

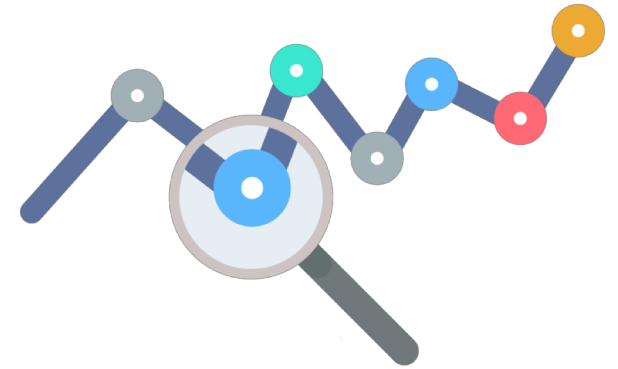
# Automated Traceability

Faruk Acibal & Dingguang Jin





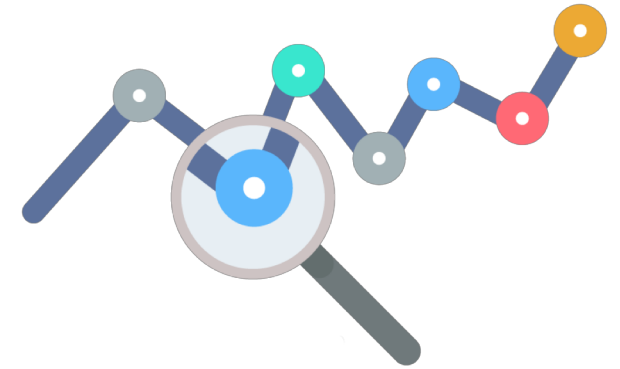
# The Problem



- Misconception: Traceability practices return little value for the effort
- Difficulties creating, assessing, using, and maintaining traceability links
- Manual traces often created in an ad hoc fashion
  - Inconsistent
  - Incomplete
  - Erroneous
  - Tedious
- Simple support for manual linking seems to not work
- Achieving traceability in projects that did not plan for it is very hard
- Manual work is cumbersome and leads to negative attitudes

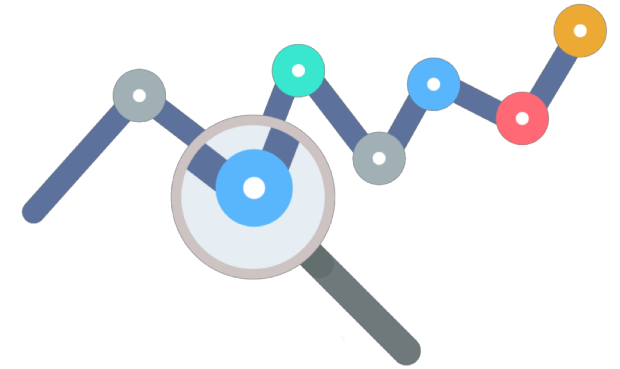


# The Solution

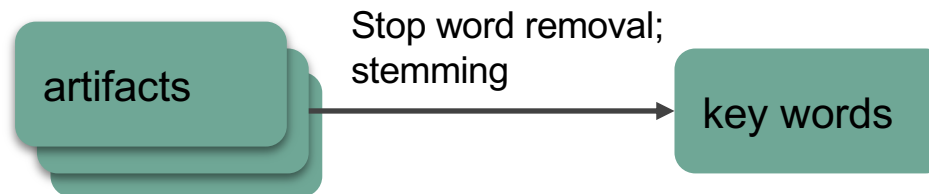


- Automation
- Information Retrieval algorithms (IR-Algorithms)
  - Vector-space model
  - Probabilistic network model
- Compute likelihood of a link Similarity score
  - Rarer terms  $\Rightarrow$  stronger indicators for potential link (tf-idf)
- Remove stop words and perform “stemming”

