A Comprehensive Characterization of NLP Techniques for Identifying Equivalent Requirements

Davide Falessi
Simula Research Laboratory (Norway)
and University of Rome TorVergata,
Viale del Politecnico 1, 00133 Rome, Italy
falessi@ieee.org

Giovanni Cantone
University of Rome TorVergata
Viale del Politecnico 1,
00133 Rome, Italy
cantone@uniroma2.it

Gerardo Canfora
RCOST – Research Centre on Software
Technology and University of Sannio
Via Traiano, 82100 Benevento, Italy
canfora@unisannio.it

Abstract. Though very important in software engineering, linking artifacts of the same type (clone detection) or of different types (traceability recovery) is extremely tedious, error-prone and requires significant effort. Past research focused on supporting analysts with mechanisms based on Natural Language Processing (NLP) to identify candidate links. Because a plethora of NLP techniques exists, and their performances vary among contexts, it is important to characterize them according to the provided level of support. The aim of this paper is to characterize a comprehensive set of NLP techniques according to the provided level of support to human analysts in detecting equivalent requirements. The characterization consists on a case study, featuring real requirements, in the context of an Italian company in the defense and aerospace domain. The major result from the case study is that simple NLP are more precise than complex ones.

1 INTRODUCTION

1.1 Context

Linking artifacts, of the same type (clone detection) or of different types (traceability recovery), is a common practice in many software engineering tasks, including verification and validation, reuse, maintenance, and reverse engineering. A relevant 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].

1.2 Study Motivation

Experience from applying Product Line Engineering shows that investing in reuse does not always pay off; a success factor is applying the right reuse strategy with the right amount of investment on the right assets (e.g. components, test cases, requirements) without underestimating non-technical aspects (e.g. current organization)[3]. When SELEX Sistemi Integrati, an Italian company in the defense and aerospace domain, started the development of five different products (systems of systems), it was clear to us the opportunity to take advantage from the products commonalities. In order to reason on what and how to reuse and to design a reference architecture [4, 5], it is mandatory to have an understanding of the variability and commonalities of the systems to develop. This activity is called domain analysis [3]. Among the several ways to enact domain analysis, linking equivalent requirements is particularly appealing as it is non-invasive for the actual organization, is scalable, and supports requirements consolidation [1, 6]. With these motivations for detecting equivalent requirements, we note that for large systems it is impossible for a human alone to manually analyze all the feasible combination of requirements pairs for a potential link. As a matter of fact, given a total number of 2,500 requirements, the number of (feasible) combination of requirements pairs to analyze is 3,123,750 (i.e., 2500!/(2500-2)!). NLP can help human in detecting equivalent requirements. In particular, NLP is used to measure the similarity among requirements; similarity is then used to rank/filter the requirement pairs according to their likelihood of being redundant. Ideally, an NPL applied to a software engineering task should reveal all and only relevant traces; in practice, every NPL will reveal some useless traces, while missing some actual ones. In fact, a plethora of NLP techniques exists and different NLP perform differently in different contexts [7]. Therefore, it is important to characterize them according to the provided level of support. The paper presents a case study characterizing several NLP techniques for identifying equivalent requirements.

ESEM'10, September 16-17, 2010, Bolzano-Bozen, Italy.

1.3 Aim and Research Questions

The aim of this paper is to characterize a set of NLP techniques according to the level of support they provide to human analysts in detecting equivalent requirements. In particular, we will focus on the following research questions:

R.Q. 1 *Which is the best NLP? What is the rank of the available NLP techniques?*

This question aims at detecting the best NLP among the available ones i.e., the NLP that, if adopted in a traceability tool, would provide the maximum benefit to the human analyst compared to other NLP. It also aims at discovering if there is a *dominant* NLP i.e., a NLP that outperforms the others in all the performance metrics.

R.Q. 2 How does the best NLP perform? Is there space for improvement? How far are we from the ideal performances?

It is important to reveal the actual limitation in the support provided by the best NLP; this highlights the current lacks, which in turn drives future research.

1.4 Structure

The paper is structured as follows: Section 2 describes the background of the study. Section 3 proposes a taxonomy of NLP techniques that will be characterized in the remainder of the paper. Section 4 describes the applied empirical procedure and

Section 5 the related results and discussion. Section 6 discusses validity limitations and Section 7 concludes the paper.

