Requirements Eng (2015) 20:281–300
DOI 10.1007/s00766-013-0199-y

ORIGINAL ARTICLE

On the role of semantics in automated requirements tracing

Anas Mahmoud · Nan Niu

Received: 4 December 2012 / Accepted: 26 December 2013 / Published online: 22 January 2014

Abstract In this paper, we investigate the potential benefits of utilizing natural language semantics in automated traceability link retrieval. In particular, we evaluate the performance of a wide spectrum of semantically enabled information retrieval methods in capturing and presenting requirements traceability links in software systems. Our objectives are to gain more operational insights into these methods and to provide practical guidelines for the design and development of effective requirements tracing and management tools. To achieve our research objectives, we conduct an experimental analysis using three datasets from various application domains. Results show that considering more semantic relations in traceability link retrieval does not necessarily lead to higher quality results. Instead, a more focused semantic support, that targets specific semantic relations, is expected to have a greater impact on the overall performance of tracing tools. In addition, our analysis shows that explicit semantic methods, that exploit local or domain-specific sources of knowledge, often achieve a more satisfactory performance than latent methods, or methods that derive semantics from external or general-purpose knowledge sources.

Keywords Information retrieval · Traceability · Semantics

1 Introduction

Requirements traceability is defined as "the ability to describe and follow the life of a requirement, in both a forward and backward direction (i.e., from its origins to its subsequent deployment and use, and through all periods of ongoing refinement and iteration in any of these phases)" [44]. The availability of traceability information has been proven vital to several software engineering activities such as program comprehension [25, 71], impact analysis [59], feature location [87], software reuse [98], and verification and validation (V&V) [56].

Traceability is often accomplished in practice by linking various software artifacts (e.g., requirements, design documents, code elements, and test cases) manually through a matrix, known as requirements traceability matrix (RTM) [89]. However, as software systems evolve over time, this process becomes labor-intensive, boring, time-consuming, and error-prone [29, 45, 56]. Consequently, effective traceability is rarely established in practice, as practitioners often fail to implement a consistent and effective manual tracing process [23, 44].

To overcome the limitations of the manual approach, modern traceability tools employ information retrieval (IR) methods for automated support [6, 8, 17, 66, 70, 73, 81, 95]. These methods aim to match a query of keywords with a set of objects in the software repository and rank the retrieved objects based on how relevant they are to the query using a predefined similarity measure [49]. The tenet underlying IR-based tracing methods is that artifacts having a high textual similarity probably share several concepts, so they are likely good candidates to be traced from one another [6, 24]. The main assumption is that a consistent terminology has been used throughout the project's lifecycle. Such terminology is embedded in the set of vocabulary used in naming code identifiers, comments, and messages, and the textual content of other software artifacts (e.g., requirements and design documents) [4]. These terms serve as signs that can be traced to produce meaningful

A. Mahmoud (✉) · N. Niu
Department of Computer Science and Engineering, Mississippi State University, Mississippi State, MS 39762, USA
e-mail: amm560@msstate.edu

tracks in the system [46]. In other words, IR methods assume that the same words are used whenever a particular concept is described [9]. However, as projects evolve, a new and inconsistent terminology gradually finds its way into the system [62], causing topically related artifacts to exhibit a large degree of variance in their textual contents [5, 12, 37]. This problem is known as the vocabulary mismatch problem and is regarded as one of the principal causes of declining accuracy in retrieval engines [40].

In an attempt to alleviate the problem of vocabulary mismatch, researchers have started investigating semantically enabled IR techniques that look beyond the lexical structure of software artifacts. Unlike lexical methods, which deal with text as strings of tokens, semantic methods capture similarity among various artifacts by exploiting the semantic knowledge embedded in their contents. In requirements engineering, several semantically enabled methods have been exploited to support various related tasks such as requirements discovery, analysis, modeling, traceability, and reuse [13, 54, 58, 68, 78]. The underlying assumption is that the overwhelming majority of requirements are written in natural language (NL) [67, 80]. Therefore, IR methods that exploit semantics in the NL component of software artifacts should be able to discover dimensions that lexical methods often overlook [54].

It is important at this point of our study to distinguish between two kinds of semantics: programming language semantics, which refer to the meaning of a program as a state transformer from inputs to outputs, and natural language semantics, which refer to the meaning inherent in the natural language component of artifacts, such as code identifiers' names and comments [12]. In this paper, the latter is our concern. In particular, the analysis in this paper addresses several research questions related to the utilization of semantics in traceability link retrieval. Such questions include how much semantics is needed? What specific effects does semantics have on the performance? What are the merits of different semantic enhancements? And what is the scope of applicability of different methods? Our work not only advances the fundamental understanding about the role of semantics in supporting automated tracing, but also enables principled ways to increase the practicality of requirements tracing and management tools. In particular, the contributions of this paper are as follows:

- A comprehensive statistical analysis of the merits of a wide spectrum of semantically enabled IR methods in identifying and capturing traceability links in software systems.
- A systematic categorization of different semantically enabled IR methods based on their internal operation, the semantic relations they target, and scope of application.
- A set of guidelines for using semantically enabled IR methods in requirements traceability tasks, including guidelines for optimizing, evaluating, and implementing such methods.

To achieve our research objectives, we analyze the performance of a plethora of semantically enabled IR methods using three traceability datasets from various application domains. These methods include basic vector space model (VSM) [92], VSM with thesaurus support (VSM-T) [56], *Part-of-Speech*-enabled VSM (VSM-POS) [17], latent semantic indexing (LSI) [27], latent Dirichlet allocation (LDA) [15], explicit semantic analysis (ESA) [41], and normalized Google distance (NGD) [20]. These methods are based on the standard VSM method which is used as a baseline in our experiment.

The rest of the paper is organized as follows. Section 2 describes the various semantically enabled IR methods investigated in this paper. Section 3 describes the research methodology and experimental design. Section 4 presents analysis results and study limitations. Section 5 discusses the main findings of the paper and their potential impact. Section 6 reviews related work. Finally, Sect. 7 summarizes the paper and discusses directions for future work.

2 Semantically enabled IR

In our analysis, we experiment with various semantically enabled IR methods that are based on the algebraic VSM. Such methods transform textual documents into more compact representations in the form of vectors. These vectors can hold various types of information, representing different aspects of the semantic knowledge embedded in a text corpus (e.g., word counts in artifacts or latent topical structures in a corpus [15]). Once the vector representation of a document is generated, a simple similarity measure (e.g., the cosine distance [106]) can be used to calculate the similarity between these vectors, thus determining the relevance of documents.

Based on the underlying IR model used, we identify three categories of VSM-based semantically enabled IR methods. These categories include semantic-augmented, latent semantic, and semantic relatedness methods. In what follows, we describe each of these categories, its main methods, and their applications in greater detail.

2.1 Vector space model

Vector space model is an algebraic model that describes and compares objects using N-dimensional vectors, where each dimension corresponds to an orthogonal feature of the object [92]. The standard VSM for IR encodes textual documents and queries as vectors of terms' weights. The normalized inner product between two documents' vectors denotes the cosine similarity between them. The basic tenet is that different words are used to express different topics. Therefore, texts sharing many words are more similar than texts with only a few words in common.

Using VSM, each document is represented as a set of terms $T = \{t_1,\ldots, t_n\}$. Each term t_i in the set T is assigned a weight w_i. The terms in T are regarded as the coordinate axes in N-dimensional coordinate system, and the terms' weights $W = \{w_1,\ldots, w_n\}$ are the corresponding values. Thus, if q and d are two artifacts represented in the vector space, then their similarity can be measured as the cosine of the angle between them (Eq. 1):

$$\mathrm{Sim}(q,d) = \frac{\sum q_i \cdot d_i}{\sqrt{\sum q_i^2 \cdot \sum d_i^2}} \tag{1}$$

where q_i and d_i are real numbers standing for the weights of term i in q and d, respectively. Word counts or term frequencies in documents are often used to assign weights to terms in the document's vector. While this method is computationally efficient, it might represent a bias toward long-text documents or frequent words in the corpus. To mitigate this risk, another weighting scheme based on term frequency and inverse document frequency (TFIDF) is used. Using this approach, q_i and d_i in Eq. 1 become $q_i = tf_i(q) \cdot idf_i$ and $d_i = tf_i(d) \cdot idf_i$, where $tf_i(q)$ and $tf_i(d)$ are the frequencies of term i in q and d, respectively. idf_i is the inverse document frequency and is computed as $idf_i = -\log_2(t/df_i)$, where t is the total number of artifacts in the corpus, and df_i is the number of artifacts in which term i occurs. TFIDF determines how relevant a given word is in a particular document. Words that are common in a single or a small group of documents tend to have higher TFIDF, while terms that are common in all documents such as articles and prepositions get lower TFIDF values. A higher TFIDF implies a stronger relationship between the term and the document it appears in, thus if that term were to appear in a query, the document would probably be a correct match.

Due to its conceptual and mathematical simplicity, basic VSM has gained a considerable popularity in IR research [13]. However, this over simplification often comes with several limitations. For instance, the bag-of-words assumption assumes term independence. This assumption discards the punctuation information and the words ordering. However, since there are strong inherent associations between terms in a language, this assumption is never satisfied [106]. This often results in loss of information and ambiguity problems as texts should ideally be compared at their topic level and not based on the specific words that were chosen to express these topics [60].

2.2 Semantic-augmented methods

This category includes IR methods that semantically *augment* the basic VSM by adding new information to the basic document's vector, thus integrating additional evidence into retrieval. We identify two methods under this category, including vector space model with thesaurus support (VSM-T) and vector space model with *Part-of-Speech* tagging (VSM-POS). Both methods have been investigated before in automated tracing research [17, 56]. Following is a description of these methods in greater detail.

2.2.1 Vector space model with thesaurus support

A very common occurring semantic relation in software artifacts is *synonymy*, or equivalent words. As mentioned earlier, as software projects evolve, developers tend to use different vocabulary, including abbreviations and acronyms, to refer to certain domain concepts [5, 28]. Basic VSM fails to capture these relations as it assumes terms' independence. A simple way to overcome this problem is to equip VSM with a dictionary or a thesaurus that keeps track of such relations (e.g., different acronyms used to describe a certain concept). Documents are then matched based on their matching keywords, as well as *synonymy* relations found in the supporting thesaurus.

The integration of a thesaurus into VSM is relatively simple. For each pair of synonyms identified (s_i, s_j), a perceived similarity coefficient α_{ij} can be assigned to indicate their equivalence [56]. For each document in the corpus, document vectors are expanded based on these synonym pairs. A similarity coefficient of $\alpha_{ij} < 1$ is usually assigned to distinguish a *synonymy* match from an exact match ($\alpha_{ij} = 1$). The similarity between two documents can then be calculated as:

$$s(q,d) = \frac{\sum q_i d_i + \sum_{(k_i,k_j,\alpha_{ij})\in T} \alpha_{ij}(q_i d_j + d_j q_i)}{\sqrt{\sum q_i^2 \cdot \sum d_i^2}} \tag{2}$$

Based on the type of the integrated thesaurus, two methods of VSM-T can be distinguished under this category:

- VSM with general-purpose thesaurus (VSM-T-WN): This method uses general-purpose dictionaries, such as WordNet, to derive synonyms. WordNet, introduced and maintained by the Cognitive Science Laboratory of Princeton University, is a large lexical database of

English verbs, nouns, and adjectives, grouped into sets of cognitive synonyms, known as *synsets* [38]. The main advantage of general-purpose dictionaries is their high coverage of terms, and the fact that they are constantly being maintained by highly trained linguists. However, with no domain-specific knowledge (context), relying on a general-purpose thesaurus to handle abbreviations and acronyms in a software system can become a challenge.