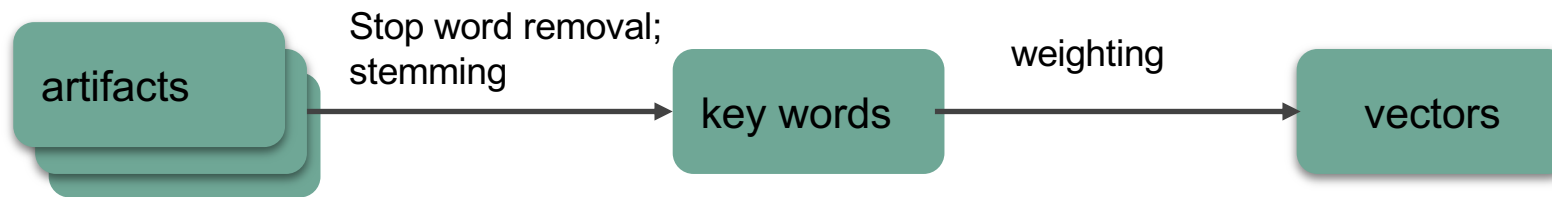
# Stop Words removal & Stemming



- Stop words: common words without strong meaning
  - The, this, in, etc.
- Stemming: unifying words
  - “Registers”, “registered”, “registering” ⇒ in the same basket



# Vector-Space Model



Use cosine to measure the similarity of each two vectors



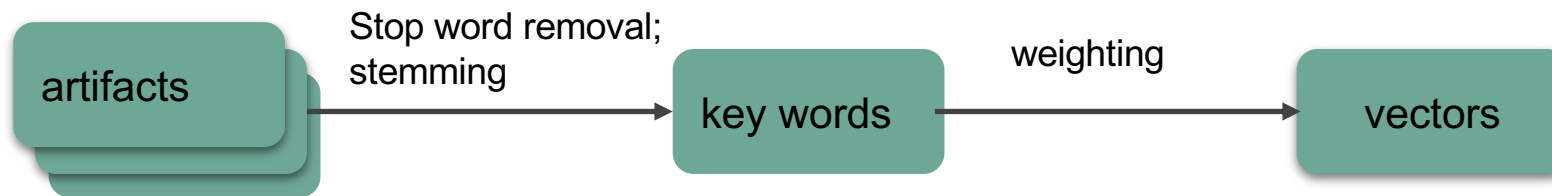
Most popular weighting scheme: tf-idf  
The weight of each term:

$$w \propto \frac{\text{how often the term is found in the document}}{\text{how often it is found in the entire collection}}$$

$$\text{sim}(d, q) = \cos(d, q) = \frac{\sum_{j=1}^N d_j \cdot q_j}{\sum_{j=1}^N d_j^2 \cdot \sum_{j=1}^N q_j^2}$$

User feedback?

# Vector-Space Model



Use cosine to measure the similarity of each two vectors



$$\text{sim}(d, q) = \cos(d, q) = \frac{\sum_{j=1}^N d_j \cdot q_j}{\sum_{j=1}^N d_j^2 \cdot \sum_{j=1}^N q_j^2}$$

**User feedback:**

One method is Rochio feedback processing method

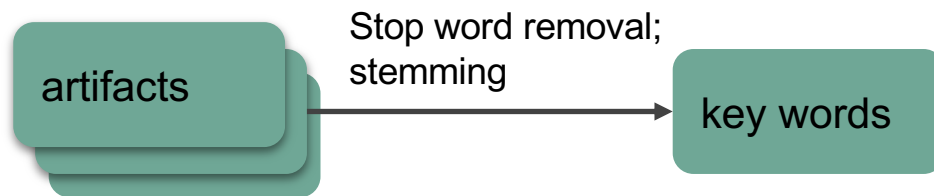
$$q_{\text{new}} = \alpha q + \left( \frac{\beta}{r} \sum_{d_j \in D_r} d_j \right) - \left( \frac{\gamma}{s} \sum_{d_k \in D_{\text{irr}}} d_k \right)$$