2 BACKGROUND

2.1 Terminology

NLP=IR=technique. 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)." [8] Natural Language Processing (NLP) is a field of computer science and linguistics that is concerned with adopting machine to parse the human language for a given purpose. In other words, the 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 techniques and NLP techniques coincide. In this paper, the term NLP refers to the technique adopted to provide the similarity between two pieces of text.

Variation points and variants. NLP can differ in several aspects and dimensions. In order to effectively analyze such differences we adopted the terminology from Software Product Line Engineering [3]. In particular, the term *variation point* refers to a given type of variability among NLP 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 (i.e., similarity metric); the variants are, for instance, the Jaccard and the Cosine metrics.

2.2 Related Work

NLP have been applied in different areas of software engineering from a long time ago [9]. Most studies adopted NLP to trace artifacts at different levels of abstraction including high level to low level requirements [10, 11], requirements to design [12], requirements to source code [13], high-level features to their implementation [14], functional requirements to Java components [15], requirements to defect reports [16, 17], and change requests to the impacted software modules [18]. Such studies differ from the present one in the fact that we try to identify equivalence among the same type of artifacts; i.e. duplicates. Runeson et al. in [17] achieved promising results in adopting NLP techniques to detect equivalent defect reports.

The present study has been mainly inspired by Natt och Dag et al. [1], who compare three different similarity metrics to assess the feasibility of 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 in the same abstraction level.
2. Semantic of link: the links between requirements are equivalences.
3. Vision: "some approaches seem promising but we believe that more effort need to be put into this field to reach consensus on which methods, techniques, approaches and tools may be appropriate for different type 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 future works recommendation: "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 main differences among the present study and the past are the number of compared NLP (242), and the use of an objective function and cross validation (see Section 4.4) to compare NLP performances.

3 A TAXONOMY OF NLP

In this section we introduce a taxonomy that reveals the variability and commonalities of available NLP techniques. Canfora and Cerulo [19] 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. The taxonomy we use in this paper comprises four dimensions, where each dimension is a variation point. The four variation points that compose our taxonomy are: *Algebraic models*, *Term extraction*, *Weighing schema*, and *Similarity metric*. According to this taxonomy, a specific NLP technique is viewed as a point in a four-dimensional space; in other words, a specific combination of four variants (one for each variation point) indentifies a given NLP. Figure 1 is novel and describes the proposed taxonomy by using a feature model [3].

The proposed taxonomy has some limitations in describing the available NLP, as any attempt to characterize a complex issue. One of the main limitations of the taxonomy is that it doesn't take into account to the "feedback" variant as proposed in [11]. This is because we want to assess the different NLP techniques, and feedback is something that can be added on top of any technique. Despite such minor weaknesses, the proposed taxonomy, compared with past studies, explains the variability among the largest number of NLP (No. 242). The following sections describe the main characteristics of the analyzed NLP. We examine variation points in Figure 1 by depth, i.e., by reviewing all the variants for each variation point. Accordingly, the names of a variation points are used as the title of sections, while the names of the related variants are printed in italic.

3.1 Algebraic Models

Vector Space Model (VSM): VSM adopts the underlying metaphor of using spatial proximity for semantic proximity; such a conceptual simplicity makes it the most adopted algebraic model in practice. 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 Weighing schema in Section 3.3. Once the texts are represented in vectors, their similarity is computed in terms of distances in such vector space.

Latent Semantic Analysis (LSA): LSA was developed during the '80 with the name *LSI* (Latent Semantic Indexing) as an evolution of the traditional VSM. The reason for such an evolution is that VSM doesn't take into consideration the synonymies among terms; in other words, despite "vessel" and "yacht" have a similar meanings, VSM is not able 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 terms occur in the same documents more often than chance.

