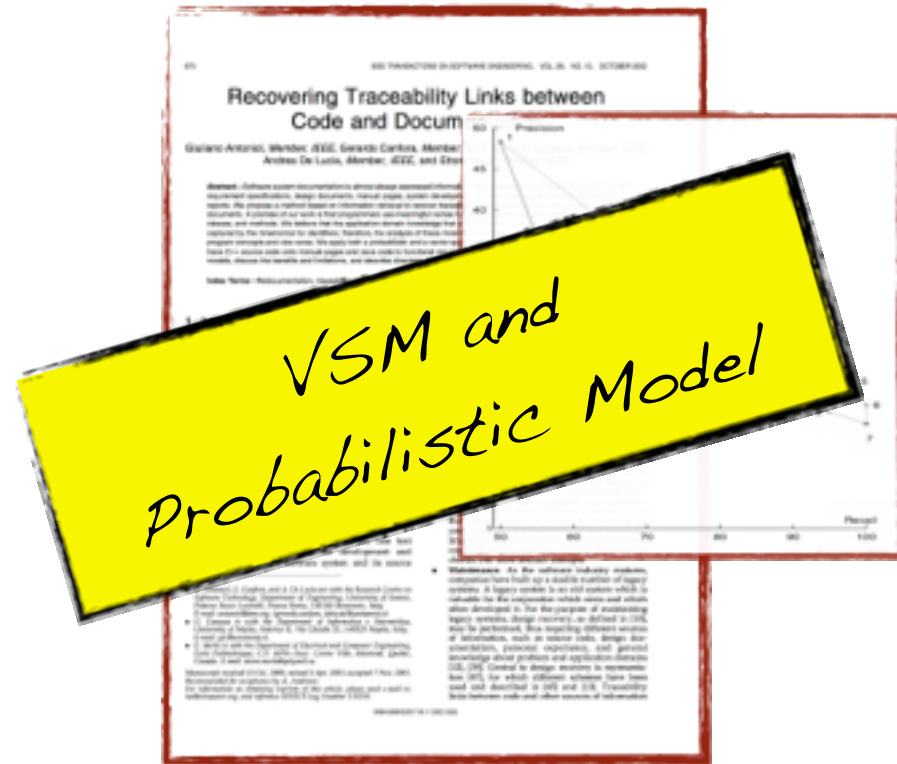




# State of the art

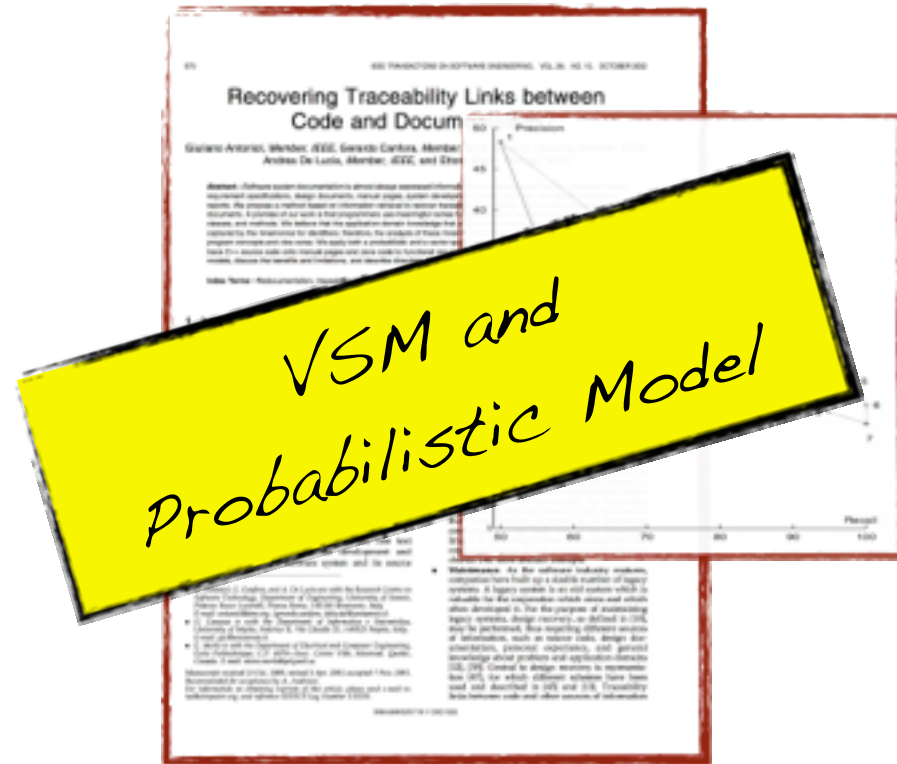


# State of the art

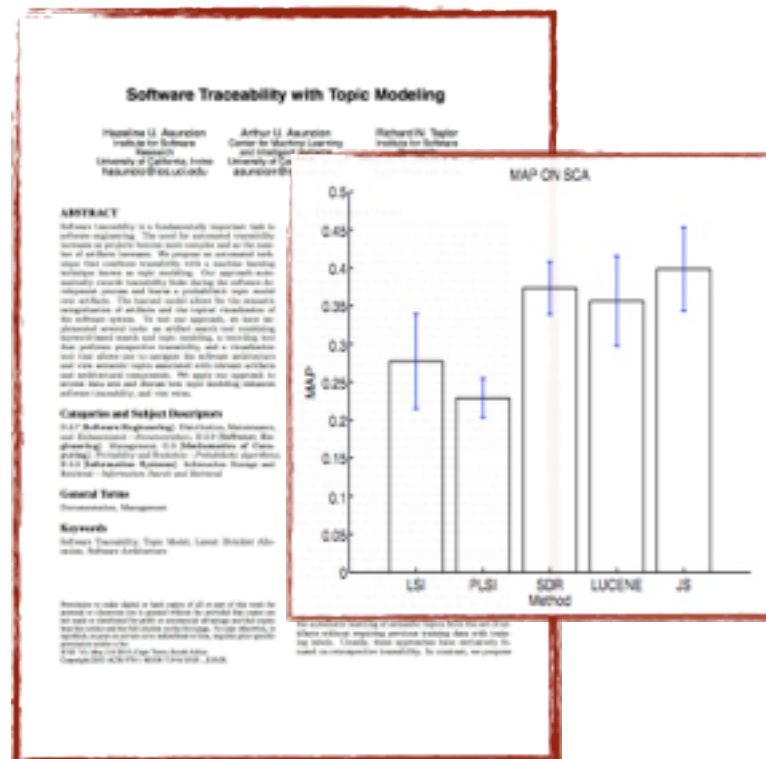
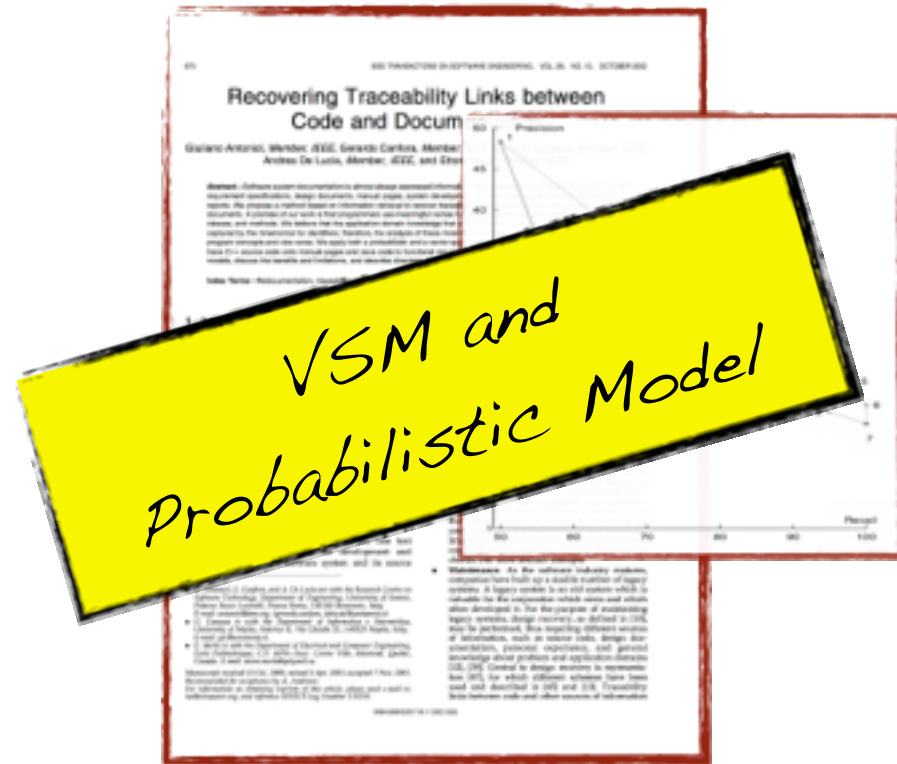




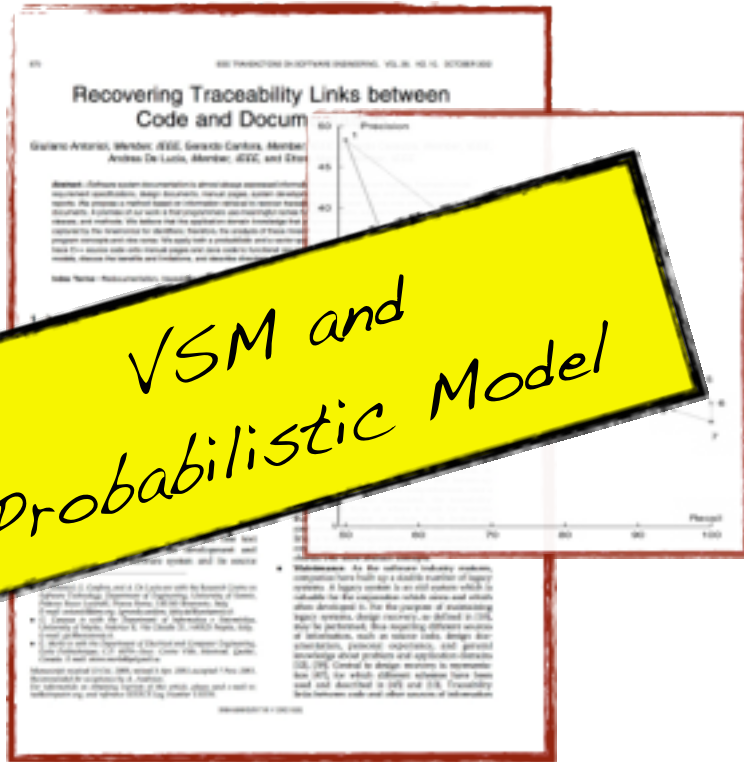
