Towards Mining Replacement Queries for Hard-to-Retrieve Traces

Marek Gibiec, Adam Czauderna, and Jane Cleland-Huang
Systems and Requirements Engineering Center (SAREC)
DePaul University
Chicago, USA
{mgibiec, aczauderna}@gmail.com; jhuang@cs.depaul.edu;

ABSTRACT
Automated trace retrieval methods can significantly reduce the cost and effort needed to create and maintain requirements traces. However, the set of generated traces is generally quite imprecise and must be manually evaluated by analysts. In applied settings when the retrieval algorithm is unable to find the relevant links for a given query, a human user can improve the trace results by manually adding additional search terms and filtering out unhelpful ones. However, the effectiveness of this approach is largely dependent upon the knowledge of the user. In this paper we present an automated technique for replacing the original query with a new set of query terms. These query terms are learned through seeding a web-based search with the original query and then processing the results to identify a set of domain-specific terms. The query-mining algorithm was evaluated and fine-tuned using security regulations from the USA government's Health Insurance Privacy and Portability Act (HIPAA) traced against ten healthcare related requirements specifications.

Categories and Subject Descriptors
D.2.1 [**Software Engineering]:** Requirements/Specifications; H.3.3 [**Information Storage and Retrieval**: Information Search and Retrieval.

General Terms
Documentation, Verification.

Keywords
Requirements traceability, trace retrieval.

1. INTRODUCTION
Requirements traceability is a critical software engineering activity that provides support for numerous tasks such as impact analysis, compliance verification, and coverage analysis [2, 9]. It has been defined as the ability to track the life of a requirement back to its source documents and forward to the downstream work products in which it is realized [9]. Traceability links are generally created and maintained by project stakeholders using spreadsheets, databases, or specialized trace features of commercial requirements management tools. However, in non-trivial projects, the number of traceability links can grow very large, and as a result the manual effort required to establish and maintain such traces is often inhibitive. For example, in a recent transportation sector project the federally required cost of tracing 30,000 requirements to over 300 sets of regulatory codes was estimated at a staggering $US 3 Million [2]. Because of this excessive effort, unless mandated by law or enforced by organizational standards, many projects have no systematic traceability process in place and software engineers and analysts are forced to manually construct traces between artifacts on an as-needed basis.

ASE'10, September 20–24, 2010, Antwerp, Belgium.

Several different researchers have attempted to address these problems by using information retrieval (IR) methods to dynamically generate traceability links. The most popular approaches have utilized either the vector space model (VSM) [10], probabilistic approaches [4], or Latent Semantic Indexing (LSI) [1,12]. The large body of research in this area has been applied across a variety of domains including safety and business critical projects, as well as both large and small sized projects. Results from these studies have demonstrated that although trace retrieval methods can significantly reduce the traceability effort in most projects, the generated traces are imprecise and require an analyst to spend time evaluating the results in order to find the correct set of links [9].

A closer analysis of these earlier results shows that the precision problems in many of the projects are caused by only a small percentage of *stubborn* traces which pull down the overall quality of the generated traces, while the majority of trace queries perform quite well. Figure 1 illustrates this state of affairs for four previously baselined datasets. It highlights the percentage of trace queries that returned low precision results. For example, the AREMA and GasCode datasets, previously used in an industrial case study [Bere'10], had only 11% and 5% of queries returning average precision values lower than 5%. Similarly, CM1, which is a rather complex NASA dataset [10] also had 11% of these low performing queries, while the HIPAA dataset used in this paper [5] had 13% in this range.

Stubborn traces occur primarily when language in the target document neither matches the language of the source document nor matches project level synonyms defined in a thesaurus [19]. As this is a common occurrence, our traceability tool, Poirot [4], includes a feature to help human users improve stubborn traces through adding additional search terms and filtering out unwanted terms. This is illustrated in Figure 2, which depicts a cloud of terms for a query from an industrial case study that involved tracing regulatory codes for the Canadian Gas Regulations (Gas-Codes) to contractual requirements [2]. In this example, the user added the term *flange*, and filtered out several terms before re-

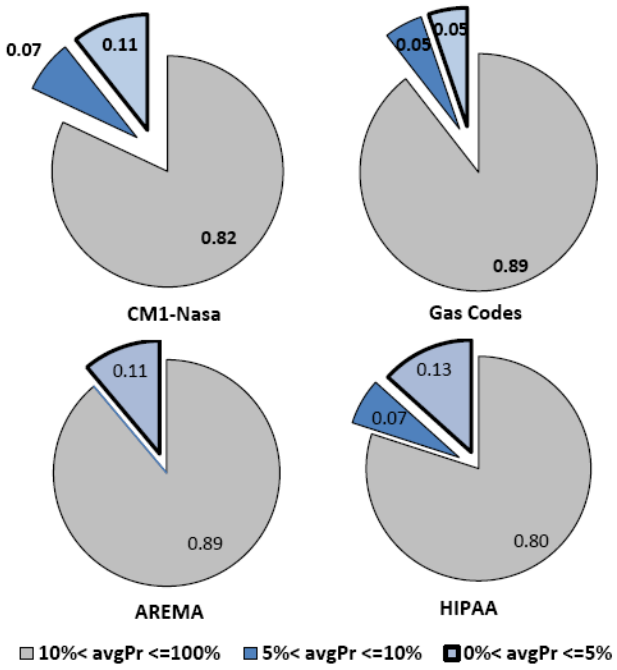


Figure 1. Stubborn Trace Queries exhibiting average precision scores < 10% in four datasets

running the query to generate an updated list of candidate links. Modifying the trace query in this way has been shown to improve the recall and precision of certain stubborn queries. Unfortunately, the effectiveness of this approach is largely dependent upon the expertise and knowledge of the person performing the trace.

To address these shortcomings we developed a web-mining approach, designed to discover a new set of query terms that can be used to replace or augment the original query. This work builds on a technique introduced in our prior work [5] in which we compared the effectiveness of two machine learning methods for improving trace results. One method utilized a manually constructed answer set to train a trace classifier, while the second approach attempted to replace this training set with terms mined from performing related web-searches. This current paper advances the second, web-searching technique in several important ways. First, the original method was only partially automated and required a human to evaluate and select appropriate documents returned as a result of the web search. This prevented the technique from being practically applied in an industrial traceability setting. It also severely limited the number of web-based documents that could be processed. Second, the prior method was very coarse grained and processed entire documents during the mining process as opposed to identifying and using only more relevant chunks of text. The work in this paper addresses these problems through automating the web-mining process, and implementing and validating a technique for identifying appropriate sections of text from within the retrieved documents. Furthermore, in the prior work, numerous decisions and parameters were established in a relatively subjective way. Although this was acceptable for a proof of concept study, it is necessary to fine-tune the algorithm so that it delivers optimal results. This paper also describes a series of experiments which were conducted to empirically discover the correct way to parameterize and execute the algorithm in order to maximize the achievable trace results.

2. TRACE RETRIEVAL

The experiments reported in this paper were conducted using a probabilistic network (PN) model, which has been shown through prior experimentation to perform equivalently to other trace retrieval models. Its use involves three primary tracing activities of pre-processing text, computing link probabilities, and analyzing results.