$\alpha, \beta, \gamma$  is constant

$D_r$  is relevant document given by user

$D_{\text{irr}}$  is irrelevant document given by user

# Probabilistic Network Model



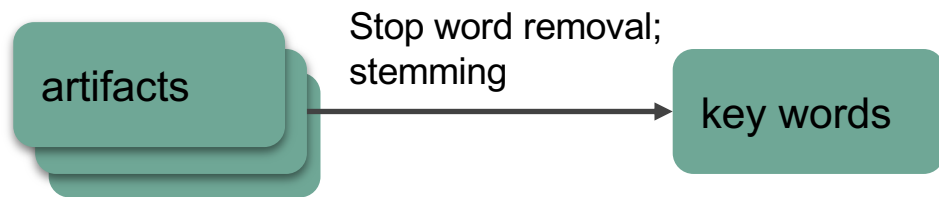
Artifacts usually have a hierarchical structure.

$$pr(a_j|q) = \left[ \sum_{i=1}^k pr(a_j|t_i) pr(q, t_i) \right] / pr(q).$$

Bayes' theorem

t is short for terms(= key words)

# Probabilistic Network Model



$$pr(a_j|q) = \left[ \sum_{i=1}^k pr(a_j|t_i) pr(q, t_i) \right] / pr(q).$$

t is short for terms(= key words)

Artifacts usually have a hierarchical structure.  
After Adding hierarchy information:

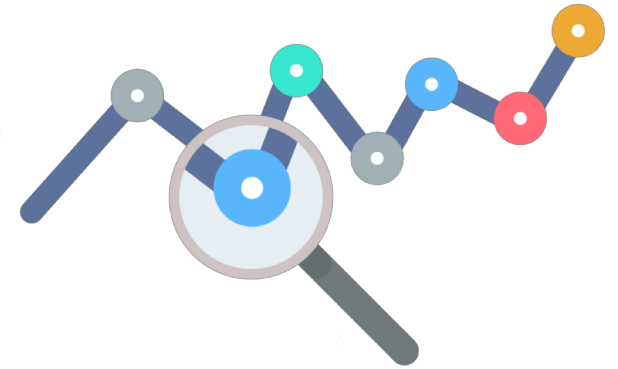
$$pr(a_j|q) = \left( \sum_{g \in pa_D(a_j)} \sum_i pr(a_j, g|t_i) pr(q, t_i) \right) / pr(q),$$

g is a parent of a.

If same term shows up in parents of  $a_j$ , then it's more probable that q and  $a_j$  should be linked.