# State of the art



# State of the art



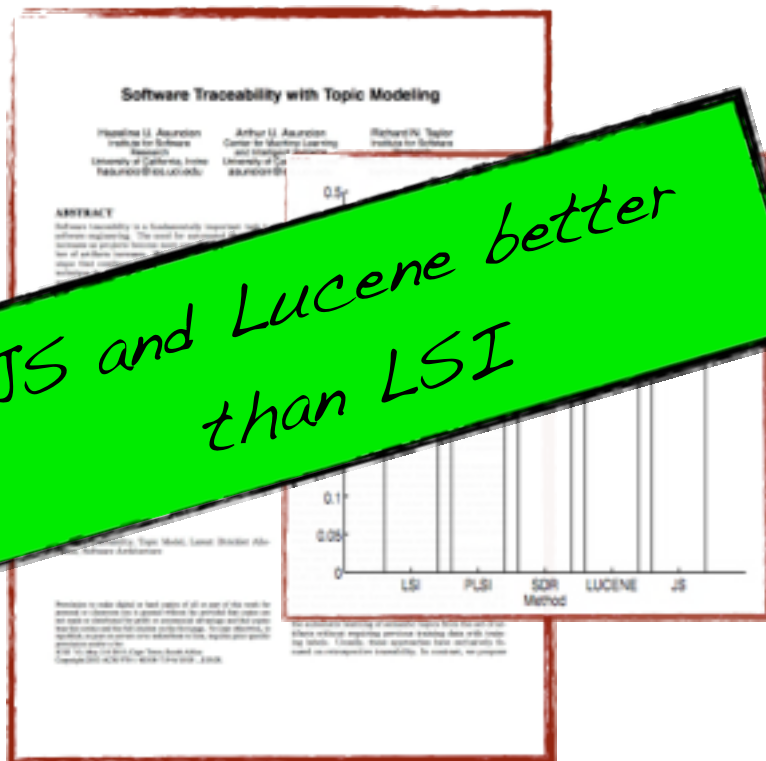
# State of the art



# VSM and Probabilistic Model



LSI better than VSM and Probabilistic Model



JS and Lucene better than LSI

# State of the art

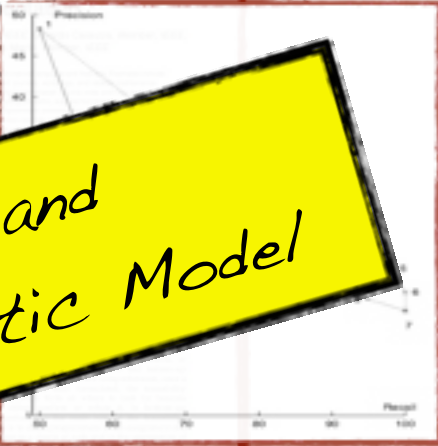
## Recovering Traceability Links between Code and Docum