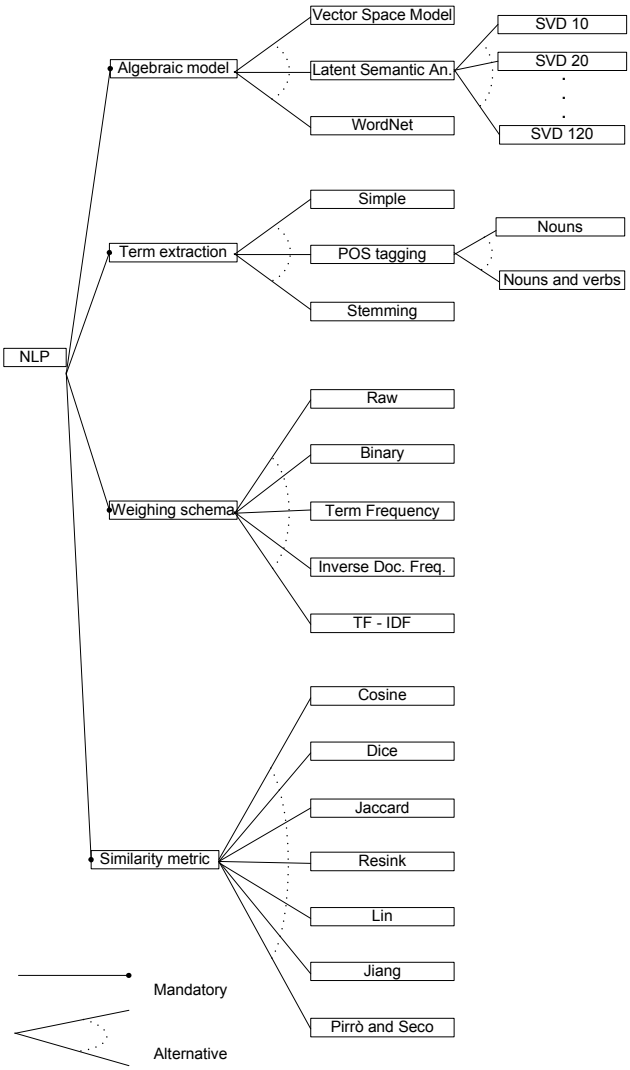

Figure 1 A taxonomy of NLP techniques.

Without going in details, we stress that LSA, given a corpus of text fragments as input, provides as output a sort of thesaurus where the similarity among words are expressed in a ratio scale. Such 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 in three matrices: a document by concept matrix, a term-by-concept matrix, and a diagonal matrix. Then it applies *Singular Value Decomposition (SVD)* to project the original term-by-document matrix into a reduced space in order to reduce the "noise" in term usage.

[1] For instance, despite the terms "monitor" and "screen" are commonly used as synonymies, they resulted as un-similar by LSA, using a given corpus of systems specifications of SELEX SI projects; this reveals that, in the application domain of SELEX SI, the term "monitor" is mostly used in the sense of "supervise" rather than "screen".

Defining the right size of the space resulting from the SVD is a hard decision because the level of details included in the space should be not too much to include noise and not too small to exclude valuable information. In our case study, we let the value range between 0 and 120 with an interval of 10.

Thesaurus-based Similarity: in this approach a thesaurus with a given set of synonyms is adopted to compare terms. WordNet [20] is the most famous example for the English language; it groups English words into sets of synonyms called synsets, provides short, general definitions, and records the various semantic relations between these synonym sets. The purpose is twofold: to produce a combination of dictionary and thesaurus that is intuitively usable, and to support automatic text analysis and artificial intelligence applications. Regarding LSA versus WordNet, if from one side LSA is domain specific, the WordNet is ready to use and richer of relations; it is therefore important to empirically evaluate which algebraic model is more suitable in the requirements engineering domain.

3.2 Term Extraction

The text fragments analyzed for similarity needs to be pre-processed before any comparison can take place.

Simple: the simplest way to preprocess the text is to apply to it a tokenization and a stop-word removal activity. During the tokenization activity, the text is transformed in a series of tokens where capitals, punctuation, and brackets are removed. Afterwards, the token are analyzed and stop words are removed. The stop words are the terms that do not contribute to the semantic of the text; examples include articles, pronouns, etc. The list of stop words needs to be defined before the text preprocessing takes place and it obviously depends from the adopted language.

Part Of Speech (POS) tagging. After the tokenization and stop word removal activities, each token is classified according to the part of the speech, i.e., the role is plays in a given text. After the POS tagging activity, each token is hence associated with a tag which may indicate that such a token is a verb, noun, adjective, etc. Afterwards, 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 the text having nouns in common will have a higher similarity measures than the ones having adjectives in common. In general, it is unknown if nouns are more important than adjectives or if such a kind of difference makes any sense for the problem at hand. It is therefore important to analyze if it pays off to enact such a POS tagging activity and, in case, which weight to assign to the different tags. We adopt two types of tagging; one where only nouns are considered and another where only nouns and verbs are considered.