2.1 The Probabilistic Network Model

To prepare text for tracing, a pre-parsing phase is conducted to remove very common words known as 'stop' words, and to stem the remaining words to their root forms. The basic retrieval algorithm uses a probabilistic network model to evaluate the relevance of a document to a specific query. The probability model is defined over a concept space U, called the Universe of Discourse, in which terms $\{t_1, t_2, \ldots, t_k\}$ represent the singleton concepts, and each document d_i and each query q_j are random variables that represent propositions within the concept space. The probability *pr(d)* of a proposition is interpreted as the degree of coverage of U by the knowledge contained in proposition *d*. The probability $pr(d_j|q)$ of a given document being linked (or relevant) to a specific query is defined as the degree of coverage for document d_j provided by *q*.

A conditional probability value *p(d|q)* is computed for each query *q* and each document *d* defined as a function of the frequency of terms co-occurring in both *q* and *d*, and computed as $p(d|q) = \sum_i^t [p(d|t_i) \times p(q,t_i)]/p(q)$ where $p(d|t_i)$ is defined as $freq(d,t_i)/\sum_i^t freq(d,t_i)$, $p(q,t_i)$ is defined as $freq(q,t_i)/n_i$ and finally p(q) is defined as $\sum_i^t p(q,t_i)$, where $freq(d, t_i)$ represents the frequency of term t_i in a document *d* and n_i is the number of documents in the searchable document collection $\{d_1, d_2, \ldots, d_n\}$ that contain term t_i. The first component $p(d|t_i)$ represents the relative frequency of term t_i in document *d* and increases with the number of occurrences of t_i in *d*. *idf* is used to reduce the contribution of common terms to the overall probability value between the query and the document. Any query-document pair receiving a probability score over a threshold value is treated as a candidate traceability link. These threshold values can be learned automatically based on the distribution of similarity scores [19].

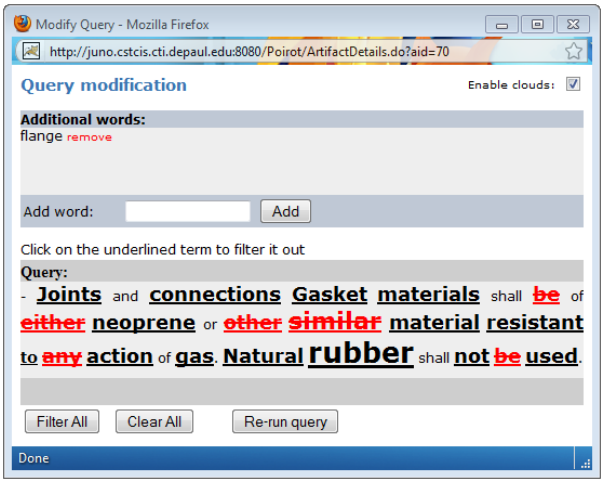


Figure 2. A Poirot Screenshot showing a phrase cloud for a query

2.2 Traceability Metrics

The experiments in this paper are reported using three metrics of recall, precision, and average precision. Recall measures the fraction of relevant links that are retrieved and is computed as:

$$Recall = \frac{Relevant\ links \cap Retrieved\ Links}{Relevant\ Links}$$

while precision measures the fraction of retrieved links that are relevant [14] and is computed as:

$$Precision = \frac{Relevant\ Links \cap Retrieved\ Links}{Retrieved\ Links}$$

Recall and precision generally trade off against each other so that as one increases, the other decreases. For experimental purposes, a threshold score is established such that all links above or at the threshold are retrieved and all links below the threshold are rejected. Because it is not always feasible to achieve identical recall values across every trace query, it can be difficult to compare recall and precision results across experiments. Furthermore, these metrics are unable to differentiate between results in which relevant links are closer to the top of the ranked listing of results, versus those in which relevant links are lower down in the list. An additional metric, Average Precision, favors results that return relevant links nearer to the top of the list, and furthermore does not depend on the creation of an arbitrary threshold value. Average precision is computed as:

$$Average\ Precision = \frac{\sum_{r=1}^{N}(P(r) \times relevant(r))}{Number\ of\ relevant\ documents}$$

where r is the rank, 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. In cases where all relevant links are retrieved at the top of the list Average Precision values will be 1.

2.3 Prior Trace Retrieval Results

There are numerous reported studies in which information retrieval methods have been used to generate traceability links. Hayes et al. [9,10] built a tool named RETRO (REquirements TRacing On-target) that implements VSM. They used RETRO to evaluate the VSM traceability model against the MODIS and CM1 datasets. Their baseline study, without use of any enhancement methods, returned maximum recall values of approximately 0.75 at precision levels of approximately 0.1 for MODIS (results are approximate as they are read from graphs and not tables) and recall of approximately 0.98 at precision of only 0.02 for CM1.

Cleland-Huang et al conducted several different experiments using their tool Poirot [4] to generate and retrieve links between regulatory codes, requirements, business goals, UML classes, java code, and test cases [15]. They reported achieving precision of 3% to 40% at recall levels of close to 90% for various datasets. DeLucia et al [6] implemented the LSI algorithm in the trace retrieval feature of their Advanced Artifact Management System (ADAMS) and provided an empirical comparison of two different recovery processes. Maletic and Marcus [12] also utilized LSI to recover links between source code and documentation and showed that it outperformed VSM. In contrast, Hayes et al reported on a study in which [10] LSI performed similarly to VSM for one dataset, but returned significantly worse results for another one. As LSI is generally accepted to perform better on larger datasets, the results may be at least partially related to the size of the datasets being traced. Researchers have also investigated traceability enhancement techniques. For example Hayes showed that using a user-defined thesaurus, improved recall and precision results. Cleland-Huang et al augmented term weights for terms and phrases found in a project glossary [4], while Zou et al used natural language processing methods to weight shared phrases more highly than simple terms [19]. Each of these approaches led to some improvements in recall and precision for certain datasets, but improvements were small and inconsistent across datasets.

Table 1. HIPAA related requirements used in this study

Description	HIPAA Related Requirements											
	Tot	AC	D	AL	P	I	PA	D	D	TS	IC	I
Care2x: Open source Hospital Info. System	6	1	1	1	0	0	1	1	1	0	0	0
CCHIT: Health Info Exchange, EHR cert.	78	17	33	1	1	5	12	2	2	2	0	3
ClearHealth: Open source HER.	11	1	4	1	0	1	0	1	1	0	1	1
Physician: Information exchange.	15	7	2	0	2	3	0	0	0	1	0	0
iTrust: Open source HER	47	3	35	1	0	0	6	0	0	0	0	2
Trial Implement-ations: Nat'l Coord. for Health Info. Tech.	31	4	6	0	0	4	13	0	0	2	0	2
PatientOS: source healthcare info. Sys.	13	1	2	3	1	0	0	3	1	1	0	1
PracticeOne: A Suite of healthcare info.Systems.	7	3	1	0	0	1	1	0	0	1	0	0
Lauesen: Sample HER reqs spec.	21	11	0	1	0	3	5	0	0	0	0	1
WorldVistA:USA Veteran Admin.	15	6	2	2	0	0	4	0	0	0	0	1
Totals	244	54	86	10	4	17	42	7	5	7	1	11

3. DATA SETS

The work described in this paper is based on a dataset containing technical requirements for the security rule of the USA's Health Insurance Portability and Accountability Act, ten health care related software requirements specifications (SRS) depicted in Table 1, and a traceability matrix showing which requirements contribute towards satisfying each of the regulations. The ten relevant HIPAA regulations addressed concerns related to Access Control (AC), Audit (AUD), Automated Logoff (AL), Emergency Access Procedures (EAP), Integrity (I), Personal Authentication (PA), Storage Encryption and Decryption (SED), Transmission Encryption and Decryption (TED), Transmission Security (TS), Integrity Controls (IC), and Unique User Identification (UUI), all of which were extracted directly from the HIPAA regulations. As an example of a HIPAA regulation, the UUI regulation states "*Assign a unique name and/or number for identifying and tracking user identity*". The associated SRSs were collected by DePaul researchers, during the summer of 2009, from a variety of sources including open source products, IT healthcare standards, requirements exemplars, and feature descriptions for commercial products. The data sets are more fully described in our prior work [5].

4. DEALING WITH STUBBORN TRACES

As previously stated, this paper describes an automated approach for addressing the problem of stubborn traces through replacing trace queries with key terms mined from documents on the World Wide Web. We first examine related work in query expansion and then present our proposed method.

Several researchers have previously explored the use of information retrieval methods to augment web-based search queries.

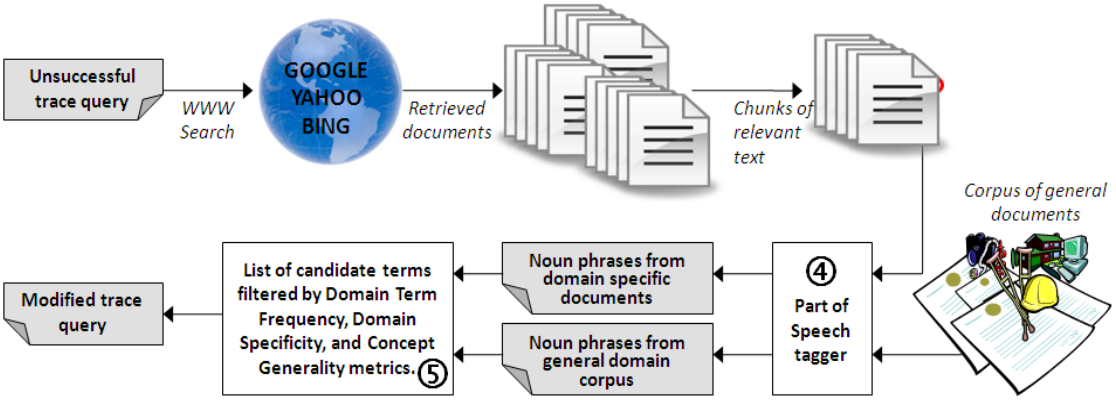


Figure 3. Query Modification Technique

However, their work generally assumes a typical web-query containing between 2.4 and 2.7 terms [7], individual users who dynamically compose queries in real-time, and a large community of users who often repeat similar queries. In contrast, trace queries are much longer in nature, typically involving requirements containing 10-100 terms. Furthermore they often describe relatively obscure concepts, are composed in advance by business analysts, requirements engineers, or even lawyers, and are used infrequently by a small group of trace users. Despite these differences there is a large body of work in the area of query expansion that is relevant to the work we propose in this paper.

For example, Hu et al, used Wikipedia to augment users' web-based queries [11]; however based on our initial analysis of the traceability problem, websites such as Wikipedia, do not currently provide sufficient information to augment a rich and diverse set of domain specific traceability queries. Other researchers have used more general web knowledge to augment queries [3,16]; however their techniques are designed with the primary intent of expanding a short query with additional and potentially disambiguating terms. In contrast, our approach is designed to add and remove terms from the original queries.

Other classes of query augmentation methods are quite interactive in nature. For example, Yurekli et al. used collaborative recommender systems to recommend query refinements to users based on other users' similar queries [18]. This approach shows some promise in the requirements domain; however it is impeded by the fact that trace users are often not the people who create the trace query in the first place and therefore may not have the knowledge needed to accept and reject recommended expansion terms.

Finally, other researchers have explored techniques that augment queries with different types of knowledge such as folksonomy tags [13] or terms from taxonomies. For example, Gabrilovich et al use initial search results to classify queries according to an extensive taxonomy and then to improve search results based on prior history of queries in the designated categories [7]. However these approaches generally depend on either very large groups of interactive users or large numbers of search queries, neither of which hold true for tracing in most software engineering projects.

4.2 Proposed Approach

Our proposed approach is illustrated in Figure 3. The text from a stubborn trace query (in our experiment we only use the HIPAA goal title) is used to seed a series of web searches using one or more standard search engines. The retrieved documents are then filtered to remove documents which are difficult to parse because they are primarily graphical in nature or which contain primarily advertisements. The remaining documents are then partitioned into chunks of text (see section 4.2.1), and a similarity score is computed between each chunk of text and the original query using the Vector Space Model (VSM). VSM is a standard approach which computes the cosine similarity between a query and document, each of which is represented as a vector of weighted terms. A more complete explanation is provided in most introductory information retrieval textbooks [14]. The most relevant chunk i.e. the chunk exhibiting highest similarity to the trace query, is then selected. All identified chunks are then processed using an algorithm we previously developed for extracting project glossary items from requirements specifications [19]. This algorithm outputs a set of candidate terms, and computes Domain Term Frequency (DTF), Domain Specificity (DS), and Concept Generality (CG) metrics which are then used to select a small set of query terms to replace the original trace query. Several components of this algorithm are now discussed in further detail.

4.2.1 Document Chunking Each of the retrieved documents is partitioned into smaller sections through splitting the document into overlapping chunks of length *chunkLength.* Chunks overlap to ensure that unfortunate partitioning decisions do not prevent the algorithm from recognizing the most relevant section of text. The degree of overlapping is determined by the *chunkOffset*, which defines the distance in characters between the starting point of the previous chunk and the starting point of the current chunk. *chunkSize* and *chunkOffset* were both measured in terms of characters in order to speed up the running time of the chunking process. The impact of various chunking parameters is experimentally evaluated in section 5.3 of this paper.

4.2.2 Term and Phrase extraction Our algorithm is designed to extract nouns and noun phrases from the retrieved documents. It utilizes a freely available parser-based part-of-speech (POS) tagger named *QTag* to identify nouns and noun phrases [17]. QTag uses a dictionary to identify the syntactic category of each token in the text and outputs a series of POS tags that represent grammatical classes of words, such as nouns, verbs and adjectives for each token in the input text. This phase of the process can produce a large number of terms and phrases numbering from 5,000 to 10,000 for a typical trace query. Sample terms, generated for HIPAA's *audit control* regulation are depicted in Table 2.