Guillermo Amorim, Member, IEEE, Gerardo Carlos, Member, IEEE, and Andre De Luis, Member, IEEE, and Eric

**Abstract** Software system documentation is a fundamental component of software development. It provides a means to communicate the design and implementation of a system to its users and maintainers. However, the lack of traceability between code and documentation is a common problem in software development. This paper presents a probabilistic model for recovering traceability links between code and documentation. The model is based on the Vector Space Model (VSM) and the Probabilistic Model (PM). The model is evaluated using a dataset of software system documentation and code. The results show that the model is able to recover traceability links between code and documentation with a high degree of accuracy.

Index Terms—Probabilistic Model, VSM, PM

VSM and Probabilistic Model

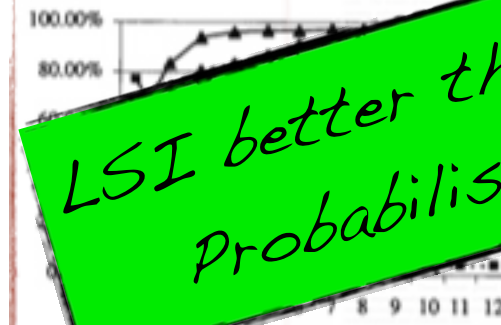


## Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing

Andrian Miron, Lucian I. Miron  
Department of Computer Science  
Babeş-Bolyai University  
Cluj-Napoca, Romania  
{miron@cs.ubb.ro, lucian@cs.ubb.ro}

**Abstract** In software development, source code and documentation are two fundamental components. They are closely related and often used together. However, the lack of traceability between source code and documentation is a common problem in software development. This paper presents a probabilistic model for recovering traceability links between source code and documentation. The model is based on the Latent Semantic Indexing (LSI) model. The model is evaluated using a dataset of software system documentation and source code. The results show that the model is able to recover traceability links between source code and documentation with a high degree of accuracy.

Index Terms—Latent Semantic Indexing, LSI, PM



LSI better than VSM and Probabilistic Model

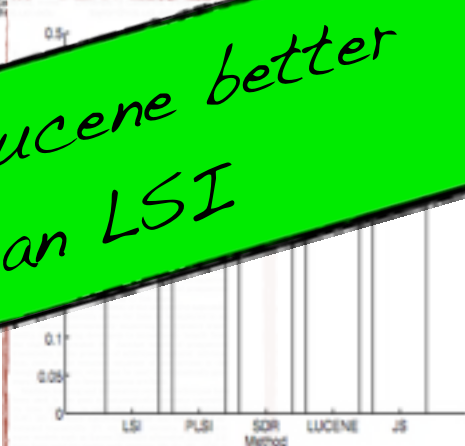
## Software Traceability with Topic Modeling

Hassane U. Aouici, Member, IEEE, Arthur U. Aouici, Member, IEEE, and Richard W. Taylor, Member, IEEE

**Abstract** Software traceability is a fundamental component of software development. It provides a means to communicate the design and implementation of a system to its users and maintainers. However, the lack of traceability between code and documentation is a common problem in software development. This paper presents a probabilistic model for recovering traceability links between code and documentation. The model is based on the Topic Modeling (TM) model. The model is evaluated using a dataset of software system documentation and code. The results show that the model is able to recover traceability links between code and documentation with a high degree of accuracy.

Index Terms—Topic Modeling, TM, PM

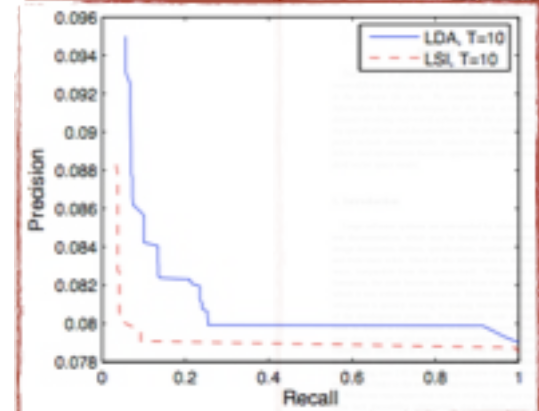
JS and Lucene better than LSI



## The 10th IEEE International Conference on Program Comprehension

### A Traceability Technique for Specifications

Abdullah Alshaykh, Member, IEEE, and Tahar Bouadja, Member, IEEE  
IBM Haifa Research Lab  
Haifa, 31099 Israel  
{alshaykh@il.ibm.com, bouadja@il.ibm.com}



**Abstract** Software traceability is a fundamental component of software development. It provides a means to communicate the design and implementation of a system to its users and maintainers. However, the lack of traceability between code and documentation is a common problem in software development. This paper presents a probabilistic model for recovering traceability links between code and documentation. The model is based on the Latent Dirichlet Allocation (LDA) model. The model is evaluated using a dataset of software system documentation and code. The results show that the model is able to recover traceability links between code and documentation with a high degree of accuracy.

