Traceability Challenge 2011: Using TraceLab to Evaluate the Impact of Local versus Global IDF on Trace Retrieval

Adam Czauderna, Marek Gibiec, Greg Leach, Yubin Li
Yonghee Shin, Ed Keenan, Jane Cleland-Huang
Systems and Requirements Engineering Center, DePaul University
jhuang@cs.depaul.edu

ABSTRACT

Numerous trace retrieval algorithms incorporate the standard tf-idf (term frequency, inverse document frequency) technique to weight various terms. In this paper we address Grand Challenge C-GC1 by comparing the effectiveness of computing idf based only on the local terms in the query, versus computing it based on general term usage as documented in the American National Corpus. We also address Grand Challenges L-GC1 and L-GC2 by setting ourselves the additional task of designing and conducting the experiments using the alpha-release of TraceLab. TraceLab is an experimental workbench which allows researchers to graphically model and execute a traceability experiment as a workflow of components. Results of the experiment show that the local idf approach exceeds or matches the global approach in all of the cases studied.

Categories and Subject Descriptors

D.2.7 [**Software Engineering**]: Distribution, Maintenance, and Enhancement; D.2.1 [**Requirements/Specifications**]: Tools

General Terms

Experimentation, Measurement, Documentation

Keywords

Traceability, TraceLab, Challenge

1. INTRODUCTION

Requirements traceability is recognized as a critical software engineering activity which establishes relationships between requirements and other artifacts such as software designs, code, and test cases. Due to the time and effort required to manually create and maintain traceability links, researchers have explored the use of information retrieval methods for automating generating candidate links. Such approaches have included the vector space model (VSM) [5], probabilistic approaches [2], or Latent Semantic Indexing (LSI) [1, 7, 8]. Many approaches incorporate a technique known as tf-idf (term frequency, inverse document frequency) which generates similarity scores between a requirement and an artifact by considering the number of shared terms and the *weight* of those terms. In this paper we explore the impact of using term weights computed locally versus those computed according to their frequency in the American National Corpus [6]. Results are evaluated against four different datasets.

Our experiment directly addresses three challenges from the Grand Challenges of Traceability (GCT). The first challenge, C-GC1 [4] is related to trace recovery. We asked the question "Can the quality of generated traceability links be improved through computing idf (inverse document frequency) values based upon the general occurrence of words in written language versus their occurrence in project specific artifacts?" The second set of challenges, L-GC1 and L-GC2 [4], are related to defining standard processes, procedures, and benchmarks for performing traceability research. To this end, we conducted our experiments using TraceLab [3], a workbench currently in the early stages of development, which is designed for modeling and executing traceability experiments. This traceability challenge represents the first time TraceLab has been used to formally conduct a traceability experiment. We therefore asked the question "Can TraceLab be used to effectively support the idf traceability experiment?"

2. DOCUMENT FREQUENCY

All of the experiments reported in this paper utilize the vector space model (VSM). In the VSM, each query q and each document d is represented as a vector of terms $T = t_1, t_2, ..., t_n$ defined as the set of all terms in the set of queries. A document d is therefore represented as a vector $\vec{d} = (w_{1,d}, w_{2,d}, ..., w_{n,d})$, where $w_{i,d}$ represents the term weight of term i for document d. A query is similarly represented as $\vec{q} = (w_{1,q}, w_{2,q}, ..., w_{n,q})$. The standard weighting scheme known as $tf - idf$ is used to assign weights to individual terms [9], where tf represents the term frequency, and idf the inverse document frequency. Term frequency is computed for document d as $tf(t_i, d) = (freq(t_i, d))/(|d|)$, where $freq(t_i, d)$ is the frequency of the term in the document, and $|d|$ is the length of the document. Inverse document frequency idf, is typically computed as:

$$idf_{ti} = log_2(n/n_i) \quad (1)$$

TEFSE'11, May 23, 2011, Waikiki, Honolulu, HI, USA

Dataset	Description	Source		Target		# of Traces
		Name	Count	Name	Count	
CM1	Space Telescope	High Level reqs.	22	Low level reqs.	53	45
Easy Clinic	Electronic health care system	Use Cases	30	Classes	47	93
E-Tour	Tour cultural sites online	Use Cases	58	Java Code	116	327
WV-CCHIT	Health Information system	Requirements	116	Regulatory Requirements	1064	587

Table 1: Datasets used in idf experiment

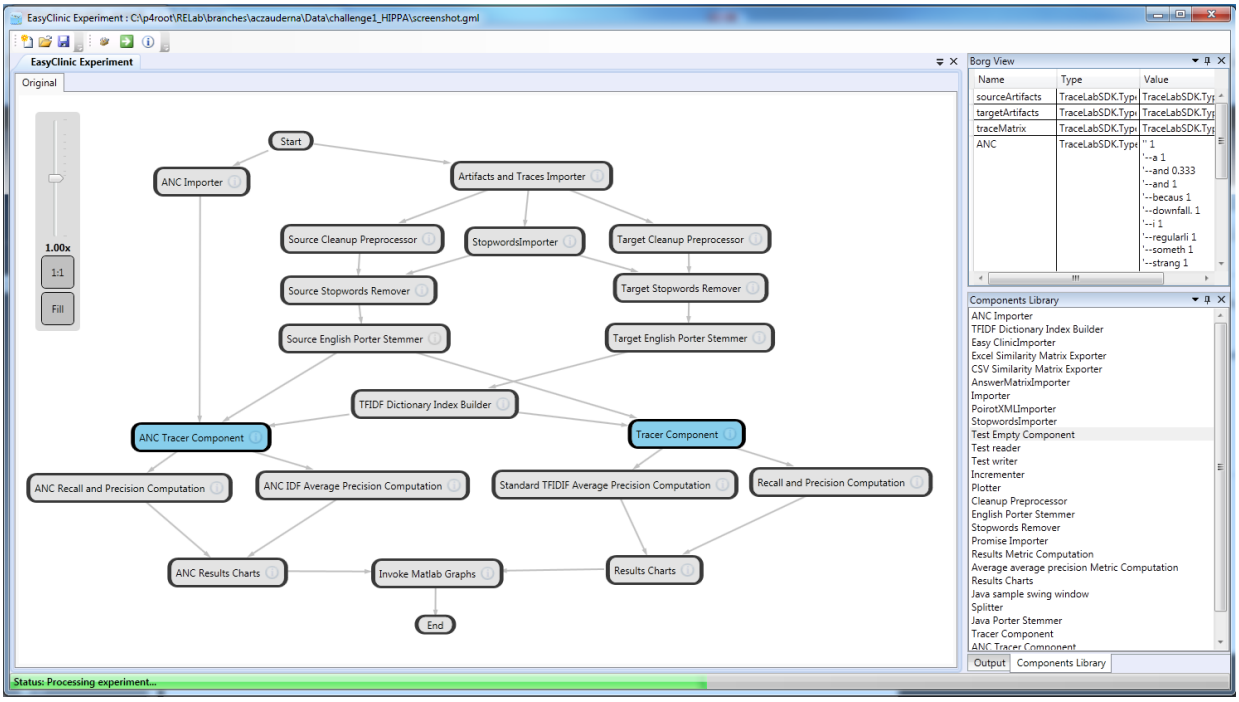


Figure 1: TraceLab workflow for comparing two IDF techniques

where n is the total number of documents in the traceable collection, and n_i is the number of documents in which term t_i occurs. The individual term weight for term i in document d is then computed as $w_{id} = tf(t_i, d) \times idf_{ti}$. A similarity score $sim(d, q)$ between document d and query q is computed as the cosine of the angle between the two vectors as

$$sim(d,q) = \frac{(\sum_{i=1}^{n} w_{i,d} w_{i,q})}{\left(\sqrt{\sum_{i=1}^{n} w_{i,d}} \cdot \sqrt{\sum_{i=1}^{n} w_{i,q}}\right)} \quad (2)$$

The typical practice of computing *idf* relative to its occurrence in project-specific requirements, enables terms which are rare in a given project to be weighted more highly than more common ones. However this weighting scheme can result in false positives (i.e. retrieving incorrect links) when terms which are actually quite common in general language usage occur disproportionately few times in the traced dataset. In this case, the terms may be assigned high weights by a local *idf* algorithm even though they are relatively unimportant in general use. We hypothesized that this problem might be mitigated through computing the *idf* of a term using the ANC as follows:

$$idf\_ANC_{ti} = 1/min(rf_{ti}, 1000) \quad (3)$$

where rf_{ti} is the relative frequency assigned to term ti in the ANC collection. Furthermore, if term ti was not found in the ANC, it was considered to be rare, and idf_ANC_{ti} was set to 1. We hypothesized that this approach would produce *idf* weightings better aligned with their normal use. However, computing *idf* based on a general corpus might also result in false negatives if generally commonplace terms take on specific meanings in a local project.

3. EXPERIMENTAL DESIGN

In the remainder of this paper we comparatively evaluate the use of *local* vs. *global* idf values, computed using equations (1) and (3) respectively. Experiments were conducted using the CM1, EasyClinic, E-Tour, and WV-CCHIT datasets depicted in Table 1. For each dataset, traces were generated using the VSM approach, first utilizing the standard *idf* method and secondly utilizing the *global* method. To execute the experiment, we modeled a workflow using TraceLab. The workflow, depicted in Figure 1, included two distinct experimental treatments, and was constructed using the following types of components. **Importer components** such as the *ANC importer* and *Artifacts and Traces* importer were used to import the data from the four case studies, and transform it into standard TraceLab data types. **Preprocessors** were used to prepare raw artifacts for trac-

```
namespace Preprocessor
{
    [Component(GuidIDString = "420775E4-1AFC-4142-9145-F32A7D1ED8C4",
        Name = "English Porter Stemmer",
        DefaultLabel = "English Porter Stemmer",
        Description = "This Pre-Processor stemms the words to their roots.  It uses the Porter stemming algorithm.",
        Author = "DePaul RE Lab Team",
        Version = "1.0",
        ConfigurationType=null)]
    [IOSpec(IOSpecAttribute.IOSpecType.Input, "listOfArtifacts", typeof(TraceLabSDK.Types.TLArtifactList))]
    [IOSpec(IOSpecAttribute.IOSpecType.Output, "listOfArtifacts", typeof(TraceLabSDK.Types.TLArtifactList))]
    public class PreprocessorStemmerComponent : BaseComponent
    {
        public PreprocessorStemmerComponent(IBorg borg) : base(borg) { }

        public override void Compute()
        {
            System.Diagnostics.Trace.WriteLine("Start component PreprocessorStemmerComponent");

            TLArtifactList listOfArtifacts = (TLArtifactList)Borg.Load("listOfArtifacts", typeof(TLArtifactList));

            foreach (TLArtifact artifact in listOfArtifacts)
            {
                artifact.ProcessedText = PreprocessorStemmer.Process(artifact.ProcessedText);
            }

            Borg.Store("listOfArtifacts", listOfArtifacts);

            System.Diagnostics.Trace.WriteLine("Completed component PreprocessorStemmerComponent");
        }
    }
}
```

Figure 2: The *stemmer* component integrated into the TraceLab environment

ing. For example the *stop words remover*, and *stemmer* represent standard information retrieval functions which were used to remove common words and stem terms to their root forms respectively. The **dictionary index builder** was used to index the terms and compute term frequencies in each artifact. The ANC and standard **Tracer** components generated candidate traceability links through computing the cosine similarity between each pair of source artifacts and target artifacts. The **Results analysis components**, such as the *ANC results chart* and the *Invoke Matlab Graphs* components computed metrics and displayed results.

Each component was written in java, C#, or C++ and then modified to make it compatible with TraceLab. Modifications include two steps of (i) defining meta data that documents the name, purpose, version, and interface of the component, and (ii)using the TraceLab SDK to define I/O calls on the TraceLab data cache. Both of these modifications are depicted for the *Stemmer* component in Figure 2. A TraceLab experiment is modeled as a precedence graph, showing the order in which components must be executed. TraceLab is multi-threaded and can therefore support concurrent execution of components. Precedence is typically established according to data dependencies. For example, in Figure 1 the *Artifact and Traces Importer* reads use cases from an xml file, transforms them into a standard TraceLab datatype, and writes the data to the TraceLab data cache. This same data structure is used and modified by downstream components such as the *stemmer* and *stopper*.

4. RESULTS

Traceability results are often evaluated using the standard metrics of *recall*, which measures the fraction of relevant documents which are correctly retrieved, *precision*, which measures the fraction of retrieved documents which are relevant, and finally *FPR* (false positive rate), which measures the number of non-relevant documents retrieved as a fraction of the total number of non-relevant documents. In this paper, we utilize recall, precision, and FPR to compute average precision, interpolated precision, and ROC as follows:

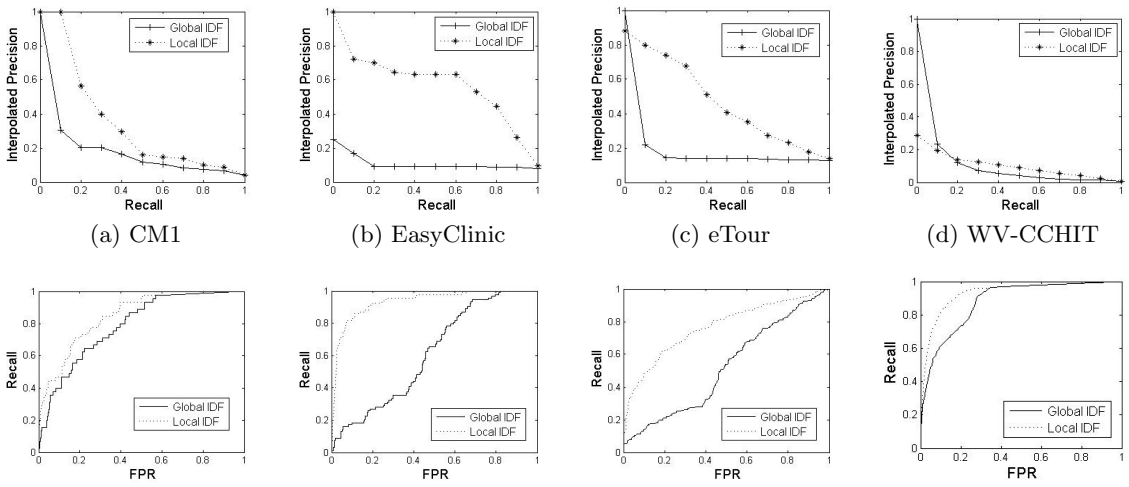
$$AveragePrecision = \frac{\sum_{r=1}^{N}(P(r) * relevant(r))}{Num.OfRelevantDocuments} \quad (4)$$

where r is the rank of the document in the ordered set of candidate traceability links, N is the number of retrieved documents, $relevant()$ is a binary function assigned 1 if the rank is relevant and 0 otherwise, and $P(r)$ is the precision computed after truncating the list immediately below that ranked position. Average Precision returns a higher value (approaching 1) when more relevant documents are retrieved towards the top of the ranked list. Interpolated precision reports precision at fixed recall levels, while compensating for peculiarities of peaks and valleys in the recall/precision graph. Interpolated Precision is computed as follows:

$$InterpolatedPrecision(r) = \max_{r'>=r} P(r') \quad (5)$$

where r is the recall at a certain recall level, $P(r')$ is the precision for any recall level $r' >= r$. Interpolated Precision is computed at 11 recall levels (0.0, 0.1, 0.2, ... 1.0) to compare models. Finally, a ROC graph shows the trade-off between recall and false positive rate.

Average precision results, reported in Table 2, show that local idf values outperformed global ones in all four cases, with most significant differences in the Easy Clinic and E-Tour datasets. The interpolated precision and ROC results resported in Figure 3, also indicate that local idf values outperformed global ones. Differences were particularly pronounced in the case of Easy Clinic and E-Tour; however this could be explained by the fact that both datasets contain Italian terms not found in the ANC. In the CM1 dataset,



(a) CM1 (b) EasyClinic (c) eTour (d) WV-CCHIT

(e) CM1 (f) EasyClinic (g) eTour (h) WV-CCHIT

Figure 3: Interpolated precision-recall graphs(a-d) and ROC graphs(e-h)

Table 2: Average Precision

Dataset	Local idf	Global idf
CMI	0.482	0.386
Easy Clinic	0.721	0.196
E-Tour	0.331	0.192
WV-CCHIT	0.296	0.294

the local approach also outperformed the global one; however this is potentially explained by the fact that CM1 contains numerous highly technical terms which are also not found in the ANC. It was due to these possible issues that we added a fourth dataset to the three standard challenge datasets. The WV-CCHIT dataset does not contain overly technical or Italian terms, and in this case the two idf methods returned almost identical average precision scores, but showed higher interpolated precision at lower levels of recall, and lower interpolated precision at higher levels.

5. CONCLUSIONS

This report evaluated the impact of computing idf values using frequencies obtained from the ANC instead of from the local dataset. Results show that the local idf approach outperformed the global idf approach in three of the four datasets, and performed equivalently in the fourth one. An analysis of the results shows that the global approach significantly under-performs with datasets containing terms not found in the ANC. Future work could therefore investigate a hybrid approach which utilizes global rankings for terms in the ANC and local weightings for other terms. Finally, this experiment demonstrated the viability of using TraceLab to model and execute a realistic traceability experiment, including making calls to Matlab in order to analyze and report results.

6. ACKNOWLEDGMENTS

The work described in this paper was primarily funded by Major Research Instrumentation grant #CNS-0959924 from the U.S. National Science Foundation.

7. REFERENCES

[1] G. Antoniol, G. Canfora, G. Casazza, A. D. Lucia, and E. Merlo. Recovering traceability links between code and documentation. *IEEE Trans. Software Eng.*, 28(10):970–983, 2002.

[2] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova. Best practices for automated traceability. *IEEE Computer*, 40(6):27–35, 2007.

[3] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. Huffman Hayes, E. Keenan, G. Leach, J. Maletic, D. Poshyvanyk, Y. Shin, A. Zisman, G. Antonio, B. Berenbach, A. Egyed, and P. Maeder. Grand challenges, benchmarks, and tracelab: Developing infrastructure for the software traceability research community. 2011.

[4] J. Cleland-Huang, A. Dekhtyar, J. Huffman Hayes, G. Antoniol, B. Berenbach, A. Egyed, S. Ferguson, J. Maletic, and A. Zisman. *Grand Challenges of Traceability*. Center of Excellence for Software Traceability, http://www.coest.org, 2006.

[5] J. Huffman Hayes, A. Dekhtyar, S. K. Sundaram, and S. Howard. Helping analysts trace requirements: An objective look. In *RE*, pages 249–259, 2004.

[6] C. Macleod, N. Ide, and R. Grishman. The american national corpus: Standardized resources for american english. In *Second Language Resources and Evaluation Conf.* American National Corpus Project, 2000.

[7] A. Marcus, J. I. Maletic, and A. Sergeyev. Recovery of traceability links between software documentation and source code. *Intn'l Journal of Software Engineering and Knowledge Engineering*, 15(5):811–836, 2005.

[8] R. Oliveto, M. Gethers, D. Poshyvanyk, and A. De Lucia. On the equivalence of information retrieval methods for automated traceability link recovery. In *Proc. of 18th IEEE Intn'l Conference on Program Comprehension (ICPC'10)*, pages 68–71, 2010.

[9] G. Salton. *Automatic text processing: the transformation, analysis, and retrieval of information by computer*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 1989.