Table 2. A Small Sample of the total 8480 candidate terms generated for HIPAA's Audit regulation

Term	DS	DS	DS	Term	DS	DS	DS
audit	67.2	0.8	5.1	work	14.7	0.2	0.5
control	9.0	0.9	5.1	board	63.7	0.2	0.5
manag	8.5	0.7	2	respons	8.2	0.3	0.5
secur	28.2	0.5	1.7	technolog	20.4	0.3	0.5
report	5.5	0.6	1.4	access	3.9	0.2	0.5
risk	473.0	0.5	1.4	record	6.7	0.2	0.5
system	16.7	0.6	1.3	time	1.2	0.3	0.5
process	5.1	0.5	1.3	audit control	1000.0	0.2	0.5
auditor	88.5	0.4	1.2	committe	1000.0	0.2	0.5
procedur	52.2	0.5	1.2	issu	65.7	0.2	0.5
account	9.4	0.5	1	transact	29.8	0.2	0.5
data	1.7	0.4	1	offic	72.6	0.2	0.5
servic	3.5	0.4	1	requir	2.2	0.3	0.4
polici	13.1	0.4	1	comput	12.6	0.2	0.4
busi	2139.0	0.4	0.9	statement	74.4	0.2	0.4
review	54.1	0.4	0.9	effect	15.7	0.2	0.4
organ	14.7	0.3	0.9	plan	20.8	0.3	0.4
activ	9.1	0.4	0.8	area	12.4	0.2	0.4
oper	1.7	0.4	0.8	assess	103.9	0.2	0.4
campani	17.9	0.3	0.8	function	3.7	0.2	0.4
softwar	6.0	0.3	0.7	employe	19.6	0.2	0.4
complianc	46.7	0.4	0.7	assur	94.0	0.3	0.4
applic	3.7	0.3	0.7	asset	1000.0	0.2	0.4
object	9.0	0.3	0.6	state	12.6	0.2	0.3
program	5.9	0.3	0.6	staff	10.9	0.2	0.3
standard	6.1	0.3	0.6	author	31.0	0.2	0.3
chang	7.4	0.2	0.6	fraud	1000.0	0.1	0.3
document	9.5	0.3	0.6	tool	9.3	0.2	0.3
govern	154.7	0.3	0.6	entiti	90.3	0.1	0.3
depart	46.9	0.2	0.6	level	5.3	0.2	0.3

Candidate terms are then filtered by establishing threshold values for the following three metrics.

- *Domain term frequency* (DTF) computes normalized term frequency information for term *t* across multiple documents as:

$$DTF(t) = \sum_{d \in D} [freq(t,d)/|d|]$$

where *freq(t, d)* is the total number of occurances of term *t* in a given document *d*, and |*d*| is the length of that document expressed as the total number of all terms in d. The normalized occurrences from all retrieved documents D are then summed.

- *Domain specificity* (DS) measures the extent to which a term or term phrase is specific to the domain document, as opposed to occurring frequently across a broad spectrum of topics. For our experiments, domain specificity *DS(t, d)* of term *t* in document *d*, was estimated by comparing the relative frequency of the term within a domain-specific document, versus its relative frequency in a general corpus of documents [19]. It is calculated as follows:

$$DS(t,d) = ln\left[\frac{freq(t,d)}{\sum_{t \in d} freq(t,d)} \Big/ \frac{freq(t,G)}{\sum_{t \in G} freq(t,G)}\right]$$

where the first component $freq(t,d)/\sum_{t \in d} freq(t,d)$ is the normalized number of occurrences of term *t* in the domain-specific document *d*, and the second component is the normalized number of occurrences of t in the general corpus of documents. Domain specificity (DS) is then calculated as the average value of all domain specificities from each document from the collection:

$$DS(t) = \frac{1}{|D|} \sum_{d \in D} DS(t,d)$$

Additionally, when a term is not found in the general corpus of documents it is considered as highly domain specific and is assigned a DS value of 1000.0

- *Concept generality* (CG) computes the fraction of domain specific documents in which a specific term occurs. Concept generality differentiates between terms that occur in multiple domain specific documents versus those that occur in only a few. The concept generality of term *t*, *CG(t)*, is computed *as* the number of documents containing term *t*, $|D_t|$ over the total number of documents: $CG(t) = |D_t|/|D|$.

Once these metrics are computed, they are used to filter out non-useful terms. The remaining terms are then ranked in descending order of term frequency. Section 6.4 of this paper describes a series of experiments we conducted to explore the optimal threshold values for each metric.

4.3 Results from Prior Experiment

In our previously reported experiments [5], these threshold values were set according to our informal observations of the initial results. Remaining phrases were sorted in descending order according to their domain term frequency and the top 10 terms were then formed into a new query used for tracing.

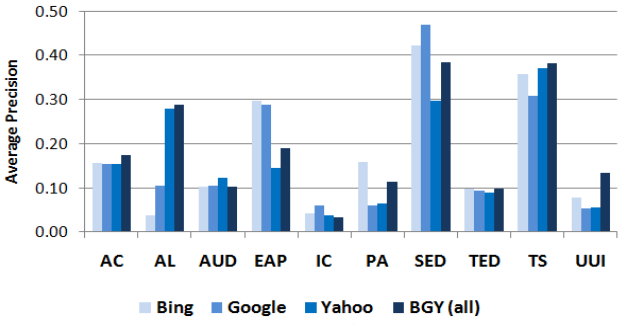
Results from this earlier experiment showed that for the five HIPAA regulations of type AL, EAP, SED, TED and TS, the basic Poirot algorithms performed very well, retrieving at least 90% of the relevant requirements at precision values of 6-78%. Neither the manually trained, nor the web-trained algorithms were able to improve on these results, and in fact both machine learning techniques had an overall negative effect on the results. On the other hand, the web-mining approach returned improved traces for the PA, AUD, AC, and I HIPAA regulations. A further analysis of the results suggested that the basic Poirot algorithm performed well for traces with low fan-out values i.e. for which a HIPAA regulation traced to only one or two requirements in the target dataset, but tended to perform significantly worse for cases with higher fan-out. This is explained by the fact that traces exhibiting high fan-out use a broader set of terms to represent the set of traceable concepts, and that the language in the regulation is often insufficient for retrieving all of the relevant documents. Query expansion techniques can potentially address this problem by discovering a broader set of terms than those found in the original query.

4.4 Open Issues

Although our previous results demonstrated that this approach could be useful, the technique was not fully automated and neither was it fine-tuned to produce optimal results. The new work described in this paper therefore fully automates the approach, and includes techniques for dispatching queries to the search engine, selecting relevant documents, parsing those documents to partition the text into smaller chunks, and finally selecting the most relevant chunk. It further includes a series of experiments that were conducted to evaluate and fine-tune the query augmentation process and related algorithms. As our approach relies quite significantly upon the results returned from standard search engines, the first experiment evaluated the impact of search engine selection on the trace results. Experiments were also conducted to evaluate the optimal number of documents to retrieve and parse, as it is important to discover whether retrieving a smaller number of documents, such as the ten documents used in our proof of concept study, or a larger number of documents provides the best

	Bing			Google		
	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec
AC	0.926	0.036	0.157	0.926	0.084	0.153
AL	0.100	0.333	0.038	0.100	1.000	0.105
AUD	0.895	0.051	0.102	0.907	0.051	0.104
EAP	1.000	0.062	0.298	1.000	0.129	0.288
IC	0.235	0.138	0.043	0.235	0.190	0.061
PA	0.595	0.088	0.157	0.548	0.061	0.060
SED	1.000	0.259	0.422	1.000	0.368	0.469
TED	0.800	0.114	0.097	0.800	0.103	0.093
TS	1.000	0.047	0.358	1.000	0.066	0.309
UUI	0.818	0.043	0.078	0.636	0.035	0.053
	Yahoo			BGY (all)		
	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec
AC	0.926	0.042	0.153	0.907	0.096	0.173
AL	0.400	0.333	0.279	0.400	0.364	0.289
AUD	0.895	0.052	0.123	0.895	0.051	0.103
EAP	1.000	0.118	0.145	1.000	0.093	0.190
IC	0.824	0.010	0.038	0.824	0.010	0.034
PA	0.571	0.059	0.063	0.571	0.068	0.113
SED	1.000	0.152	0.297	1.000	0.179	0.385
TED	0.800	0.108	0.089	0.800	0.114	0.097
TS	1.000	0.052	0.371	1.000	0.053	0.382
UUI	0.455	0.044	0.054	0.818	0.041	0.133

a. Recall, Precision (Prec), and Average Precision (Avg Prec)



b. Average Precision of trace results

Figure 4. Impact of Search Engine selection

results. Furthermore, our previous study used all the text from the retrieved document, whereas it is likely that a single document may cover multiple topics, and results might be improved by identifying one or more chunks of more relevant data from within each document. Experiments were conducted to evaluate the impact of chunk-size and offset values on the trace results. Finally, the set of candidate terms and term phrases are filtered using the metrics of domain term frequency, domain specificity, and concept generality and a series of experiments were conducted to empirically explore these relationships in order to identify the optimal values for the datasets studied.

5. EXPERIMENTAL EVALUATION

This section of the paper describes the experiments that were conducted to comparatively evaluate different approaches. Recall, precision, and average precision values were computed for each HIPAA goal, traced against all ten of the datasets.

5.1 Use of Different Search Engines

The first experiment was designed to comparatively evaluate the impact of using three popular search engines, namely Bing, GOOGLE, and Yahoo, as well as a combination of the three. For each of these experiments 100 documents were retrieved per trace query. Each of the retrieved documents was automatically partitioned into chunks of 2000 characters, and the chunk that exhibited greatest relevance to the query was mined for query expansion terms. The results are depicted in Figure 4 and show that different search engines performed well on different queries. As a result, we decided to utilize the combination of all three search engines as the default setting for future experiments.

5.2 Number of documents retrieved

A series of experiments were conducted to evaluate the impact of selecting 10, 20, 40, 60 80 and 100 documents from each search engine. As three search engines were used, the actual number of unique documents retrieved, averaged 22, 45, 100, 160, 190, and 220 documents for each experiment. In many cases documents could not be read (our tool only supports html, doc and some pdf files) or had almost no textual content, so fewer documents were actually parsed. Furthermore, as the Google Search Engine API only supports results in sets of 8, up to a maximum of 64 distinct links, the number of web documents retrieved from Google results were 16, 24, 40, 64, 64, and 64 respectively. Results are depicted in Figure 5 and varied somewhat by query, depending on how many relevant high quality documents were available.

5.3 Chunk Size

Experiments were also conducted to evaluate the impact of chunk size on the trace query results. For these experiments, 60 documents were retrieved from each of the search engines. Chunk sizes were evaluated at 200, 400, 800, 1000, 1600, and 3,200 characters. As in previous experiments, queries were issued for each HIPAA goal, and recall, precision, and average precision results were computed. These results are reported in Figure 6 and show that in six of the ten trace queries, results were improved with smaller chunks, while in two queries results were improved with larger chunks. In the final two queries, no definite trend was identified based on chunk size. These results suggest that chunk size should not be fixed in length but should be customized according to the characteristics of each query and document. Investigating this conjecture is left for future work.

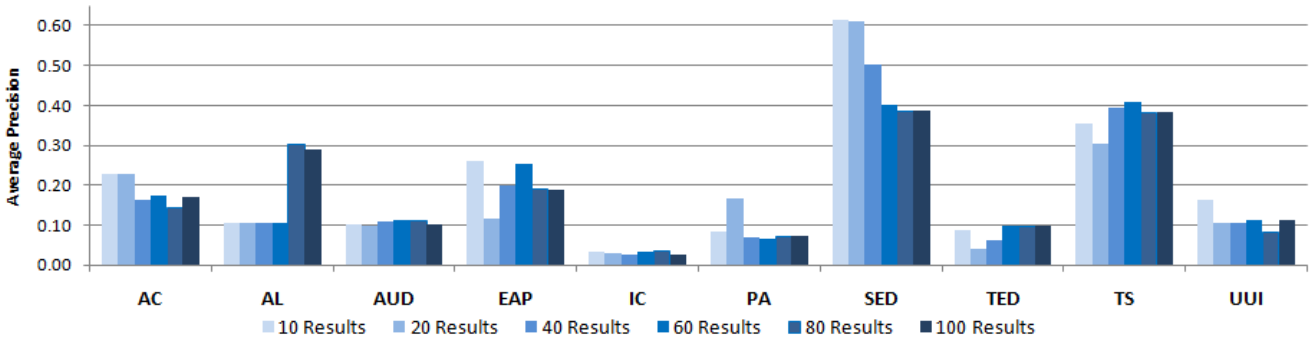
5.4 Threshold values for term selection

The final experiment focused on determining appropriate threshold values for CG, DS, and DTF metrics. Candidate lists of terms and phrases were generated using a combination of the three search engines, 100 documents per query, chunk sizes of 2000, and chunk offset values of 200. As there were three parameters that are all quite closely interdependent, a series of 512 experiments were run to evaluate various combinations of each parameter across the following ranges: Concept generality ranged from 0.02–0.6, Domain Specificity from 10-140, and Term Frequency from 0.1-0.9. A total of 512 different combinations were investigated using an inductive experimental style which tested combinations of low, medium, and high values for each metric, and then further explored areas of high-performing combinations.

Average Precision values were recorded for all 512 experiments. Results are shown in Figure 7. Domain Specificity exhibited best results at threshold values of 20 to 60. Setting the DS threshold too high caused useful terms to be excluded. Similarly Domain Term Frequency performed best at values between 0.1 and 0.4, and caused a gradual decline in Average precision values when it was set to higher values. The Concept Generality threshold did not appear to have a significant impact on the quality of the trace

	10 Results			20 Results			40 Results			60 Results			80 Results			100 Results		
	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec
AC	0.926	0.039	0.229	0.926	0.037	0.229	0.907	0.073	0.162	0.926	0.090	0.175	0.926	0.049	0.143	0.907	0.050	0.172
AL	0.100	1.000	0.105	0.100	1.000	0.105	0.100	1.000	0.105	0.200	0.010	0.105	0.400	0.400	0.303	0.400	0.364	0.289
AUD	0.907	0.052	0.100	0.895	0.051	0.099	0.895	0.051	0.108	0.895	0.051	0.111	0.895	0.051	0.111	0.895	0.051	0.103
EAP	1.000	0.083	0.262	1.000	0.089	0.116	1.000	0.105	0.199	1.000	0.108	0.254	1.000	0.093	0.191	1.000	0.091	0.189
IC	0.353	0.061	0.033	0.294	0.063	0.031	0.235	0.073	0.027	0.471	0.035	0.033	0.824	0.010	0.035	0.824	0.010	0.026
PA	0.571	0.075	0.085	0.595	0.081	0.166	0.571	0.068	0.070	0.571	0.063	0.066	0.548	0.061	0.071	0.571	0.064	0.071
SED	1.000	0.333	0.614	1.000	0.304	0.612	1.000	0.318	0.501	1.000	0.184	0.403	1.000	0.179	0.385	1.000	0.179	0.385
TED	1.000	0.082	0.086	0.800	0.053	0.042	0.800	0.095	0.062	0.800	0.118	0.098	0.800	0.114	0.097	0.800	0.114	0.097
TS	1.000	0.057	0.353	1.000	0.066	0.304	1.000	0.058	0.395	1.000	0.045	0.410	1.000	0.053	0.380	1.000	0.053	0.382
UUI	0.455	0.035	0.163	1.000	0.023	0.106	0.818	0.035	0.106	0.545	0.030	0.111	0.636	0.032	0.083	0.818	0.041	0.111

a. Recall, Precision (Prec), and Average Precision (Avg Prec)