Index Terms—Latent Dirichlet Allocation, LDA, PM

### 3. Related Work

Many researches have explored the effectiveness of different Information Retrieval (IR) methods for automatically detecting links between software artifacts in general, and between requirements and code in particular. Decker et al. [1] consider Latent Semantic Indexing (LSI) [2] for this purpose. Their conclusion is that LSI effectively discovers dependencies between the software artifacts used. Amorim et al. [3] compare a probabilistic IR model to a vector space model for tracing code to documents. They applied the algorithms in [1] and [3] and have shown that the LSI model is more effective than the vector space model. In order to trace requirements and functional requirements, respectively. They conclude that both methods are useful for tracing dependencies between documents, neither of them is significantly more effective than the other. Amorim et al. [3] applied LSI for the traceability between the code documents as Amorim et al. They conclude that LSI is more effective for this purpose, achieving both higher precision and recall values. De Luis et al. [4] developed a traceability recovery tool based on LSI. They evaluated the effectiveness of their tool during the development of software projects for multiple and concluded that their tool is helpful. For not sufficiently precise. Indurkhya and Minton [5] explore a combination of Formal Concept Analysis (FCA) and LSI for detecting dependencies in source code. First, a query is generated from the FCA and then the results are filtered by LSI and a subset of the



# State of the art

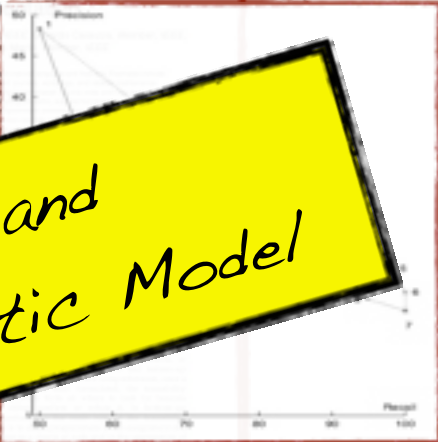
## Recovering Traceability Links between Code and Docum

Giuseppe Amato, Member, IEEE, Gerardo Caruso, Member, IEEE, and Andrea De Luca, Member, IEEE, and Stefano

**Abstract** Software system documentation is a critical asset for software development. It provides a structured and organized view of the system, its components, and its behavior. However, the manual process of maintaining and updating the documentation is often tedious and error-prone. This paper presents a novel approach to recovering traceability links between source code and documentation. The approach is based on a probabilistic model that captures the relationships between the two. The model is trained on a set of annotated documents and code snippets. The results show that the proposed approach outperforms existing methods in terms of precision and recall.

Index Terms—Software development, documentation, traceability, probabilistic model.

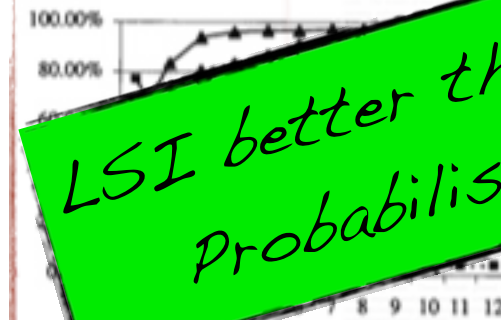
VSM and Probabilistic Model



## Recovering Documentation-to-Source-Code Traceability Links using Latent Semantic Indexing

Andrian Miron, London I. Miron, Department of Computer Science, York University, Toronto, Canada, andro@yorku.ca, liron@yorku.ca

**Abstract** The information retrieval techniques, latent semantic indexing (LSI) and its probabilistic variant, latent semantic analysis (LSA), have been widely used in the past to recover traceability links between source code and documentation. However, these techniques are not designed to handle the complex relationships between the two.



LSI better than VSM and Probabilistic Model

## Software Traceability with Topic Modeling

Hassane U. Aouici, Institute for Software Research, University of California, Irvine, haouici@uci.edu

Arthur U. Aouici, Center for Machine Learning and Intelligent Systems, University of California, Irvine, aaouici@uci.edu

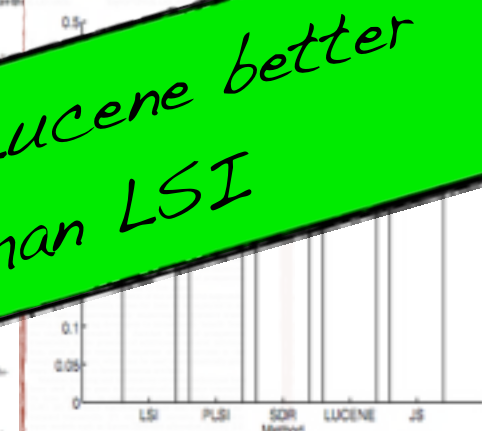
Richard W. Taylor, Institute for Software Research, University of California, Irvine, rtaylor@uci.edu

**ABSTRACT**

Software traceability is a fundamental requirement for software development. It provides a structured and organized view of the system, its components, and its behavior. However, the manual process of maintaining and updating the documentation is often tedious and error-prone. This paper presents a novel approach to recovering traceability links between source code and documentation. The approach is based on a probabilistic model that captures the relationships between the two. The model is trained on a set of annotated documents and code snippets. The results show that the proposed approach outperforms existing methods in terms of precision and recall.

Index Terms—Software development, documentation, traceability, probabilistic model.

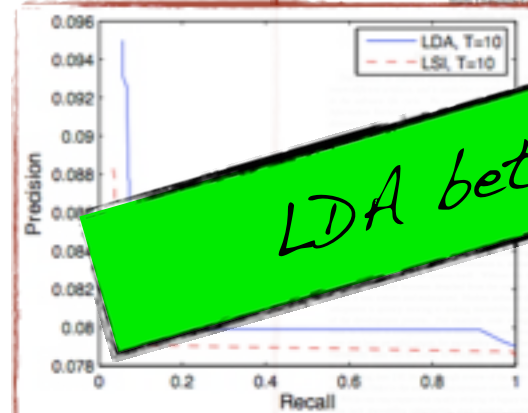
JS and Lucene better than LSI



The 14th IEEE International Conference on Program Comprehension

## A Traceability Technique for Specifications

Abdullah Alshaykh, Mohamed Elmaghrabi, and Tahar Kechel, IBM Research Lab, York University, Canada, alshaykh@yorku.ca, elmaghrabi@yorku.ca, kechel@yorku.ca



LDA better LSI

This paper presents a novel approach to recovering traceability links between source code and documentation. The approach is based on a probabilistic model that captures the relationships between the two. The model is trained on a set of annotated documents and code snippets. The results show that the proposed approach outperforms existing methods in terms of precision and recall.