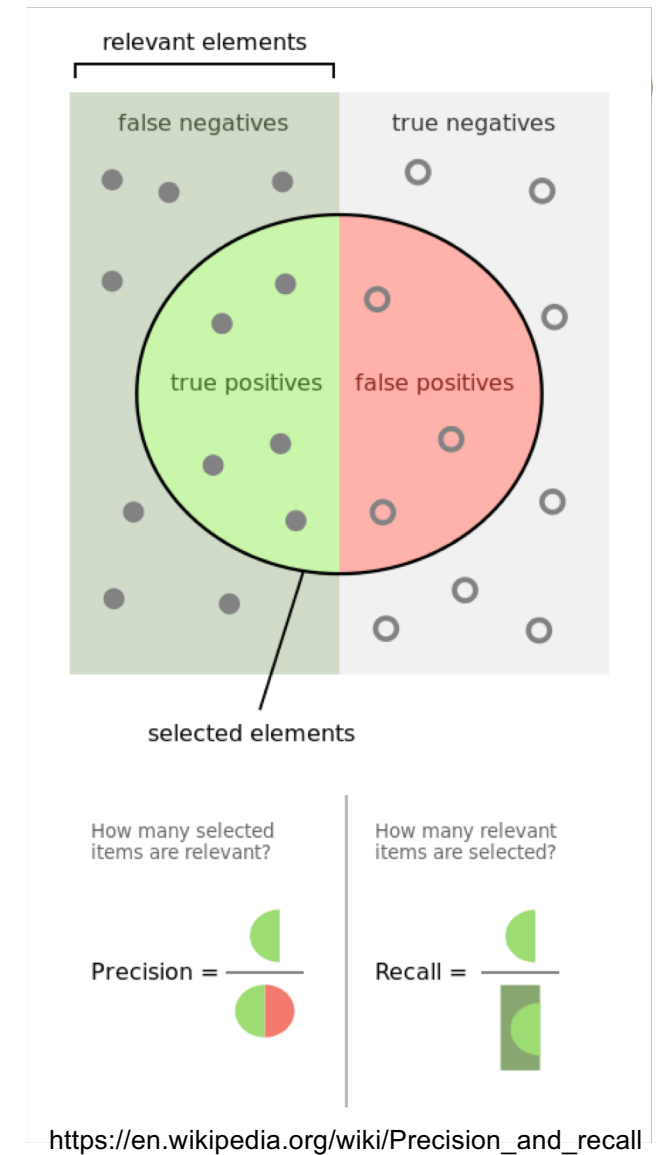
## Additional things to consider



- Let user propose omissions (since usually recall is around 90%)
- Nice to also produce confidence level of predictions
- Developers use many third-party tools to manage artifacts. In order to trace them, should provide customized adapters in an enterprise-level
- Stakeholders are interested in different subsets of artifacts and have different traceability needs
- Not only back end matters, front end also matters

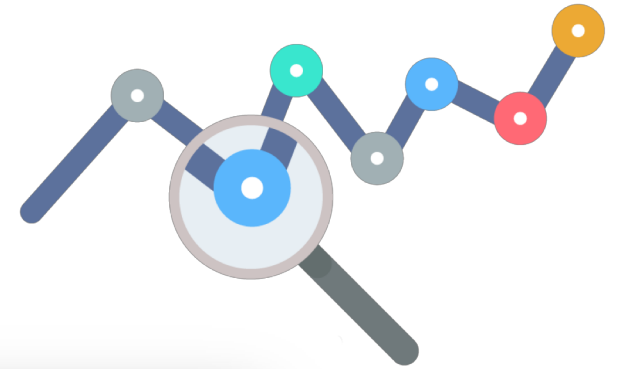
# Recall & Precision

- Precision
  - Percentage of retrieved documents that are relevant
- Recall
  - Percentage of relevant documents that were retrieved
- Recall more important in industry





# Best Practices



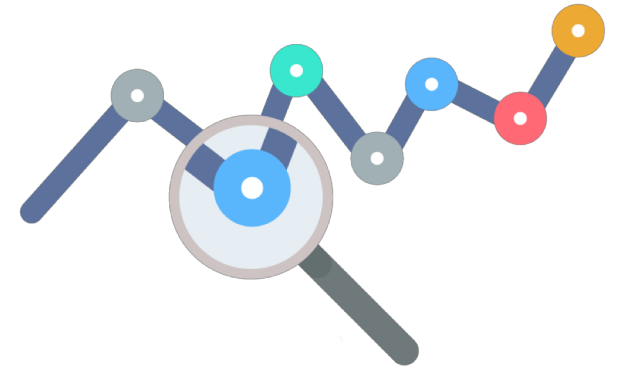
Traceability Environment

Creating Traceable Artifacts

Introducing Automated Trace  
Processes

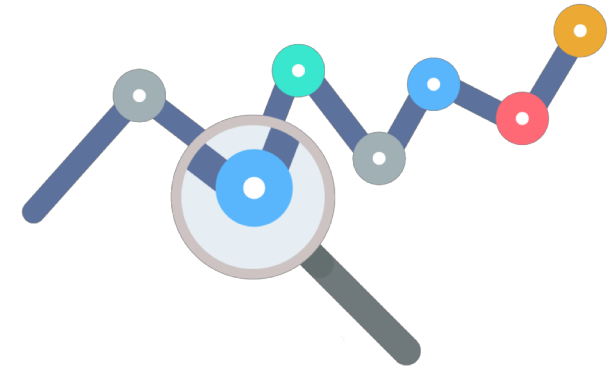


# Traceability Environment

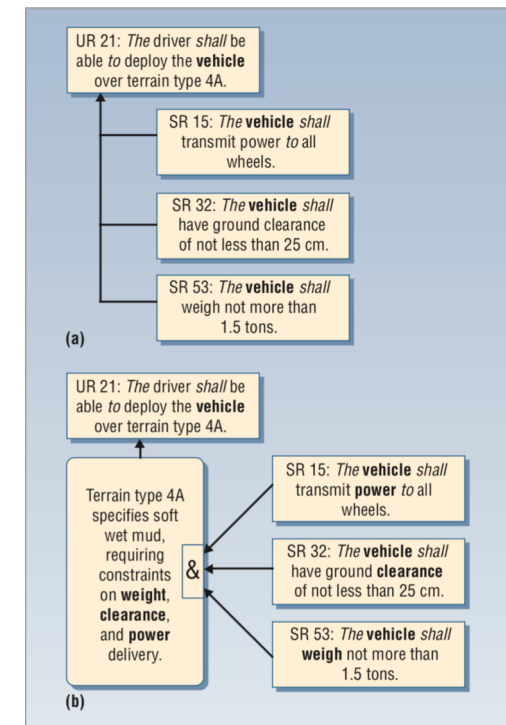


- Trace for a purpose
  - First define what needs to be traced and how
- Define a suitable trace granularity
  - Package, class, or method level  $\Rightarrow$  improving the granularity has diminishing returns
- Support in-place traceability
  - Let developers easily define traces during their work with artifacts

# Creating Traceable Artifacts

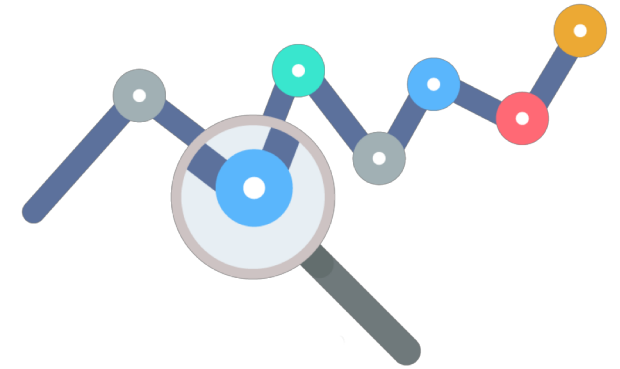


- Using a well-defined project glossary
  - Increases term usage  $\Rightarrow$  better (automated) traceability
- Writing quality requirements
- Constructing meaningful hierarchy
  - Closer ancestor  $\Rightarrow$  assume stronger information / connections
- Bridging the intradomain semantic gap
  - Intradomain synonyms
- Creating rich content
  - Add rationales & domain knowledge





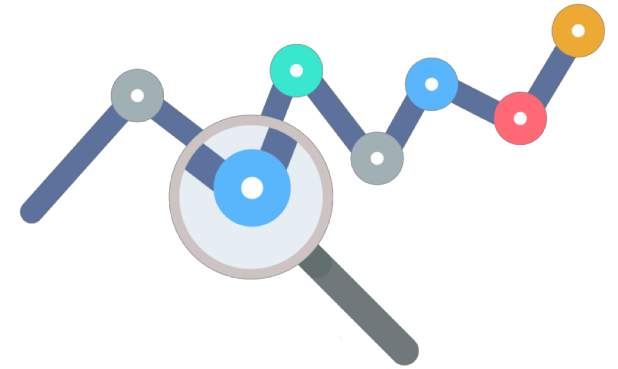
# Introducing Automated Trace Processes



- Using a process improvement plan
- 3 Scenarios
  - Organization with no traceability in place  $\Rightarrow$  cost effective solution
  - Organization with limited traceability  $\Rightarrow$  broaden the scope
  - Organization with traceability  $\Rightarrow$  replace with more cost effective ways
- 20-35% precision on 90% recall possible



# Literature



- Hayes, Jane Huffman, et al. "REquirements TRacing On target (RETRO): improving software maintenance through traceability recovery." Innovations in Systems and Software Engineering 3.3 (2007): 193-202.
- Cleland-Huang, Jane, et al. "Best practices for automated traceability." Computer 40.6 (2007).



## Thanks and further



1. User feedback have huge potential to be further incorporated into the model *Adaptive User Feedback for IR-based Traceability Recovery 2015*
2. We have considered one feature of artifacts, its hierarchy. What about others? Like versions. What should be done to trace links as software evolves?
3. The method we have seen is categorized as trace recovery--done afterwards. Another categorization is trace capture. Those links are used daily by developers. So trace capture is very reasonable, yet nearly no research is on it.

Trace Recovery

Trace Capture