- VSM with domain-specific thesaurus (VSM-T-DT): Such methods use a domain-specific thesaurus to handle synonym pairs derived from the project domain's taxonomy. These domain-specific thesauri can deal with cases such as acronyms and abbreviations. However, they can become quickly out-of-date, as keeping track of the changes in the project's vocabulary over time can become an exhaustive task.

2.2.2 *Vector space model with part-of-speech tagging*

Part of speech (POS) refers to the syntactic role of terms in written text (e.g., nouns, verbs, adjectives). Research in natural language processing (NLP) has revealed that some parts of speech carry more information value than others. For instance, it has been reported that nouns and verbs are better discriminators, or more descriptive, to the content of a document than other parts of speech [19, 34]. These observations have been recently integrated into the IR paradigm. The main assumption is that favoring certain parts of speech over treating all terms at the same level of importance should improve the overall accuracy of retrieval engines [64, 84, 90].

We build upon these observations to derive a VSM model with *Part-of-Speech* tagging support (VSM-POS) for traceability link retrieval. To calculate the similarity between two artifacts in the system (q and d), initially POS analysis is conducted to identify different parts of speech (e.g., verbs and nouns) in q and d. This process can be achieved using various text tagging tools such as TreeTagger [94]. Only terms which belong to a particular part of speech are then considered in retrieval.

In this paper, we only consider the two linguistic forms of verbs (VSM-POS-V) and nouns (VSM-POS-N) in building term vectors of software artifacts. These two particular parts of speech have been found to carry higher information values than other parts, capturing main actions and objects in software artifacts [35, 57, 68, 96]. Therefore, indexing based on these two linguistic forms is expected to filter out unwanted noise resulting from keywords that do not significantly contribute to the artifact's topic.

Limitations of methods based on POS often stem from the complications associated with text tagging, or automatically identifying different parts of speech in a text. While generating linguistic parse trees can be relatively simple for artifacts expressed in natural language (e.g., requirements documents), this process can become more complicated for semi-formal or formal artifacts, such as source code or design documents [1, 14]. In addition, selecting an optimal linguistic form or a combination of forms that best achieves desired performance levels can be computationally exhaustive [64].

2.3 Latent semantic methods

While semantic-augmented methods help to exploit basic semantic aspects of artifacts, they reveal a little about the inter- or intra-semantics in a corpus. To count for such information, another set of methods are often used. Such methods use statistical and probabilistic models to automatically discover latent semantic structures in text corpora. In particular, instead of representing a document as a vector of independent terms, latent methods represent documents and terms as combinations of implicit semantic schemes that are often hidden from other methods. Two methods can be identified under this category: LSI and LDA. In what follows, we briefly introduce both methods and discuss their limitations and applications in greater detail.

2.3.1 *Latent semantic indexing*

Latent semantic indexing is a statistical method for inducing and representing aspects of the meanings of words and passages reflective in their usage. LSI is based on the assumption that there is some underlying (*latent*) structure in words that is partially concealed by the variability of words used to express a certain concept [27]. In particular, using statistically derived conceptual indices, LSI tries to overcome the problem of vocabulary mismatch by capturing the semantic relations of *synonymy* (equivalent words) and *polysemy* (multiple meanings) in software artifacts [91].

LSI is a corpus-based technique that uses singular value decomposition (SVD) to estimate the structure in word usage across documents in the corpus [30]. It starts by constructing a term-document matrix (A) for terms and documents in the corpus. This matrix is usually huge and sparse. Word counts are often used to build this matrix. SVD is then applied to decompose A into three new matrices $A = USV^T$ where T stands for transpose. Dimensionality reduction is then performed to produce reduced approximations of $<U, S, V^T>$ by keeping the top K eigenvalues of these matrices. A dimensionality reduction technique takes a set of objects that exist in a high-dimensional space and represents them using low dimensions, often in a 2D or 3D space. These reduced matrices can be described as $<U_k, S_k, V_k^T>$. The best value of K that fits a certain corpus can be obtained experimentally; however, a value in the range of [100, 300] is frequently used [27]. From the newly reduced space, the equation $V = A^T US^{-1}$ can be derived. Assuming A is a matrix with $n > 1$ documents, for a given document vector d in A, d can be expressed as $d = d^T US^{-1}$. In LSI, the query is also treated as a document, which is the case in traceability, where the query itself is a requirement or a piece of code. The query q can be expressed in the new coordinates of the reduced space as $q = q^T US^{-1}$. In the K-reduced space, q and d can be represented as $d = d^T U_k S_k^T$ and $q = q^T U_k S_k^T$, respectively. The similarity of q and d can then be calculated as the cosine measure:

$$\mathrm{sim}(q, d) = \mathrm{sim}(q^T U_k S_k^{-1},\ d^T U_k S_k^{-1}) \tag{3}$$

In other words, retrieval in LSI is performed using the database of singular values and vectors obtained from the SVD analysis. Therefore, a query and a document can have a high cosine similarity even if they do not have any overlapping terms, as long as their terms are semantically similar in the latent semantic space.

LSI has been employed in a wide range of software engineering activities such as categorizing source code files [71], detecting high-level conceptual code clones [72], and recovering traceability links between documentation and source code [73]. The drawbacks of LSI include its huge storage requirements, the computational costs of performing SVD, and the assumption of normally distributed data, which might be inappropriate for handling the word count data of the term-by-document matrix. In addition, it is often difficult to add new documents to the corpus, and determining the optimal K can be computationally exhaustive [32].

2.3.2 *Latent Dirichlet allocation*

LDA was first introduced by Blei et al. [15] as a statistical model for automatically discovering topics in large corpus of text documents. The main assumption is that documents in a collection are generated using a mixture of latent topics, where a topic is a dominant theme that describes a coherent concept of the corpus's subject matter.

A topic model can be described as a hierarchical Bayesian model that associates with each document d in the collection D a probability distribution over a number of topics K. In particular, each document d in the collection $(d_i \in D)$ is modeled as a finite mixture over K drawn from a Dirichlet distribution with parameter α, such that each d is associated with each $(t_i \in K)$ by a probability distribution of θ_i. On the other hand, each topic t in the identified latent topics $(t_i \in K)$ is modeled as a multidimensional probability distribution, drawn from a Dirichlet distribution β, over the set of unique words in the corpus (W), where the likelihood of a word from the corpus $(w_i \in W)$ to be assigned to a certain topic t is given by the parameter ϕ_i.

LDA takes the documents collection D, the number of topics K, and α and β as inputs. Each document in the corpus is represented as a bag of words $d = <w_1, w_2, \ldots, w_n>$. Since these words are observed data, Bayesian probability can be used to invert the generative model and automatically learn ϕ values for each topic t_i, and θ values for each document d_i. In particular, using algorithms such as Gibbs sampling [85], a LDA model can be extracted. This model contains for each t the matrix $\phi = \{\phi_1, \phi_2, \ldots, \phi_n\}$, representing the distribution of t over the set of words $<w_1, w_2, \ldots, w_n>$, and for each document d the matrix $\theta = \{\theta_1, \theta_2, \ldots, \theta_n\}$, representing the distribution of d over the set of topics $<t_1, t_2, \ldots, t_n>$. Once these matrices (vectors) are produced, a similarity measure such as the cosine distance can be used to compute the similarity of two documents by comparing their topic distribution vectors to produce a ranked list of topically similar documents [53].

Selecting the number of topics (K) that best fits a certain text corpus is computationally expensive. In NLP tasks, often a heuristic of 50–300 topics is empirically specified depending on the size of the collection [15, 48, 105]. In some other cases, such values are determined automatically. For example, Teh et al. [101] proposed a nonparametric model known as hierarchical Dirichlet processes which extends LDA and seeks to learn the optimal K automatically. However, while such heuristics and methods achieve satisfactory performance in NLP tasks, they are not necessarily optimal for software systems [74, 82].

Several methods have been proposed in the literature to approximate near-optimal combinations of LDA parameters (α, β, K) in software systems. Such methods can be (a) manual, based on a domain expert understanding of the system [3, 74], (b) experimentally determined, in which LDA parameters are tuned until a configuration that achieves acceptable performance over a certain quality measure is reached [8, 11], or (c) automatically generated using statistical methods or machine learning approaches [47, 82].

LDA has been used to support several software engineering tasks, such as mining semantic topics from source code [63], analyzing software evolution [103], and automated tracing [8, 81]. Several drawbacks of LDA stem from its mathematical complexity. For example, it can be easily misguided by uninformative words, and also the use of the Dirichlet distribution limits LDA ability in modeling data with high diversity [82]. Another limitation is the fact that the number of topics that are naturally present in the corpus should be specified ahead, which is, similar to specifying the K in LSI, can be computationally exhaustive [15].

2.4 Semantic relatedness methods

Semantic relatedness (SR) methods try to quantify the degree to which two concepts semantically relate to each other by exploiting different types of semantic relations connecting them [9]. The main intent is to mimic the human mental model when computing the relatedness of words. The human brain establishes semantic relatedness between words based on their meaning, or context of use [52]. For example, both words <*cow*, *horse*> refer to a mammal that has four legs, and thus they can be considered related. Also, the words <*horse*, *car*> both refer to a means of transportation for humans, thus they can be considered related from that perspective. Another aspect the brain examines is the frequent association between words. Words that often appear together are likely to be related. For example, the words <*table*, *chair*> appear together frequently, giving the human brain an indication of relatedness.

A wide range of methods for measuring SR have been proposed in the literature. These methods infer words relatedness by exploiting massive amounts of textual knowledge to leverage all the possible relations that contribute to words similarity (e.g., Table 1). Such information is usually available in external knowledge sources including linguistic knowledge bases (LKB), such as WordNet [38], collaborative knowledge bases (CKB), such as the online encyclopedia *Wikipedia* [99], or general Web search results, such as Google search [36].

SR has been applied to several NLP applications such as automated spelling correction [16], text retrieval [39], word sense disambiguation [83], question answering [2], and automatic speech recognition [88]. In what follows, we describe two methods of semantic relatedness that have been heavily investigated in related literature. These methods include ESA and NGD.

2.4.1 *Explicit semantic analysis*

ESA represents the meaning of a text as a high-dimensional weighted vector of concepts, derived from *Wikipedia* [41]. In details, given a text fragment $T = < t_1, \ldots, t_n >$, and a space of *Wikipedia* articles (C), initially a weighted vector V is created for the text, where each entry of the vector v_i is the TFIDF weight of the term t_i in T. Using a centroid-based classifier [51], all *Wikipedia* articles in C are ranked according to their relevance to the text. Let k_j be the strength of association of term t_i with *Wikipedia* article c_j, where $c_j \in < c_1, c_2, \ldots, c_n >$ (N is the total number of *Wikipedia* articles), the semantic interpretation vector S for text T is a vector of length N, in which the weight of each concept is defined as:

$$S_i = \sum_{w_i \in T} v_i . k_j \tag{4}$$

Entries of this vector reflect the relevance of the corresponding articles to text T. The relatedness between two texts can then be calculated as the cosine between their corresponding vectors.

Among the different *Wikipedia*-based measures proposed in the literature, ESA has been proven to achieve the highest correlation with human judgment [76, 99]. In addition, ESA compares text fragments. This makes it a suitable approach for traceability tasks or even requirements engineering tasks in general [70]. In fact, due to its flexibility, ESA has been extended to work in cross-lingual retrieval settings, which can be considered as an extreme case of vocabulary mismatch [97]. Limitations of ESA can stem from the complexity of its implementation, as it requires downloading the whole *Wikipedia*, which requires substantial space requirements, in addition to the computational capabilities required for indexing such a large amount of data [41].

2.4.2 *Normalized Google distance*

The fuzzy set theory suggests that the degree of keywords' co-occurrence can be considered as a measure of their semantic relatedness [9]. Based on that, the NGD provides a method to estimate confidence scores between words using words' co-occurrences collected over Web search results (e.g., Google).

Formally, for each two terms being matched, a Google search query is initiated. The semantic relatedness between two terms $s(t_1, t_2)$ is then measured using the normalized Google similarity distance (NGD) introduced by Cilibrasi and Vitanyi [20] as:

$$s(t_1, t_2) = 1 - \frac{\log(\max(D_1, D_2)) - \log(|D_1 \cap D_2|)}{\log(|D|) - \log(\min(D_1, D_2))} \tag{5}$$

where D_1 and D_2 are the numbers of documents containing t_1 and t_2, respectively, and $|D_1 \cap D_2|$ is the number of documents containing both t_1 and t_2. The assumption is that pages that contain both terms indicate relatedness, while pages that contain only one of the terms suggest the opposite. The denominator is used to normalize the similarity scores of vectors to fit in the range [0–1].

For example, the NGD between the two terms, *patient* and *hospital*, can be calculated as follows, a Google search is

Table 1 Semantic relations

Relation	Description	Example
Synonymy	Equivalent	<*sick*, *ill*>
Polysemy	Multiple meanings	<*charge*>
Hyponymy	Type-of	<ambulance, car>
Antonymy	Opposite	<male, female>
Meronymy	Part-of	<*room*, *hotel*>
Statistical	Co-occurrence	<*patient*, *hospital*>

initiated for the terms *patient* and *hospital* separately. The search process returns 573,000,000 and 1,200,000,000 hits for both terms, respectively (i.e., D_1 and D_2). Next, a search using the phrase *patient hospital* is requested. Google returns 335,000,000 hits (pages in which both *patient* and *hospital* appear) representing $|D_1 \cap D_2|$. Using Eq. 5, NGD (*patient*, *hospital*) $= 0.446$, given that Google search engine indexes approximately ten billion pages ($D = 10^{10}$).

In NGD, the smallest the distance, the closer the terms, hence $NGD(x, x) = 0$, and the distance between two completely unrelated terms (e.g., $|D_1 \cap D_2| = \phi$) is equal to ∞. To quantify the similarity between different artifacts in the system, we initially calculate the pairwise NGD similarity between all the unique terms in the corpus. These values are normalized to fit in the interval [0–1], producing NGD‘. The value $sNGD = 1 - NGD‘(x,y)$ is then used to indicate the pairwise term similarity rather than dissimilarity (i.e., 1 means an exact match) [43]. These values are stored in a thesaurus similar to the synonyms thesaurus introduced earlier (VSM-T). Similarity between any two artifacts in the system can then be calculated using Eq. 2, where α_{ij} is equal to the sNDG value between the terms t_i and t_j.

NGD has been successfully applied in several NLP tasks such as search query prediction [18] and concepts mapping [43]. However, the quality of NGD can be highly affected by the noise usually returned by search engines due to the inherent ambiguity of some terms, and the lack of context when matching individual terms.

2.5 Summary

The collection of methods presented in this section covers a wide spectrum of semantically enabled IR methods that have been intensively used in software engineering research in general, and traceability research in particular. Some of these methods are focused on certain semantic relations such as *synonymy* (e.g., VSM-T), while other methods expand the range of relations to cover more semantic relations such as *polysemy*, *hyponymy*, and *meronymy*, and statistical associations such as co-occurrence of terms (e.g., LDA, LSI, ESA, and NGD). In addition, some of these methods use local knowledge sources, such as a domain thesaurus or the internal textual structure of the corpus, to derive their similarity scores (e.g., VSM-T-TD, LSI, and LDA), while other methods use external sources, such as WordNet and *Wikipedia*, to estimate similarity (e.g., VSM-T-WN, ESA, and NGD).

Furthermore, the presented methods can be divided into explicit and latent. Explicit methods are explicit in the sense that they manipulate concepts grounded in human cognition (e.g., ESA and NGD) or import semantics explicitly from an external source such as a domain thesaurus (e.g., VSM-TD), or the grammatical structure of documents (e.g., VSM-POS). On the other hand, latent methods use statistical methods to derive latent semantic structures hidden in the natural language component of the system (e.g., LDA and LSI). Table 2 summarizes the attributes of the different methods described in this section.

3 Experimental settings

The main objective of our experimental analysis is to systematically compare the performance of the various methods proposed earlier (Table 2) in capturing various requirements traceability links in software systems.

Table 2 Categories of semantically enabled IR methods used in our analysis

	Description	Semantics	Knowledge source	Related work
Baseline				
Vector Space Model	VSM	N/A	N/A	[92]
Semantic augmented with thesaurus				
VSM with domain thesaurus	VSM-T-TD	Synonyms	Domain Thesaurus	[56, 104]
VSM with general-purpose thesaurus	VSM-T-WN	Synonyms	WordNet	
Semantic augmented with POS				
VSM with Part-of-Speech tagging (nouns)	VSM-POS-N	Nouns	OpenNLP	[19, 57]
VSM with Part-of-Speech tagging (verbs)	VSM-POS-V	Verbs	OpenNLP	[19, 57]
Latent semantic				
Latent Semantic Indexing	LSI	Synonyms, Polynyms	Corpus	[27]
Latent Dirichlet allocation	LDA	Topics modeling	Corpus	[15]
Semantic relatedness				
Explicit Semantic Analysis	ESA	Synonymy, hyponymy Antonym, meronymy	Wikipedia	[41]
Normalized Google Distance	NGD	Co-occurrence	Google	[20]

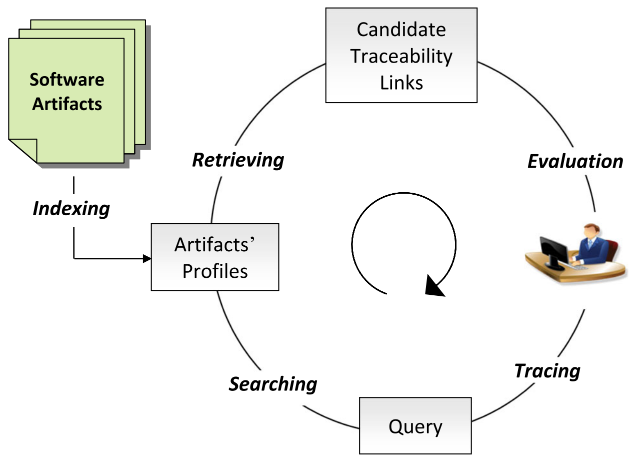

Fig. 1 IR-based automated tracing loop

Figure 1 depicts our experimental procedure, which also describes how the problem is formulated. In general, the IR-based tracing problem can be described as a loop with three main steps:

- *Indexing*: The indexing process starts by extracting the textual content of software artifacts (e.g., comments, code identifiers, requirements text). Lexical processing is then applied to extract individual tokens from the text (e.g., splitting code identifiers into their constituent words). In some cases, stemming is performed to reduce words to their inflectional roots (e.g., *"patients"* → *"patient"*). In our analysis, we use Porter stemming algorithm [86]. The output of the process is a compact content descriptor, or a *profile*, which is usually represented as keywords components matrix or a VSM [69].
- *Retrieval*: The various semantically enabled IR methods described earlier are used to capture requirements traceability links in the system. Links with similarity scores above a certain threshold (cutoff) value are called candidate links [56].
- *Evaluation*: A set of primary and secondary measures are used to assess the different performance aspects of the underlying IR methods [100]. In particular, the performance of the various methods of each category is compared to the VSM baseline, and the best performing methods of different categories are compared to each other.

3.1 Evaluation

3.1.1 Quality measures

Precision (P) and recall (R) are the standard measures often used to assess the performance of IR methods. Recall measures coverage and is defined as the percentage of correct links that are retrieved. Precision, on the other hand, measures accuracy and is defined as the percentage of retrieved candidate links that are correct. Formally, if A is the set of correct links and B is the set of retrieved candidate links, then recall (R) and precision (P) can be defined as:

$$R = |A \cap B| / |A| \tag{6}$$

$$P = |A \cap B| / |B| \tag{7}$$

3.1.2 Browsability measures

Browsability is the extent to which a presentation eases the effort for the analyst to navigate the candidate traceability links. For a tracing tool or a method that uses a ranked list to present traceability links, it is important to not only retrieve the correct links, but also to present them properly. Being set-based measures, precision and recall do not reflect any information about the list browsability. To convey such information, other measures are usually used.

Assuming h and d belong to two sets of system artifacts, where $H = \{h_1, \ldots, h_n\}$ and $D = \{d_1, \ldots, d_m\}$. Let C be the set of true links connecting d and h, and $L = \{(d, h)|\text{sim}(d, h)\}$ is a set of candidate traceability links between d and h generated by the IR-based tracing tool, where sim(d, h) is the similarity score between d and h. L_T is the subset of true positives (correct links) in L, a link in this subset is described as (d, h). L_F is the subset of false positives in L, a link in this subset is described using the notion (d', h'). Based on these assumptions, secondary measures can be described as:

- Mean average precision (MAP) is a measure of quality across recall levels [7]. For each query, a cutoff point is taken after each true link in the ranked list of candidate links. The precision is then calculated. Correct links that were not retrieved are given a precision of 0. The precision values for each query are then averaged over all the relevant (correct) links in the answer set of that query ($|C|$), producing average precision (AP). The MAP is calculated as the average of AP for all queries in each dataset [102]. MAP gives an indication of the order in which the returned documents are presented. For instance, if two IR methods retrieved the same number of correct links (similar recall), then the method that place more true links toward the top of the list will have a higher MAP. Equation 4 describes MAP, assuming the dataset has Q traceability queries.

$$MAP = \frac{1}{|Q|}\sum_{j=1}^{|Q|}\frac{1}{|C_j|}\sum_{k=1}^{|L_{T_j}|}\text{Precision}(L_{T_{j_k}}) \tag{8}$$

- DiffAR measures the contrast of the list. It can be described as the difference between the average similarity of true positives and false positives in a ranked list. A list with higher *DiffAR* has a clearer distinction between its correct and incorrect links, hence, is considered superior. Equation 7 describes *DiffAR*.

$$DiffAR = \frac{\sum_{i=1}^{|L_T|} \mathrm{sim}(h_i, d_i)}{|L_T|} - \frac{\sum_{j=1}^{|L_F|} \mathrm{sim}(h'_j, d'_j)}{|L_F|} \tag{9}$$

- Lag can be described as the average of the number of false positives with higher similarity score that precede each true positive in the ranked list. In other words, the average number of incorrect links that appears before each correct link in the list. Equation 8 describes Lag.

$$Lag = \frac{\sum_{i=1}^{|L_T|} Lag(h_i, d_i)}{|L_T|} \tag{10}$$

Lag gives an indication of how separated true positives from false positives in a list. A higher lag means that true links are scattered all over the list, which is considered a sign of poor performance.

3.2 Datasets

Three datasets are used to conduct the experiment in this paper: *CM-1*, *eTour*, and *iTrust*. Following is a description of these datasets and their application domains.

- *iTrust*: An open source medical application developed by software engineering students at North Carolina State University (USA).[1] It provides patients with a means to keep up with their medical history and records and to communicate with their doctors [75]. The dataset (source code: v15.0, Requirements: v21) contains 314 requirement-to-code links. The links are available at method level. To conduct our analysis, the link granularity is abstracted to class level based on a careful analysis of the system.
- *eTour*: An electronic tourist guide application developed by final-year students at the University of Salerno (Italy).[2] *eTour* was selected as experimental object in this experiment because its source code contains a combination of English and Italian words, which is considered an extreme case of vocabulary mismatch. The dataset contains 394 requirement-to-source code links at class-level granularity.
- *CM-1*: Consists of a complete requirements (high-level) document and a complete design (low-level) document for a NASA scientific instrument. The project source code was written in C with approximately 20 K lines of code. It has 235 high-level requirements and 220 design elements. The traceability matrix contains 361 actual requirement-to-design traces.

Table 3 Experimental datasets

Dataset	LOC (K)	COM (K)	Source	Target	Links
iTrust	18.3	6.3	Req	SC	314
eTour	17.5	7.5	Req	SC	394
CM-1	20	N/A	Req	Design	361

Table 3 shows the characteristics of each dataset. The table shows the size of the system in terms of lines of source code (LOC), lines of comments (COM), source and target of traceability links [e.g., use case (UC), source code element (SC), or requirement (Req.)], and the number of correct traceability links (Links).

3.3 Implementation

ESA implementation was guided through several online implementations [3] including preprocessing tools for parsing *Wikipedia* dumps (e.g., *WikiPrep*) and carrying out ESA analysis. *Wikipedia* 2009 dumps were used in our implementation. To implement NGD, the client library *Google.NET* [4] was used to initiate Google queries and interpret returned responses. For the implementation of VSM-POS we used *SharpNLP*,[5] a port of the Java *OpenNLP* library written in *C#*. For VSM-WN we used WordNet.Net,[6] a free open source. Net framework library for WordNet written in *C#*. For the implementation of LSI we used *Bluebit Matrix Calculator*,[7] a high performance matrix algebra for .NET programming which provides routines for SVDs, eigenvalues, and eigenvectors problems. *JGibbLDA*,[8] a Java implementation of LDA is used for topic modeling. This particular implementation uses Gibbs Sampling for parameter estimation and inference [48].

4 Results and analysis

We use two sample artifacts (*q*, *d*) from the *iTrust* dataset as an illustrative example to guide our analysis. These artifacts are shown in Fig. 2a and b, respectively. *q* represents a requirement of the system (req. 3.1.1), and it describes a basic *login* functionality. *d* is a method that verifies user's login information. There is a valid trace link

[1] http://agile.csc.ncsu.edu/iTrust/wiki/doku.php.
[2] http://www.cs.wm.edu/semeru/tefse2011/Challenge.htm.

[3] http://www.cs.technion.ac.il/~gabr/resources/code/.
[4] https://developers.google.com/gdata/client-cs.
[5] http://sharpnlp.codeplex.com/.
[6] http://opensource.ebswift.com/WordNet.Net/.
[7] http://www.bluebit.gr/net/.
[8] http://jgibblda.sourceforge.net/.

Fig. 2 Example 1: Traceability link

3.1 Functions

…

…

3.1.3 Login:

Each patient should authenticate himself using his credentials. In case of forgotten password see 3.1.4.

3.1.4 Forget Password:

The system should provide a functionality for recovering user's password...

(a) SRS document

```
//Validate Patient's email and password
public bool Login_OnClick(object sender)
{
    if
     (ValidateUser(EmailText.Text))
      &&
      (ValidateUser(PasswordText.Text))
          return true;
       else
        {
          Exception.Text ="Login failed";
          return false;
        }
}
```

(b) Code Element

<login>
<patient>
<Authenticate>
<credential>
<forget>
<password>

(c) Profile of requirement 3.1.3

<patient>
<email>
<password>
<login>
<user>
<validate>
<fail>

(d) Profile of Method Login_OnClick

between q and d. Some methods require artifacts to be indexed before matching them. Both q and d were indexed using the indexing process described earlier [69]. The output of the indexing process is shown in Fig. 2c and d.

Performance of different methods is presented in precision/recall curves over various threshold levels (<.1, .2,…, 1>) [56]. A higher threshold level means a larger list of candidate links, i.e., more links were considered in the analysis. Wilcoxon signed-rank test is used to measure the statistical significance of the results. This is a nonparametric test that makes no assumptions about the distribution of the data [31]. This test is applied over the combined samples from two related samples or repeated measurements on a single sample (before and after effect). The IBM SPSS Statistics software package is used to conduct the analysis. We use $\alpha = 0.05$ to test the significance of the results. Note that different IR methods are applied independently, so there is no interaction effect between them.

4.1 Semantic-augmented methods

We start our analysis by examining the performance of the semantic-augmented methods including VSM with thesaurus support (VSM-T) and VSM with *Part-of-Speech* tagging (VSM-POS).

4.1.1 Vector space model with thesaurus support

Two methods are investigated under his category: VSM-T-DT and VSM-T-WN. We propose an optimization algorithm in order to specify acceptable approximations for α in Eq. 2. This algorithm is based on maximizing the recall. The main assumption is that IR-based tracing tools favor recall over precision. This is mainly because commission errors (false positives) are easier to deal with than omission errors (false negatives) [56]. The algorithm starts from $\alpha = 0$, gradually increasing this value by .05 each time, and monitoring the recall over constant threshold levels. The value of α that achieves the highest average recall at lowest threshold level (i.e., highest possible precision) is considered a local maximum. We run this algorithm over our three experimental datasets. Results show that average similarity coefficients of $\alpha = .43$ and $\alpha = .81$ achieve the best recall in VSM-T-WN and in VSM-TD, respectively. However, while this kind of optimization achieves acceptable performance levels, more sophisticated approaches, such as assigning different weights based on human judgment or statistical similarity analysis over WordNet, can be used. However, such analysis is beyond the scope of this paper.

Figure 3 shows the performance of the two methods in comparison with the VSM baseline in all three experimental datasets. Statistical analysis results are shown in the *Semantic-Augmented (Thesaurus)* section of Table 5. Analysis shows that the VSM baseline starts with relatively higher precision and recall at lower threshold levels. VSM-T-TD is able to catch up halfway through, keeping up the good performance until almost achieving a 100 % recall at higher threshold levels, while basic VSM stopped at 93.3 % recall. Figure 3 also shows the fast drop in the precision of VSM, while VSM-TD shows a more gradual decrease in the precision with the increase in the recall. Results also show the poor performance of VSM-T-WN, which performs significantly worst than VSM and VSM-T-TD in all three datasets. While VSM-T-WN is able to hit a 100 % recall at higher threshold levels, it retrieves so many

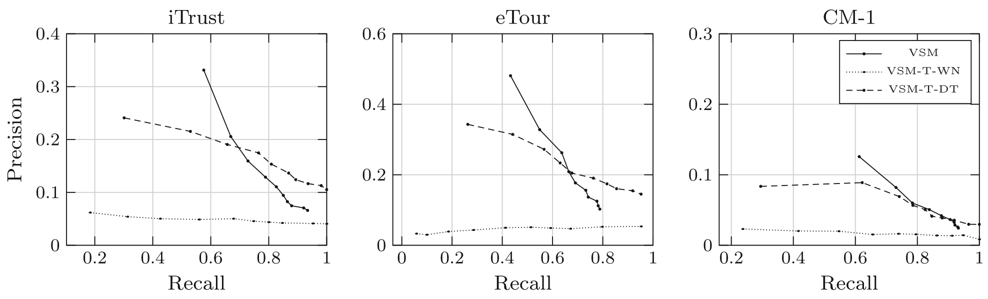

Fig. 3 VSM-T methods performance

false positives taking the precision down to significantly lower levels.

The above analysis shows that the explicit introduction of synonyms in VSM improves the overall recall. It also has a positive effect on the accuracy by keeping acceptable precision levels at higher recall levels. The poor performance of VSM-T-WN in comparison with VSM-T-TD can be explained based on the fact that WordNet is a general-purpose thesaurus; no domain knowledge is available to guide the synonym extraction process. Therefore, this method introduces a high noise-to-signal ratio that leads to retrieving a large number of false positives. In contrast, the domain thesaurus in WSM-T-TD was generated using terms from within the corpus, so noise levels were kept under control. Also, the domain knowledge helped to deal with non-English words that the English dictionary WordNet fails to handle, especially in the *eTour* dataset where Italian words were used.

To further confirm our findings, we refer to our example in Fig. 2. The similarity scores between q and d given by VSM, VSM-T-TD, and VSM-T-WN were .54, .63, and .47, respectively. Using VSM-T-WN, q's vector has been expanded with the following synonyms:

- *credential: certificate*
- *authenticate: formalize, corroborate*
- *validate: formalize, corroborate*
- *password: watchword, word, parole, countersign*
- *user: exploiter*
- *fail: miscarry, neglect, die, go, break, break, flunk, bomb*
- *forget: bury, block, leave.*

This list shows that WordNet introduces so many domain irrelevant terms (e.g., *parole*, *miscarry*). While such enrichment might have a positive influence on the recall, especially in retrieving some of the hard-to-trace requirements [42], it often causes a significant drop in the accuracy, which is reflected in the fast drop in the precision values of VSM-T-WN at higher threshold levels. This is also clearly shown by the *DiffAR* values in Table 4 which show that VMS-T-WN was the least successful in distinguishing between true and false links.

In contrast, using VSM-T-TD, the following synonym pairs were manually identified based on the domain's context: <*password*, *credential*>, <*email*, *credential*>, <*authenticate*, *validate*>, and <*patient*, *user*>. It is important to point out here that VSM-T-TD is also prone to noise. For example, the synonym pair <*patient*, *user*> might cause some confusion, as in the *iTrust* dataset, the term *user* might refer to individuals other than patients, such as visitors or doctors. However, regardless of that small amount of noise, VSM-T-TD still gives a higher similarity score between q and d, which is desirable since (q, d) is actually a correct link. In addition, the *DiffAR* values of VSM-T-TD, shown in Table 4, show that this method achieves an acceptable distinction between true and false links in comparison with VSM and VSM-T-WN.

4.1.2 VSM with part-of-speech tagging

Under this category we analyze the performance of VSM-POS-N, in which only nouns are considered in the indexing process, and VSM-POS-V, in which only verbs are considered. POS is applied before indexing to preserve the grammatical structure of the text. In case of source code, this process depends heavily on the availability of free text comments that can be correctly parsed. After indexing,

Table 4 DiffAR values taken at 0.7 threshold

	Baseline	Thesaurus support		POS support		Latent semantic		Semantic relatedness	
	VSM	VSM-T-TD	VSM-T-WN	VSM-POS-N	VSM-POS-V	LSI	LDA	NGD	ESA
iTrust	0.3	0.21	0.17	0.33	0.33	0.01	0.01	0.02	0.11
eTour	0.22	0.16	0.1	0.39	0.32	0.01	0.01	0.07	0.12
CM-1	0.1	0.09	0.06	0.21	0.2	0.01	0.01	0.01	0.09

documents are matched using the standard cosine similarity (Eq. 1).

Performance precision/recall curves for running the two VSM-POS methods over our three experimental datasets are shown in Fig. 4. Statistical analysis results are shown in the *Semantic-Augmented (POS)* section of Table 5. Results show that traceability link retrieval is heavily affected by the grammatical filters. In both cases, considering only one part of speech has a significantly negative effect on recall in all three datasets. Even though considering only nouns has relatively less negative impact on the performance than considering only verbs, it still fails to match the baseline's recall. The relatively better performance of both methods in *CM-1* can be explained based on the fact that *CM-1* is a requirement-to-design dataset, and free text is used to describe artifacts at both sides of the traceability link. This allowed the POS tagger to generate more accurate lists of candidate links for this particular dataset in comparison with the two other datasets. Results also show that considering only nouns in the indexing process achieves a significantly higher recall than indexing verbs only. This suggests that nouns carry more information value when retrieving traceability links. However, such information value is not sufficient enough to achieve optimal recall levels.

In general, it can be concluded that this kind of augmentation fails to achieve a satisfactory performance in the domain of automated tracing. However, if high precision levels are favored over recall, these methods can be useful as they tend to filter out a large portion of unwanted noise, usually caused by some irrelevant terms generated by the indexing process. This behavior is clearly reflected in the *DiffAR* values (Table 4) which show that VSM-POS methods generate the highest values in terms of distinguishing between true positives and false positives. However, this success does not give them an edge over the baseline, as they significantly fail to outperform basic VSM.

Considering our example in Fig. 2, VSM-POS-V reduces q and d vectors to <*login*, *Authenticate*, *forget*> and <*login*, *validate*, *fail*>, respectively. In contrast, VSM-POS-N reduces q to <*patient*, *credential*, *password*> and d to <*patient*, *user*, *email*, *password*>. Using VSM-POS-V, q and d only match at <*login*> taking the similarity down to 0.224, and VSM-POS-N has two matches <*patient*, *password*> also taking the similarity score of the true (q, d) link down to 0.336.

4.2 Latent semantic methods

Under this category, we analyze the performance of the latent methods LSI and LDA. To approximate an optimal value of K, LSI is ran iteratively while the K value is gradually increased by 5 after each iteration. K values that produce globally better precision/recall, averaged over all the instances of each dataset, are kept. Running this optimization procedure over our three experimental datasets produced K values of 35, 40, and 45 for *iTruts*, *eTour*, and *CM-1*, respectively.

We follow a similar experimental approach to approximate an optimal number of topics (K) for LDA. In particular, K is initially set to 40 topics. The document-topic distribution matrix of each artifact in the system is then generated. A cosine comparison (Eq. 2) is conducted to capture matching in the latent topic structures of different artifacts, generating candidate traceability links. The value of K is then increased by 40, and the process is repeated. This particular step size of 40 is the minimum value that yields noticeable changes in the recall. As mentioned earlier, we tie optimality to recall in our analysis. Therefore, we follow a hill-climbing approach to monitor the changes in the recall, and best recall values were detected at K values of 160, 180, and 180 for *iTrust*, *eTour*, and *CM-1*, respectively. At this range of K, topics tend to be more distinguishable from each other, which makes these particular values nearly optimal for traceability analysis.

It is important to point out that the complexity of the study grows exponentially with the inclusion of other LDA parameters such as α and β. Therefore, at this stage of our analysis, we fix these values. This strategy is often used in related research to control for such parameters' effect [47, 65, 103]. In particular, values of $\alpha = 50/K$ and $\beta = 0.1$ are

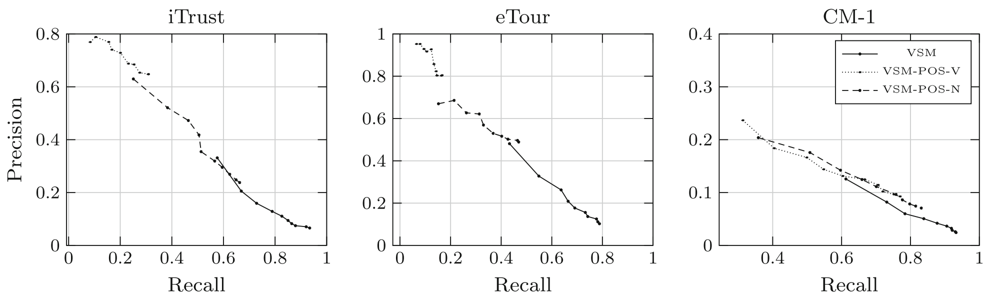

Fig. 4 VSM-POS methods performance

Table 5 Wilcoxon signed-rank test results ($\alpha = .05$)

	iTrust		eTour		CM-1	
	Recall (Z, *p value*)	Precision (Z, *p value*)	Recall (Z, *p value*)	Precision (Z, *p value*)	Recall (Z, *p value*)	Precision (Z, *p value*)
Semantic augmented						
VSM × VSM-T-TD	(−.357, .721)	(−1.785, .074)	(−.255, .799)	(−1.580, .114)	(−1.784, .074)	(−1.785, .074)
VSM × VSM-T-WN	(−2.293, .022)	(−2.803, <.005)	(−2.293, .022)	(−2.803, <.005)	(−2.191, .028)	(−2.803, <.005)
VSM-T-TD × VSM-T-WN	(−2.666, .008)	(−2.803, <.005)	(−2.666, .008)	(−2.803, <.005)	(−2.666, .008)	(−2.803, <.005)
Semantic augmented						
VSM × VSM-POS-N	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)
VSM × VSM-POS-V	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)
VSM-POS-N × VSM-POS-V	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)
Latent semantic						
VSM × LSI	(−.866, .386)	(−2.803, <.005)	(−.459, .646)	(−2.803, <.005)	(−2.090, .037)	(−2.803, <.005)
VSM × LDA	(−1.478, .139)	(−2.803, <.005)	(−.051, .959)	(−2.803, <.005)	(−1.784, .074)	(−2.803, <.005)
LSI × LDA	(−2.490, .013)	(−2.380, .017)	(−1.599, .110)	(−.652, .515)	(−.652, .515)	(−.178, .859)
Semantic relatedness						
VSM × ESA	(−1.580, .114)	(−.765, .444)	(−.866, .386)	(−2.803, <.005)	(−.968, .333)	(−2.803, <.005)
VSM × NGD	(−1.682, .093)	(−2.803, <.005)	(−.663, .508)	(−2.803, <.005)	(−1.479, .139)	(−2.803, <.005)
ESA × NGD	(−.652, .515)	(−2.803, <.005)	(−2.666, .008)	(−2.803, <.005)	(−2.547, .011)	(−2.803, <.005)
Inter-category						
VSM-T-TD × VSM-POS-N	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.701, .007)	(−2.803, <.005)
ESA × VSM-T-TD	(−2.429, .015)	(−2.599, .009)	(−2.666, .008)	(−2.803, <.005)	(−2.666, .008)	(−2.803, <.005)
ESA × VSM-POS-N	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−1.988, .047)
LSI × VSM-T-TD	(−2.192, .028)	(−2.803, <.005)	(−1.599, .110)	(−2.803, <.005)	(−.652, .515)	(−2.803, <.005)
LSI × VSM-POS-N	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)	(−2.803, <.005)
ESA × LSI	(−2.668, .008)	(−2.803, <.005)	(−2.547, .011)	(−2.803, <.005)	(−2.666, .008)	(−2.803, <.005)

used. These heuristics have been shown to achieve satisfactory performance in the literature [48, 105].

The performance of LSI and LDA over our three datasets in comparison with the baseline is shown in Fig. 5. Statistical analysis results are shown in the *Latent Semantics* section of Table 5. The results show that, in comparison with VSM, in all three datasets, both methods achieve relatively better recall. However, the only statistically significant improvement in recall over the baseline is achieved by LSI over *CM-1*. The results also show that this improvement in the recall has taken the precision down to significantly lower levels in all three datasets. Further more, a closer look at the recall and precision curves shows that, in *iTrust* and *CM-1*, LSI manages to outperform LDA at lower threshold levels. This difference in the performance is more obvious in the *iTrust* dataset where LSI does significantly better than LDA in terms of precision and recall. In the *eTour* dataset, both LDA and LSI achieve an interchangeably good performance before reaching the maximum recall and minimum precision point, i.e., all the links are retrieved (considered relevant).

Applying the latent methods over *q* and *d* in Fig. 2 shows that both methods produce relatively low similarity scores. LSI returns a similarity score of .032, and LDA returns a score of .007. These similarity scores were generated at class-granularity level. For instance, in LDA, the topic distribution of the class that contains the function `Login_OnClick` (Fig. 2a) was matched with the topic distribution of requirement 3.1.3 (Fig. 2a).

The relatively higher score of LSI might be explained based on its operation, such as a detected synonym relation between <*credential*, *password*>. However, there is no clear indication if that is actually the case, or just a mathematical coincidence. In general, the poor performance of latent methods can be ascribed to their internal operation. When dealing with software artifacts, the amount of knowledge available in a corpus is not expressive enough to produce meaningful document-topic or

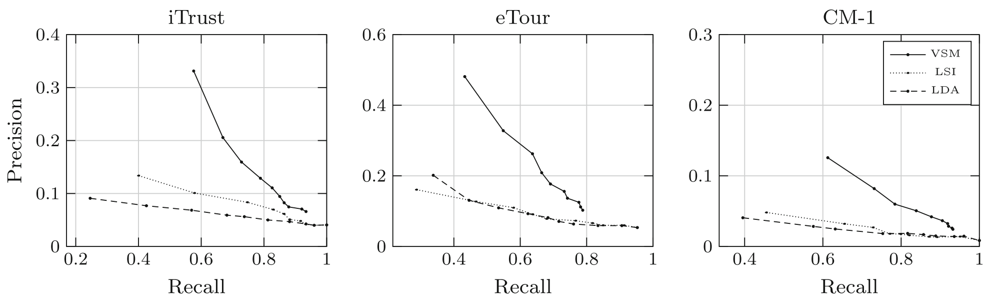

Fig. 5 Latent semantic methods performance

document-term matrices for LDA and LSI, which makes these methods prone to mathematical noise. This behavior was clearly reflected in the *DiffAR* values shown in Table 4. In all three datasets, both latent methods are the least successful in distinguishing between true and false links, achieving the smallest *DiffAR* values among other investigated methods.

4.3 Semantic relatedness methods

Under this category, the methods of ESA and NGD are investigated. Performance of both methods in comparison with the baseline is shown in Fig. 6. Statistical analysis results are shown in the *Semantic Relatedness* section of Table 4. Results show that in all three datasets, both methods are able to hit a 100 % recall at higher threshold levels. However, this improvement over the baseline's recall is statistically insignificant. On the other hand, the precision is affected negatively due to the high number of false positives, which is more obvious in NGD where the precision at 100 % recall hits a minimum.

The diagrams also show that ESA achieves significantly better precision and recall than NGD in all datasets. This can be explained based on the fact that *Wikipedia*, being a semi-structured source of knowledge, cancels a high ratio of noise usually returned by search engines, thus achieving a higher precision. Another potential reason for the relatively poor performance of NGD is the oversimplification of the problem. While ESA utilizes a smarter approach for extracting relatedness measures, NGD simply relies on hit counts to derive similarity, ignoring several inherent problems related to term ambiguity.

To gain more insights into these methods, we refer to our example in Fig. 2. ESA compares vectors of text, thus the domain knowledge is somewhat preserved through the context. For example, the word *fail* in the context <*user*, *login*, *fail*> obviously refers to a failure in the login process. In contrast, due to the lack of context in NGD, terms such as *user* and *fail* can refer to so many types of failure. This behavior was reflected in the *DiffAR* values shown in Table 4, which shows that ESA is more successful in differentiating between true links and false links.

4.4 Inter-category comparison

We conduct a comprehensive analysis to compare the performance of the best performing methods from each category. Results are shown in Fig. 7. Pairwise statistical analysis is shown in the *Inter-Category* part of Table 5. In terms of recall, in all three datasets, VSM-T-DT, ESA, and LSI were able to reach a maximum recall at higher threshold levels, with VSM-T-DT achieving significantly higher precision, followed by ESA, which in turn significantly outperforms LSI. In terms of precision, VSM-POS-N achieves highest precision, outperforming all other methods significantly. However, it fails to achieve an acceptable recall.

In terms of browsability measures, Figs. 8 and 9 show the *MAP* and *Lag* results of the best performing methods from the different categories of semantically enabled IR methods. In general, results show that methods that achieve a reasonable performance in terms of quality, such as VSM-T-TD and ESA, also tend to achieve a good performance in terms of browsability.

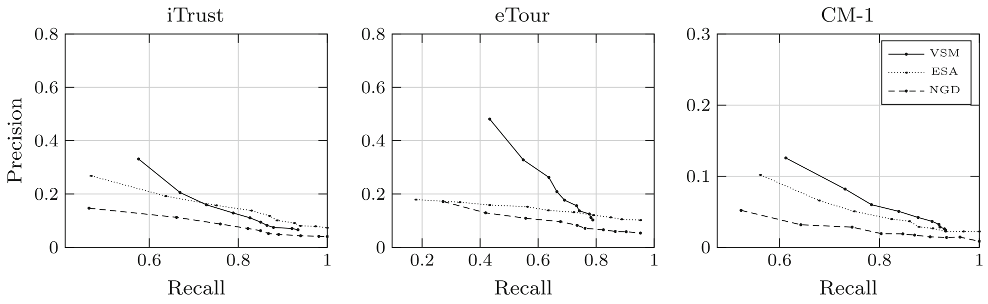

Fig. 6 Semantic relatedness methods performance

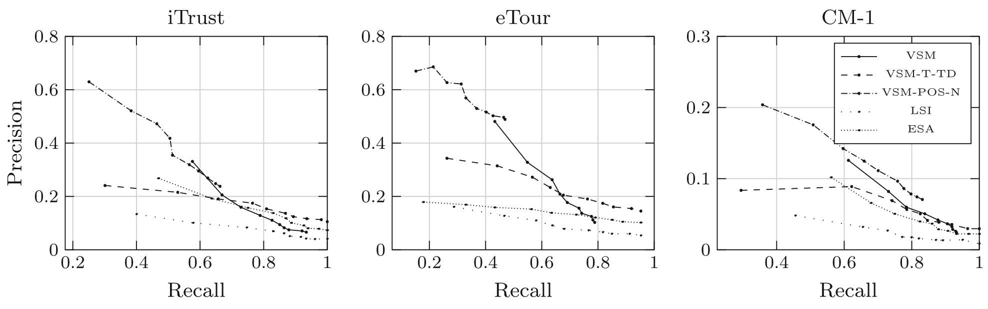

Fig. 7 Overall performance

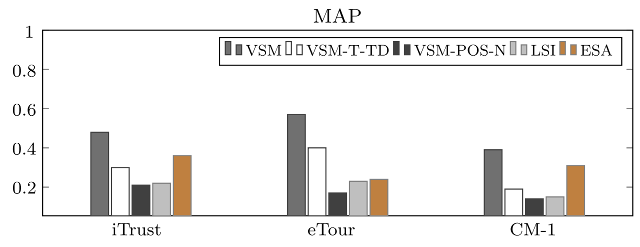

Fig. 8 MAP values in iTrust, eTour, and CM-1

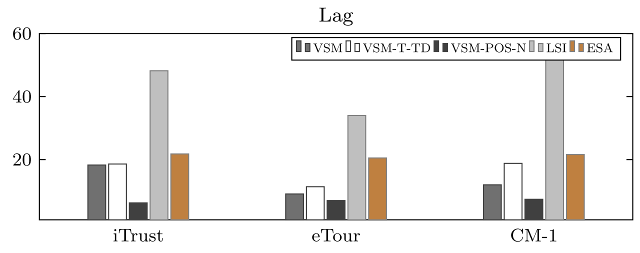

Fig. 9 Lag values in iTrust, eTour, and CM-1

In particular, in terms of *MAP*, Fig. 8 shows that VSM-T-DT and ESA achieve an interchangeably good performance over the experimental datasets, ESA outperforms VSM-T-TD in *iTrust* and *CM-1* while VSM-T-TD outperforms ESA in *eTour*. The results also show the constantly poor performance of LSI and VSM-POS-N, achieving a significantly lower MAP in all datasets. LSI tends to scatter true positives all over the ranked list of candidate links, with higher concentration of these links at the bottom of the list, thus taking the average precision (AP) value down for each query. VSM-POS-N, on the other hand, captured a smaller number of correct links, thus increasing the number of links with 0 precision (false negatives) in Eq. 8.

Similar patterns are observed in the *Lag* results, shown in Fig. 9. The method that achieved the highest precision (VSM-POS-N) acquired the lowest *Lag* values (i.e., smaller number of false positives separate true positives). Results also show that VSM-T-TD achieves a comparable performance to the baseline, also outperforms ESA and LSI in all three datasets. While ESA manages to keep very close performance in both *iTrust* and *CM-1*, in the *eTour* dataset ESA performance is significantly worst. In contrast, LSI achieves the worst performance in all three datasets. This means that LSI tends to scatter the correct links all over the list with higher numbers of false positives separating them.

Finally, while identifying a winning technique is not a main goal of our analysis, in terms of the primary and secondary performance measures, our overall analysis results lean toward announcing VSM-T-TD as the most reliable semantically enabled IR method for traceability link recovery.

4.5 Limitations

Several factors can affect the validity of our study. Construct validity is the degree to which the variables accurately measure the concepts they purport to measure [26]. In our experiment, there are minimal threats to construct validity as standard IR measures (recall and precision), which are used extensively in requirements traceability research, are used to assess the different methods investigated in our paper. These measures are also complemented by another set of secondary measures (*MAP*, *DiffAR*, and *Lag*) that are used to provide more insights into the browsability of the generated lists of candidate traceability links. We believe that these two sets of measures sufficiently capture and quantify the different performance aspects of the various methods evaluated in this study.

Threats to external validity impact the generalizability of results. In particular, the findings of this study might not generalize beyond the underlying experimental settings [26]. A major threat to the external validity comes from the datasets used in our experiment. In particular, two of these datasets were developed by students and may not be representative of a program written by industrial professionals. Also, all three of our datasets are limited in size, which raises some scalability concerns. However, we believe that the use of three datasets, from different application domains, helps to mitigate these threats. Finally, specific design decisions and heuristics used during the experiment can also limit the study's applicability. Such decisions include using *Wikipedia* 2009 in ESA, using TFIDF weights in our baseline, the decision of only considering verbs and nouns in VSM-POS, and the heuristic values of α and β used in LDA.

Internal validity refers to factors that might affect the causal relations established in the experiment. A potential threat to our study's internal validity comes from our specific implementation of the different methods investigated in this paper. However, we believe that using freely available open source tools and libraries in our implementations helps to mitigate this threat. It also makes it possible to independently replicate our study. In addition, an experimental bias might stem from the fact that some of the procedures in our experiment were carried out manually. For instance, the local domain thesaurus in VSM-T-TD was created manually by our researchers, based on their understanding of the system, which might raise some subjectivity concerns that can affect the internal validity of our study.

5 Discussion and impact

Capturing and maintaining accurate lists of requirements traceability links is vital to managing requirements in the multiple phases of the software development process [44, 56]. In this paper, we investigated the performance of several semantically enabled IR methods in bridging the semantic gap that often appears in the system as a direct result of software evolution [61, 62]. In particular, we experimentally assessed the effect of different semantic schemes on the performance of various IR-based traceability methods.

Our results revealed that explicit semantic methods (VSM-T, VSM-POS, ESA, and NGD) tend to do a better job in recovering traceability links than latent methods (LSI and LDA). Though latent methods able to achieve higher recall levels, they often fail to compete with the precision of other methods. This can be explained based on the fact that lexicons and syntax of NL documents differ from those of software artifacts. Source code is more constrained and less varied than natural language, which makes it more regular and repetitive [54, 63]. This limits the ability of latent methods to extract hidden semantics schemes using statistical and probabilistic models. In fact, latent methods were initially developed to work with contents of large document collections [15, 48, 77, 107]. However, software systems are often much smaller than NL text corpora, depriving such methods of context, and causing them to behave randomly, even when calibrated using settings that usually achieve adequate performance over natural language corpora [54, 93]. For instance, poor parameter calibration or wrong assumptions about the nature of the data often lead a method such as LDA to generate several irrelevant or duplicated topics [82].

Probably the most interesting observation in our analysis is that considering more semantic relations in retrieval does not necessarily lead to a better tracing performance. Instead, a local and a more focused semantic support is expected to serve the automated tracing problem better. This was clearly reflected in the performance of VSM-T and ESA, while both methods achieved a relatively good performance, VSM-T managed to keep a significantly higher precision in all three datasets, also significantly higher recall in both *eTour* and *CM-1*. In particular, our analysis shows that methods, which only consider the semantic relation of *synonymy*, tend to be the most reliable for traceability link recovery. This can be explained based on the fact that software artifacts are not as semantically rich or complex as free text. In fact, it has been observed that developers tend to shy away from using fancy language when writing specifications. Instead, software artifacts are usually expressed in a simplified form of the natural language, with a smaller vocabulary set and simplified grammars [33]. In addition, source code developers tend to abbreviate names; causing concepts to be denoted through their full name as well as multiple abbreviations [5, 28], thus increasing the density of synonymy relations in software systems.

Our results also show that external sources of knowledge such as *Wikipedia* or WordNet tend to increase the level of noise in retrieval. This can be explained based on the fact that often a coherent vocabulary structure, derived from the system application domain, is used through out the project's lifecycle. Therefore, as shown in our analysis, using external and general-purpose sources of knowledge to enrich the system's vocabulary will most likely corrupt that coherent structure with unrelated terms, thus causing a significant decline in the precision of IR methods. In addition, as observed earlier, synonyms generated by abbreviating domain names seem to be the most occurring semantic relation during software evolution. Being domain-specific, such synonyms are often not included in general-purpose dictionaries or knowledge sources.

Finally, in terms of tool support, our results reveal how different methods, at different levels of semantic support, might be useful in certain contexts of requirements management. For example, in tools where accuracy is the main concern, methods that achieve significant precision levels might be useful (e.g., VSM-POS). However, in safety-critical systems, which imposes special demands on ensuring quality and reliability of the system, methods that achieve higher recall levels might be more appropriate [21]. In addition, if practicality is a major concern, then methods that utilize external knowledge sources such as *Wikipedia* or WordNet should be avoided. Such methods require relatively higher time and space requirements to function properly (e.g., initiating multiple Web search requests or long search queries in NGD, or downloading and indexing *Wikipedia* in ESA).

6 Related work

Several IR-based traceability link recovery methods have been proposed in the literature. Next we selectively review some of the seminal work in this domain over the last decade. Initial work on IR-based traceability was conducted by Antoniol et al. [6]. The authors used probabilistic network model (PN) and basic VSM to recover traceability links between source code and free text documents. This work provided an initial evidence of the practicality of IR methods as a potential solution for the automated tracing problem. Marcus and Maletic repeated the same case study using LSI [73]. They compared the performances of LSI with VSM and PN. Results showed that LSI can achieve a comparable performances without the need for stemming.

Huffman-Hayes et al. [55] used two different variants of VSM including retrieval with key phrases and VSM with thesaurus support, to recover traceability links between requirements. The former approach was found to improve recall; however, it affected precision negatively. On the other hand, VSM with thesaurus support resulted in improved recall and precision. A more recent work by the same authors addressed issues related to improving the overall quality of the automated tracing process [56]. In particular, the authors analyzed the performance of several IR methods including VSM, VSM with thesaurus support, and LSI, and incorporating relevance feedback from human analysts in the retrieval process. Results showed that using analysts' feedback to tune the weights in the term-by-document matrix of the VSM improved the final tracing results.

Settimi et al. [95] compared the performance of several IR techniques in tracing UML models. In particular, the authors applied VSM and one of its variants that uses pivot normalization to trace requirements to UML artifacts, code, and test cases. The results raised some concerns about the adequacy of the IR-based paradigm in solving the traceability problem. However, they found that such methods can be used to alleviate some of the manual effort in requirements tracing tasks. Similarly, Oliveto et al. [81] compared the performance of several IR-based traceability recovery methods including the Jensen–Shannon (JS) method, VSM, LSI, and LDA. Results showed that JS, VSM, and LSI were almost equivalent in that they captured almost the same information. However, LDA achieved lower accuracy.

Cleland-Huang et al. [22] introduced three enhancement strategies (hierarchical modeling, logical clustering of artifacts, and semi-automated pruning) to improve the performance of the probabilistic network model. Results showed that such strategies can be used effectively to improve trace retrieval results and increase the practicality of tracing tools. Similar to this work, in our previous work [79], we proposed an approach based on the cluster hypothesis to improve the quality of candidate link generation for requirements tracing. The main assumption was that correct and incorrect links can be grouped into high-quality and low-quality clusters, respectively. Result accuracy can thus be enhanced by identifying and filtering out low-quality clusters. Evaluating this approach over multiple datasets showed that it outperformed a baseline pruning strategy.

Lormans and van Deursen [66] used a new strategy, based on LSI, to trace requirements to test case specifications and design artifacts. Their experimental analysis on three case studies provided an evidence that LSI can increase the insight in a system by means of reconstructing the traceability links between the different artifacts. Later, Asuncion et al. [8] employed topic modeling through the use of latent Dirichlet allocation (LDA) to recover different types of traceability links. Results showed that the combination of traceability with topic modeling can be useful in practice.

Gibiec et al. [42] used a Web-based query expansion algorithm to bridge the vocabulary gap in the system. Evaluating this approach over a group of healthcare datasets showed its ability to improve the traceability of hard-to-trace requirements. Similarly, in our earlier work [70], we introduced semantic relatedness as an approach for traceability link recovery. Results showed that the *Wikipedia*-based ESA achieves a balance between LSI and VSM. It significantly outperforms the recall of VSM and the precision of LSI in most cases, showing more stable performance at different threshold levels. In addition, we conducted a preliminary analysis of VSM with *Part-of-Speech* tagging in recovering traceability links [68]. Results showed that POS could not beat basic VSM in terms of precision and recall. However, a more recent work on POS was conducted by Capobianco et al. [17]. Analysis of this approach over five software artifact repositories indicated that noun-based indexing of software artifacts significantly improves the accuracy of IR-based traceability recovery methods.

7 Conclusions and future work

7.1 Summary

In this paper, we conducted an experimental analysis to assess the performance of various semantically enabled IR methods, including semantic-augmented, latent semantic, and semantic relatedness methods, in capturing requirements traceability links in software systems.

The performance of the different methods in terms of results' quality and browsability was compared to the basic VSM as an experimental baseline. Results showed that explicit semantic methods (VSM-T, VSM-POS, ESA, and

NGD) tend to outperform latent methods (LSA and LDA). In addition, results revealed that methods that use selective indexing based on the lexical form of terms (VSM-POS) cancel a considerable amount of textual noise, thus achieve the best precision. However, such methods often suffer on the recall as a considerable amount of information is lost when filtering out other parts of speech. Results also showed that considering more semantic relations in retrieval does not necessarily lead to improved performance. Instead, a more focused explicit semantic support, in particular synonyms from a domain-specific thesaurus, is expected to achieve more adequate performance levels.

7.2 Future work

The line of work in this paper has opened several research directions to be pursued in our future work. These directions can be described as following:

- Automated tracing methods: Our work in this paper is limited to VSM-based methods. In our future work, we are interested in exploring other semantically enabled methods that apply different semantic schemes to the problem. For instance, we are interested in assessing the performance of ontology-based traceability tools which have been shown to achieve satisfactory performance in the domain of automated tracing [50, 108].
- Requirements engineering tasks: In this paper, we have limited our analysis to the requirements traceability problem. In our future work, we are interested in studying the performance of semantically enabled methods in supporting other requirements engineering tasks in which the IR paradigm is often employed. Such tasks include, for example, reusable requirements retrieval [68], requirements discovery [13], and evolution [10].
- Tool support: Our analysis in this paper suggested several guidelines for the design and development of practical automated tracing tools. It is in the scope of our future work to implement these findings in a working prototype to support various requirements engineering tasks besides traceability. In addition, a working prototype will allow us to conduct human studies to assess the usability of different methods in practice.

Acknowledgments This work is supported in part by the US NSF (National Science Foundation) Grant CCF1238336.

References

1. Abebe S, Tonella P (2010) Natural language parsing of program element names for concept extraction. In: International conference on program comprehension, pp 156–159
2. Ahn D, Jijkoun V, Mishne G, Mller K, de Rijke M, Schlobach S (2004) Using Wikipedia at the TREC QA track. In: Interactive poster and demonstration sessions, pp 73–76
3. Andrzejewski D, Mulhern A, Ben BL, Zhu X (2007) Statistical debugging using latent topic models. In: European conference on machine learning, pp 6–17
4. Anquetil N, Fourrier C, Lethbridge T (1999) Experiments with clustering as a software remodularization method. In: Working conference on reverse engineering, pp 235–255
5. Anquetil N, Lethbridge T (1998) Assessing the relevance of identifier names in a legacy software system. In: Conference of the centre for advanced studies on collaborative research, pp 4–14
6. Antoniol G, Caprile B, Potrich A, Tonella P (2000) Design-code traceability for object-oriented systems. Ann Softw Eng 9(1–4):35–58
7. Aslam J, Yilmaz E, Pavlu V (2005) A geometric interpretation of r-precision and its correlation with average precision. In: Annual international ACM SIGIR conference on research and development in information retrieval, pp 573–574
8. Asuncion H, Asuncion A, Taylor R (2010) Software traceability with topic modeling. In: International conference on software engineering, pp 95–104
9. Baeza-Yates R, Ribeiro-Neto B (1999) Modern information retrieval. Addison-Wesley, Reading, MA
10. Ben Charrada E, Koziolek A, Glinz M (2012) Identifying outdated requirements based on source code changes. In: International requirements engineering conference, pp 61 –70
11. Biggers L, Bocovich C, Capshaw R, Eddy B, Etzkorn L, Kraft N (2012) Configuring latent Dirichlet allocation based feature location. Empir Softw Eng 1–36
12. Biggerstaff T, Mitbander B, Webster D (1994) Program understanding and the concept assignment problem. Commun ACM 37(5):72–82
13. Binkley D, Lawrie D (2010) Information retrieval applications in software development. In: Computer technologies and information sciences, chap 37
14. Binkley D, Lawrie D (2011) Maintenance and evolution: information retrieval applications. In: Laplante PA (ed) Encyclopedia of software engineering, pp 454–463
15. Blei D, Ng A, Jordan M (2003) Latent Dirichlet allocation. J Mach Learn Res 3:993–1022
16. Budanitsky A, Hirst G (2006) Evaluating wordnet-based measures of lexical semantic relatedness. Comput Linguist 32(1):13–47
17. Capobianco G, De Lucia A, Oliveto R, Panichella A, Panichella S (2013) Improving IR-based traceability recovery via noun-based indexing of software artifacts. J Softw Maint Evolut Res Pract 25(7):743–762
18. Chen P, Lin S (2010) Automatic keyword prediction using Google similarity distance. Expert Syst Appl 37(3):1928–1938
19. Chowdhury A, McCabe M (1998) Improving information retrieval systems using part of speech tagging. Technical report, ISR, Institute for Systems Research
20. Cilibrasi R, Vitanyi P (2007) The google similarity distance. IEEE Trans Knowl Data Eng 19(3):370–383
21. Cleland-Huang J, Heimdahl M, Huffman-Hayes J, Lutz R, Mäder P (2012) Trace queries for safety requirements in high assurance systems. In: International conference on requirements engineering: foundation for software quality, pp 179–193
22. Cleland-Huang J, Settimi R, Duan C, Zou X (2005) Utilizing supporting evidence to improve dynamic requirements traceability. In: International conference on requirements engineering, pp 135–144
23. Cleland-Huang J, Settimi R, Romanova E (2007) Best practices for automated traceability. Computer 40(6):27–35

24. De Lucia A, Fasano F, Oliveto R, Tortora G (2007) Recovering traceability links in software artifact management systems using information retrieval methods. ACM Trans Softw Eng Methodol 16(4):13–50
25. De Lucia A, Oliveto R, Zurolo F., Di Penta M (2006) Improving comprehensibility of source code via traceability information: a controlled experiment. In: International conference on program comprehension, pp 317–326
26. Dean A, Voss D (1999) Design and analysis of experiments. Springer, Berlin
27. Deerwester S, Dumais S, Furnas G, Landauer T, Harshman R (1990) Indexing by latent semantic analysis. J Am Soc Inf Sci 41(6):391–407
28. Deiβenböck F, Pizka M (2005) Concise and consistent naming. In: International workshop on program comprehension, pp 97–106
29. Dekhtyar A, Huffman-Hayes J, Antoniol G (2007) Benchmarks for traceability? In: International symposium on grand challenges in traceability
30. Demmel J, Kahan W (1990) Accurate singular values of bidiagonal matrices. J Sci Stat Comput 11(5):873–912
31. Demšar J (2006) Statistical comparisons of classifiers over multiple data sets. J Mach Learn Res 7:1–30
32. Dumais S (1993) Lsi meets trec: a status report. In: Harman D (ed) The first text retrieval conference (TREC1), National Institute of Standards and Technology Special Publication, pp 137–152
33. Etzkorn L, Davis C (1995) An approach to object-oriented program understanding. In: IEEE workshop on program comprehension, pp 14–15
34. Evans D, Zhai C (1996) Noun-phrase analysis in unrestricted text for information retrieval. In: Annual meeting on association for computational linguistics, pp 17–24
35. Falleri J, Huchard M, Lafourcade M, Nebut C, Prince V, Dao M (2010) Automatic extraction of a wordnet-like identifier network from software. In: International conference on program comprehension, pp 4–13
36. Fang J, Guo L (2011) Calculation of relatedness by using search results. In: International workshop on intelligent systems and applications, pp 1–4
37. Feilkas M, Ratiu D, Jurgens E (2009) The loss of architectural knowledge during system evolution: an industrial case study. In: International conference on program comprehension, pp 188–197
38. Fellbaum C (1998) WordNet: an electronic lexical database. MIT Press, Cambridge, MA
39. Finkelstein L, Gabrilovich E, Matias Y, Rivlin E, Solan Z, Wolfman G, Ruppin E (2002) Placing search in context: the concept revisited. ACM Trans Inf Syst 20(1):116–131
40. Furnas G, Deerwester S, Dumais S, Landauer T, Xarshman R, Streeter L, Lochbaum K (1988) Information retrieval using a singular value decomposition model of latent semantic structure. In: Annual international ACM SIGIR conference on research and development in information retrieval, pp 465–480
41. Gabrilovich E, Markovitch S (2007) Computing semantic relatedness using Wikipedia-based explicit semantic analysis. In: International joint conference on artificial intelligence, pp 1606–1611
42. Gibiec M, Czauderna A, Cleland-Huang J (2010) Towards mining replacement queries for hard-to-retrieve traces. In: International conference on automated software engineering, pp 245–254
43. Gligorov R, Aleksovski Z, Kate W, Harmelen F (2007) Using Google distance to weight approximate ontology matches. In: International conference on world wide web, pp 767–776
44. Gotel O, Finkelstein A (1994) An analysis of the requirements traceability problem. In: International conference on requirements engineering, pp 94–101
45. Gotel O, Finkelstein A (1995) Contribution structures. In: International symposium on requirements engineering, pp 100–107
46. Gotel O, Morris S (2011) Out of the labyrinth: leveraging other disciplines for requirements traceability. In: IEEE international requirements engineering conference, pp 121–130
47. Grant S, Cordy J (2010) Estimating the optimal number of latent concepts in source code analysis. In: International working conference on source code analysis and manipulation, pp 65–74
48. Griffiths T (2004) Steyvers M finding scientific topics. In: The National Academy of Sciences, pp 5228–5235
49. Grzywaczewski A, Iqbal R (2012) Task-specific information retrieval systems for software engineers. J Comput Syst Sci 78(4):1204–1218
50. Guo Y, Yang M, Wang J, Yang P, Li F (2009) An ontology-based approach for traceability recovery. In: International symposium on knowledge acquisition and modeling, pp 160–163
51. Han E, Karypis G (2000) Centroid-based document classification: Analysis and experimental results. In: European conference on principles of data mining and knowledge discovery, pp 424–431
52. Hata M, Homae F, Hagiwara H (2009) Semantic relatedness between words in each individual brain: an event-related potential study. Neurosci Lett 501(2):72–77
53. Hazen T (2010) Direct and latent modeling techniques for computing spoken document similarity. In: Spoken language technology workshop, pp 366–371
54. Hindle A, Bird C, Zimmermann T, Nagappan N (2012) Relating requirements to implementation via topic analysis: do topics extracted from requirements make sense to managers and developers? In: International conference on software maintenance, pp 243–252
55. Huffman-Hayes J, Dekhtyar A, Osborne J (2003) Improving requirements tracing via information retrieval. In: International conference on requirements engineering, pp 138–147
56. Huffman-Hayes J, Dekhtyar A, Sundaram S (2006) Advancing candidate link generation for requirements tracing: the study of methods. IEEE Trans Softw Eng 32(1):4–19
57. Jurafsky D, Martin J (2000) Speech and language processing. Prentice Hall, Englewood Cliffs NJ
58. Kit L, Man C, Baniassad E (2006) On finding duplication and near-duplication in large software systems. In: Annual ACM SIGPLAN conference on object-oriented programming systems, languages, and applications, pp 383–396
59. von Knethen A (2002) Automatic change support based on a trace model. In: International workshop on traceability in emerging forms of software engineering
60. Kuhn A, Ducasse S, Gírba T (2007) Semantic clustering: Identifying topics in source code. Inf Softw Technol 49(3):230–243
61. Lawrie D, Feild H, Binkley D (2007) Extracting meaning from abbreviated identifiers. In: International working conference on source code analysis and manipulation, pp 213–222
62. Lehman M (1984) On understanding laws, evolution, and conservation in the large-program life cycle. J Syst Softw 1(3):213–221
63. Linstead E, Rigor P, Bajracharya S, Lopes C, Baldi P (2007) Mining concepts from code with probabilistic topic models. In: International conference on automated software engineering, pp 461–464
64. Lioma C, Blanco R (2009) Part of speech based term weighting for information retrieval. In: Advances in information retrieval, pp 412–423
65. Liu Y, Poshyvanyk D, Ferenc R, Gyimóthy T, Chrisochoides N (2009) Modelling class cohesion as mixtures of latent topics. In: International conference on software maintenance, pp 233–242

66. Lormans M (2006) Can lsi help reconstructing requirements traceability in design and test. In: European conference on software maintenance and reengineering, pp 47–56
67. Luisa M, Mariangela F, Pierluigi I (2004) Market research for requirements analysis using linguistic tools. Requir Eng 9(1):40–56
68. Mahmoud A, Niu N (2010) Using semantics-enabled information retrieval in requirements tracing: An ongoing experimental investigation. In: Annual computer software and applications conference, pp 246–247
69. Mahmoud A, Niu N (2011) Source code indexing for automated tracing. In: International workshop on traceability in emerging forms of software engineering, pp 3–9
70. Mahmoud A, Niu N, Xu S (2012) A semantic relatedness approach for traceability link recovery. In: International conference on program comprehension, pp 183–192
71. Maletic J, Marcus A (2000) Using latent semantic analysis to identify similarities in source code to support program understanding. In: International conference on tools with artificial intelligence, pp 46–53
72. Marcus A, Maletic J (2001) Identification of high-level concept clones in source code. In: International conference on automated software engineering, pp 107–114
73. Marcus A, Maletic J (2003) Recovering documentation-to-source-code traceability links using latent semantic indexing. In: International conference on software engineering, pp 125–135
74. Maskeri G, Sarkar S, Heafield K (2008) Mining business topics in source code using Latent Dirichlet allocation. In: ISEC, pp 113–120
75. Meneely A, Smith B, Williams L (2012) iTrust electronic health care system: a case study, chap. software and systems traceability. Springer, Berlin
76. Milne D, Witten I (2008) An effective, low-cost measure of semantic relatedness obtained from wikipedia links. In: AAAI workshop on wikipedia and artificial intelligence, pp 25–30
77. Nallapati R, Cohen W, Lafferty J (2007) Parallelized variational EM for Latent Dirichlet allocation: an experimental evaluation of speed and scalability. In: International conference on data mining workshops, pp 349–354
78. Niu N, Easterbrook S (2008) Extracting and modeling product line functional requirements. In: International requirements engineering conference, pp 155–164
79. Niu N, Mahmoud A (2012) Enhancing candidate link generation for requirements tracing: the cluster hypothesis revisited. In: IEEE international requirements engineering conference, pp 81–90
80. Nuseibeh B, Easterbrook S (2000) Requirements engineering: a roadmap. In: Conference on the future of software engineering, pp 35–46
81. Oliveto R, Gethers M, Poshyvanyk D, De Lucia A (2010) On the equivalence of information retrieval methods for automated traceability link recovery. In: International conference on program comprehension, pp 68–71
82. Panichella A, Dit B, Oliveto R, Di Penta M, Poshyvanyk D, De Lucia A (2013) How to effectively use topic models for software engineering tasks? An approach based on genetic algorithms. In: International conference on software engineering, pp 522–531
83. Patwardhan S, Banerjee S, Pedersen T (2005) Senserelate::targetword: a generalized framework for word sense disambiguation. In: Interactive poster and demonstration sessions, pp 73–76
84. Pedersen J, Silverstein C, Vogt C (2000) Verity at trec-6: out-of-the-box and beyond. Inf Process Manage 36(1):187–204
85. Porteous I, Newman D, Ihler A, Asuncion A, Smyth P, Welling M (2008) Fast collapsed gibbs sampling for latent Dirichlet allocation. In: ACM SIGKDD international conference on knowledge discovery and data mining, pp 569–577
86. Porter F (1997) An algorithm for suffix stripping. Morgan Kaufmann Publishers Inc., Los Altos, CA, pp 313–316
87. Poshyvanyk D, gal Guhneuc Y, Marcus A (2007) Feature location using probabilistic ranking of methods based on execution scenarios and information retrieval. IEEE Trans Softw Eng 33(6):420–432
88. Pucher M (2007) Wordnet-based semantic relatedness measures in automatic speech recognition for meetings. In: Annual meeting of the ACL on interactive poster and demonstration sessions, pp 129–132
89. Ramesh B, Jarke M (2001) Towards reference models for requirements traceability. IEEE Trans Softw Eng 27(1):58–93
90. Rao A, Lu A, Meier E, Ahmed S, Pliske D (2000) Query processing in trec-6. Inf Process Manage 36(1):179–186
91. Rosario B (2000) Latent semantic indexing: an overview. INFOSYS 240 Spring Paper, University of California, Berkeley
92. Salton G, Wong A, Yang C (1975) A vector space model for automatic indexing. Commun ACM 18(11):613–620
93. Sarukkai R (2002) Foundations of web technology. The Springer International Series in Engineering and Computer Science, New York, pp 106–108
94. Schmid H (1994) Probabilistic part-of-speech tagging using decision trees. In: International conference on new methods in language processing, pp 44–49
95. Settimi R, Cleland-Huang J, Ben Khadra O, Mody J, Lukasik W, DePalma C (2004) Supporting software evolution through dynamically retrieving traces to uml artifacts. In: International workshop on the principles of software evolution, pp 49–54
96. Shepherd D, Fry Z, Hill E, Pollock L, Vijay-Shanker K (2007) Using natural language program analysis to locate and understand action-oriented concerns. In: International conference on aspect-oriented software development, pp 212–224
97. Sorg P, Cimiano P (2012) Exploiting wikipedia for cross-lingual and multilingual information retrieval. Data Knowl Eng 74(0):26–45
98. Spanoudakis G, Zisman A (2004) Software traceability: a roadmap. Handb Softw Eng Knowl Eng 3:395–428
99. Strube M, Ponzetto S (2006) Wikirelate! computing semantic relatedness using wikipedia. In: National conference on artificial intelligence, pp 1419–1424
100. Sundaram S, Huffman-Hayes J, Dekhtyar A, Holbrook E (2010) Assessing traceability of software engineering artifacts. Requir Eng J 15(3):313–335
101. Teh Y, Jordan M, Beal M, Blei D (2006) Hierarchical Dirichlet processes. J Am Stat Assoc 101(476):1566–1581
102. Teufel S (2007) An overview of evaluation methods in trec ad hoc information retrieval and trec question answering. In: Dybkjaer L, Hemsen H, Minker W (eds) Evaluation of text and speech systems, pp 163–186
103. Thomas S, Adams B, Hassan A, Blostein D (2010) Validating the use of topic models for software evolution. In: IEEE working conference on source code analysis and manipulation, pp 55–64
104. Tsatsaronis G, Varlamis I, Vazirgiannis M (2010) Text relatedness based on a word thesaurus. Commun ACM 37(1):1–40
105. Wei X, Croft B (2006) LDA-based document models for ad-hoc retrieval. In: ACM SIGIR, pp 178–185
106. Wong S, Ziarko W, Raghavan V, Wong P (2012) On modeling of information retrieval concepts in vector spaces. In: ACM transactions database systems, pp 299–321
107. Zhai K, Boyd-Graber J, Asadi N, Alkhouja M (2012) Mr. LDA: a flexible large scale topic modeling package using variational inference in MapReduce. In: International conference on world wide web, pp 879–888
108. Zhang Y, Witte R, Rilling J, Haarslev V (2006) An ontology-based approach for traceability recovery. In: International workshop on metamodels, schemas, grammars, and ontologies for reverse engineering, pp 36–43