# State of the art

VSM and Probabilistic Models

LSI better than VSM and Probabilistic Model

Constraining results! No clear winner among the IR methods

JS and Lucene better than LSI

LDA better LSI



# Our previous study...

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

Rocco Oliveto\*, Malcolm Getters<sup>†</sup>, Denys Poshyvanyk<sup>‡</sup>, Andrea De Lucia\*

\*Department of Mathematics and Informatics, University of Salerno, via Ponte don Melillo, Fisciano (SA), Italy

<sup>†</sup>Computer Science Department, The College of William and Mary, Williamsburg, VA 23185, USA

<sup>‡</sup>roliveto@unisa.it, mgetters@cs.wm.edu, denys@cs.wm.edu, adelucia@unisa.it

**Abstract**—We present an empirical study to statistically analyze the equivalence of several traceability recovery methods based on Information Retrieval (IR) techniques. 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 the Jensen-Shannon (JS) method, Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Latent Dirichlet Allocation (LDA). The results show that while JS, VSM, and LSI are almost equivalent, LDA is able to capture a dimension unique to the set of techniques which we considered.

**Keywords**—Traceability Recovery; Information Retrieval; Empirical Studies.

### I. INTRODUCTION

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 (e.g., [3], [4]). 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 Model (VSM) and probabilistic model [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 [5], VSM, LSI, and Latent Dirichlet Allocation (LDA) [6]. The first three methods were selected because they are widely used and seem to be the techniques that give the best results [5], [3], [4]. LDA is not as widely used for traceability link recovery though it has been used recently [16]. However, we also experiment such a technique for traceability recovery because LDA is able to capture some aspects missed by other IR methods, such as LSI, when it is used in other contexts [7].

The empirical analysis has been conducted on two software repositories, namely EasyClinic and eTour. The studied IR methods have been used to recover traceability links between the use cases and the source code of the two software systems. The results prove that the accuracy of LDA is lower than previously used methods. However, while JS, VSM, and LSI are almost equivalent, LDA is able to capture some information missed by the other exploited IR methods. These considerations suggest that probably LDA can be used as a method to augment canonical methods—e.g., JS, VSM, and LSI—aiming at improving their accuracy.

The rest of the paper is organized as follows. Section II discusses related work, while Section III briefly describe an IR-based traceability recovery process. Sections IV and V provide details on the design of the case study and report the results achieved, respectively. Section VI gives concluding remarks.

### II. RELATED WORK

The use of IR methods for traceability recovery was introduced by Amoriot *et al.* [3]. They apply probabilistic and Vector Space Models [1] to trace source code onto software documentation. Later, other IR methods (e.g., LSI, JS method and Numerical Analysis) have been proposed to recover links between different types of artifacts [4], [5], [8]. In particular, IR methods have been used to recover traceability between requirements [9], between requirements and design artifacts [10], between maintenance requests and software documents [11], and between other types of artifacts (e.g., use cases and UML diagrams) [12], [13].

All these reported case studies compare different IR-based traceability recovery approaches using recall and precision.

Which is the best IR method?

Is the ranking of these methods different?

# Our previous study...

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

Rocco Oliveto\*, Malcolm Getters<sup>†</sup>, Denys Poshyvanyk<sup>‡</sup>, Andrea De Lucia\*

\*Department of Mathematics and Informatics, University of Salerno, via Ponte don Melillo, Fisciano (SA), Italy

<sup>†</sup>Computer Science Department, The College of William and Mary, Williamsburg, VA 23185, USA

<sup>‡</sup>roliveto@unisa.it, mgetters@cs.wm.edu, denys@cs.wm.edu, adelucia@unisa.it

**Abstract**—We present an empirical study to statistically analyze the equivalence of several traceability recovery methods based on Information Retrieval (IR) techniques. 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 the Jensen-Shannon (JS) method, Vector Space Model (VSM), Latent Semantic Indexing (LSI), and Latent Dirichlet Allocation (LDA). The results show that while JS, VSM, and LSI are almost equivalent, LDA is able to capture a dimension unique to the set of techniques which we considered.

**Keywords**—Traceability Recovery; Information Retrieval; Empirical Studies.

### I. INTRODUCTION

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 (e.g., [3], [4]). 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 Model (VSM) and probabilistic model [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 [5], VSM, LSI, and Latent Dirichlet Allocation (LDA) [6]. The first three methods were selected because they are widely used and seem to be the techniques that give the best results [5], [3], [4]. LDA is not as widely used for traceability link recovery though it has been used recently [16]. However, we also experiment such a technique for traceability recovery because LDA is able to capture some aspects missed by other IR methods, such as LSI, when it is used in other contexts [7].

The empirical analysis has been conducted on two software repositories, namely EasyClinic and eTour. The studied IR methods have been used to recover traceability links between the use cases and the source code of the two software systems. The results prove that the accuracy of LDA is lower than previously used methods. However, while JS, VSM, and LSI are almost equivalent, LDA is able to capture some information missed by the other exploited IR methods. These considerations suggest that probably LDA can be used as a method to augment canonical methods—e.g., JS, VSM, and LSI—aiming at improving their accuracy.

The rest of the paper is organized as follows. Section II discusses related work, while Section III briefly describe an IR-based traceability recovery process. Sections IV and V provide details on the design of the case study and report the results achieved, respectively. Section VI gives concluding remarks.

### II. RELATED WORK

The use of IR methods for traceability recovery was introduced by Amoriot *et al.* [3]. They apply probabilistic and Vector Space Models [1] to trace source code onto software documentation. Later, other IR methods (e.g., LSI, JS method and Numerical Analysis) have been proposed to recover links between different types of artifacts [4], [5], [8]. In particular, IR methods have been used to recover traceability between requirements [9], between requirements and design artifacts [10], between maintenance requests and software documents [11], and between other types of artifacts (e.g., use cases and UML diagrams) [12], [13].

All these reported case studies compare different IR-based traceability recovery approaches using recall and precision.

Which is the best IR method?

Is the ranking of these methods different?