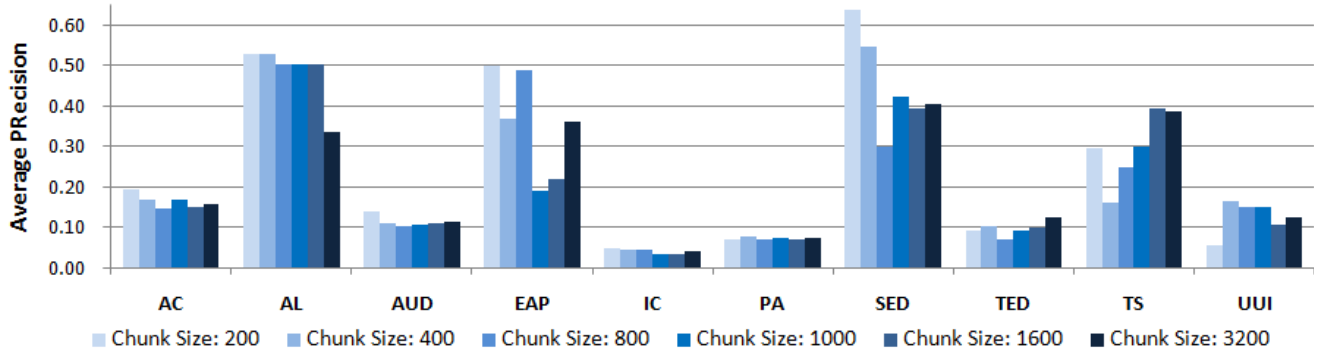


b. Average Precision of trace results

Figure 5. Impact of Retrieved Document Count

	Chunk Size: 200			Chunk Size: 400			Chunk Size: 800			Chunk Size: 1000			Chunk Size: 1600			Chunk Size: 3200		
	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec	Recall	Prec	Avg Prec
AC	0.907	0.046	0.193	0.907	0.043	0.168	0.926	0.035	0.144	0.907	0.036	0.166	0.926	0.035	0.151	0.926	0.045	0.158
AL	0.700	0.115	0.527	0.700	0.130	0.527	0.500	1.000	0.502	0.500	1.000	0.502	0.500	1.000	0.502	0.400	0.333	0.336
AUD	0.895	0.051	0.137	0.895	0.052	0.108	0.895	0.051	0.102	0.895	0.052	0.104	0.895	0.051	0.111	0.895	0.052	0.113
EAP	1.000	0.080	0.499	1.000	0.091	0.368	1.000	0.087	0.488	1.000	0.105	0.189	1.000	0.063	0.217	1.000	0.062	0.359
IC	0.412	0.039	0.046	0.588	0.030	0.043	0.353	0.044	0.043	0.353	0.046	0.033	0.235	0.070	0.032	0.353	0.044	0.041
PA	0.571	0.059	0.068	0.571	0.071	0.076	0.571	0.071	0.070	0.571	0.071	0.072	0.548	0.068	0.070	0.571	0.063	0.071
SED	1.000	0.318	0.637	1.000	0.292	0.547	1.000	0.226	0.300	1.000	0.212	0.420	1.000	0.184	0.392	1.000	0.189	0.405
TED	0.800	0.093	0.092	0.800	0.103	0.103	0.800	0.080	0.069	0.800	0.111	0.091	0.800	0.118	0.098	0.800	0.129	0.124
TS	1.000	0.032	0.296	1.000	0.029	0.160	1.000	0.037	0.247	1.000	0.035	0.299	1.000	0.113	0.394	1.000	0.063	0.386
UUI	0.818	0.020	0.057	0.818	0.039	0.162	0.818	0.039	0.150	0.818	0.039	0.150	0.818	0.036	0.106	0.727	0.047	0.124

a. Recall (rec), Precision (Pr), and Average Precision (Av. Prec)



b. Average Precision by Chunk Size.

Figure 6. Impact of Chunk Size on Trace Query results

results. The top scoring results are shown in Table 3, and show that maximum average precision was achieved for this dataset at following thresholds: CG=0.12, DS=40, and DTF = 0.1 or 0.2.

Domain Specificity appears to be the dominant factor in improving trace results. Setting it too high appears to exclude some useful terms while setting it too low introduces terms that are too

Table 3. Achieved Average Precision

Results	Average precision	CG	DS	DTF
Top 8 results	0.260181856	0. 12	40	0.1
	0.260181856	0.12	40	0.2
	0.258481429	0.12	40	0.3
	0.258018459	0.12	40	0.4
	0.25669283	0.06	40	0.4
	0.25669283	0.09	40	0.4
	0.250239424	0.35	40	2.9
	0.250239424	0.41	40	2.9
Selection of lower ranked results (sampled every 100th result from ranked list)	0. 239754669	0.53	60	0.5
	0. 226525938	0.04	50	0.25
	0. 1953757	0.44	15	0.5
	0. 180683942	0.47	50	0.2
	0. 109055538	0.04	80	0.9

general in nature. Term frequency serves as a second internal discriminator. Optimal results were achieved by lowering this threshold in order to include terms that occur less frequently, but which are representative of the domain.

6. TOOL SUPPORT

For experimental purposes we developed a java based tool named WATson (**W**eb **A**ugmented **T**race**S ON** demand) that imports two xml files containing source and target artifacts, and three text files containing stop words, correct trace information for evaluative purposes, and program settings. For each input query, Watson issues queries to GOOGLE, Bing, and Yahoo, collects and filters the results, extracts the most relevant chunks of text from each result, computes candidate terms and term phrases and then selects the ten terms which exhibit appropriate levels of term frequency, domain specificity, and concept generality. WATson outputs a new xml file containing the reconstituted queries which are passed to a Report Generator. This generator utilizes libraries from Poirot to generate traces and compute standard metrics. WATson is currently being integrated into Poirot to provide analysts with support for stubborn trace queries.

7. STUBBORN TRACES

WATson was evaluated against the stubborn traces of the HIPAA project. Unlike the previous experiments which traced each HIPAA regulation to requirements from all ten datasets, this experiment explored the quality of each individual trace query. It therefore exposed the details of individual trace queries, providing visibility into successful and stubborn traces. The ten HIPAA regulations and ten SRS documents created a potential for 100 different trace queries. However several SRS documents did not have any relevant traces for certain HIPAA regulations, and so the actual number of queries with relevant traces was 59. Each of these 59 queries was executed using both the basic PN algorithm and the query expansion algorithm.

7.1 HIPAA Query Expansion

To illustrate the effect of query expansion on the HIPAA regulatory codes, Figure 8 shows stemmed terms from both the original

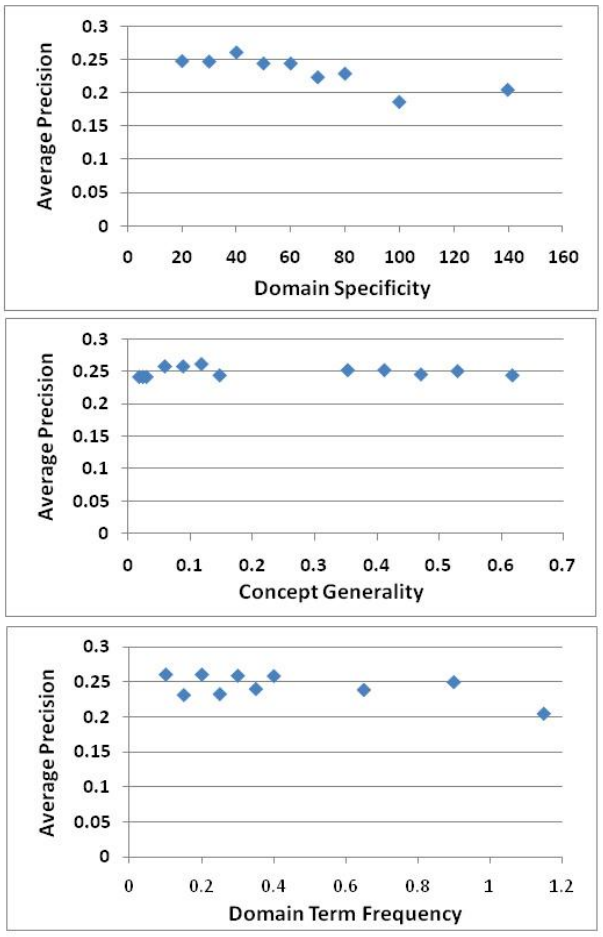


Figure 7. Maximum Average Precision Achieved at various threshold values

regulation and the replacement query generated by WATson, and shows that query terms can be retained, removed, or added by the algorithm. For example in the AC regulation, terms such as *access* and *control* were retained, terms such as *permiss* and *ident* were added, and finally terms such as *technic* and *program* were removed. The Integrity regulation is worth mentioning because it is the only regulation for which WATson did not return a reasonable set of replacement terms. In fact this regulation was not clearly defined in the original HIPAA regulations and created problems throughout the experiment for both the automated techniques and for human analysts who had to manually create traceability links. Although not reported in this paper we also conducted a series of experiments that combined new and original queries; however early results did not show improvements with this approach, and further investigation is left for future work.

7.2 HIPAA Query Expansions

Results are reported in Figure 9 displayed in ascending order according to the average precision of the basic Poirot results. In other words queries that were more stubborn-to-trace using the basic Poirot algorithm appear on the left hand side of the graph, while those that returned high average precision values are shown on the right hand side. The darker bars on the left hand side that

Reg	Original Query	Modified Query
AC	access allow control electron grant health implement inform maintain person polici procedure program protect right softwar specifi technic	access control door reader permiss control system com role lock ident kei
AL	automat electron implement inact logoff predetermin procedur session termin time	logoff post shutdown com auto logon inact forum password login
AUD	activ audit contain control electron examin hardwar health implement inform mechan procedur protect record software	audit risk auditor procedur review complianc govern depart board audit control
EAP	access electron emerg establish health implement inform necessari need obtain procedur protect	emerg procedur health plan implement depart emerg access fire offic build
IC	control detect dispos electron ensur health implement improperli inform integr measur modif	integr
PA	access authent claim electron entiti health implement inform person procedur protect seek verifi	authent entiti health procedur ident implement certif identif password kei
SED	decrypt electron encrypt health implement inform mechan protect	encrypt kei algorithm password disk bit drive certif cipher decrypt
TED	appropri deem electron encrypt health implement inform mechan protect	encrypt decrypt kei algorithm password cipher string byte bit
TS	access commun electron guard health implement inform measure network protrect secur techni transmiss	secur encrypt transmiss kei health risk integr email data com
UUI	assign ident identif identifi number track uniqu user	identif password authent kei login mail card com

Figure 8. Key terms in Original vs. Modified Queries

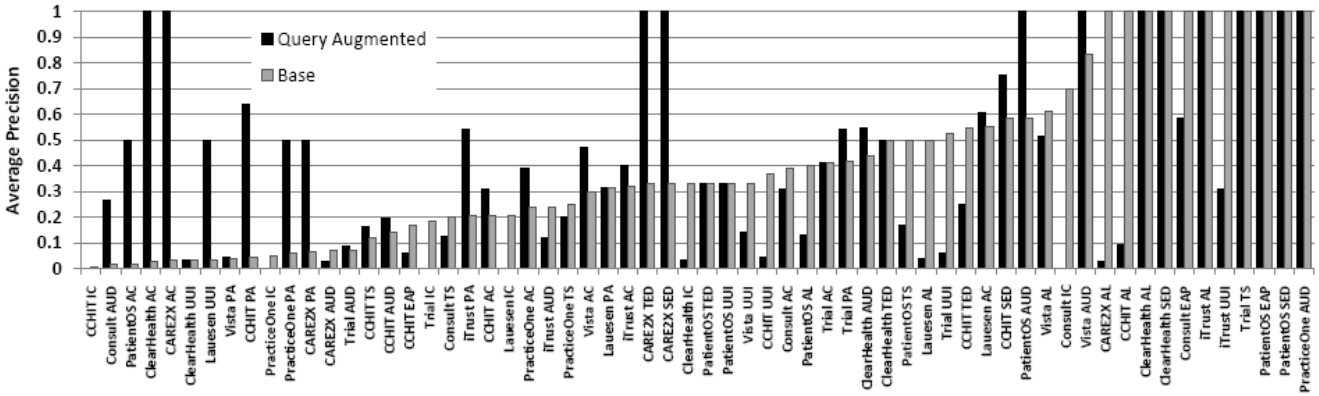


Figure 9. The Effect of Query Modification on Stubborn Trace Queries

rise above the gray bars represent the significant improvements achieved through using query modification techniques on the stubborn queries that did not perform well in the initial trace.

There are several interesting observations that can be made. First, a Pearson correlation analysis shows only weak correlation of 0.37 between the average precision results returned by the two methods. This suggests that the augmented query technique is not simply returning good results for the same traces as the base method. In fact the graph shows that there are several traces that performed well using the base method that did not perform well under the augmented query method.

On the other hand, there are several traces that performed very poorly using the base method, which were significantly improved by the augmented query approach. If stubborn queries are defined as those which returned average precision scores of 0.1 or lower from the base algorithm, then the experimental results show that of the 14 traces in this category, 8 augmented queries performed significantly better than the baseline, 3 performed marginally better or equivalently, while 3 performed marginally worse. Furthermore, out of the total 57 traces, the query augmentation technique improved 24 and matched the quality of an additional 8.

It is equally clear from these results, that this approach does not uniformly improve every trace query. The augmented query approach returned only perfect scores (i.e. all relevant traces were found in the topmost positions of the candidate links), for 7 of the 11 queries exhibiting perfect results with the basic method. In fact in three of these cases, average precision dropped below 50%, including two cases in which it dropped all the way down to 3%. On the other hand WATson returned perfect scores for 4 queries that did not perform so well using the basic trace method. WATson is not designed to replace the basic approach, but instead to provide the user with a realistic option for improving stubborn traces. This is especially useful for tracing regulatory codes such as HIPAA regulations, as this is an inherently labor intensive task in which the analyst has to first determine whether the regulation is relevant, and if it is found to be relevant to demonstrate that the requirements sufficiently satisfy it. In this type of scenario the analyst could attempt the base trace first, and if results are not satisfactory, could then generate a replacement query.

8. CONCLUSIONS

This paper has described a fully automated approach for modifying the terms found in trace queries and then re-issuing the query using the new set of terms. The experimental results have clearly shown that in the HIPAA dataset a significant portion of previously stubborn trace queries were improved through use of this method. Although our current algorithm cannot predict which queries could benefit from the use of this method, the approach provides

analysts with a potentially useful option to help them improve results for unsatisfactory traces.

The experiments reported in this paper represent an initial exploration of the use of query expansion techniques to support requirements traceability. However there are several threats to validity that must be considered. The most obvious threat is that the experiments were all conducted in the healthcare domain, and focused on tracing HIPAA regulations. We therefore cannot claim generality of the approach. This is especially true because information retrieval methods are quite sensitive to the nuances of various contexts, and so could behave quite differently in different domains. Secondly, our approach makes the assumption that documents can be found on the web to support each of the search queries. However this may not always be the case, in which case no expansion terms will be recommended. Finally, the experiments involved exploring the combination of many different factors, and although we approached this in a systematic way, it is possible that some important combinations went unexplored. Nevertheless, this work represent a first step in focusing on the problem of stubborn traces and makes a unique contribution to traceability research through demonstrating that query modification can be used effectively to improve many of these traces.

The results from these experiments introduce further questions that will be explored in future work. For example, we need to develop techniques for more dynamically sizing chunks in the retrieved documents according to the relevance of the retrieved data; however over constraining these chunks to sections of text that closely match the initial query might undermine the goal of finding alternate query terms. Finally, in order to draw general conclusions we need to conduct a further series of experiments to evaluate the query augmentation technique against a much broader set of datasets representing a variety of domains.

9. ACKNOWLEDGMENTS

Funding for this work was partially provided by grant # CCF 0810924 from the National Science Foundation.

10. REFERENCES

[1] Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E., (2002), "Recovering Traceability Links between Code and Documentation", *IEEE Transactions on Software Engineering*, vol. 28, no. 10, October, pp. 970-983.

[2] Berenbach, B., Gruseman, D., Cleland-Huang, J., "Application of Just In Time Tracing to Regulatory Codes", Conf. on Systems Eng. Research, Hoboken, NJ, March, 2010.

[3] Broder, A., Fontoura, M., Gabrilovich, E., Joshi, A., Josifovski, V. and Zhang, T. Robust Classification of Rare Queries using Web Knowledge. *30th Intn'l ACM SIGIR Conf on Research and Development in Inf. Retrieval*, July, 2007

[4] Cleland-Huang, J., Settimi, R., Duan, C., and Zou, X., (2005b), "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability", IEEE International Conference on Requirements Engineering (RE), pp. 135-144.

[5] Cleland-Huang, J., Czauderna, A., Gibiec, M., Emenecker, J., "A Machine Learning Approach for Tracing Regulatory Codes to Product Specific Requirements", International Conf on Software Eng., Cape Town, South Africa, May, 2010.

[6] DeLucia, A., Oliveto, R., and Tortora, G., (2004), "IR-Based Traceability Recovery Processes: An Empirical Comparison of 'One-Shot' and Incremental Processes", IEEE/ACM International Conf on Automated Software Engineering, L'Aquila, Italy, 2008: pp. 39-48

[7] Gabrilovich, E., Fountoura, B., Joshi, M., Josifovski, V., Riedel, L., Zhang, T., "Classifying Search Queries Using the Web as a Source of Knowledge, ACM Transactions on the Web, 04/2009, Volume 3, Issue 2, p.1--28, (2009)

[8] Gotel, O. C. Z. and Finkelstein, A. C. W., (1994), "An Analysis of the Requirements Traceability Problem", in Proceedings of 1st IEEE Intn'l Conf. on Requirements Eng., Colorado Springs, CO, USA, April 18-22, pp. 94-101.

[9] Hayes, J. H., Dekhtyar, A., Sundaram, S., and Howard, S., (2004), "Helping Analysts Trace Requirements: An Objective Look", IEEE Requirements Engineering Conference (RE'04), Kyoto, Japan, September pp. 249-261.

[10] Hayes, J. H., Dekhtyar, A., and Sundaram, S., (2006), "Advancing Candidate Link Generation for Requirements Tracing: The Study of Methods", *IEEE Transactions on Software Engineering*, vol. 32, no. 1, January, pp. 4-19.

[11] Hu, J., Wang, G., Lochovsky, F., Sun, J. and Chen, Z. Understanding user's query intent with wikipedia. *18th International Conference on World Wide Web*, *Madrid, Spain, April 20-24, 2009* (WWW'09). 471-480.

[12] Maletic, J. I. and Marcus, A., "Using Latent Semantic Analysis to Identify Similarities in Source Code to Support Program Understanding", in Proceedings of 12th IEEE Int'l Conf. on Tools with Artificial Intelligence, Vancouver, British Columbia, 2000, Nov 13-15, pp. 46-53.

[13] Nauman, M., Khan, S., "Using PersonalizedWeb Search for Enhancing Common Sense and Folksonomy Based Intelligent Search Systems," IEEE/WIC/ACM International Conference on Web Intelligence (WI'07), 2007, pp.423-426.

[14] Salton, G. *Automatic Text Processing: The Transformation, Analysis and Retrieval of Information by Computer*. Addison-Wesley, 1989.

[15] Settimi, R., Cleland-Huang, J., Khadra, O. B., Mody, J., Lukasik, W., and DePalma, C., (2004), "Supporting Software Evolution through Dynamically Retrieving Traces to UML Artifacts", 7th Intn'l Workshop on Principles of Software Evolution (IWPSE) Kyoto, Japan, Sept 6-7, pp. 49-54.

[16] Shen, D., Sun, J., Yang, Q. and Chen, Z. Building bridges for Web Query Classification. *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, *SIGIR-06* (2006).

[17] Tufis D, Mason O, "Tagging Romanian texts: a case study for QTAG, a language independent probabilistic tagger", Proceedings of the International Conference on Language Resources & Evaluation, Granada, Spain, 1998.

[18] Yurekli, B., Capan, G., Yilmazel, B., Yilmazel, O., "Guided Navigation Using Query Log Mining through Query Expansion," International Conference on Network and System Security, 2009, pp.560-564.

[19] Zou, X., Settimi, R., and Cleland-Huang, J., "Improving Automated Trace Retrieval: A Study of Term-based Enhancement Methods" Empirical Software Engineering, July, 2009.