Stemming: After the POS tagging activity, the tokens can be converted in his morphological stems. For example, "monitoring", "monitor", "monitored", and "monitors" are all converted in the token "monitor". The advantage provided by stemming the words is mitigating the influence of morphological variants on similarity measure. Moreover, the stemming activity is based on rules or statistics and its application to neo-Latin languages is quite complex. In the present paper we adopted as stemming mechanism the rule based algorithm that Martin Porter proposed in [21].

3.3 Term Weighting

The mechanisms to assign different weights to terms are based on the occurrences of a term into the analyzed fragments (see Table 1).

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

Binary: It assigns a weight of 0 in case the term doesn't occur and 1 vice versa.

Term Frequency (TF): It assigns a weight proportional to the frequency of the term occurrences 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 included. This approach aims to avoid that long text fragments provide terms with higher frequency but low semantic relevance.

Inverse Document Frequency (IDF): It assigns a weight depending on the number of given texts that include the term (rated to the total number of texts). Although IDF is strongly correlated with the inverse of TF, the two variables are not completely predictable from one another [22]. The underlying idea for IDF weighting is the observation that the documents related to a given domain share a lot of 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 few information in the documents in which it is used.

TF-IDF: it computes the weight as the multiplication of TF and IDF with the aim of achieving the benefits from 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 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.

Table 1: 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)	$TF(x,y) \times IDF(x)$

3.4 Similarity Metric

3.4.1 Vector similarity metrics

There are different ways to compute the distance of two vectors; they are called 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. 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"[23].

Dice: The Dice coefficient normalizes for length by dividing by the total number of non-zero entries. The scale factor 2 gives a measure that ranges from 0.0 to 1.0, with 1.0 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 [24].

Jaccard: The Jaccard coefficient penalizes a small number of shared entries (as a proportion of all non-zero entries) more than the Dice coefficient does. Both measures range from 0.0 (no overlap) to 1.0 (perfect overlap), but the Jaccard coefficient gives lower values to low-overlap cases.

Cosine: The cosine is identical to the Dice coefficient for vectors with the same number of non-zero entries but it penalizes less in cases where the numbers of non-zero entries are very different to 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 want to say they are dissimilar just because of that.

Table 2: Vector-similarity metrics.

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

3.4.2 WordNet similarity metrics

WordNet has its own set of metrics [25]. 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 described in Table 3, 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.

In the present paper we consider the following WordNet similarity metrics [25]: Resnik, Lin, Jiang, and Pirrò and Seco as described in Table 4 where:

- *IC* is the information content
- *LCS* is the least common subsume

[2] For instance, the similarity between the terms "monitor" and "screen" changes according to its direction. In fact, the term "screen" can be quite always substituted to "monitor" but not vice versa because "monitor" can have several further meaning like "observing".

- *W(x,y)* is the set of words included in x and y.

Table 3 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 4 WordNet similarity metrics.

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

4 METHOD

4.1 Context

Finmeccanica is a large Italian industrial group operating globally in the aerospace, defense, and security sectors, and is one of the world's leading groups in the fields of helicopters and defense electronics. It has revenues of 15 Billion Euros and invests 1.8 Billion Euros (12% of turnover) a year in R&D activities. SELEX Sistemi Integrati (also known as SELEX SI) is the Finmeccanica Company focusing on the design of systems of systems; it aims to be the European leader in the definition and integration of sensors and systems for defense, coastal/maritime surveillance, and air traffic management. The context of this study is the development of systems of systems. In addition to being large, distributed, adaptive and complex, a system of systems is structured into components (i.e. systems) that can work independently of each other, though their cooperation provides functionality that are greater than the sum of their functionalities. The requirements taken into account in this paper 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 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 Goal, Questions, and Metrics

4.2.1 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. The addressed research questions are described in Section 1.3.

4.2.2 Performance metrics

Research questions (see Section 1.3) concern the characterization of NLP in terms of performance in retrieving equivalent requirements. Since NLP can support the analyst in different ways, then the performance related to the above research questions is measured by adopting several metrics. In the following we describe three main types of support, the rationale, and the related metrics; Table 5 reports the formula of each performance metric.

- *Filtering*: As stated above, the number of possible requirements combinations is too large to consider the possibility that a human analyst can 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 a NLP technique to produce/retrieve an effective list of candidate traces we adopted *ROC area, Precision* and *Recall* metrics. We selected these metrics for two main reasons: i) they look at different aspects of performances, and ii) they are the best current practice.
- *Ranking:* