Empirical Principles and an Industrial Case Study in Retrieving Equivalent Requirements via Natural Language Processing Techniques

D. FALESSI, Member, IEEE, G. CANTONE, G. CANFORA, Member, IEEE Computer Society

Abstract— Though very important in software engineering, linking artifacts of the same type (clone detection) or different types (traceability recovery) is extremely tedious, error-prone, and effort-intensive. Past research focused on supporting analysts with techniques based on Natural Language Processing (NLP) to identify candidate links. Because many NLP techniques exist and their performance varies according to context, it is crucial to define and use reliable evaluation procedures. The aim of this paper is to propose a set of seven principles for evaluating the performance of NLP techniques in identifying equivalent requirements. In this paper we conjecture, and verify, that NLP techniques perform on a given dataset according to both ability and the odds of identifying equivalent requirements correctly. For instance, when the odds of identifying equivalent requirements are very high, then it is reasonable to expect that NLP techniques will result in good performance. Our key idea is to measure this random factor of the specific dataset(s) in use and then adjust the observed performance accordingly. To support the application of the principles we report their practical application to a case study that evaluates the performance of a large number of NLP techniques for identifying equivalent requirements in the context of an Italian company in the defense and aerospace domain. The current application context is the evaluation of NLP techniques to identify equivalent requirements. However, most of the proposed principles seem applicable to evaluate any estimation technique aimed at supporting a binary decision (e.g. equivalent/non-equivalent), with the estimate in the range [0,1] (e.g. the similarity provided by the NLP), when the dataset(s) is used as a benchmark (i.e. test bed), independently of the type of estimator (i.e. requirements text) and of the estimation method (e.g. NLP).

Index Terms—Empirical software engineering, traceability recovery, natural language processing, equivalent requirements, metrics and measurement.

1 Introduction

1.1 Context and Motivation

Linking artifacts of the same type (clone detection) or different types (traceability recovery) is a common practice in many software engineering tasks, including verification and validation, reuse, maintenance and reverse engineering. An important case is the identification of equivalent requirements to avoid assigning the same requirement to different developers, thus performing redundant tasks and originating duplications in the code base [1]. This usually happens when requirements are numerous and many stakeholders are involved in the analysis process [2]. An example of equivalent requirements is "The system shall provide a means for communicating SOS messages among and within systems of type B for allowing effective assistance." and "In order to enable assistance, the system must support the communication of SOS messages among and within B systems."

When SELEX Sistemi Integrati [3], an Italian company in the defense and aerospace domain, planned the development of five different products, taking advantage of the product commonalities was a first choice. Detecting equivalent requirements is crucial to reason on what and how to reuse [4-10]; however, humans alone cannot afford to analyze all the feasible combinations of requirement pairs for potential links (around three million in SELEX Sistemi Integrati projects).

Natural Language Processing (NLP) provides an automated approach to measure the similarity among requirement pairs; similarity is then adopted as a criterion to rank/filter the requirement pairs according to their likelihood of being equivalent.

Ideally, an NLP technique applied to a software engineering task should reveal all and only true traces. In practice, every NLP technique will reveal some false positive traces, while missing some real ones. There is a plethora of NLP techniques [11], with various techniques performing differently according to the context [12]. Therefore, it is important to define and use reliable evaluation procedures.

1.2 Aim and Research Questions

The aim of this paper is to propose a set of seven principles for evaluating the performance of NLP techniques, which aims to enhance result validity. To support the application of the principles we report their practical application to a case study that evaluates the performance of a large number of NLP techniques for retrieving equivalent requirements in the context of SELEX Sistemi Integrati. In particular, we adopted the

- *D. Falessi is with Certus Software V&V Center at Simula Research Laboratory (Norway) and University of Rome "TorVergata", DISP, viale del Politecnico 1, 00133 Rome, Italy. E-mail: d.falessi@ieee.org*
- *G. Cantone is at the University of Rome "TorVergata", DISP, viale del Politecnico 1, 00133 Rome, Italy.E-mail: cantone@uniroma2.it*
- *G. Canfora is at the University of Sannio, DING-RCOST, piazza Roma, 82100 Benevento, Italy. E-mail: canfora@unisannio.it*

Manuscript received: XX- December-2010.

proposed empirical principles to answer the following research questions:

RQ 1. *What is the ranking of the available NLP techniques? Which is the best NLP technique (if any)?*

RQ 2. How does the best NLP technique perform? Is there room for improvement? How far is its performance from the ideal performance?

RQ 3. Does performance significantly vary among NLP techniques?

RQ 4. How does an optimal combination of available NLP techniques perform?

RQ 5. Do the NLP techniques actually differ from one another?

The current application context is the evaluation of NLP techniques for equivalent requirements identification. However, most of the empirical principles seem applicable to evaluate any technique with a view to supporting a binary decision (e.g. equivalent/non-equivalent, correct/incorrect, etc.), that provides values in the range [0, 1] (e.g. the similarity provided by the NLP), and when a dataset(s) is used as benchmark (i.e. test bed).

1.3 Terminology

NLP vs. IR. Information Retrieval (IR) means "finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)." [13] NLP is a field of computer science and linguistics that is concerned with adopting machines to parse the natural language for a given purpose. In other words, IR refers to a problem at hand, while NLP refers to the solution in the domain of natural language. In our context, the natural language of two artifacts is analyzed to support a retrieval problem; hence, the terms IR and NLP may overlap.

NLP technique. Given a pair of text fragments (in our case study requirements), we refer to an NLP technique, or *technique* for short, as a way to analyze the two fragments and provide a similarity measurement as a value ranging between 0 and 1, with a maximum of two decimal places. This similarity measurement represents the likelihood of the two text fragments expressing equivalent concepts.

Performance of an NLP technique. An analyst using an NLP technique for evaluating similarity of artifacts can approach it with different focuses, such as completeness and correctness. Each quality focus can be measured by several metrics, e.g., precision and recall for correctness [14]. In this paper we use the term *performance* when we refer to the quality of an NLP technique in general terms, without assuming any finer-grain focus. Otherwise, we use the name of the adopted performance metric.

Variation points and variants. NLP techniques can differ in several aspects and dimensions. In order to effectively analyze such differences we adopted the terminology from Software Product Line Engineering [9, 15]. In particular, the term *variation point* refers to a given type of variability among NLP techniques whereas the term *variant* refers to an available option to realize a given variation point. An example of variation point is the formula to compute the fraction of common words; the variants are, for instance, the Jaccard and the Cosine metrics [16].

1.4 Structure

The remainder of the paper is organized as follows: Section 2 describes the background of the study presenting both related work and a classification of available NLP techniques. Section 3 discusses the principles for evaluating the performance of NLP techniques. Section 4 describes a case study that applies the proposed principles in an industrial context. Section 5 concludes the paper.

2 Background

2.1 Related Studies

The role of NLP in requirements engineering is depicted by Ryan in [17] who describes NLP as a promising approach to support the requirements engineering process due to the increasing complexity of the systems to develop and hence of the requirements to manage.

NLP has been applied in different areas of software engineering [12, 18, 19] for many years [20, 21]. Some studies have adopted NLP to transform requirements into analysis models [22, 23]. Others have analyzed requirements to identify ambiguity [24-27] and check their composability in the context of product families [28, 29].

Several studies have adopted NLP to trace artifacts at different levels of abstraction, including high-level to low-level requirements [14, 30-32], requirements to design [33-36], requirements to source code [37-39], high-level features for their implementation [40], functional requirements to Java components [41, 42], requirements to test case descriptions [43, 44], requirements to defect reports [45-47], and requirement changes to the impacted software modules [48]. Such studies differ from the present case study in that we try to identify equivalence among the same type of artifacts; i.e., duplicates.

Hayes et al. [49] proposed a framework for characterizing experiments in requirements tracing and apply it to four related work. We based the present paper upon their suggestion to investigate the use, meaning, and usefulness of metrics, and to base the study upon robust datasets (see Sections 3.5 and 3.6).

Runeson et al. [46] achieved promising results in adopting NLP techniques to detect equivalent defect reports.

Hayes et al. [50], based on personal experience, proposed specific thresholds to define the performance of an NLP technique as acceptable, good, or excellent. They judged such a proposal as "drawing a line in the sand" and explicitly showed a "desire to find empirically where the lines are drawn". Building on their first attempt, we propose the activity of adjusting scores; this provides an objective way to evaluate the performance of NLP techniques. It is objective because it is independent of any human experience or judgment.

The case study proposed in this paper was inspired by Natt och Dag et al. [1], who compared three NLP

techniques to assess the feasibility of a fully-automated linking approach. In particular, we share with [1] the following commonalities:

1. Objects to retrieve: the compared text fragments are industrial requirements expressed in natural languages; moreover, such requirements are supposed to be on the same abstraction level.
2. Semantics of links: the links between requirements are equivalences.
3. Vision: "some approaches seem promising but we believe that more effort needs to be put into this field to reach consensus on which methods, techniques, approaches and tools may be appropriate for different types of developing organizations." [1]
4. Type of results: "a benchmark is provided to which further effort may be compared." [1]
5. We try to address their recommendation for future work: "it is of great interest to compare different approaches and combinations of approaches. The implementation cost and computational effort needed for statistical methods, linguistic methods and other computational models (such as LSA) are of great interest…"[1].

The last three works we mentioned above [1, 46, 50] are of significant high value to the software engineering community given their high rigor and contribution to the state of the art. However, their empirical procedure should be enhanced by the empirical principles we propose in the next section. We note that, after this enhancement, some of their results would most likely be the same. To provide a short example, the NLP technique that proved the best in [1, 46, 50] would remain the best also when applying the proposed empirical principles. However, several validity aspects of such results would significantly improve. For instance, the results presented in [1, 46, 50] are neither reported nor adjusted based on the difficulty of the adopted dataset(s). As detailed in Section 3.5, the performance of the best NLP technique(s) observed in [1, 46, 50], though best, would clearly differ, in an unknown manner, in every dataset with a different difficulty level. Therefore, according to the results in [1, 46, 50] the reader can clearly pick the best NLP technique but its performance in the reader's context is very questionable. This specific validity threat can be mitigated via the proposed empirical principle called "Analyzing the impact of difficulty" (see section 3.5). Section 3 will provide a list of empirical principles and the threats to validity they inhibit.

2.2 A Classification of NLP Techniques

NLP techniques shifted from being a research matter to a practical instrument due to their improved quality, open-source availability, and the computational resource nowadays available from any desktop computer.

Canfora and Cerulo [11] proposed a taxonomy of information retrieval models and tools with the aim of clarifying the terminology, often confusing and misleading as different terms are frequently used to denote the same, or similar, tasks. In this section we introduce a classification that reveals the variability and commonalities of available NLP techniques. Our classification comprises four dimensions; each dimension defines a variation point. The four variation points comprising our classification are: *algebraic models, term extraction, weighting schema,* and *similarity metric*. According to this classification, a specific NLP technique is viewed as a point in four-dimensional space; in other words, a specific combination of four variants (one for each variation point) identifies a given NLP technique. Figure 1 describes the proposed classification by using a feature model [51]. Not all variants, for different variation points, are compatible with each other; we report incompatibilities with dashed edges in Figure 1. Specifically, WordNet [52], which is a thesaurus-based variant of the algebraic model variation point, has its own set of similarity metrics [53]. The total number of (compatible) NLP techniques considered in our case study is 242.

As with any attempt to characterize a complex issue, the proposed classification has some limitations in describing the available NLP techniques. One of the main limitations of the classification is that it does not take account of the "feedback" variant as proposed elsewhere [37, 50, 54]. This is because we want to assess the different NLP techniques, and feedback is something that can be added on top of any technique. Further NLP techniques not considered in the present study include collocation analysis, word sense disambiguation [55], Latent Dirichlet Allocation [56], Relational Topic Models [57], and probabilistic IR techniques [42].

Despite such minor weaknesses, the proposed classification, compared with previous studies, explains the variability among the largest number of NLP techniques. Moreover, in the present work the NLP techniques are both reported and evaluated (Section 4).

The following subsections describe the main characteristics of the analyzed NLP techniques. We examine variation points in Figure 1 by depth, i.e., by reviewing all the variants for each variation point. Accordingly, the names of variation points are used as the title of sections, while the names of the related variants are printed in italics.

2.2.1 Algebraic Model

Algebraic models evaluate the semantic similarity among words. The vector space model (VSM) uses spatial proximity for semantic proximity; such conceptual simplicity makes it one of the most widely adopted algebraic models in practice [16]. The VSM proscribes the use of synonyms when comparing two words. If we wish to consider synonymies, the issue is how to compute the level of similarity. This can be pre-defined in a thesaurus (as in the case of WordNet), or it can be computed on a corpus of elements (as in the case of Latent Semantic Analysis - LSA).

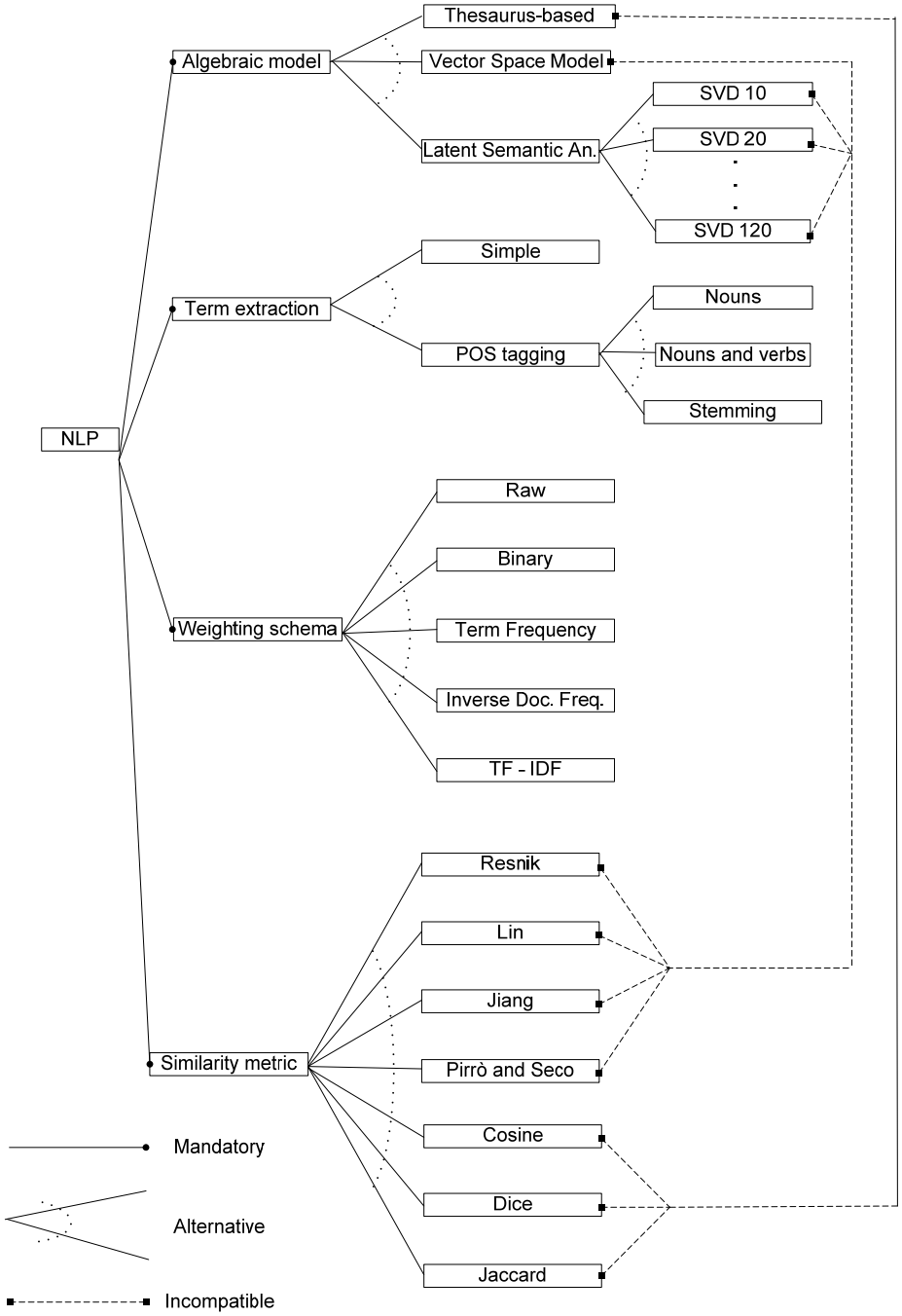

Figure 1 A classification of NLP techniques, evaluated in Section 4, consisting of four main components: algebraic models, term extraction, weighting schema, and similarity metric.

2.2.1.1 *Vector Space Model (VSM)*

The VSM represents text documents in terms of vectors in a multi-dimensional space; each dimension of the space corresponds to a term. A piece of text D_i is represented by a vector $D_i = (t_{1i}, t_{2i}, \ldots, t_{ni})$, where t_{ji} represents the weight of the term t_j of D_i. There are different ways to compute such a weight, described as realization mechanisms for weighting schema in Section 2.2.3. Once the texts are represented in vectors, their similarity is computed in terms of distances in such a vector space. Different ways to compute such distances are discussed in Section 2.2.4.1.

2.2.1.2 *Thesaurus-based*

In this approach a thesaurus with a given set of synonyms is adopted to compare terms. WordNet [52] is a well-known example for the English language. Note that WordNet is not just a thesaurus and hence it may be used, and categorized, not only as an algebraic model but also as a knowledge base of individual word semantics.

2.2.1.3 *Latent Semantic Analysis (LSA)*

LSA was developed during the 1980s with the name *LSI* (Latent Semantic Indexing), evolving from the traditional VSM [16]. VSM does not take into consideration synonymies among words. For instance, although "boat" and "ship" have similar meanings, VSM is unable to take this kind of similarity into consideration. LSA is able to compute/reveal synonymies among words according to their co-occurrence in a given corpus of text fragments, i.e., the fact that two or more words occur in the same documents more often than chance. We stress that LSA, given a corpus of text fragments as input, provides as output a thesaurus where similarity among words is expressed in a ratio scale. Therefore, such a thesaurus is domain-dependent, and hence LSA is able to capture similarities according to the language adopted in a given application context[1]. LSA requires significant computational resources and extensive corpora to produce reliable results. LSA decomposes a document-by-term matrix into three matrices: a document-by-concept matrix, a term-by-concept matrix, and a diagonal matrix. It then uses the document-by-concept matrix (this is much smaller than the document-by-term matrix as there are far fewer concepts than terms, thus being more efficient). Finally, it applies *Singular Value Decomposition (SVD)* [6] to project the original term-by-document matrix into a reduced space in order to reduce the "noise" in term usage. Defining the right size of the space resulting from the SVD is a hard decision because the level of detail included in the space should be not so great as to include noise nor too small to exclude valuable information. In our approach, we let the value range between 0 and 120 with an interval of 10. We chose such an interval as a tradeoff between precision (which is high with a high interval) and analyzability (which is high with a low interval). Indeed, the actual interval of 10 entails the analysis of 242 NLP techniques, whereas with an interval of 1 the number of NLP techniques to analyze would have risen to 1880.

2.2.2 Term Extraction

This variation point defines how the text fragments analyzed for similarity are pre-processed before any comparison can take place. The different variants are incremental steps rather than alternative options.

2.2.2.1 *Simple*

The simplest way to preprocess the text is by tokenization and stop-word removal. Tokenization transforms the text into a series of tokens where capitals, punctuation, and brackets are removed. Afterwards, the terms that do not contribute to the semantics of the text are removed; examples include articles, pronouns, etc. The list of stop words needs to be defined before text preprocessing takes place.

2.2.2.2 Part Of Speech (POS) Tagging

After tokenization and stop word removal, each token is classified according to the part of speech, i.e., the role it plays in a given text. Each token is associated with a tag indicating whether the token is a verb, noun, adjective, etc. Afterwards the different tags can be used in different ways:

1) Term weighting: the different tags are weighted according to the expected amount of semantic contribution in the given application domain. For instance, nouns can be deemed as more important than adjectives, and hence texts with nouns in common will have higher similarity values than those with adjectives in common. In general, it is not known whether nouns are more important than adjectives or whether such a difference makes any sense for the problem at hand. It is therefore important to analyze whether it pays off to enact such POS tagging and, if it does, what weight to assign to the different tags. Based on past works on software engineering, we consider here two types of tagging; one where only nouns are considered (like in [58]) and another where only nouns and verbs are considered.

2) Stemming: the tokens are converted into the corresponding morphological stems. For example, "monitoring", "monitor", "monitored", and "monitors" are all converted into the token "monitor". The advantage of stemming the words is that it mitigates the influence of morphological variants on similarity measures. Moreover, stemming is based on rules or statistics and its application to neo-Latin languages is quite complex. Available empirical evidence does not support the hypothesis that stemming improves the quality of the estimated similarity [59, 60]. However, because results generalizability is questionable, it makes sense to take the approach into account. In the present paper as a stemming mechanism we adopted the rule-based algorithm proposed by Martin Porter [61].

2.2.3 Term Weighting

This variation point comprises ways to assign different weights to terms based on their occurrences in the analyzed text fragments (see Table 1).

2.2.3.1 *Raw frequency*

The weight of a term is computed as the number of times that the term occurs in the text.

2.2.3.2 *Binary*

A weight of 0 is assigned if the term does not occur, and 1 vice versa.

2.2.3.3 *Term Frequency (TF)*

A weight is assigned which is proportional to the frequency of the term occurring in the given text fragments. In other words, it is computed by dividing the raw value by the length of text fragments in which it is

[1] For instance, although the words "monitor" and "screen" can be used as synonyms, they proved un-similar by LSA, using a given corpus of systems specifications of Selex SI projects. This suggests that, in the application domain of Selex SI, the term "monitor" is mostly used in the sense of "supervise" rather than "screen".

included. This approach aims to avoid long text fragments providing terms with higher frequency but low semantic relevance. Table 1, first row, reports the formula to compute the TF, where:

- *n(x,y)::* the number of occurrences of the considered term *Tx* in the document *Dy*
- $\sum_k n(k,y)$:: the sum of occurrences of all terms in document *Dy, i.e.,* the size of the document | *Dy*|

2.2.3.4 *Inverse Document Frequency (IDF)*

The IDF assigns a weight depending on the number of given texts that include the term (related to the total number of texts). Table 1, second row, reports the formula to compute the IDF where:

- |*D*|:: total number of documents in the corpus
- |*d : t(x)* ∈ d|:: number of documents where the term *t(x)* appears.

Although IDF is strongly correlated with the inverse of TF, the two variables are not completely predictable from one another [62]. The underlying idea for IDF weighting is the observation that the documents related to a given domain share many words; therefore, such frequent words do not provide a lot of semantic value; i.e., they are unable to discriminate among the different documents. For instance, according to IDF, in the automotive domain the term "car" should have a low weight because it adds little information in documents in which it is used [63, 64].

2.2.3.5 TF_IDF

The weight is computed as the multiplication of TF and IDF so as to achieve the benefits of both TF and IDF. Hence, TF_IDF assigns to a term a weight that is: i) high, when the term occurs many times within a small number of documents (thus lending a high discriminating power to those documents); ii) low when the term occurs few times in a document, or occurs in many documents (thus offering a less pronounced relevance signal); iii) lowest when the term occurs in virtually all documents [13].

2.2.4 Similarity Metric

This variation point comprises the ways to compute the similarity level of two text fragments given the similarity levels of the terms. To avoid confusion, we note that the term *performance metric* refers to a specific performance aspect (e.g., recall) whereas *similarity metric* refers to a specific formula to compute the fraction of common words between two text fragments.

2.2.4.1 *Vector Similarity Metrics*

There are different ways to compute the distance of two vectors, known as vector similarity metrics. In this study we consider the Dice, Jaccard and Cosine metrics as described in Table 2 for vectors (or sets) *x* and *y* of terms [16]. According to Salton, "the choice of a particular vector-estimated similarity for a certain application is not prescribed by any theoretical considerations and is left to the user" [59].

2.2.4.1.1 *Dice*

TABLE1
Term Weighting Formulas

Measure	Formula
TF(x,y)	$\frac{n(x,y)}{\sum_k n(k,y)}$
IDF(x)	$\log \frac{\lvert D \rvert}{\lvert d : t(x) \in d \rvert}$
TF_IDF(x,y)	$TF(x,y) \times IDF(x)$

TABLE2
Vector Similarity Metrics

Measure	Formula
$S_{Dice}(x,y)$	$\frac{2 \lvert x \cap y \rvert}{\lvert x \rvert + \lvert y \rvert}$
$S_{Jaccard}(x,y)$	$\frac{\lvert x \cap y \rvert}{\lvert x \rvert + \lvert y \rvert - \lvert x \cap y \rvert}$
$S_{Cosine}(x,y)$	$\frac{\lvert x \cap y \rvert}{\sqrt{\lvert x \rvert \lvert y \rvert}}$

The Dice coefficient normalizes for length by dividing by the total number of non-zero entries. The scale factor 2 (Table 2, first row) gives a measurement that ranges from 0.00 to 1.00 with 1.00 indicating identical vectors. In such a way, as compared to Euclidean distance, the Dice distance ($1 - S_{Dice}$) retains sensitivity in more heterogeneous data sets and gives less weight to outliers [65].

2.2.4.1.2 *Jaccard*

The Jaccard coefficient penalizes a small number of shared entries (as a proportion of all non-zero entries) more than the Dice coefficient. Both measures range from 0.00 (no overlap) to 1.00 (perfect overlap), but the Jaccard coefficient gives lower values to low-overlap cases (Table 2, second row).

2.2.4.1.3 *Cosine*

The cosine is identical to the Dice coefficient for vectors with the same number of non-zero entries but it penalizes less where the numbers of non-zero entries are very different from one another. This property of the cosine is important in statistical NLP since we often compare words or objects that we have different amounts of data for, but we do not wish to say they are dissimilar just because of that. Note that the adopted definition of the cosine metric (Table 2, third row) is the most intuitive and assumes that the vectors consist only of 0 and 1. Further definitions of the cosine metric can be found in [66, 67].

2.2.4.2 WordNet Similarity Metrics

WordNet has its own set of metrics [53]. In the present paper we consider the following WordNet similarity metrics: Resnik [68], Lin [69], Jiang [70], and Pirrò and Seco [71] as described in Table 3 where:

- *IC* is the information content
- *LCS* is the least common subsume
- *W(x,y)* is the set of words included in x and y.

Such metrics, however, cannot be directly exploited in our work because the WordNet system provides only a unidirectional measure[2]. In order to measure the similarity of text fragments, the unidirectional estimated similarity is transformed into a bidirectional similarity measure as suggested by Corley and Mihalcea in [72] and described in Table 4 where:

- *SWORDNET(x,y)* is the estimated similarity between the text fragments *x* and *y*.
- *Suni(x,y)* is the unidirectional similarity measure,
  - *n* is the number of words of *x*,
  - *m* is the number of words of *y*,
  - *Wi* is the word number *i*,
  - *Sz(x,y)* is a given WordNet similarity metric.

2.3 Performance Metrics

In the context of software engineering, specifically traceability recovery, the common procedure to analyze the performance of NLP techniques consists of a post mortem analysis on given dataset(s), where the underlying principle is that a pair of artifacts should be traced upon each other if the similarity is higher than a threshold [1, 38, 39, 46, 50, 54, 58, 73, 74]. The dataset(s) contains a large set of pairs of software artifacts; for each pair, the dataset contains the text of each artifact (upon which the similarity is computed) and the "actual traces". The latter are usually defined by human experts and used as oracle. Each NLP technique is applied to measure the similarity of each pair. Then the "candidate traces" are computed, for each NLP technique, by applying a threshold on the measured similarity of each pair. Finally, for each NLP technique the performance is computed by comparing the actual and candidate traces. The less the difference between the actual and candidate traces, the higher the performance of the NLP technique.

The term *threshold* means a limit over which a trace is deemed candidate or not candidate if vice versa. This limit is usually expressed as a value of the similarity metric. A further yet effective way to express this limit is the specific rank of the trace [50]; i.e. only a pre-specified number of top ranked traces are deemed candidate. In this paper we refer to threshold with the former meaning; however, most of the reasons apply to the latter case as well.

[2] For instance, the similarity between the terms "manuscript" and "document" changes according to its direction: the term "manuscript" can be replaced by "document" but not vice versa because a "document" may also be electronic.

TABLE3
WordNet Similarity Metrics

Measure	Formula
$S_{Resink}(x,y)$	$Max\ IC(c)\ where\ c \in W(x,y)$
$S_{Lin}(x,y)$	$\frac{2\ IC\ (LCS\ )}{IC\ (x) + IC\ (y)}$
$S_{Jiang}(x,y)$	$1 - \frac{IC(x) + IC(y) - 2IC(LCS)}{2}$
$S_{Pirrò\&Seco}(x,y)$	$3\ IC\ (LCS\ ) - IC\ (x) - IC\ (y)$

TABLE4
WordNet Similarity Measurement

$S_{WORDNET}(x,y)$	$\frac{Suni(x,y) + Suni(y,x)}{2}$
$S_{uni}(x,y)$	$\frac{\sum_n \sum_m Max(Sz(Wn,Wm)) \times idf(Wn)}{\sum_n idf(Wn)}$

TABLE 5
Performance Metrics for Different Types of Support

Type of support	Metric	Range	Formula
Filtering	Precision	[0,1]	*(true positive) / (false positive + true positive)*
	Recall	[0,1]	*(true positive) / (false negative + true positive)*
	ROC area	[0,1]	*True postive rate vs. False positive rate*
Ranking	Lag-global	[0, ∞]	$\frac{(\text{Rank of the last positive} - \text{Number of positive})}{\text{Number of positive}}$
Suggesting	Credibility	[0,1]	*Average of (1- |similarity measure – oracle|)*

Because an analyst can use a technique in different ways, each with some different focuses, several metrics have to be adopted to measure performance related to the above research questions. In the following we describe three main types of support, the rationale, and the related metrics as adopted in our case study (Section 4); Table 5 provides the metrics definitions. Because these metrics will be used in the case study presented in Section 4, we also report here their rationale.

- *Filtering*: As stated above, the number of possible requirement combinations is too large for a human to check all of them. Hence, it is important to investigate the possibility of filtering them according to a given threshold. In order to characterize the ability of an NLP technique to produce/retrieve an effective list of candidate traces, *ROC area, precision* and *recall* metrics can be adopted. We chose these metrics because they are the best current practice to look at different aspects of performance [14]. The ROC area is rarely used in software engineering and it derives from the acronym Receiver Operating Characteristic, a term used in signal detection to characterize the tradeoff between hit rate and false alarm rate over a noisy channel [75]. In practice, the ROC area represents the area under the curve of true positive rate and false positive rate as

the prediction threshold sweeps through all the possible values. Note that the values of precision and recall are not independent of each other, but relate to the same (optimal) threshold. Given a specific threshold, the F-measure [50] can be easily computed and therefore in this context it is neglected to avoid redundancy.

- *Ranking:* Albeit filtered, the number of available requirement pairs is extremely high. Therefore, it is important that NLP techniques support analysts in focusing their attention on the requirement pairs that are most likely to be equivalent. Hayes et al. [50] proposed the *Lag* metric as a measure of ranking effectiveness. The *Lag* represents, in the average among actual equivalent requirement pairs, the number of actual non-equivalent requirement pairs with a similarity (i.e., a rank) higher than actual equivalent requirement pairs, on average among requirement pairs. In other words, it represents - based on the rank provided by an NLP technique - the average number of non-equivalent requirement pairs that need to be analyzed before analyzing each equivalent pair. In this work we use Lag-global (Table 5) which slightly differs from that original Lag presented in [50]. Specifically, the Lag-global refers to a single global ranking of all candidate traces, whereas the original Lag refers to the average of the rankings, of candidate traces, of specific requirements. In this paper we adopt the Lag-global because in PROUD [2] all candidate traces are ranked together whereas in RETRO [50] each high-level requirement to trace has its own ranking.
- *Suggesting:* When the human analyst has to decide about the equivalence or otherwise of a given requirement pair, the similarity measure provides practical support in suggesting such a classification (equivalent vs. non-equivalent). The underlying principle is that when the (measured) similarity is low, humans tend to classify such a pair as non-equivalent, and vice versa. In particular, results from a previous study [2] support the hypothesis that the *Credibility* of the similarity measure shown significantly impacts on the quality and speed of human analysts in linking requirements. The credibility metric measures the correlation between the similarity as perceived by a human and as measured by a given NLP technique; the more credible the technique, the more it reduces the time to classify a given requirement pair and the more it improves the quality of the classification. In order to analyze the performance of the different NLP techniques, credibility can be computed for each NLP technique on each requirement pair of the industrial dataset. For a single requirement pair, credibility is formally defined as 1-|similarity measure – human classification| where human classification is the subjective judgment of an expert regarding its equivalence: 1 when the requirement pair is classified as equivalent and 0 otherwise. The more correlated the similarity measurements and expert judgment, the higher the credibility, within a [0, 1] range. The credibility of a given NLP technique is then computed as the average among the requirement pairs in the industrial dataset..

3 Empirical Principles for Evaluating NLP Techniques

Some good examples of studies evaluating the performance of NLP techniques include [1, 38, 39, 46, 50, 54, 58, 73, 74]. This set of references does not aim to be exhaustive. Indeed, our aim in this paper is not to provide any systematic review of the literature.

Building on past studies, and by leveraging on recent advances of empirical principles for evaluating NLP techniques, this method aims to inhibit threats to validity. Although empiricists usually need to trade off the different aspects of validity (e.g., control over realism) [76] the proposed method aims to enhance several aspects of validity without negatively influencing any other. In the remainder of this section we refer to the aspects of validity as reported by Wohlin et al. [77]: conclusion validity, external validity, internal validity, and construct validity.

Each sub-section below reports a specific principle. Because a principle can be implemented differently according to the application context, we report below the general motivation, the proposed solution, and examples of implementation strategies. Then we conclude by discussing the strengths and limitations.

3.1 Principle 1: Definition of an Objective Function and Adoption of an Optimal Threshold

3.1.1 Motivation

In order to analyze the performance of NLP techniques, a dataset is adopted, similarity measurements are computed, and thresholds are applied to define which links are candidates of interest.

An NLP technique performs differently on different thresholds: a high threshold is prone to false negatives, whereas a low threshold is prone to false positives. These two types of errors differently affect diverse performance metrics; for instance, recall is more sensitive to false negatives than precision (see Table 5). As a result, for each NLP technique, a given threshold implies a given tradeoff among performance metrics. Moreover, for a given performance metric, different NLP techniques may perform differently on different thresholds; this can happen for several reasons. For instance, an NLP technique may be overoptimistic and hence prone to false positives even with high thresholds. Therefore the results of comparisons of NLP techniques greatly depend on the adopted thresholds.

The common evaluation procedure consists of applying a set of thresholds and observing the resulting performance. Thresholds can be chosen by common sense and without any rigorous criteria. In this paper we claim that such an approach does not effectively support the investigation of the question "Which is the best NLP technique in supporting the given software engineering

task?". In particular, the use of a casual set of thresholds entails the following problems:

1. The optimal threshold for a given NLP technique is not considered and hence its performance is underestimated. For instance, in [1], the Dice and Cosine variants outperform Jaccard regarding "True positive rate", and vice versa regarding "False positive rate". Because this result is based on a limited, though large, set of thresholds (9), for a threshold not taken into account the Jaccard variant could outperform both Dice and Cosine, or vice versa.
2. Consciously or unconsciously, the researcher can select the set of thresholds for driving the results. S/he may for example enlarge the interval among thresholds to prevent an intermediate threshold being taken into account and the preferred NLP technique being then outperformed by another.
3. The comparison is based on an unclear objective and hence the achieved result is unreliable. If the objective upon which the NLP techniques are compared is unclear, then it is more difficult to understand in which contexts the results apply. For instance, in a given context, it may be preferred to have false negatives rather than false positives, or vice versa.

In conclusion, adoption of a casual set of thresholds significantly threatens conclusion validity; this would eventually negatively influence both practitioners in adopting the wrong NLP technique and researchers in underestimating the real performance of a given NLP technique.

3.1.2 Proposed solution

We maintain that it is important to compare the different NLP techniques based on their performance according to their own optimal threshold. The number of thresholds adopted in past studies [1, 38, 39, 46, 50, 54, 58, 73, 74] is very limited; generally, no more than 10. An exception to this trend is [78] where a very extensive set of thresholds is adopted. Unfortunately, no study reasoned on an optimal threshold. The optimal threshold may change among NLP techniques and should be computed according to a pre-specified objective function that, in turn, depends upon the specific software engineering task at hand. For instance, a recall of 100% is a suitable objective for safety critical systems when traceability is used for the activity of validation and verification. However, such an objective is not suitable if traceability is used for the reuse task; in this case, the effort to be spent tracing artifacts needs to be balanced by the effort reduction expected from the reuse strategy.

One option to define the objective function is to weight false positive and false negative errors; another option is to fix the desired amount of recall (e.g., 100% as in [58]). A finer-grain way to express the objective function is presented by Zou et al. [79]. Yet another option is the use of the F-measure, as proposed by van Rijsbergen [80] where F-*b* measures the effectiveness of retrieval with respect to a user who attaches *b* times as much importance to recall as precision. The main limitation of F-*b* measures is to take into account only precision and recall; further metrics including Lag-global, Credibility, and ROC area seem not covered.

3.1.3 Implementation strategies

Simple logistic regression is a procedure that computes the linear equation that best predicts the dependent variable according to the independent variable [81]. Simple logistic regression is applied when there is one ratio variable (i.e., the similarity measurement provided by an NLP technique on a requirement pair) and one nominal variable with two values (i.e., equivalent/non-equivalent). We may view simple logistic regression as a procedure to find the optimal threshold, among all the possible thresholds, for a given NLP technique, for a given objective function.

Weka [82] is an open source tool containing a collection of machine learning algorithms for data mining tasks. The algorithms can either be applied directly to a dataset or called from Java applications. Weka contains tools for data pre-processing, classification, regression, clustering, association rules, and visualization. Weka is an example of an open source tool to compute simple logistic regression. It takes as input the dataset as oracle and the similarity measures provided by the NLP techniques. The user defines the objective function by weighting false positive and false negative types of error. Then Weka applies all the possible thresholds and reports the results (i.e. performance metrics) achieved by the threshold maximizing the objective function. For further details see [75, 82].

3.1.4 Discussion

Independently of the approach for defining the objective function, it is better to avoid the use of a casual set of thresholds for the reasons described above. The rationale for choosing a specific objective function should be clearly explained and motivated by the researchers. This is expected to help readers realize the extent to which results are applicable to their own context.

We note that this principle does not apply to evaluation procedures that use as a threshold a predefined number of ranked traces. In this case, the objective is already explicit as it coincides with the exact number of top-ranked traces.

It is of course reasonable to compare the NLP techniques on several objectives; again, it is important to explain why the NLP techniques should be compared according to such objectives and not to others.

3.2 Principle 2: Cross-validation of the Performance of NLP Techniques

3.2.1 Motivation

Publishing and using datasets is largely advised [83]. The PROMISE repository is a reliable existing resource; in particular, the CM-1 dataset consists of a complete requirements (high-level) document and a complete design (low-level) document for a NASA scientific instrument [50]. However, due to non-disclosure agreements between researchers and industry, few benchmarks are currently publicly available. The shortage

of datasets forces researchers to adopt a small number of datasets (usually one, three at the most). Therefore, the given characteristics of the adopted dataset(s) threaten the external generalization of the results.

If the optimal threshold is computed on a given dataset and hence the performance of the NLP technique is computed on the same dataset, the results tend to be optimistic [84].

3.2.2 Proposed solution

The adoption of cross-validation is a standard approach for assessing prediction models in order to obtain realistic and general results. There are several methods for cross-validation and which is the best is still an open question [75]. However, there is general agreement that, given a finite number of available datasets, cross-validation is a reliable procedure to maximize external validity of findings.

The underlying principle of cross-validation is to modify the dataset on which the optimal threshold is computed by randomly re-sampling the whole dataset. A standard cross-validation approach is called 10-fold [75]. In this approach, the dataset is divided randomly into 10 subparts, with the constraints to have approximately the same cardinality and the same proportion of categories of the dataset. Each subpart, in turn, is taken away from the dataset, and the remaining data are used to train the model which is then applied on the subtracted part. In the NLP context, training the model means selecting the threshold that optimizes the fulfillment of the objective function. Finally, performance is computed by averaging the performances resulting from the ten-step procedure.

3.2.3 Implementation strategies

To implement cross-validation, we recommend the use of WEKA as it provides an easy and user-friendly environment to enact cross validation which allows the user to select the desired amount of folds (e.g. 10). WEKA takes as input the dataset as oracle and the similarity measures provided by the NLP techniques. Once the type of validation has been defined, WEKA provides as output the cross-validated scores of the performance metrics. For further details see [75, 82].

3.2.4 Discussion

Examples of studies adopting cross-validation for evaluating the performance of NLP techniques include [85-87]. Note that cross-validation is useful only when training is involved; in our case study (Section 4) cross-validation is required because the metrics are computed over an optimal (trained) threshold. When using pre-defined thresholds, the cross-validation is useful only if the NLP techniques under evaluation incorporate some training mechanisms (e.g. feedback).

3.3 Principle 3: Statistically Testing the Difference among Performances of NLP Techniques

3.3.1 Motivation

The empirical software engineering literature clearly agrees on the importance of supporting a given claim by means of statistical tests [77]. The use of statistical tests, though common in the software maintenance community, is not yet widespread overall; this absence significantly threatens conclusion validity.

The underlying principle of statistically testing the significance of a difference is to check whether the observed difference could reasonably occur ''just by chance'' due to the random sample used for developing the model; if this is not the case, one can infer that there is evidence for a significant difference [88]. According to Mittas and Angelis [89] "this policy can lead to unstable and erroneous conclusions since a small change in the data is able to turn over the best model selection." In other words, a possible implication of not statistically testing the superiority of a given technique (over another) is that when a small amount of data is changed, the ''best'' model may no longer be the ''best''. Hence, there is a need for formal comparison of NLP techniques on the basis of inferential statistical techniques, such as hypothesis testing and confidence intervals [89].

3.3.2 Proposed solution

Statistically testing performance in terms of precision and recall can be easily done by means of the Z-test for two proportions or Fisher's exact test (in the case of few observations) [90]. Other metrics like the ROC area [14] require a more complex procedure [91].

3.3.3 Implementation strategies

Commercial tools to run statistical tests include JMP ® and SPSS ®. See [92] for a detailed review.

3.3.4 Discussion

Statistically testing the superiority of a given NLP technique supports practitioners in avoiding overestimation of performance differences and helps researchers to enact meta-analytic studies [93]. Statistical performance testing should be performed according to the NLP techniques' optimal threshold(s) (see Section 3.1). Therefore, the result of the statistical test depends on the given objective function upon which the optimal threshold is computed. Further details on the current needs of statistical tests on comparative studies can be found in [89, 94].

Finally, power analysis [95-97] can be helpful both before and after evaluation of the NLP techniques. Before evaluation, power analysis helps estimate the size of the dataset(s) to achieve appropriate statistical power. After evaluation, power analysis helps check whether a resulting low statistical result is due to a small dataset.

3.4 Principle 4: Statistically Testing the Equivalence among NLP Techniques

3.4.1 Motivation

The procedure to statistically test that two NLP techniques are equivalent differs from the procedure to test that they are different. The difference between the two procedures can be underestimated when interpreting the results of statistical tests. In particular, the fact that two NLP techniques do not statistically differ does not imply that they are statistically equivalent. There is the

case where two distributions (i.e., the performances) are not diverse enough to be judged statistically different and, at the same time, they are not equal enough to be judged statistically equivalent; one of the possible causes is the small size of the dataset.

3.4.2 Proposed solution

Principal Component Analysis (PCA) provides a way to assess whether or not a given NLP technique is redundant (i.e., correlated) to a given set of NLP techniques; e.g. one NLP can be equivalent to a combination of other NLP techniques. PCA [98] transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components (PCs). In particular, the first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible. Associated with each PC is its eigenvalue, which is a measure of the variance of the PC.

As a stopping rule, a PC is deemed important (i.e., uncorrelated with other PCs) only if its eigenvalue is greater than 1.0 [99].

3.4.3 Implementation strategies

PCA is available in any advanced statistical tool including JMP ® and SPSS ®.

3.4.4 Discussion

In our context, PCA is a way to assess whether or not a given NLP technique is redundant (i.e., correlated) to a given set of NLP techniques; e.g. one NLP may be equivalent to a combination of other NLP techniques. In practice, before claiming that a new NLP technique outperforms past ones, it is recommended that researchers analyze whether or not such a new NLP technique is equivalent to any combination of already proposed ones (i.e., if it has an eigenvalue higher than 1.0). This research procedure has provided important advances in the software metrics field [100, 101].

3.5 Principle 5: Analyzing the impact of difficulty

3.5.1 Motivation

Performance evaluation of NLP techniques is based on the use of datasets. Since the datasets in question are scant in number, it is questionable whether results from a given context are applicable to others. In fact, descriptive and statistical analyses are the appropriate way to analyze performance differences of NLP techniques. However, while comparing the output, these analyses do not take the input (i.e. the dataset(s)) into account.

Here we conjecture and test that NLP techniques, identifying equivalent requirements, perform on a given dataset according to both ability and the odds of making a correct identification. Adopting the metaphor of a golfer, it is easier to score if the hole is larger, though her/his ability would also influence the resulting overall performance. In the context of NLP, if we consider a dataset where 1% of data belong to the category to retrieve, then achieving high precision would be very *difficult* for any NLP technique. In this case, a technique resulting in low precision may be erroneously deemed as not supportive for the task. However, in practice, the technique may provide significant support to the human, saving a significant amount of time in analyzing that dataset [2]. Another possible wrong conclusion is that as this technique shows low precision, it is deemed worse than another technique which shows on another dataset a higher precision (e.g. in a dataset where 99% of data belong to the category to retrieve). However, when applied on the same dataset, the former technique may provide a support that is orders of magnitude higher than the latter technique.

3.5.2 Proposed solution

For any given performance metric, we define *difficulty* as a characteristic of the dataset representing the extent to which the retrieval task is arduous. In a dataset where 99% of data belong to the category to retrieve, the difficulty is low because even a random approach would retrieve a high proportion of data of the category concerned.

Difficulty changes according to the performance metric adopted; for instance, in a given dataset it can be very easy to have high precision but very difficult to have high recall. For the specific performance metric, we propose to measure difficulty as the performance of a random approach to retrieve links in the given dataset; the higher the performance, the lower the difficulty. Equation 1 reports the formula to compute difficulty of a given dataset.

- *Best Score*: the ideal best value for that given metric and dataset. For instance, the best score of Lag-global is 0 whereas for Precision it is 1.0.
- *Random Score* is the value of a given metric (e.g., precision) achieved by a non-intelligent (i.e., random) estimator of similarity, on a given dataset.

The rationale of equation 1 is to measure difficulty as the distance between the best score and the random score. Because some metrics improve with quality (e.g. precision and recall) and others vice versa (e.g. Lag-global), we used the absolute operator to make sure that difficulty is in the range $[0, \infty)$.

We propose to analyze and report NLP performance by analyzing the extent to which NLP performance varies over a wide spectrum of difficulty levels. This would allow readers to better compare different NLP techniques when measured on different datasets as the impact of difficulty has been mitigated, thus avoiding the mistake of considering one NLP worse than another only because it is measured on a more difficult dataset.

Some may wonder why we chose the random algorithm among several available rudimentary algorithms as the "ubiquitous grep". We note that we propose to adjust performance metrics to allow a better interpretation of the support, provided by a given NLP technique, to the human analyst. In this context, the use of randomly generated scores clearly represents the situation where the human analyst has no retrieval support. On the contrary, adopting as a baseline the score provided by a

rudimentary algorithm, e.g. ubiquitous grep, would represent a situation where the human analyst is supported by a retrieval tool encompassing an algorithm. At this point, we wonder why a rudimentary algorithm (e.g. ubiquitous grep) should be chosen by those who need to spend effort implementing a retrieval algorithm, when several more intuitively supportive algorithms are available online (e.g. the 242 NLP techniques analyzed in this work).

3.5.3 Implementation strategies

A possible way to implement this principle is to create sub-datasets, from the available dataset(s), with a wide spectrum of difficulty. The performance of the same NLP technique would then be reported and analyzed over different difficulty levels.

An available procedure to create sub-datasets with a wide spectrum of difficulty levels is to randomly sample the whole dataset with a wide spectrum of proportions of the category to retrieve (i.e. equivalent requirements pairs). We focus on a wide spectrum of proportions of equivalent pairs because the difficulty of all performance metrics is reasonably related to the specific proportion of true links. An example of a wide spectrum of proportions is 10%, 20%, 30%, 40%, 50%, 60% 70%, 80%, and 90%. Several sub-datasets should be sampled for each proportion since the sampling strategy, adopted to create a sub-dataset, is reliable only over multiple applications. Regarding the size of the sub-datasets, this should be the same among proportions. Moreover, in theory, the higher the size of a sub-dataset, the higher its reliability. However, because the dataset from which the sub-datasets are sampled is of limited size, the size of sub-datasets is limited by the number of elements of a given category in the dataset. For instance, considering the above example of spectrum of proportions, and considering a dataset with size 1000 and 36% of true links, then there are only 360 available true links to build the sub-datasets (i.e., 1000 * 36%). This means that, the size of the sub-datasets with 90% of proportion of true links is limited to 400 links; i.e. 360 true links plus 40 false links. A strategy to improve this limit in the size of sub-datasets is sampling with replacement [102]. Because this sampling strategy has some limitations, the size of the sub-datasets should be a tradeoff. The size of the sub-datasets should undoubtedly be slightly higher than the minimum between the number of false links and the number of true links in the dataset.

An available procedure to compute the random score, for each pair of elements in the dataset, is to randomly generate a similarity score in the range [0,1] using an appropriate distribution (e.g., uniform). All major statistical tools provide a random generator. For instance, the random generator in Excel ® has been extensively validated [100-102].

Note that the best ideal score does not need to be computed on a given dataset; it depends on the meaning and range of the metric. For example, the ideal best score of recall is 1, independently of the dataset.

$$Difficulty = \left| \; BestScore - RandomScore \; \right|$$

Equation 1: Equation to compute the difficulty of a dataset for a given metric.

3.5.4 Discussion

Given the novelty of the present principle, this section includes a validation part and is structured in sub-sections.

3.5.4.1 Validation

The aim of this section is assess whether the difficulty of the dataset in use to evaluate NLP techniques impacts the evaluation results. Therefore, we aim to investigate the following null hypothesis:

$H_{Importance0}$*: The performance of an NLP technique is independent of the difficulty of the dataset in use.*

3.5.4.1.1 Method

The dependent variable is the performance of NLP techniques. Due to feasibility constraints we focused only on one performance metric: precision. We chose precision because it is one of the most representative and common in the field.

The independent variable is the difficulty of the dataset as defined by Equation 1. In order to have a set of datasets covering a wide spectrum of difficulty, we created nine types of sub-datasets from the dataset described in Section 4.2.3.2. Each of these types of sub-datasets was created 30 times by randomly sampling the whole dataset with the constraints to have 200 requirement pairs, and the following proportions of equivalent pairs: 10%, 20%, 30%, 40%, 50%, 60% 70%, 80%, and 90%. This is because the difficulty is reasonably related to the proportion of true links.

Due to feasibility constraints, we consider 20 NLP techniques. We chose the following NLP techniques so as to represent the proposed classification (see Section 2.2):

1. VSM, BINARY, COSINE, 0, Stanford (nouns)
2. VSM, RAW, COSINE 0, Stanford (nouns)
3. VSM, BINARY, COSINE0, Stanford (nouns and verbs)
4. VSM, RAW, COSINE0, Stanford (nouns and verbs)
5. VSM, BINARY, JACCARD0, Porter stemmer
6. VSM, RAW, DICE0, Simple
7. VSM, TF_IDF, COSINE0, Simple
8. VSM, IDF, JACCARD0, Simple
9. WordNet, RESNIK, IDF, WN_SIMILARITY0, Stanford (nouns)
10. WordNet, JIANG, IDFm WN_SIMILARITY0, Stanford (nouns)
11. WordNet, RESNIK, IDF, WN_SIMILARITY0, Stanford (nouns and verbs)
12. WordNet, LIN, IDFWN_SIMILARITY0, Stanford (nouns and verbs)
13. WordNet, LIN, IDFWN_SIMILARITY0Simple
14. WordNet, PIRRO_SECO, IDFWN_SIMILARITY0, Simple
15. LSA, TF_IDF, COSINE, 50, Simple

16. LSA, BINARY, COSINE, 60, Simple
17. LSA, RAW, COSINE, 60, Simple
18. LSA, RAW, COSINE, 80, Simple
19. LSA, TF_IDF, COSINE, 80, Simple
20. LSA, BINARY, COSINE, 90, Simple

The measurement procedure consists in computing the difficulty of the dataset and the precision for each of the above-mentioned NLP techniques, adopting 0.50 as the threshold between positive and negative candidate links. Thus, this procedure relies on more than a million observations (i.e., 21 techniques * 9 types of dataset * 30 datasets * 200 links).

3.5.4.1.2 Results

Figure 2 plots the precision, for each of the 20 NLP techniques, according to difficulty levels. Different colors relate to different NLP techniques.

Instead of reporting actual observations, we preferred to report their linear least squares regression so as to support easy interpretation of the tendency of observations of the same NLP technique. Linear least squares regression is the most commonly adopted technique to model a linear relation between two variables [103]. Specifically, the unknown parameters of the linear model are estimated by minimizing the sum of the squared deviations between the data and the model. Although non-linear regression models could also have been used, they would not allow easier interpretation of trends among NLP techniques. Because providing accurate predictions lies beyond the scope of the present paper, we refer the study of the most appropriate regression model to future work.

From Figure 2, we can clearly see that, for all the 20 NLP techniques, the performance depends on difficulty. Specifically, as expected, the higher the difficulty, the lower the score.

We analyzed the statistical impact of difficulty on precision, via the Wilcoxon-Mann-Whitney test, on average among NLP techniques and for each of them. All p-values proved less than 10^-4 and therefore the impact of difficulty on performance is statistically significant. Consequently, what is evident from Figure 2 is statistically confirmed: the difficulty of the dataset influences evaluation results.

This is particularly important as, due to practical constraints, NLP evaluations are based on a limited number of dataset(s) and therefore the external validity is questionable.

3.5.4.1.3 Limitations

The main limitation of the proposed validation may seem to be the use of only on dataset upon which the different sub-datasets have been sampled. Conversely, the use of one dataset allowed us to reduce the differences among sub-datasets and gain in results validity. Specifically, the sub-datasets are very homogeneous (since they are written by the same humans, in the same domain, etc.) and hence the different difficulty is more likely to be the main cause of observed differences in performance of the same technique among sub-datasets.

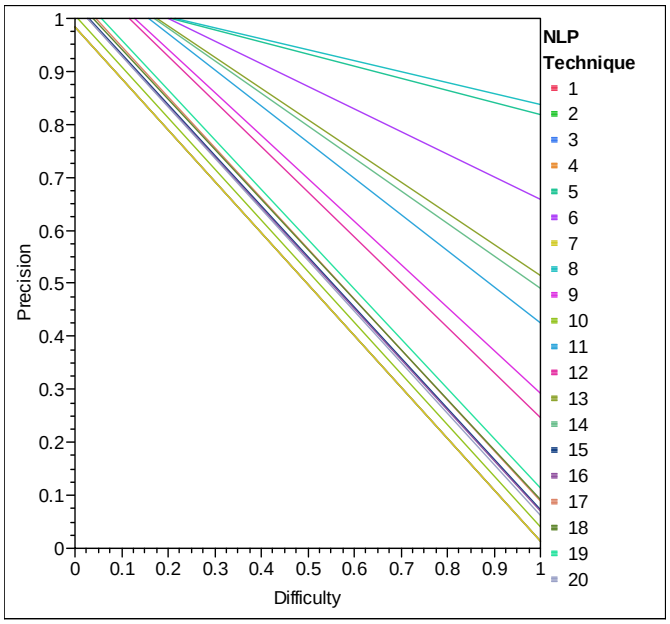

Figure 2. Precision of NLP techniques related to difficulty.

Further limitations are the number of performance metrics and NLP techniques adopted. Note that we cannot demonstrate that difficulty influences the performances of all NLP techniques, in all metrics, in all datasets. The proposed validation certainly provides some genuine evidence suggesting the application of the intuitive proposed principle of reporting NLP performance over different difficulty levels. Moreover, no side effects are either observed or expected from taking difficulty into consideration.

Replication is suggested and fairly straightforward. For instance, it is very easy to set up a simulation and observe that for a random estimator, the precision is higher in a dataset balanced over true links than vice versa.

3.5.4.2 Conclusions

According to the above results, the evaluation of NLP techniques is misleading when based on different difficulty. Specifically, any technique becomes better or worse than any other when compared on a dataset with different difficulty. For instance, the dominant NLP technique, i.e. NLP 8 (green dotted line at the top), is better than the worse NLP, i.e., NLP 7 (orange dotted line at the bottom), when considering the same difficulty. However, NLP 7 is better than NLP 8 when NLP 7 is measured over a difficulty around 0.05 and NLP 8 over a difficulty around 0.95. This implies that NLP 7, though much worse, can be erroneously preferred over NLP 8 when measured on different datasets.

An alternative option for alleviating the influence of difficulty on results is to perfectly balance the dataset(s) in use by resampling. However, this approach has two main drawbacks:

1) Too many data are wasted: Resampling does not work well when difficulty is very high or very low (i.e., when it is particularly influential on the results). Consider for example a dataset where only 5% of data belong to the category to retrieve; application of a resampling procedure, for balancing categories, would eliminate 90% of data (i.e., 100% - 5% -5%). This procedure significantly decreases result validity because the results are based on a considerably smaller (re-sampled) dataset. Therefore, while resampling is an option that completely alleviates the effects of difficulty on the results, it seriously reduces validity. Moreover, the higher the difficulty, the higher the need for alleviating its impact on results. This is because the higher the difficulty, the more data are eliminated by balancing the dataset. Thus, in conclusion, resampling is not a reliable option to alleviate difficulty.
2) Generalizability: real application contexts usually have unbalanced datasets. Therefore, evaluation results on balanced datasets are usually not generalizable.

Albeit rigorous and valid, the proposed procedure to measure and report difficulty may lead to the following mistaken conclusions:

1) The adjusted results might be deemed completely generalizable. This is wrong because the difficulty is only one aspect of the dataset that has been alleviated. External generalizability of results is always unpredictable. Indeed, irrespective of the number of adopted datasets, the results may completely differ in other datasets. Therefore, NLP techniques may perform differently on different datasets with the same difficulty, due to intrinsic reasons, i.e., a specific NLP technique is more or less suitable for different types of language.
2) Deeming the proposed difficulty as the only type of difficulty a dataset may have. This is wrong as our concept of difficulty concerns only the concept of "odds" characterizing a given dataset, i.e. the proportion of true links. Another well known concept of difficulty is the intrinsic difficulty of a "sublanguage" to be analyzed via NLP [104]. For instance, it is well known that a "sublanguage" of a natural language is easier to analyze than generalized, unrestricted natural language [105]. Different sublanguages can be more or less easy to analyze according to different characteristics, such as the size of the restriction on the (extremely wide) natural language. Therefore, in two datasets with the same difficulty in terms of odds, the same NLP technique would perform significantly better in a dataset characterized by a significantly easier sublanguage. See [106] for further types of difficulties not taken into consideration in the proposed adjustment procedure.
3) Relating difficulty to required effort. This is wrong as a dataset may be an order of magnitude larger than another dataset with the same difficulty. However, adopting the same technique, the latter dataset will certainly require much more effort to be analyzed.

3.6 Principle 6: Analyzing Boundaries by adjusting scores

3.6.1 Motivation

According to Figure 2, the meaning of the performance score may be misleading. For instance, we can judge the support of NLP 8 as unsatisfactory if analyzed on a dataset with a difficulty around 0.95; i.e. precision around 0.85. Conversely, because a random approach would provide a precision of 0.05 (i.e. random score= best score - difficulty), then NLP 8 should be deemed useful in practice because it offers a precision that is 17 times higher than a random approach (i.e. 0.85/0.05).

Even when NLP performance is computed over the same dataset, i.e. over the same difficulty, the results may be misleading as prone to overestimation and underestimation of the differences in the performance among NLP techniques. For instance, suppose the difference between two NLP techniques, in a given metric (e.g., precision), is 0.05. It is then difficult to judge the two NLP techniques as different in practice or not.

3.6.2 Proposed solution

We propose to *adjust* the performance of NLP techniques according to both the upper and lower performance boundaries. Equation 2 describes a formula to compute an adjusted score for a given current score, regardless of the performance metric used (e.g., recall, precision, etc.).

- *Adjusted Score:* the value resulting from neutralizing the effect of difficulty on a performance measurement, as obtained by a given metric.
- *Current score*: the value of a given metric computed on a given dataset.

The numerator in Equation 2 represents the improvement provided by the evaluated NLP technique over a random technique; the denominator represents the maximum possible improvement. Therefore, performances of NLP techniques expressed in the adjusted score proposed in Equation 2 are in the range $[-\infty, 1]$ and must be interpreted as in the following:

- *Adjusted score = 0*: the performance is equal to the ideal non-intelligent NLP technique. In other words, the NLP technique with such an adjusted score provides no support.
- *Adjusted score > 0*: the performance is better than the ideal non-intelligent NLP technique. The closer the adjusted score to 1, the closer the support provided by the NLP technique is to the maximum; the closer the adjusted score to 0, the nearer the support is to a non-intelligent technique. For instance, an NLP technique with an adjusted score of 0.30 is more similar to a non-intelligent technique than to the ideal one.
- *Adjusted score < 0*: the performance is worse than the ideal non-intelligent technique. In other words, adopting an NLP technique with a negative adjusted score would be counterproductive.

$$AdjustedScore = \frac{CurrentScore - RandomScore}{BestScore - RandomScore}$$

Equation 2: Equation to compute the adjusted score of performance of an NLP technique.

3.6.3 Implementation strategies

Equation 2 is very straightforward to compute once the NLP techniques have been applied to sub-datasets with different difficulty levels as described in Section 3.5.3.

3.6.4 Discussion

Because the adjustment procedure is a scaling operation, then adopting standard or adjusted metrics provides the same ranking of NLP techniques. Moreover, the meanings of adjusted scores are formally defined and do not require any form of validation.

Despite just a scaling operation, adjusting metrics supports a better interpretation of results: it avoids both overestimation and underestimation of the differences in performance among NLP techniques. For instance, suppose the difference between two NLP techniques, in a given metric (e.g., recall), is 0.05. When computed on the adjusted score, then the two NLP techniques are clearly almost equivalent on that metric.

As regards conclusion validity, the level of support provided by NLP techniques is more clearly expressed via adjusted scores. This supports the analysis of the actual limits of NLP techniques by observing the distance from the ideal performance.

A further option to analyze NLP performance in respect to boundaries is by means of a ranking effectiveness graph. Here the abscissa represents the percentage of analyzed artifacts and the ordinate represents the percentage of retrieved artifacts. The performance of the actual NLP technique is then plotted together with the performance of the ideal best technique and the non-intelligent technique. Consequently, the distance of the actual NLP technique from the non-intelligent technique is an objective measure of the support provided by the given NLP technique. See Figure 10 for an example.

3.7 Principle 7: Validation of the Random Score

3.7.1 Motivation

In the previous subsection, we stressed the importance of alleviating the effect of difficulty on results by means of adjusted scores. However, the procedure of adjusting metrics includes the generation of a random distribution: the random score, i.e., a random similarity measurement. This random generation entails the threat to validity related to the chance of the random score not being random. Indeed, there is the possibility that the generated random similarity measurements are extremely similar (or extremely different) to the actual similarity; in such a case, the related adjusted metrics would lead research analysis to the wrong conclusions by underestimating (resp. overestimating) the performance of NLP techniques. In other words, while adjusting metrics enhances the conclusion and the external validity, getting it wrong would support erroneous conclusions.

3.7.2 Proposed solution

We propose here to validate the quality of the random scores generated; this is supposed to enhance conclusion validity.

3.7.3 Implementation strategies

An effective and easy procedure to check the reliability of the random scores consists of: 1) repeating the random generation many times (e.g., 1000), 2) measuring the score related to a given aspect(s) of performance (e.g., precision), and 3) computing the random score as the average among such generations. Afterwards, the reliability of the adopted random scores is measured as the standard deviation among such generations; the lower the standard deviation, the higher the reliability of the adopted random scores as well as the validity of the adjustment procedure.

3.7.4 Discussion

In general, it is better to adopt different random algorithms for the different generations. If the generations are done by the same random algorithm, and the latter is affected by a given bias (e.g., it may tend to produce low scores), the resulting random score would be affected by the same bias and the standard deviation would not detect it (this happens because the same error would be repeated). However, since the random algorithms provided by state-of-the-art statistical tools are reliable, then an error in random generation is expected to be caused only by chance and not by a shortcoming in the algorithm, which would produce a constant bias. In conclusion, we claim that generating the several random scores by adopting one random algorithm is a reliable procedure under three conditions: i) the random algorithm belongs to a state-of-the-art statistical tool, ii) the quality of the random score is validated by means of standard deviation, and iii) the number of generations is high.

4 An Industrial Case study on Detecting Equivalent Requirements

4.1 Context

The context of this study is the development of systems of systems. A system of systems is large, distributed, adaptive and complex; it is also structured into components (i.e., systems) that can work independently of each other though their cooperation provides a functionality that is greater than the sum of their functions [107].

Requirements taken into account in this paper are in English and they can be characterized, according to MIL STD 498, as System Segment Specification (SSS); they specify the requirements for a system or subsystem and the methods to be used to ensure that each requirement has been met. An example of a realistic, sanitized, and

equivalent requirement pair is "The system shall provide a mechanism to interact among and within systems of type B for allowing effective cooperation" and "The system shall support the interaction among and within systems of type B; this aims to support their effective cooperation."

4.2 Method

4.2.1 Problem statement and research objective

Experience from applying Product Line Engineering shows that investing in reuse does not always pay off [108]; the success factor consists in applying the right reuse strategy [109] with the right amount of investment [110] on the right assets (e.g., components, test cases, requirements) [111] without underestimating non-technical aspects (e.g., current organization). When SELEX Sistemi Integrati [3], an Italian company in the defense and aerospace domain, planned the development of five different products, taking advantage of the product commonalities was a first choice. In order to reason on what and how to reuse, it is mandatory to have an understanding of the variability and commonalities of the systems to develop; this activity is called domain analysis [9]. Detecting equivalent requirements is the key to domain analysis [4-8, 10]; however, humans alone cannot afford to analyze all the feasible combinations of requirement pairs for a potential link. Indeed, given a total number of 2500 requirements, the number of (feasible) combinations of requirement pairs to analyze is 3,123,750 (i.e., 2500!/(2500-2)!*2!).

NLP can support analysts in detecting equivalent requirements: it is used to measure the similarity among requirements; similarity measurements are used to rank/filter the requirement pairs according to their likelihood of being redundant. The aim of the case study is to characterize a set of NLP techniques according to the provided level of support to the human analyst in detecting equivalent requirements.

4.2.2 Goal, Questions, and Metrics

4.2.2.1 Goal

The goal of the case study is to analyze the NLP techniques, for the purpose of characterization, with respect to their performance (i.e., level of provided support) in retrieving equivalent requirements from the point of view of the human analyst in an industrial context.

4.2.2.2 Questions

In order to characterize the NLP state of the art, in the specific task of detecting equivalent requirements, we investigate the following research questions:

RQ 1. What is the ranking of the available NLP techniques? Which is the best NLP technique (if any)?

This question aims to detect the best NLP technique among the available ones, i.e., the technique that, if adopted in a traceability tool, would provide the maximum benefit to the human analyst compared to other techniques.

RQ 2. How does the best NLP technique perform? Is there room for improvement? How far is its performance from the ideal performance?

It is important to reveal the actual limitation in the support that the best NLP technique provides; this highlights the current shortcomings, which in turn drives future research.

RQ 3. Does performance significantly vary among NLP techniques?

In general, suggesting that a given technology should not be used is as important as suggesting the contrary. Therefore, it is important to analyze the benefits of adopting any NLP technique, and, in the event, to identify counterproductive techniques.

RQ 4. How does an optimal combination of available NLP techniques perform?

Natt och Dag et al. claim that "different linguistic models might together contribute to better precision" [112]. Therefore, it is important to analyze:

- The difference, in performance, between the best single NLP technique and the optimal combination of several NLP techniques. While there is no doubt that an optimal combination of NLP techniques cannot be worse than any single technique (including the best one), it is questionable whether combining the NLP techniques is worthwhile, i.e., if a combination of NLP techniques significantly outperforms the best single NLP technique.
- The actual limitations of NLP techniques. If the optimal combination of NLP techniques is far from an ideal technique, it is possible to achieve significant advances. Conversely, if the optimal combination of NLP techniques is very close to the ideal technique, the NLP field should be considered mature and there is a low possibility that new NLP techniques would significantly contribute to the field.

RQ 5. Do NLP techniques actually differ from one another?

Analyzing the equivalence among NLP techniques is as interesting as analyzing their differences. For instance, when in the event of equivalence, practitioners can make the informed decision that selecting one or another NLP technique does not change the expected benefits, researchers may realize that a new NLP technique is equivalent to others. Judging two NLP techniques as equivalent, when they are not, would cause as much damage as incorrectly deeming one NLP technique better than another. To our knowledge, no past study has statistically tested the equivalence among NLP techniques.

In this paper we analyze a large number (242) of NLP techniques. Since we exclude any discussion or evaluation of human feedback, the paper can be considered a study of methods rather than a study of humans [50].

4.2.3 Data Collection, Measurement, and Pre-processing procedure

4.2.3.1 Overview

Figure 3 describes the flow of data in the case study, from top-left, clockwise. Data collection was performed at SELEX SI and automated by means of an open source tool

called PROUD [2]. The measurement procedure was performed by using a new open source tool called ANTARCTICA (Automatic characterizatioN of naTural lAnguage pRoCessing meThods for estimating semantiC similarity) which applied the available NLP techniques [113]. In order to pre-process the similarity measurements that ANTARCTICA produced, we adopted Weka [82]. The data were analyzed by means of a statistical commercial tool (JMP™). Finally, the case study results were packaged and presented to SELEX SI with the aim of improving the process of detecting equivalent requirements.

4.2.3.2 Data Collection

The dataset adopted in this study coincides with [2]. Table 6 reports the main characteristics of our case study at SELEX SI regarding five different projects with 500 requirements each on average. Among a population of nearly three million possible requirement pairs, we sampled a total of 983; 138 requirement pairs were equivalent. The sampling procedure was driven by: 1) the realistic amount of human effort that can be spent in such a tedious task, 2) the aim to detect all the equivalent requirements. For further details about the sampling procedure and the adopted dataset please see [2].

4.2.3.3 Measurement Procedure

In order to characterize the 242 NLP techniques, we developed ANTARCTICA, which works as a post mortem analysis tool; it takes as input the log file as produced by PROUD and provides as output a benchmark, i.e., the similarity measurement, for each requirement in the dataset, for each of the NLP techniques described in Section 2.2. Therefore, ANTARCTICA does not directly support human analysts; rather it provides a useful means to ascertain which NLP technique would best support them. The data were analyzed according to the proposed empirical principles as discussed in the following section.

4.2.3.4 Principles implementation details

The case study was analyzed by applying the empirical principles discussed in Section 3. Specifically:

-*Principle 1. Optimal threshold*: We measured the performances of the 242 NLP techniques on the threshold maximizing the objective function. We defined the objective function as having the same weight to false positive and false negative types of error. Therefore, though there was one objective function shared among the NLP techniques, the resulting threshold was specific to each technique. Then we applied simple Logistic Regression by using Weka.

-*Principle 2. Cross-validation*: In order to obtain more realistic results, we applied the standard ten-fold cross validation procedure by using Weka.

-*Principle 3. Statistically testing difference*: To compare the performance between the optimal combination and the best single NLP technique we applied a one-tailed Z-score for a proportion test, on precision and recall metrics. The results are described in section 4.3.4.

-*Principle 4. Statistically testing equivalence:* By using JMP®, we applied Principal Component Analysis to the similarity measurements achieved by the 242 NLP techniques on each requirement pair in the dataset. Our results are described in section 4.3.5.

-*Principle 5. Impact of difficulty:* To compute how the performances of the best and the optimal combination of NLP techniques vary with difficulty we applied a procedure similar to that reported in section 3.5.1.3. Specifically, we measured the performance (i.e., Credibility, Lag-global, precision, recall, and the ROC area) on different sub-dataset types of the same dataset. Each type of sub-dataset comprises 200 links, has a specific proportion of true links (10%, 20%, 30%, 40%, 50%, 60%, 70%, 80%, 90%), and was created 30 times by randomly resampling the (big) industrial dataset (see Table 6). The adopted threshold is 0.75. This analysis relies on 162,000 observations, i.e., 3 NLP techniques * 9 types of dataset * 30 datasets * 200 links. Performance of the best single and the optimal combination of NLP techniques is reported in sections 4.3.2 and 4.3.4, respectively.

-*Principle 6. Adjusting scores:* To compute the random score we randomly generated a similarity score in the range [0,1], for each pair in the dataset, using the uniform distribution, via Excel ®. Our results are described in sections 4.3.2, 4.3.3, and 4.3.4.

-*Principle 7. Random score validation:* We validate the random score by generating the random scores 1000 times and measuring the related standard deviation. Our results are discussed in section 4.3.2.

Table 7 reports the difficulty (see Section 3.5) of the different performance metrics of the adopted dataset. According to Table 7, it is easier to achieve high precision and recall than the ROC area and Credibility; therefore, the same eventual differences among NLP techniques need to be emphasized in their importance in precision or recall over ROC area and Credibility. This is due to a smaller variability range.

4.3 Results and Interpretations

4.3.1 RQ 1: What is the ranking of the available NLP techniques? Which is the best technique (if any)?

4.3.1.1 Results

The evaluation procedure consists of computing, for each NLP technique, the following metrics: precision, recall, ROC, Lag-global and Credibility. A table format is not suitable to rank the NLP techniques given the large number of diverse data to display (242 techniques x 5 metrics); thus, our approach is to rank the variants. Since the same variant is adopted by several NLP techniques (in combination with other variants), we computed the maximum score achieved by a given variant in all the different combinations. We adopted a Kiviat diagram for each variation point, a color for each variant, and a dimension for each of the adopted performance metrics.

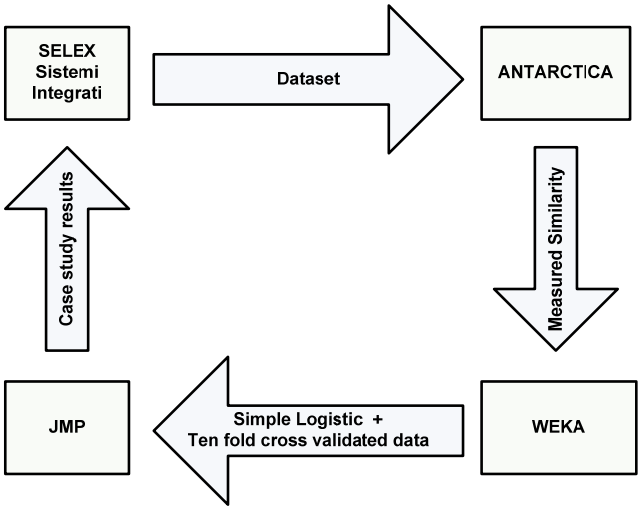

Figure 3. Flow of data from data sampling to the results; clockwise from top-left.

TABLE 6
Case Study Characteristics

Application domain	Systems of Systems
Industry	SELEX SI
Number of projects	5
Total number of requirements	2483
Total number of possible requirements pairs	3081403
Sample size (classified pairs)	983
Equivalent requirements pairs	138

TABLE 7
Difficulty of the Adopted Dataset

Metric	Difficulty
Precision	*0.26*
Recall	*0.14*
ROC area	*0.50*
Lag-global	*4.34*
Credibility	*0.50*

We stress that the performance of each NLP technique was computed as a whole technique. To improve readability the results are reported in terms of variants and variation points.

Figure 4 describes the performance of the variants; the greater the area of a given variant, the better the performance. For each variation point, a best variant exists only if the area includes all the others, i.e., it is better than (or equal to) all the other variants in all dimensions (performance metrics). The best NLP technique consists of four best variants, one for each variation point.

Because the Kiviat diagram requires metrics ranging between 0 and 1 according to their goodness, we normalized the Lag-global metric by dividing the Lag-global score of each NLP technique by the lowest score among the techniques; as a result, NormLag measures the same concept as Lag-global but ranging from 0 (worst NLP technique) to 1 (best NLP technique).

4.3.1.2 Discussion

According to Figure 4, we can make the following considerations:

- *Algebraic model*: VSM, Thesaurus-based Similarity, and LSA exhibit similar performance with respect to precision, recall, and the ROC area. VSM clearly outperforms Thesaurus-based Similarity and LSA in both credibility and Lag-global metrics. Therefore, VSM is the best variant of the algebraic model variation point.
- *Weighting*: all the variants performed very similarly in credibility, precision, recall and the ROC area. Raw frequency clearly outperforms all the other variants in Lag-global metric; therefore, raw frequency is the best variant of the Weighting variation point.
- *Term extraction*: again, all the variants performed very similarly in credibility, precision, recall and the ROC area. As regards Lag-global, Stanford (verbs and nouns) clearly outperforms all the other variants, and is thus the best of the Term extraction variation points.
- *Similarity metric*: all the variants performed very similarly in precision, recall and the ROC area. Cosine clearly outperforms the other variants in both credibility and Lag-global performance metrics. Therefore, Cosine is the best variant of the *similarity* metric variation point.

Therefore, the NLP technique consisting of VSM, raw frequency, Cosine and Stanford (verbs and nouns) is the best; this technique, if adopted in a traceability tool, is expected to provide the maximum benefit to the human analyst compared to other NLP techniques.

Interestingly, the use of synonymies proved deleterious. The most reasonable deduction is that in a mature organization, professionals tend to adopt the same terms when they express the one and same technical concept. Therefore, techniques that are able to detect similarity even when the terms are not identical resulted in worse performances, introducing more noise rather than making a positive contribution. This reasoning can be linked to the concept of sublanguages as the language in the adopted dataset clearly proved to be a subset of the English language [104, 105]. Results from the case study suggest the conjecture that, for homogeneous artifacts, if the terms are different, it is better to judge their semantic as different rather than consider possible semantic similarity. Preliminary empirical validation for such a conjecture is that the simplest NLP techniques are usually adopted to retrieve duplicates [1, 46, 112, 114].

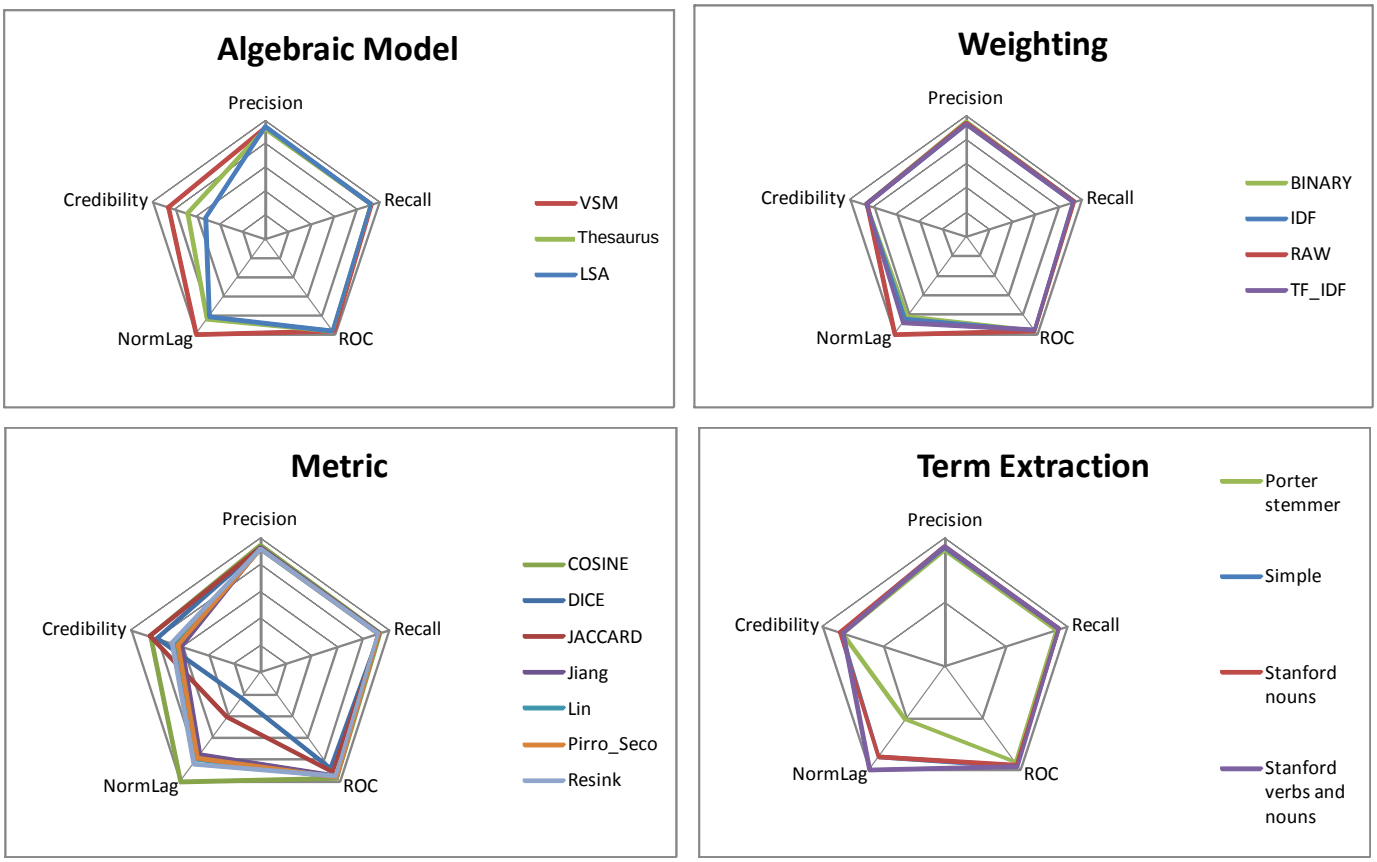

Figure 4 Kiviat diagrams, one per variation point, of the maximum performance of each NLP technique variant.

Conversely, one might expect the use of complex NLP techniques to be more convenient for heterogeneous artifacts (e.g., requirements from different projects or organizations or application domains or companies with weakly shared culture and vocabulary), because they might use polysemy and synonymy, and their vocabulary might include hyponyms and hyperonyms. Again, based on the literature, a preliminary empirical validation for this further conjecture is that complex NLP techniques seem to perform better than simpler ones when adopted to retrieve (trace) artifacts developed by different organizations, or related to different application domains, or belonging to different abstraction levels [14, 30, 33-35, 37-39, 41, 42, 45].

In other words, different datasets, of homogeneous artifacts, can be seen as different sublanguages where the spectrum of the English language has been appropriately fragmented. Future studies can focus on analyzing "how" this spectrum has been fragmented according to different types of artifacts. For instance, past studies focused on understanding the sublanguage characterizing the comments of C++ software packages [115, 116]. Future studies may focus on the sublanguage(s) characterizing requirements; this would provide insights into the language adopted by requirements analysts and would in turn suggest effective strategies to improve NLP techniques.

Among the adopted performance metrics, only Lag-global and Credibility were able to discriminate NLP techniques. Therefore, we maintain that future evaluation of NLP techniques should also adopt Lag-global and Credibility performance metrics together with more standard and common metrics like precision, recall, and the ROC area.

4.3.2 RQ 2: How does the best NLP technique perform? What is the top performance? Is there room for improvement?

4.3.2.1 Results

Our evaluation procedure starts by reporting the performance (Table 8) and adjusted scores (Table 9) of the best NPL technique. To investigate the variation in performance according to difficulty we applied the procedure described in Section 4.2.3.4. Figures 5, 6, 7, 8 and 9 report the performance of the best NLP technique in precision, recall, the ROC area, credibility, and NormLag, respectively. Each figure reports both actual observations and the linear least squares regression.

TABLE 8

Performance of the Best Single NLP Technique

NLP	Precision	Recall	ROC Area	Lag-global	Credibility
VSM, RAW, COSINE, Stanford nouns and verbs	0.935	0.936	0.967	0.754	0.850

TABLE 9

Adjusted Scores of the Best NLP Technique

NLP	Precision	Recall	ROC Area	Lag-global	Credibility
VSM, RAW, COSINE, Stanford nouns and verbs	0.751	0.543	0.934	0.827	0.700

TABLE 10

Performances of Ideal NLP Technique

Ideal Performances						
		Precision	Recall	ROC Area	Lag-global	Credibility
Best		1.000	1.000	1.000	0.000	1.000
Worst		0.000	0.000	0.000	4.370	0.000
Non-intelligent (random)	Average	0.739	0.860	0.500	4.340	0.500
	Std. Deviation	0.000	0.000	0.000	N/A	0.009

Figure 10 shows the ranking effectiveness by relating the percentage of requirement pairs that needs to be analyzed (abscissa) to retrieve a given percentage of equivalent pairs (ordinate), according to the provided ranking. The steeper the slope, the better the ranking provided by the given NLP technique. In Figure 10, starting from the top, the red line shows the performance of the best ideal ranking, the violet line shows that of the best NLP technique, and the sky blue line reports the average performance among NLP techniques; this will be considered when discussing the next research question. The dark blue line shows the performance of a non-intelligent ranking. Therefore, the closer an NLP technique is to the red line and farther from the blue line, the better its ranking. Finally, the green line shows the performance of the ideal worst ranking; it represents the lower bound in ranking effectiveness.

4.3.2.2 Discussion

According to Table 9, the best NLP technique performed quite well. The adjusted scores are higher than 0.5 in all the performance metrics; this means that the performance of the best NLP technique is nearer the ideal perfection rather than non-intelligence in all the different aspects of performance. In particular, the ROC area is very close to a perfect performance. Moreover, according to the last row in Table 10, the standard deviation is low and therefore the adopted random scores (i.e., average value) show high reliability.

Regarding the impact of difficulty on performance, we note that in general, these results are useful for future aggregation of results rather than drawing conclusions on the best optimal NLP technique. However, on comparing

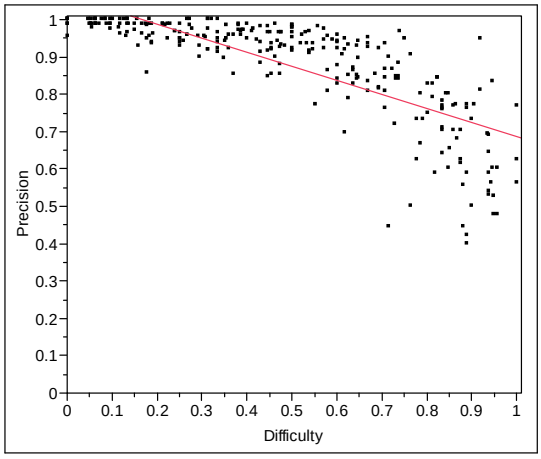

Figure 5 Precision of the best NLP technique related to difficulty

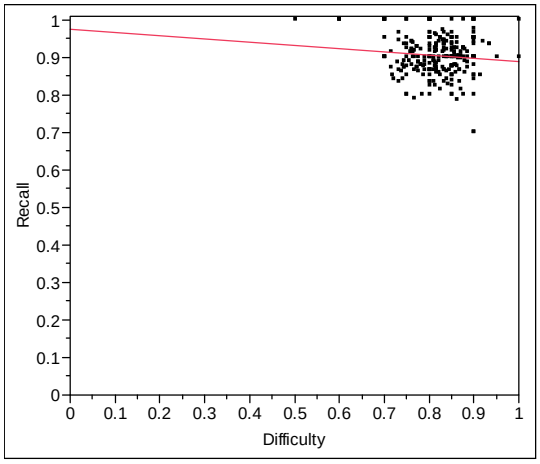

Figure 6 Recall of the best NLP technique related to difficulty

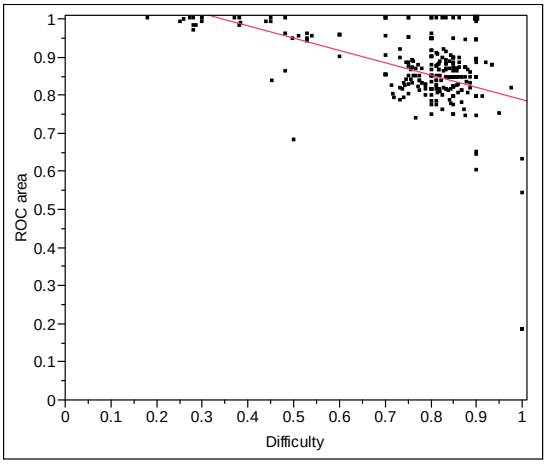

Figure 7 ROC area of the best NLP technique related to difficulty

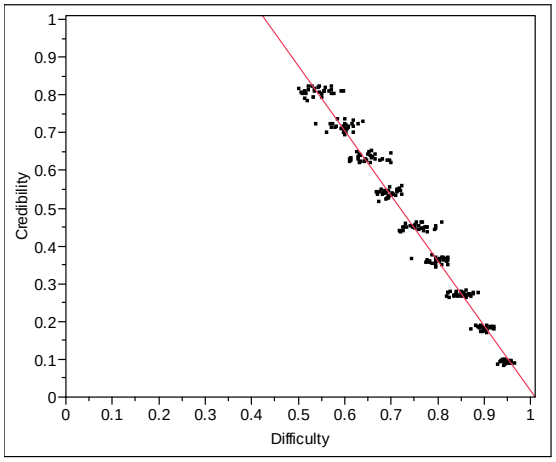

Figure 8 Credibility of the best NLP technique related to difficulty

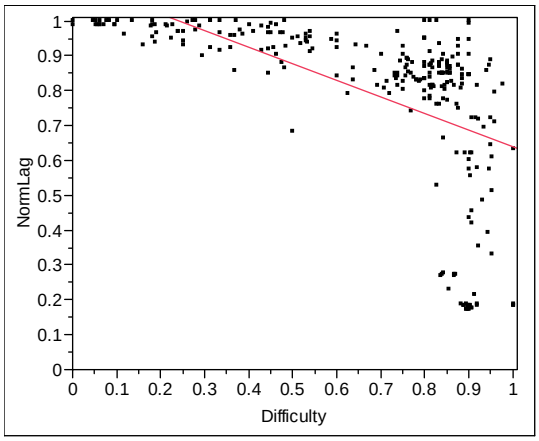

Figure 9 NormLag of the best NLP technique related to difficulty

Figures 5, 6, 7, 8 and 9 we can draw one very interesting conclusion: difficulty influenced all the five performance metrics but to a different extent. For instance, difficulty impacts credibility (Figure 8) more than recall (Figure 6). This paves the way to future empirical studies investigating the different extent to which difficulty impacts different metrics.

As far as suggesting is concerned, the credibility values reported in Table 8 and the results discussed in [2] propose that providing users with the similarity measurement produced by the best NLP technique improves both speed and quality of human classifications: it is supposed to reduce the time required to classify a given requirement pair by 1.6 seconds (-25%), while correctness is supposed to increase by 5%. As regards ranking effectiveness, the best NLP technique proved close to the best ideal performance (see the proximity between the red and violet lines in Figure 10). Table 9 shows that the Lag-global value of the best NLP technique is much better than a non-intelligent technique; in particular, random ranking has a Lag-global value 15 times higher than the best NLP technique. Hence, *ranking the requirement pairs according to the best NLP technique drastically reduces the effort required to retrieve equivalent requirements compared to random ranking*. In summary, the best NLP technique performed very well in ranking equivalent requirement pairs.

4.3.3 RQ 3: Does performance significantly vary among NLP techniques?

4.3.3.1 Results

Our evaluation procedure consists of showing the distributions among the NLP techniques, of the different performance metrics, for standard and adjusted scores. These are shown in Figures 11 and 12, respectively.

4.3.3.2 Discussion

Figure 11 shows that, while the various techniques have a very low variability in terms of precision and recall, this is not the case of ROC nor to some extent of other performance metrics. In particular, regarding ranking, we can infer from Figure 10 the extent to which NLP techniques are, on average (sky blue), significantly better than a random technique (dark blue).

As regards suggestion, reference [2] reports that the similarity measurement supports the traceability process only if credibility is higher than 0.65. Hence, according to Figure 11, only 15% of the available techniques are expected to improve speed and quality of human classification while the remaining 85% of NLP techniques would be even counterproductive. This result suggests that it is important to adopt an NLP technique according to the nature of the artifacts being compared. Otherwise, its usage may be counterproductive.

As far as ROC, recall and credibility are concerned, Figure 12 shows that more than 10% of NLP techniques do not perform better than a random technique (i.e., the adjusted score is no greater than zero in the 10th percentile).

Moreover, according to Figure 12, among the different metrics, most NLP techniques performed poorly in credibility and recall: 90% of NLP techniques have an adjusted score lower than 0.45 and are therefore closer to a random technique rather than to a perfect technique. Vice versa, regarding ROC, precision, and Lag-global, because the 50 percentiles in Figure 12 show adjusted scores higher than 0.5, most of the NLP techniques are more similar to the perfect technique rather than a random technique.

4.3.4 RQ 4: How does an optimal combination of available NLP techniques perform?

4.3.4.1 Results

The optimal combination of NLP techniques was computed with simple logistic regression. Ten-fold cross validation was then applied to measure the performance. Finally, the optimal combination of NLP techniques is compared with the best NLP technique by means of a statistical test. Table 11 describes the coefficients to combine the various NLP techniques optimally and linearly; NLP techniques with the coefficient of zero are omitted.

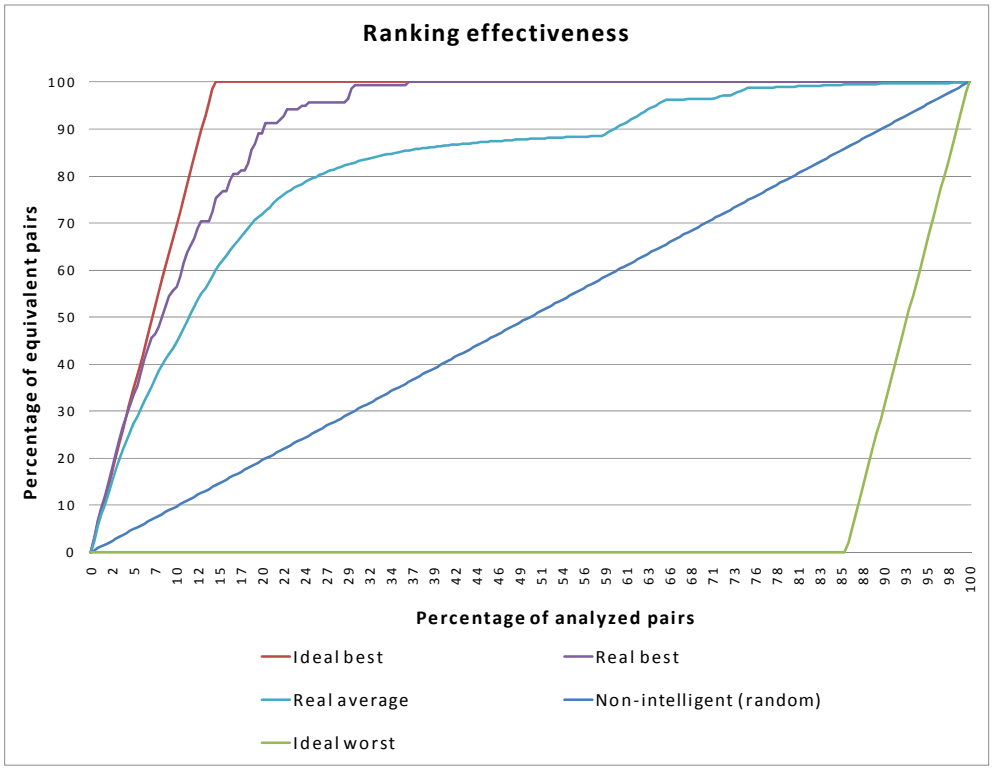

Figure 10 Ranking effectiveness of NLP techniques

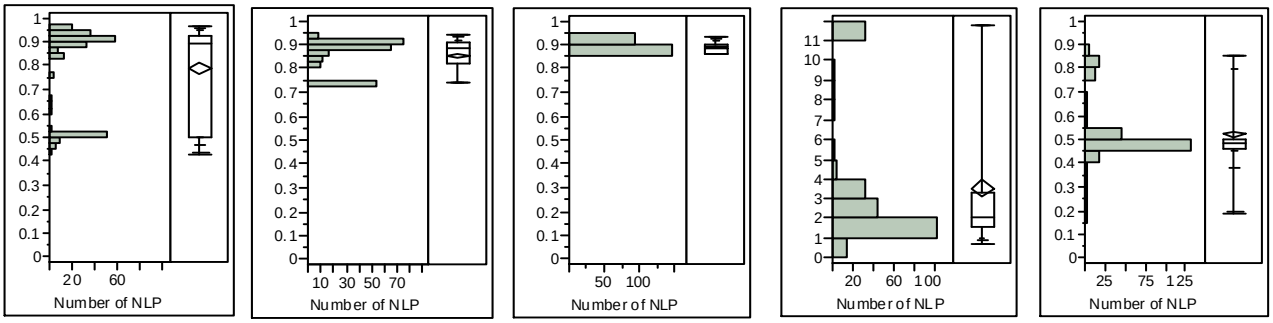

Figure 11. Distributions of performance of NLP techniques in terms of ROC, precision, recall, Lag-global and credibility

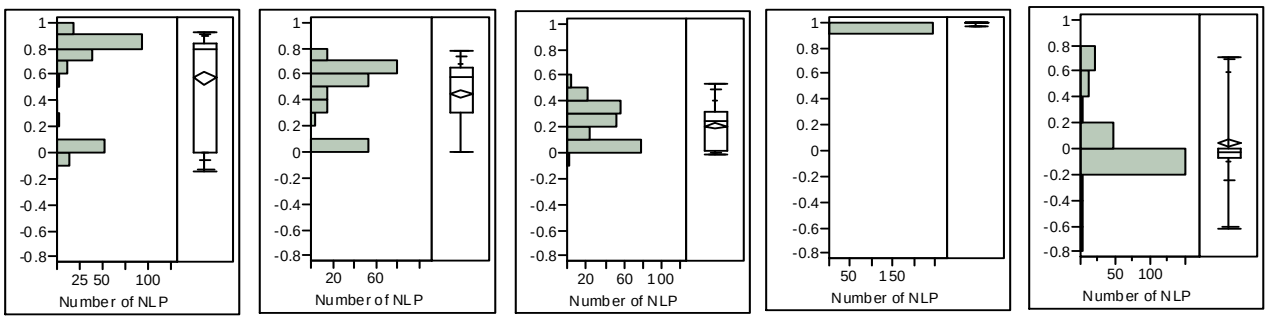

Figure 12. Distributions of adjusted performance of NLP techniques in terms of ROC, precision, recall, Lag-global and credibility

Table 12 reports the performance of such an optimal combination. Table 13 reports the result of the application of a one-tailed Z-score for a proportion test between the optimal combination and the best single NLP technique, on precision and recall metrics. Table 14 reports the adjusted scores related to the performance of a combination of NLP techniques.

To investigate the variation in performance according to difficulty we applied the procedure described in Section 4.2.3.4. Figures 13, 14, 15, 16 and 17 report performance of the optimal combination of NLP techniques in precision, recall, ROC, credibility, and NormLag, respectively. Each figure reports both actual observations and the linear least squares regression.

4.3.4.2 Discussion

On comparing Table 8 and Table 12 we can observe that an optimal combination of NLP techniques outperforms the best single technique in all the performance metrics. Since this result is mathematically obvious, it is worth analyzing the extent of improvement. Table 13 shows that there is a statistically significant improvement in combining NLP techniques via logistic regression in respect to the single best technique. Therefore, we can conclude that the adoption of logistic regression to combine NLP techniques significantly improves the benefits of adopting a single technique.

Regarding the impact of difficulty on results, on comparing Figure 13, 14, 15, 16 and 17 with Figure 5, 6, 7, 8 and 9 respectively, it may be observed that the optimal combination of NLP techniques dominates the best technique in the whole spectrum of difficulty and in all metrics. Because the lines representing the different metrics are less sharp in the case of the optimal combination of NLP techniques, we can conclude that the optimal combination of NLP techniques is more robust to difficulty. This is reasonable as the combination intuitively provides a damper over difficulty.

Table 14 shows that the performance of the optimal combination is similar to the best ideal technique. According to Table 14, with respect to the performance metrics taken into consideration, the optimally combined NLP technique has the lowest adjusted score in recall. The best single NLP technique also has the lowest adjusted score in recall. Hence, among the performance measures, recall seems to need the highest improvement; it is therefore advisable that further research effort be focused on improving recall (by new NLP techniques or new methods to combine them) rather than other performance metrics. This is also because humans are generally acknowledged as being poor at dealing with errors of omission.

TABLE 11

The Optimal Combination of NLP Techniques

NLP	Coefficient
LSATF_IDFCOSINE10Stanford (nouns)	1.17
LSABINARYCOSINE20Stanford (nouns)	-2.96
LSATF_IDFCOSINE20Stanford (nouns)	2.26
LSATF_IDFCOSINE30Stanford (nouns)	0.86
LSABINARYCOSINE120Stanford (nouns)	-1.09
LSABINARYCOSINE10Stanford (nouns and verbs)	-1.55
LSAIDFCOSINE10Stanford (nouns and verbs)	-0.39
LSATF_IDFCOSINE20Stanford (nouns and verbs)	-4.95
LSARAWCOSINE50Stanford (nouns and verbs)	1.66
LSABINARYCOSINE70Stanford (nouns and verbs)	-1.92
LSARAWCOSINE80Stanford (nouns and verbs)	2.69
LSATF_IDFCOSINE10Porter stemmer	2.65
LSAIDFCOSINE10Porter stemmer	0.65
LSABINARYCOSINE20Porter stemmer	1.71
LSARAWCOSINE70Porter stemmer	-1.81
LSABINARYCOSINE100Porter stemmer	-1.84
LSABINARYCOSINE10Simple	-0.43
LSABINARYCOSINE20Simple	0.85
LSARAWCOSINE40Simple	2.05
LSATF_IDFCOSINE120Simple	0.34
VSMBINARYDICE0Stanford (nouns)	0.64
VSMBINARYJACCARD0Stanford (nouns)	-0.83
VSMTF_IDFCOSINE0Stanford (nouns)	-2.24
VSMBINARYDICE0Stanford (nouns and verbs)	-0.11
VSMBINARYJACCARD0Stanford (nouns and verbs)	-0.07
VSMRAWCOSINE0Stanford (nouns and verbs)	-4.76
VSMBINARYJACCARD0Porter stemmer	-0.3
VSMRAWCOSINE0Porter stemmer	0.3
VSMIDFCOSINE0Porter stemmer	0.53
VSMBINARYDICE0Simple	0.36
VSMRAWCOSINE0Simple	-1.56
VSMTF_IDFCOSINE0Simple	-3.32
WordNet/JIANGIDFWN_SIMILARITY0Stanford (nou	1.28
WordNet/RESNIKIDFWN_SIMILARITY0Simple	-1.29

TABLE 12

Performance of the Optimal Combination of NLP techniques

NLP	Precision	Recall	ROC Area	Lag-global	Credibility
Optimal Combination of NLP (Simple Logistic Regression)	0.962	0.961	0.967	0.19	0.93

TABLE 13

Comparison of Performance Between the Optimal Combination and the Best Single NLP Techniques

Metric	P-value
Precision	<< 0.001
Recall	<< 0.001

TABLE 14

Adjusted Scores of Performance of the Optimal Combination of NLP Techniques

NLP	Precision	Recall	Roc Area	Lag-global	Credibility
Optimal Combination of NLP (Simple Logistic Regression)	0.854	0.721	0.934	0.957	0.860

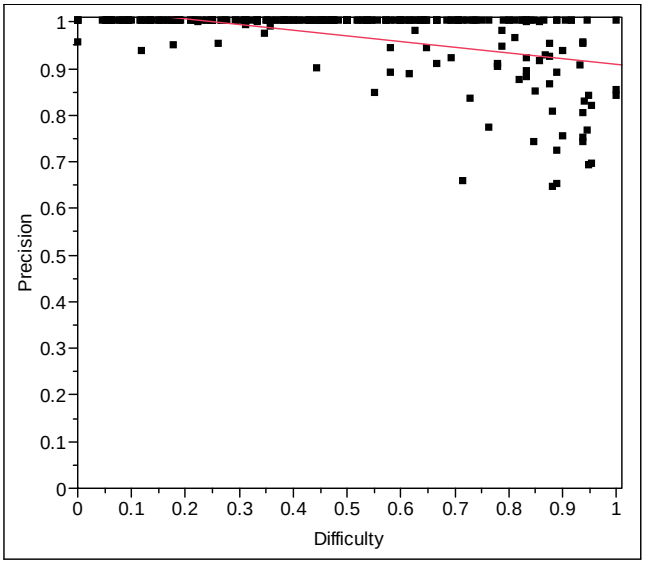

Figure 13. Precision of the optimal combination of NLP techniques related to difficulty.

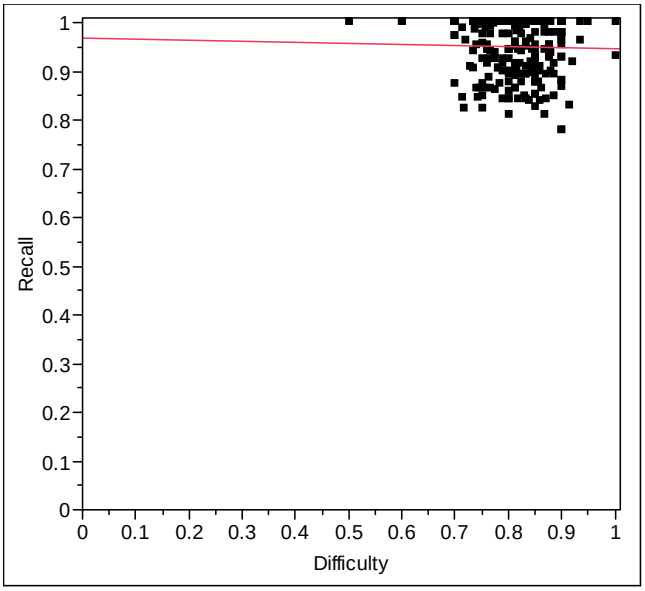

Figure 14. Recall of the optimal combination of NLP techniques related to difficulty.

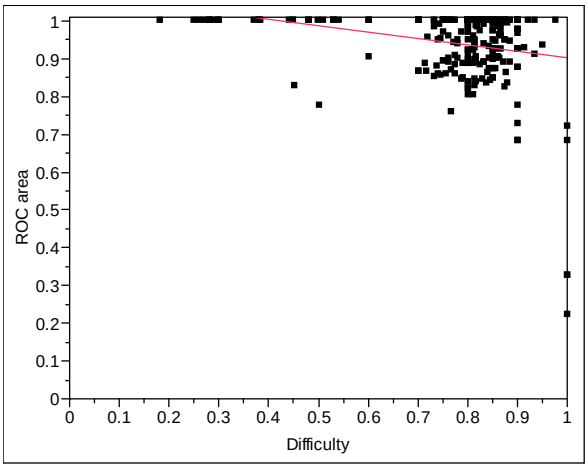

Figure 15 ROC area of the optimal combination of NLP techniques related to difficulty

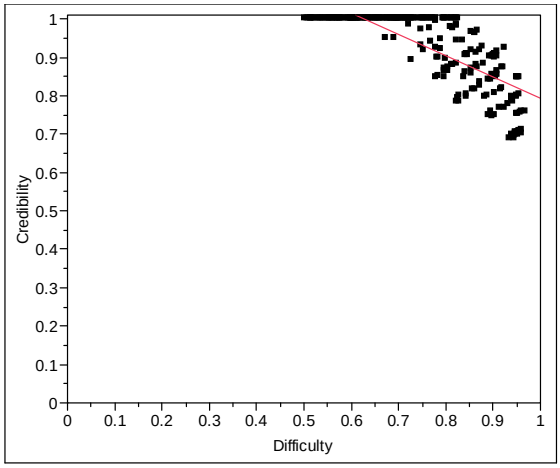

Figure 16 Credibility of the optimal combination of NLP techniques related to difficulty

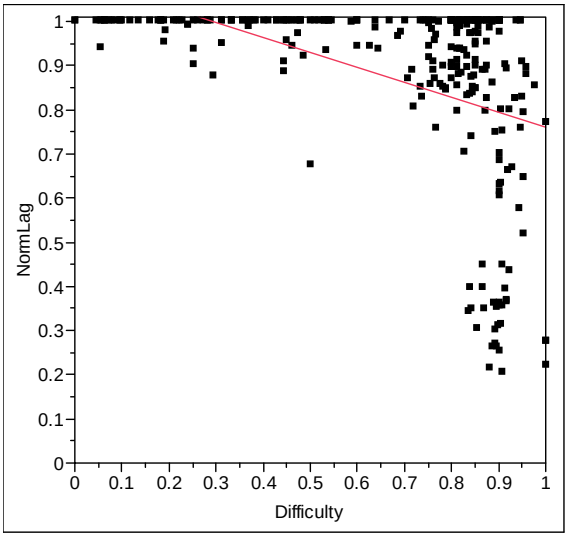

Figure 17 NormLag of the optimal combination of NLP techniques related to difficulty

4.3.5 RQ 5: Do the NLP techniques actually differ from one another?

4.3.5.1 Results

Figure 18 and Table 15 describe the results of applying PCA to NLP techniques; in our case, components refer to NLP techniques.

For all the components required to adequately represent the total variance (eigenvalue > 1.0), Table 15 reports the value of the eigenvalues (first column), the rank in descending order (second column), the percentage of variation captured (third column), and the cumulative percentage for a given number of factors (fourth column). Therefore, the first row indicates the maximum amount of variance (i.e., the highest eigenvalue) of the variables that can be explained by a single underlying component. The number of factors with eigenvalues higher than 1.0 represents the number of NLP techniques required to adequately represent the total variance of the variables; the number of factors with eigenvalues equal to or lower than 1.0 represents the number of NLP techniques that are redundant, i.e., they are equivalent to a given combination of other NLP techniques. Figure 18 describes the distribution of eigenvalues among components.

4.3.5.2 Discussion

As can be seen in Figure 18, most of the components contribute only minimally to the overall variance, having a very low eigenvalue. In particular, according to Table 15 (first and last columns), the first six factors account for more than half the total variance (Cum percent > 50). We also note that only 29 NLP techniques are non-redundant (eigenvalue > 1.0), i.e., 12% of the techniques provide useful information with respect to the others. This result should be analyzed by considering that the 242 NLP techniques are indeed different combinations of 18 variants.

4.3.6 Validity Discussion

The present section describes the major threats to validity as synthesized and structured by Wohlin et al. [77].

4.3.6.1 Conclusion Validity

As regards the *violated assumption of statistical tests,* our analysis is based on a large dataset, i.e., 983 observations per NLP technique. Regarding *low statistical power*, the resulting Z-values are so extremely low that they exclude the occurrence of any problem as far as statistical power is concerned.

Regarding *reliability of measures*, the adopted metrics are the state of the practice. In particular, we did not use the F-measure, i.e., a combination of precision and recall for a given threshold, for two main reasons:

1) Because this study is based on only one (optimal) threshold, the F-measure is neglected to avoid redundancy.

2) We used the ROC area metric which already takes into account a balance between precision and recall. Moreover, the ROC area has the advantage of being independent of the adopted threshold.

TABLE 15
PCA Results: Factors Required to adequately Represent the Total Variance (eigenvalue > 1.0)

Eigenvalue	Rank	Percent	Cum Percent
100.583	1	41.736	41.736
7.282	2	3.022	44.758
5.259	3	2.182	46.940
3.959	4	1.643	48.583
3.371	5	1.399	49.981
3.077	6	1.277	51.258
2.649	7	1.099	52.357
2.598	8	1.078	53.435
2.479	9	1.029	54.464
2.071	10	0.859	55.323
1.967	11	0.816	56.140
1.746	12	0.724	56.864
1.740	13	0.722	57.586
1.549	14	0.643	58.229
1.408	15	0.584	58.813
1.359	16	0.564	59.377
1.309	17	0.543	59.920
1.269	18	0.526	60.447
1.248	19	0.518	60.965
1.214	20	0.504	61.468
1.181	21	0.490	61.958
1.157	22	0.480	62.439
1.143	23	0.474	62.913
1.112	24	0.462	63.374
1.110	25	0.460	63.835
1.087	26	0.451	64.286
1.081	27	0.448	64.734
1.071	28	0.444	65.178
1.064	29	0.441	65.620
1.049	30	0.435	66.055

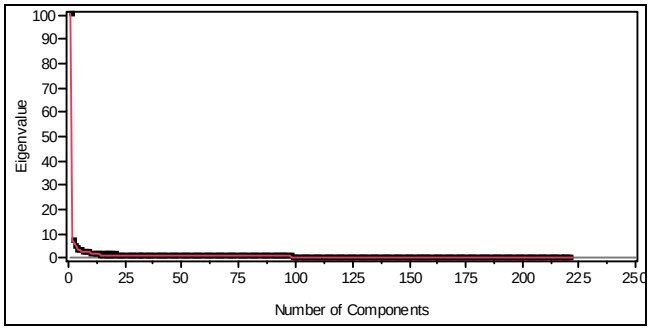

Figure 18. PCA results: distribution of eigenvalues among components (NLP technique combinations)

With regard to "selectivity" as proposed in [50], the use of adjusted scores already considers the improvement of an NLP technique over a random approach. The "average precision" metric can be defined as the precision computed with an optimal threshold which has the highest weight on false negative types of error. We discarded this metric because in the context of reuse the effort to spend on tracing artifacts needs to balance the effort reduction expected by the reuse strategy. Finally, a new metric, i.e. credibility [2], is presented and adopted for the first time.

4.3.6.2 Internal Validity

Due to the characteristic of the present case study, internal validity threats do not apply. This refers in particular to the following threats*: History, Maturation, Testing, Instrumentation, Statistical regression, Selection of subjects, Mortality,* and *Casual influence*.

4.3.6.3 Construct Validity

Regarding *mono-operation bias*, the adopted dataset contains a large sample from real industrial requirements. Additionally, the sampling strategy aims to trade off the time constraints and the generalizability of the requirements set (see [2]). As regards *mono-method bias*, we used at least five different metrics to answer each research question. Moreover, we analyze the results by adjusting the scores for the reasons described in Section 3.5. In order to deal with *restricted generalizability across constructs*, we drove the research questions according to a given objective function for the reasons described in Section 3.1. The adoption of such an objective function allowed an optimal threshold to be used for each NLP technique; this excluded the effect of *experimenter expectancies* on results. We note that Hayes et al. [50] proposed a way to compute TF that differs from that presented in Section 2. It consists in normalizing raw frequency by max(n(x,y)) (i.e., setting TF to 1 for the most frequent terms in the document). This implementation of TF is closer to the raw frequency, and may exhibit better behavior, making TF-IDF a better method than raw frequency. However, given the already large set of NLP techniques considered, we defer this analysis to future work.

4.3.6.4 External Validity

Because the adopted dataset consists of requirements that are up to date and belong to real industrial systems, there should be a low level of threat on *interaction of setting and treatment*. Additionally, the enacted cross-validation enhanced this kind of validity. Moreover, neither the *interaction of history and treatments* nor the *interaction of selection and treatment* apply to the case study. Finally, because the analyzed text belongs to a single company, the vocabulary is likely to be clear, concise, and specific. This assumption may not hold if the NLP is applied across several companies/countries.

5 Conclusions and Future Work

Because many NLP techniques exist and their performance varies according to context, it is crucial to define and use reliable evaluation procedures. In this paper we conjectured, and assessed, that NLP techniques, identifying equivalent requirements, perform on a given dataset according to both ability and the odds of making correct identification. For instance, when the odds of retrieving the right category are very high, then it is reasonable to expect that NLP techniques will result in good performance. In this paper we proposed a set of seven principles for evaluating the performance of NLP techniques in identifying equivalent requirements. To support the application of the principles we reported their practical application to a case study that evaluated the performance of a large number of NLP techniques for identifying equivalent requirements in the context of an Italian company in the defense and aerospace domain.

The current application context was specifically the evaluation of NLP techniques to identify equivalent requirements. However, most of the proposed principles seem applicable to evaluate any estimation technique to support a binary decision (e.g. equivalent/non-equivalent), with the estimate in the range [0,1] (e.g. the similarity provided by the NLP), when a dataset(s) is used as a benchmark (i.e. test bed), regardless of the type of estimator (i.e. requirements text) and estimation method (e.g. NLP).

In the future, we will steer our research towards context-dependency. We argue that the performance of NLP techniques depends upon the type of text to work with; i.e. the sublanguage characterizing the dataset. It would therefore be advisable to try out the different NLP techniques on legacy data to make an informed decision on which NLP technique to adopt without succumbing to the appeal of state-of-the-art techniques. Indeed, since there are different NLP techniques, it is reasonable to select the one that works well on legacy data. Another worthwhile approach is to combine NLP techniques. Several problems remain unresolved, including the quantity of data required to select the right NLP technique and how to validate the training data.

Regarding the empirical procedure to evaluate NLP techniques, we will investigate different ways to measure several aspects of difficulty and to mitigate its impact on the results. We will also investigate the sensitivity to difficulty of different performance metrics.

Acknowledgement

We thank the several reviewers that provided hundreds of useful and rigorous comments; these significantly contributed to improving the quality of the paper. We would like to thank Lionel Claude Briand for his valuable suggestions in the early phase of this work. The authors thank the Engineering Department of SELEX SI for their insightful interactions and support provided, especially Emanuela Barbi and Vincenzo Sabbatino. We are grateful to Carlo Ieva for developing ANTARTICA (section 4.2.3.1) and Alessandro D'Angiò for implementing the

simulation (section 4.3). This work was partially supported by SELEX SI under the research grant: SSI-DISP/07/08.

References

[1] J. Natt och Dag, B. Regnell, P. Carlshamre, M. Andersson, and J. Karlsson, "A Feasibility Study of Automated Support for Similarity Analysis of Natural Language Requirements in Market-Driven Development," *Requirements Engineering journal,* vol. 7, 2002.

[2] D. Falessi, L. C. Briand, and G. Cantone, "The Impact of Automated Support for Linking Equivalent Requirements Based on Similarity Measures," *Simula Research Laboratory Technical Report 2009-08,* 2009.

[3] http://www.selex-si.com/SelexSI/EN/index.sdo.

[4] N. Niu and S. Easterbrook, "On-Demand Cluster Analysis for Product Line Functional Requirements," presented at the Proceedings of the 2008 12th International Software Product Line Conference, 2008.

[5] E. Stierna and N. Rowe, "Applying information-retrieval methods to software reuse: a case study," *Inf. Process. Manage.,* vol. 39, pp. 67-74, 2003.

[6] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler, "An Exploratory Study of Information Retrieval Techniques in Domain Analysis," presented at the Proceedings of the 2008 12th International Software Product Line Conference, 2008.

[7] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell, "Speeding up Requirements Management in a Product Software Company: Linking Customer Wishes to Product Requirements through Linguistic Engineering," presented at the Proceedings of the Requirements Engineering Conference, 12th IEEE International, 2004.

[8] N. Niu and S. Easterbrook, "Extracting and Modeling Product Line Functional Requirements," presented at the Proceedings of the 2008 16th IEEE International Requirements Engineering Conference, 2008.

[9] P. Clements and L. Northrop, *Software Product Lines: Practice and Patterns*. Boston: Addison-Wesley, 2002.

[10] I. John, "Capturing Product Line Information from Legacy User Documentation " in *Software Product Lines*, T. Kakola and J. C. Duenas, Eds., ed: Springer Berlin 2006, pp. 127-159.

[11] G. Canfora and L. Cerulo, "A Taxonomy of Information Retrieval Models and Tools," *Journal of Computing and Information Technology,* vol. 12, 2007.

[12] A. De Lucia, F. Fasano, R. Oliveto, and G. Tortora, "Recovering traceability links in software artifact management systems using information retrieval methods," *ACM Transaction on Software Engineering Methodologies,* vol. 16, p. 13, 2007.

[13] C. Manning, P. Raghavan, and H. Schtze, *Introduction to Information Retrieval*: Cambridge University Press, 2008.

[14] A. De Lucia, R. Oliveto, and G. Tortora, "Assessing IR-based traceability recovery tools through controlled experiments," *Empirical Software Engineering; An International Journal,* vol. 14, pp. 57-92, 2009.

[15] K. Pohl, G. Böckle, and F. van der Linden, *Software Product Line Engineering: Foundations, Principles and Techniques* Springer, 2005.

[16] C. Manning and H. Schuetze, *Foundations of Statistical Natural Language Processing*: The MIT Press, 2000.

[17] K. Ryan, "The role of natural language in requirements engineering," presented at the Proceedings of IEEE International Symposium on Requirements Engineering, 1993.

[18] D. Binkley and D. Lawrie, "Information Retrieval Applications in Software Maintenance and Evolution," in *Encyclopedia of Software Engineering*, e. P. Laplante, Ed., ed: Taylor & Francis LLC, 2010.

[19] B. Güldali, H. Funke, M. Jahnich, S. Sauer, and G. Engels, "Semi-automated Test Planning for e-ID Systems by Using Requirements Clustering," presented at the Proceedings of the 2009 IEEE/ACM International Conference on Automated Software Engineering, 2009.

[20] W. B. Frakes and B. A. Nejmeh, "Software reuse through information retrieval," *SIGIR Forum,* vol. 21, pp. 30-36, 1987.

[21] Y. S. Maarek, D. M. Berry, and G. E. Kaiser, "An Information Retrieval Approach for Automatically Constructing Software Libraries," *IEEE Transactions on Software Engineering,* vol. 17, pp. 800-813, 1991.

[22] T. Yue, L. Briand, and Y. Labiche, "A systematic review of transformation approaches between user requirements and analysis models," *Requirements Engineering,* pp. 1-25, 2010.

[23] B. Paech and C. H. Martell, *Innovations for Requirement Analysis. From Stakeholders' Needs to Formal Designs: 14th Monterey Workshop 2007, Monterey, CA, USA, September 10-13, 2007. Revised Selected Papers*: Springer-Verlag, 2008.

[24] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, "Requirements for tools for ambiguity identification and measurement in natural language requirements specifications," *Requir. Eng.,* vol. 13, pp. 207-239, 2008.

[25] C. Francis, B. Nuseibeh, A. de Roeck, and A. Willis, "Identifying Nocuous Ambiguities in Natural Language Requirements," 2006, pp. 59-68.

[26] H. Yang, A. Willis, A. D. Roeck, and B. Nuseibeh, "Automatic detection of nocuous coordination ambiguities in natural language requirements," presented at the Proceedings of the IEEE/ACM international conference on Automated software engineering, Antwerp, Belgium, 2010.

[27] J. H. Weber-Jahnke and A. Onabajo, "Finding Defects in Natural Language Confidentiality Requirements," presented at the Proceedings of the 2009 17th IEEE International Requirements Engineering Conference, RE, 2009.

[28] K. Lauenroth and K. Pohl, "Towards automated consistency checks of product line requirements specifications," presented at the Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, Atlanta, Georgia, USA, 2007.

[29] J. Savolainen and J. Kuusela, "Consistency Management of Product Line Requirements," presented at the Proceedings of the Fifth IEEE International Symposium on Requirements Engineering, 2001.

[30] A. De Lucia, R. Oliveto, and P. Sgueglia, "Incremental Approach and User Feedbacks: a Silver Bullet for Traceability Recovery," presented at the Proceedings of the 22nd IEEE International Conference on Software Maintenance, 2006.

[31] H. Sultanov and J. H. Hayes, "Application of Swarm Techniques to Requirements Engineering: Requirements Tracing," 2010.

[32] S. K. Sundaram, J. H. Hayes, A. Dekhtyar, and E. A. Holbrook, "Assessing traceability of software engineering artifacts," *Requirements Eng. Journal,* vol. in press, 2010.

[33] C. Duan and J. Cleland-Huang, "Clustering support for automated tracing," presented at the Proceedings of the twenty-second IEEE/ACM international conference on Automated software engineering, Atlanta, Georgia, USA, 2007.

[34] X. Zou, R. Settimi, and J. Cleland-Huang, "Term-based Enhancement Factors for Improving Automated Requirement Trace Retrieval," presented at the ACM International Symposium on Grand Challenges of Traceability, 2007.

[35] M. Lormans and A. van Deursen, "Can LSI help Reconstructing Requirements Traceability in Design and Test?," presented at the Proceedings of the Conference on Software Maintenance and Reengineering, 2006.

[36] E. A. Holbrook, J. H. Hayes, and A. Dekhtyar, "Toward Automating Requirements Satisfaction Assessment," in *Requirements Engineering Conference, 2009. RE '09. 17th IEEE International*, 2009, pp. 149-158.

[37] M. Di Penta, S. Gradara, and G. Antoniol, "Traceability Recovery in RAD Software Systems," presented at the Proceedings of the 10th International Workshop on Program Comprehension, 2002.

[38] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability," presented at the Proceedings of the 13th IEEE International Conference on Requirements Engineering, 2005.

[39] A. Marcus and J. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," presented at the Proceedings of the 25th International Conference on Software Engineering, Portland, Oregon, 2003.

[40] D. Poshyvanyk, Y.-G. Gueheneuc, A. Marcus, G. Antoniol, and V. Rajlich, "Feature Location Using Probabilistic Ranking of Methods Based on Execution Scenarios and Information Retrieval," *IEEE Transactions on Software Engineering,* vol. 33, pp. 420-432, 2007.

[41] G. Antoniol, A. Cimitile, and G. Casazza, "Traceability Recovery by Modeling Programmer Behavior," presented at the Proceedings of the Seventh Working Conference on Reverse Engineering (WCRE'00), 2000.

[42] G. Antoniol, G. Canfora, G. Casazza, A. De Lucia, and E. Merlo, "Recovering Traceability Links between Code and Documentation," *IEEE Transactions on Software Engineering,* vol. 28, pp. 970-983, 2002.

[43] E. J. Uusitalo, M. Komssi, M. Kauppinen, and A. M. Davis, "Linking Requirements and Testing in Practice," presented at the Proceedings of the 2008 16th IEEE International Requirements Engineering Conference, 2008.

[44] J. H. Hayes, A. Dekhtyar, and D. S. Janzen, "Towards traceable test-driven development," presented at the Proceedings of the 2009 ICSE Workshop on Traceability in Emerging Forms of Software Engineering, 2009.

[45] Yadla S., J. H. Hayes, and A. Dekhtyar, "Tracing requirements to defect reports: an application of information retrieval techniques," *A NASA Journal, Information Systems Software Engineering,* 2005.

[46] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of Duplicate Defect Reports Using Natural Language Processing," presented at the Proceedings of the 29th international conference on Software Engineering, 2007.

[47] X. Wang, L. Zhang, T. Xie, J. Anvik, and J. Sun, "An approach to detecting duplicate bug reports using natural language and execution information," presented at the International Conference on Software engineering, Leipzig, Germany, 2008.

[48] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia, "Identifying the Starting Impact Set of a Maintenance Request: A Case Study," presented at the Proceedings of the Conference on Software Maintenance and Reengineering, 2000.

[49] J. H. Hayes and A. Dekhtyar, "A Framework for Comparing Requirements Tracing Experiments," *International Journal on Software Engineering and Knowledge Engineering,* vol. 15, 2005.

[50] J. H. Hayes, A. Dekhtyar, and S. K. Sundaram, "Advancing candidate link generation for requirements tracing: the study of methods," *IEEE Transactions on Software Engineering,* vol. 32, pp. 4-19, 2006.

[51] K. Kang, S. Cohen, J. Hess, W. Novak, and A. Peterson, "Feature-Oriented Domain Analysis (FODA) Feasibility Study," Technical Report CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.

[52] C. Fellbaum, *WordNet: An Electronic Lexical Database*: The MIT Press, 1998.

[53] X.-Y. Liu, Y.-M. Zhou, and R.-S. Zheng, "Measuring Semantic Similarity in Wordnet," presented at the Machine Learning and Cybernetics, 2007 International Conference on, 2007.

[54] A. De Lucia, R. Oliveto, and P. Sgueglia, "Incremental Approach and User Feedbacks: a Silver Bullet for Traceability Recovery?," presented at the Proceedings of the 22nd IEEE International Conference on Software Maintenance, 2006.

[55] P. Sawyer, P. Rayson, and K. Cosh, "Shallow Knowledge as an Aid to Deep Understanding in Early Phase Requirements Engineering," *IEEE Transactions on Software Engineering,* vol. 31, pp. 969-981, 2005.

[56] D. Blei, Andrew Ng, and M. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research,* vol. 3, 2003.

[57] J. Chang and D. Blei, "Relational Topic Models for Document Networks " *Artificial Intelligence and Statistics*, 2009.

[58] G. Capobianco, A. De Lucia, R. Oliveto, A. Panichella, and S. Panichella, "On the role of the Nouns in {IR}-based Traceability Recovery," presented at the Proc. of the Int'l Conf. on Program Comprehension, 2009.

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

[60] D. Hull, J. Pedersen, and Hinrich Schutze, "Method combination for document filtering," presented at the Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval, Zurich, Switzerland, 1996.

[61] M. Porter, "An algorithm for suffix stripping," *Program (reprinted in Readings in Information Retrieval, Morgan Kaufmann, 1997),* vol. 14, pp. 130-137, 1980.

[62] K. Church and W. A. Gale, "Inverse Document Frequency (IDF): A Measure of Deviations from Poisson," presented at the Proceedings of the Third Workshop on Very Large Corpora, 1995.

[63] K. Sparck Jones, "A statistical interpretation of term specificity and its application in retrieval," in *Document retrieval systems*, ed: Taylor Graham Publishing, 1988, pp. 132-142.

[64] S. Robertson, "Understanding Inverse Document Frequency: On theoretical arguments for IDF," *Journal of Documentation,* vol. 60, 2004.

[65] B. McCune, J. Grace, and D. Urban, *Analysis of Ecological Communities*: MjM Software Design, 2002.

[66] R. A. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval*: Addison-Wesley Longman Publishing Co., Inc., 1999.

[67] W. B. Frakes and R. Baeza-Yates, *Information retrieval: data structures and algorithms*: Prentice-Hall, Inc., 1992.

[68] P. Resnik, "Using information content to evaluate semantic similarity," presented at the Proceeding of the 14th International Joint Conference on Artificial Intelligence, Montreal, 1995.

[69] D. Lin, "An information-theoretic definition of similarity," presented at the Proceeding of the 15th International Conference on Machine Learning, 1995.

[70] J. Jiang and D. Conrath, "Semantic similarity based on corpus statistics and lexical taxonomy," presented at the Proceeding of International Conference on Research in Computational Linguistics, Taiwan, 1997.

[71] G. Pirró and N. Seco, "Design, Implementation and Evaluation of a New Semantic Similarity Metric Combining Features and Intrinsic Information Content " presented at the On the Move to Meaningful Internet Systems: OTM 2008, 2008.

[72] C. Corley and R. Mihalcea, "Measuring the semantic similarity of texts," in *ACL Workshop on Empirical Modeling of Semantic Equivalence and Entailment (EMSEE '05)*, Stroudsburg, PA, USA, 2005.

[73] X. Zou, R. Settimi, and J. Cleland-Huang, "Improving automated requirements trace retrieval: a study of term-based enhancement methods " *Empirical Software Engineering,* vol. 15, 2009.

[74] J. Cleland-Huang, R. Settimi, O. BenKhadra, E. Berezhanskaya, and S. Christina, "Goal-centric traceability for managing non-functional requirements," presented at the Proceedings of the 27th international conference on Software engineering, St. Louis, MO, USA, 2005.

[75] I. Witten and E. Frank, *Data Mining: Practical Machine Learning Tools and Techniques*, 2005.

[76] D. Falessi, M. Ali Babar, G. Cantone, and P. Kruchten, "Applying Empirical Software Engineering to Software Architecture: Challenges and Lessons Learned," *Empirical Software Engineering,* vol. 15, pp. 250-276, 2010.

[77] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, and A. Wesslen, *Experimentation in Software Engineering: an Introduction*: Springer, 2000.

[78] A. Dekhtyar, J. H. Hayes, and J. Larsen, "Make the Most of Your Time: How Should the Analyst Work with Automated Traceability Tools," in *Third Int'l Workshop Predictive Modeling in Software Eng. (PROMISE '07)*, 2007.

[79] X. Zou, R. Settimi, J. Cleland-Huang, and C. Duan, "Thresholding Strategy in Requirements Trace Retrieval," presented at the CTI Research symposium, 2004.

[80] C. J. van Rijsbergen, *Information Retrieval*. London, 1979.

[81] D. T. Larose, *Data Mining Methods and Models*: John Wiley \& Sons, Inc., 2007.

[82] http://www.cs.waikato.ac.nz/~ml/weka/.

[83] A. Dekhtyar, J. Hayes, and G. Antoniol, "Benchmarks for Traceability?," presented at the International Symposium on Grand Challenges in Traceability (GCT'07/TEFSE'07), Lexington, KY, 2007.

[84] B. Kitchenham, "A Procedure for Analyzing Unbalanced Datasets," *IEEE Transactions on Software Engineering,* vol. 24, pp. 278-301, 1998.

[85] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," presented at the Proceedings of the 32nd ACM/IEEE International Conference on Software Engineering - Volume 1, Cape Town, South Africa, 2010.

[86] M. Gibiec, A. Czauderna, and J. Cleland-Huang, "Towards mining replacement queries for hard-to-retrieve traces," presented at the Proceedings of the IEEE/ACM international conference on Automated software engineering, Antwerp, Belgium, 2010.

[87] M. Stevenson and Y. Wilks, "The interaction of knowledge sources in word sense disambiguation," *Comput. Linguist.,* vol. 27, pp. 321-349, 2001.

[88] D. J. Sheskin, *Handbook of Parametric and Nonparametric Statistical Procedures*: Chapman \& Hall/CRC, 2007.

[89] N. Mittas and L. Angelis, "Comparing cost prediction models by resampling techniques," *Journal of Systems and Software,* vol. 81, pp. 616-632, 2008.

[90] F. Yates, "Contingency tables involving small numbers and the $\chi 2$ test," *Journal of the Royal Statistical Society,* Supplement 1, 1934.

[91] P. M. DeLuca, A. Wambersie, and F. G. Whitmore, "Receiver Operating Characteristic Analysis in Medical Imaging," *Journal of the ICRU,* vol. 8, p. 3, 2008.

[92] J. A. Wass, "Comparative Statistical Software Review," *http://www.scientificcomputing.com/comparative-statistical-software.aspx*.

[93] A. B. Kitchenham, E. Mendes, and H. G. Travassos, "Cross versus Within-Company Cost Estimation Studies: A Systematic Review," *IEEE Transactions on Software Engineering,* vol. 33, pp. 316-329, 2007.

[94] B. Kitchenham and E. Mendes, "Why comparative effort prediction studies may be invalid," presented at the Proceedings of the 5th international workshop on Predictor models in software engineering, co-located with ICSE2009, Vancouver, Canada, 2009.

[95] J. Miller, "Statistical significance testing: a panacea for software technology experiments?," *J. Syst. Softw.,* vol. 73, pp. 183-192, 2004.

[96] J. Miller, J. Daly, M. Wood, M. Roper, and A. Brooks, "Statistical power and its subcomponents -- missing and misunderstood concepts in empirical software engineering research," *Information and Software Technology,* vol. 39, pp. 285-295, 1997.

[97] J. Cohen, *Statistical Power Analysis for the Behavioral Sciences*: Psychology Press, 1988.

[98] G. Dunteman, *Principal Component Analysis*: SAGE Publications, 1989.

[99] L. C. Briand, Jurgen Wüst, J. W. Daly, and D. V. Porter, "Exploring the relationship between design measures and software quality in object-oriented systems," *Journal of Systems and Software,* vol. 51, pp. 245-273, 2000.

[100] L. C. Briand, J. Wust, and H. Lounis, "Replicated Case Studies for Investigating Quality Factorsin Object-Oriented Designs," *Empirical Software Engineering,* vol. 6, pp. 11-58, 2001.

[101] E. Arisholm, L. Briand, and A. Foyen, "Dynamic Coupling Measurement for Object-Oriented Software," *IEEE Transactions on Software Engineering,* vol. 30, pp. 491-506, 2004.

[102] S. Thompson, *Sampling*: Wiley-Interscience, 1992.

[103] D. G. Kleinbaum, L. L. Kupper, and K. E. Muller, *Applied regression analysis and other multivariable methods*. Bostono, MA, USA: PWS Publishing Co., 1988.

[104] R. Kittredge and J. Lehrberger, *Sublanguage: Studies on Language in Restricted Semantic Domains*: Walter De Gruyter Inc, 1982.

[105] J. Lehrberger, "Sublanguage Analysis," in *Analyzing Language in Restricted Domains*, R. Grishman and R. Kittredge, Eds., ed: Psychology Press, 1986.

[106] TEFSE, "Grand Challenges of Traceability," *http://www.cs.wm.edu/semeru/tefse2011/Challenge.htm,* 2011.

[107] A. P. Sage and C. D. Cuppan, "On the Systems Engineering and Management of Systems of Systems and Federations of Systems," *Information-Knowledge-Systems Management,* vol. 2, pp. 325-345, 2001.

[108] P. Clements and C. Kreuger, "Point - Counterpoint: Being Proactive Pays Off - Eliminating the Adoption Barrier," *IEEE Software,* vol. 19, p. 4, 2002.

[109] J. Bosch, "On the Development of Software Product-Family Components," in *Proceedings of the Third International Conference on Software Product Lines*, R. L. Nord, Ed., ed Boston, MA, USA: Springer LNCS 3154, 2004.

[110] P. Clements, J. D. McGregor, and S. G. Cohen, "The Structured Intuitive Model for Product Line Economics (SIMPLE)," CMU/SEI-2005-TR-003, 2005.

[111] I. John, J. Knodel, T. Lehner, and D. Muthig, "A practical guide to product line scoping," presented at the Software Product Line Conference, 2006 10th International, 2006.

[112] J. Natt och Dag, V. Gervasi, S. Brinkkemper, and B. Regnell, "A Linguistic-Engineering Approach to Large-Scale Requirements Management," *IEEE Software,* vol. 22, pp. 32-39, 2005.

[113] "http://eseg.uniroma2.it/tools/ANTARCTICA/index.htm."

[114] J. Natt och Dag, T. Thelin, and B. Regnell, "An experiment on linguistic tool support for consolidation of requirements from multiple sources in market-driven product development," *Empirical Software Engineering,* vol. 11, pp. 303-329, 2006.

[115] B. Vinz and L. Etzkorn, "Comments as a Sublanguage: A Study of Comment Grammar and Purpose," in *International Conference on Software Engineering Research and Practice (SERP'08)*, Las Vegas, Nevada, USA, 2008.

[116] L. Etzkorn, C. Davis, and L. Bowen, "The Language of Comments in Computer Software: A Sublanguage of English," *Journal of Pragmatics,* vol. 33, 2001.

Davide Falessi is an adjunct research scientist in the Certus Software V&V Center at Simula Research Laboratory (Norway), and an adjunct lecturer at the Department of Informatics, Systems, and Production engineering (DISP) of the University of Rome "Tor Vergata". He currently serves as a program committee member in several international conferences and workshops including ESEM, WICSA, SPLC, ICSR, SEKE, PROFES, SHARK, and QMSPLE. His main research interest is in devising and empirically assessing scalable solutions for the development of complex software-intensive systems with a particular emphasis on architecture, requirements, and quality. He received the PhD and the "Laurea" degrees in Computer and Information Engineering from the University of Rome "TorVergata". Contact him at d.falessi@ieee.org.

Giovanni Cantone is a professor of experimental software engineering and software systems engineering in the DISP of the University of Rome "Tor Vergata". He organized and chaired the 1st Empirical Software Engineering Intl. Week, co-chaired the 2nd ACM-IEEE Symposium on Empirical Software Engineering, and organized 9th Intl. PROFES Conference. He is founding member and current member of the International Software Engineering Research Network. Cantone has been serving as a program committee member in several international conferences and workshops including CLEI, ESEM, ETSM, ESELAW, LSO, SAM, SEKE, SMEF, and WETSoM. He has been guest-editor of Real-time Systems and Advances in Software Engineering journals. Giovanni published more than 100 research papers; his current research interests include using experimentation and other empirical paradigms in software and system engineering. He received the "Laurea" degree in Electronic Engineering from the University of Naples Frederick the 2nd. Contact him at cantone@uniroma2.it.

Gerardo Canfora is a professor of computer science at the Faculty of Engineering of the University of Sannio, Italy. He serves on the program and organizing committees of a number of international conferences. He was general chair of WCRE'06 and CSMR'03, and program co-chair of WETSoM'10, ICSM'01 and ICSM'07, IWPSE'05, CSMR'04 and IWPC'97. Canfora is co-editor of the "Journal of Software: Evolution and Processes" (former: "Journal of Software Maintenance, Research and Practice); from 2000 to 2004 he was an associate editor for IEEE Transactions on Software Engineering. Canfora authored more than 150 research papers; his research interests include software maintenance and evolution, empirical software engineering, and service-oriented computing. He received the "Laurea" degree in Electronic Engineering from the University of Naples Frederick the 2nd. Contact him at canfora@unisannio.it.