LDA had lowest accuracy

LDA is orthogonal as compared to VSM, JS, and LSI



# Motivation



*Is possible to improve the accuracy  
of topic-based model?*

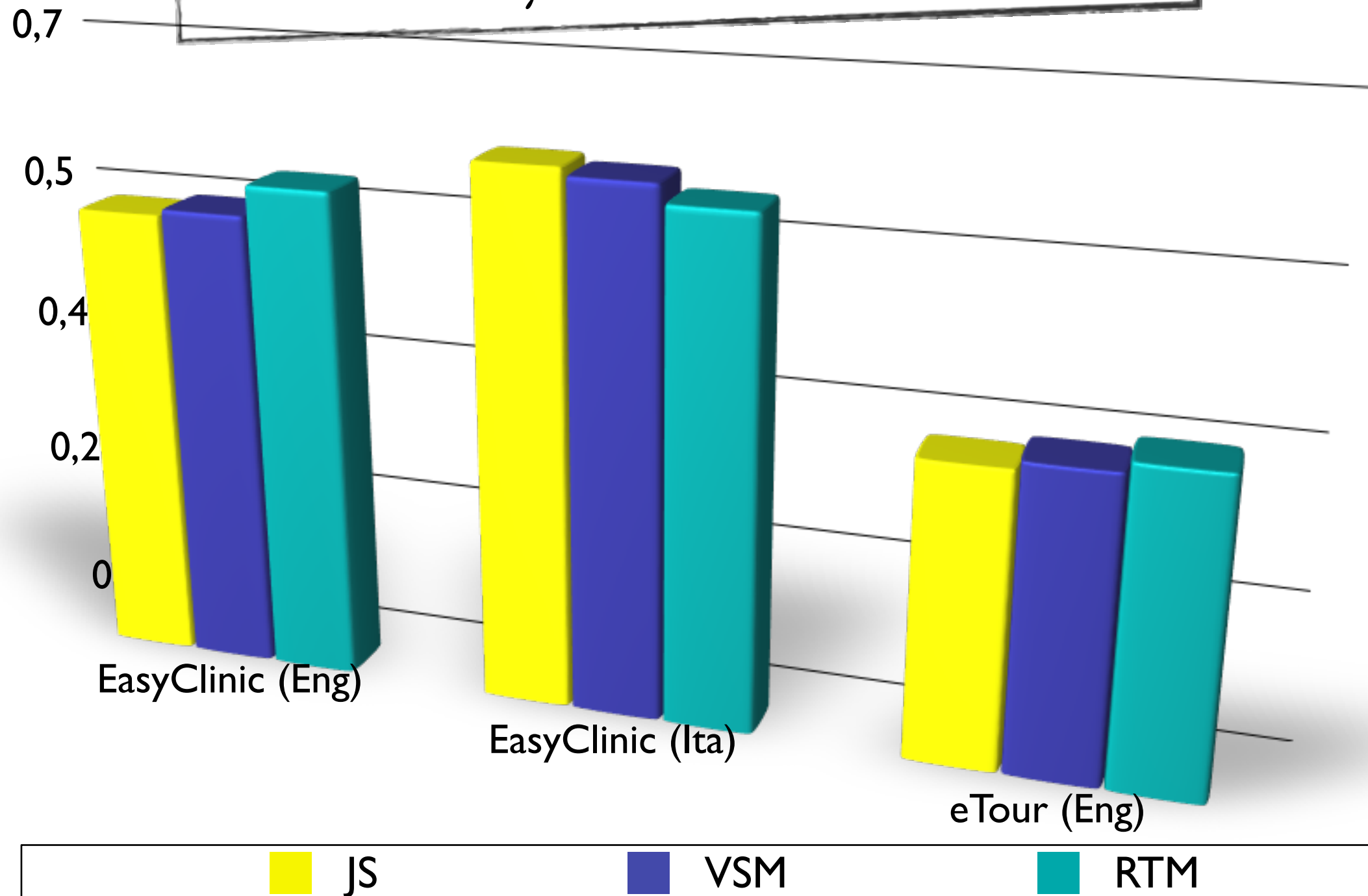
*Is it still possible to capture with other  
topic-based models information missed by  
other IR models?*

# Relational Topic Model (RTM)

Is possible to improve the accuracy  
of topic-based model?

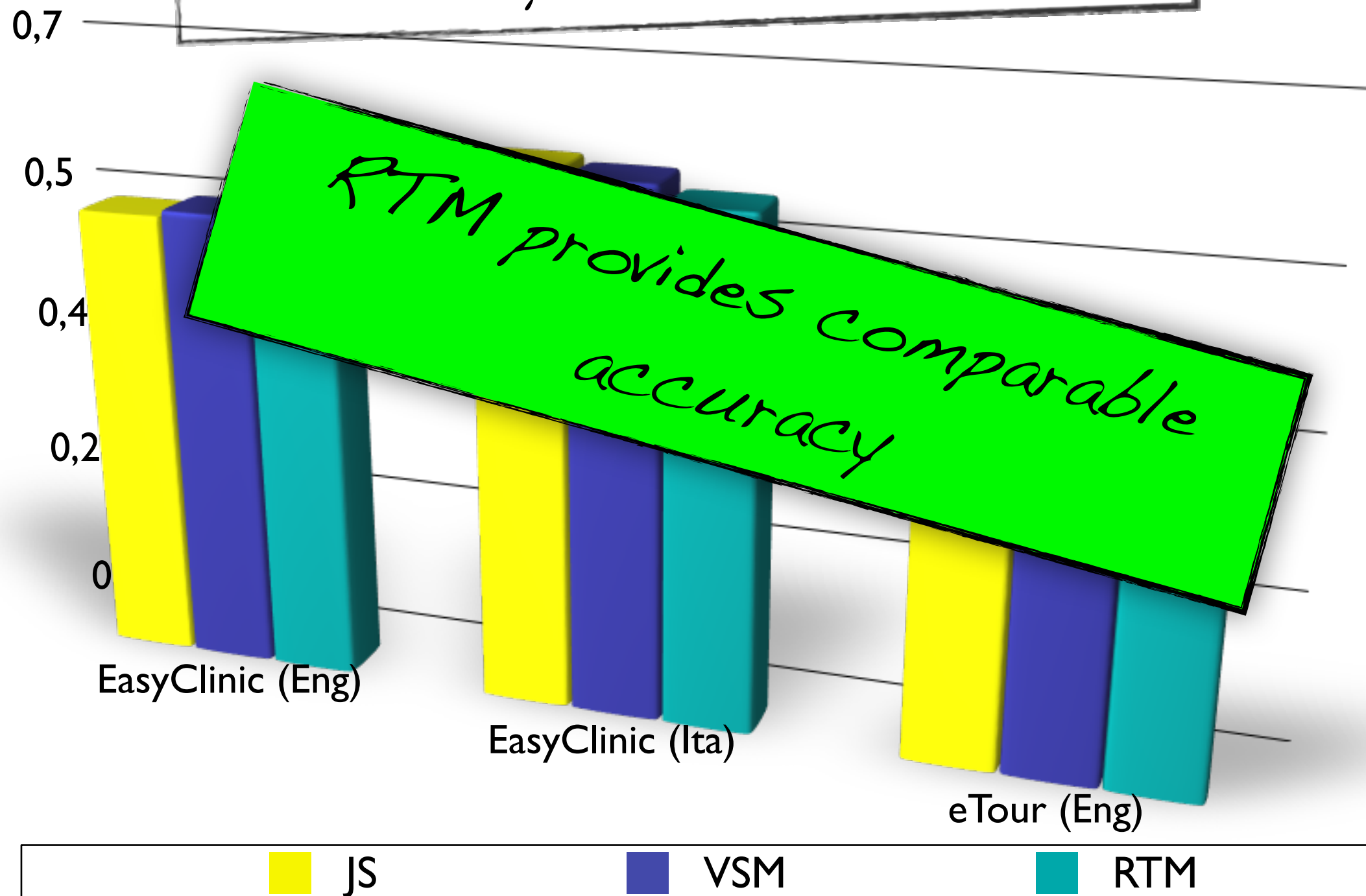
# Relational Topic Model (RTM)

*Is possible to improve the accuracy of topic-based model?*



# Relational Topic Model (RTM)

*Is possible to improve the accuracy of topic-based model?*





# Orthogonality of Techniques

Is still possible to capture with other topic-based models information missed by other IR models?

# Orthogonality of Techniques

	PC <sub>1</sub>	PC <sub>2</sub>	PC <sub>3</sub>
Proportio	68.51%	31.18%	0.31%
Cumulativ	68.51%	99.69%	100%
JS	0.99	0.11	0.07
VSM	0.99	0.08	-0.06
RTM	0.29	0.95	0.00

Is still possible to capture with other topic-based models information missed by other IR models?

# Orthogonality of Techniques

	25	50	100
correctINT(JS,VSM)	91%	94%	85%
correctDIFF(JS,VSM)	4%	5%	7%
correctDIFF(VSM,JS)	4%	0%	7%
correctINT(JS,RTM)	23%	28%	35%
correctDIFF(JS,RTM)	41%	35%	22%
correctDIFF(RTM,JS)	35%	36%	41%
correctINT(VSM,RTM)	23%	29%	32%
correctDIFF(VSM,RTM)	41%	32%	24%
correctDIFF(RTM,VSM)	35%	38%	42%

*Is still possible to capture with other topic-based models information missed by other IR models?*

# Orthogonality of Techniques

	25	50	100
correctINT(JS,VSM)	91%	94%	85%
correctDIFF(JS,VSM)	4%	5%	7%
correctDIFF(VSM,JS)	4%	0%	7%
correctINT(JS,RTM)	23%	28%	35%
correctDIFF(JS,RTM)	41%	35%	22%
correctDIFF(RTM,JS)	35%	36%	41%
correctINT(VSM,RTM)	23%	29%	32%
correctDIFF(VSM,RTM)	41%	32%	24%
correctDIFF(RTM,VSM)	35%	38%	42%

*Is still possible to capture with other topic-based models information missed by other IR models?*

# Orthogonality of Techniques

	25	50	100
correctINT(JS,VSM)	91%	94%	85%
correctDIFF(JS,VSM)	4%	5%	7%
correctDIFF(VSM,JS)	4%	0%	7%
correctINT(JS,RTM)	23%	28%	35%
correctDIFF(JS,RTM)	41%	35%	22%
correctDIFF(RTM,JS)	35%	36%	41%
correctINT(VSM,RTM)	23%	29%	32%
correctDIFF(VSM,RTM)	41%	32%	24%
correctDIFF(RTM,VSM)	35%	38%	42%

*Is still possible to capture with other topic-based models information missed by other IR models?*

# Orthogonality of Techniques

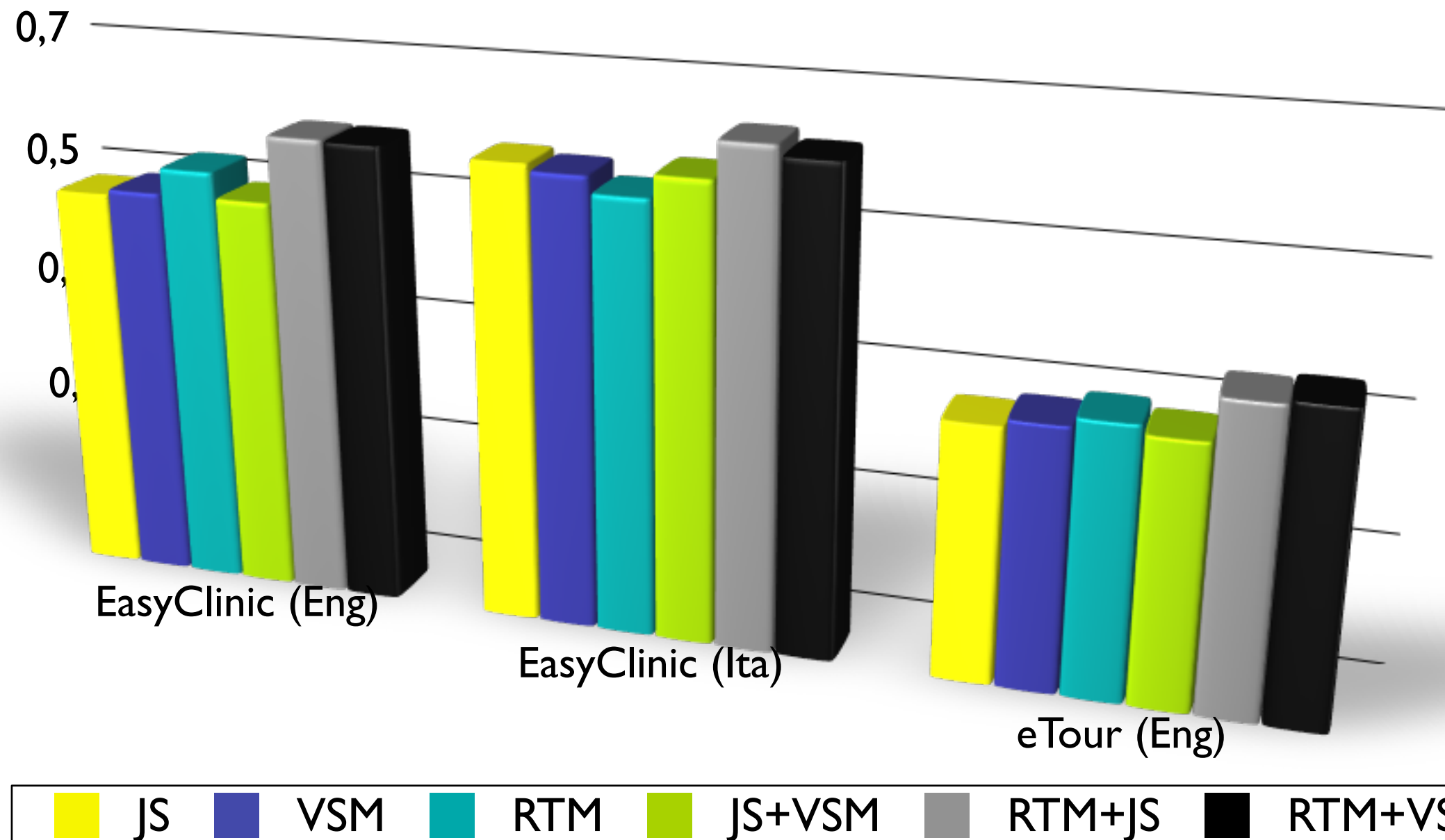
	25	50	100
correct <sub>INT</sub> (JS,VSM)	91%	94%	85%
correct <sub>INT</sub> (RTM,VSM)	4%	5%	7%
correct <sub>INT</sub> (RTM,JS)	0%	0%	7%
correct <sub>INT</sub> (VSM,RTM)			35%
correct <sub>DIFF</sub> (JS,RTM)			
correct <sub>DIFF</sub> (RTM,JS)	35%		
correct <sub>INT</sub> (VSM,RTM)	23%	29%	29%
correct <sub>DIFF</sub> (VSM,RTM)	41%	32%	24%
correct <sub>DIFF</sub> (RTM,VSM)	35%	38%	42%

RTM provides orthogonal results

Is still possible to capture with other topic-based models information missed by other IR models?

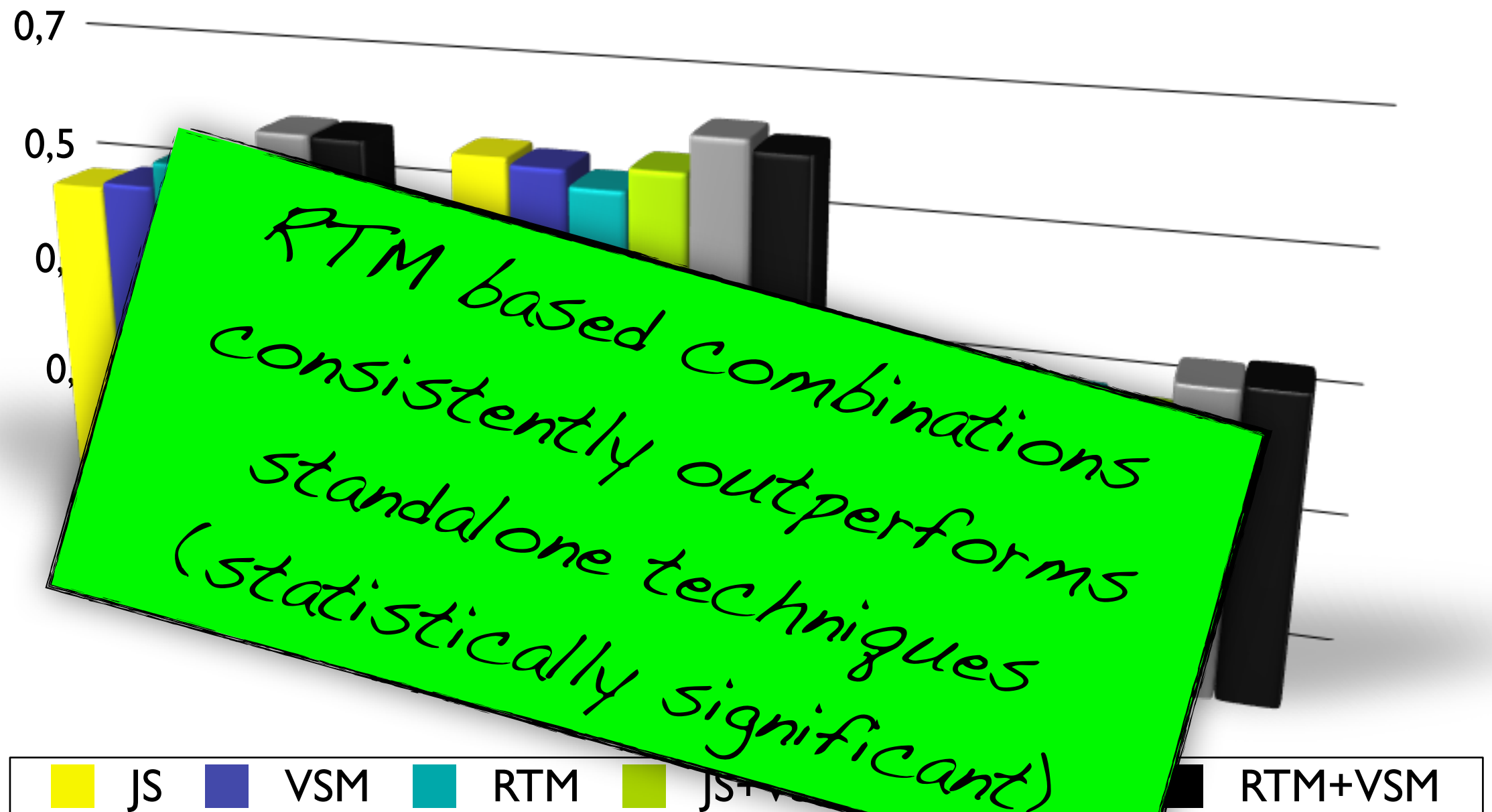


# Combining Orthogonal Techniques



*Is still possible to capture with other topic-based models information missed by other IR models?*

# Combining Orthogonal Techniques



Is still possible to capture with other topic-based models information missed by other IR models?

# Other Results

There is a statistically significant interaction between IR method and artifact type

There is a statistically significant interaction between IR method and language

# Conclusions

RTM provides accuracy comparable to existing IR techniques

RTM is able to capture orthogonal information

RTM combined with other IR techniques provides improved accuracy for traceability recovery

Conclusions

Thank You!

Questions?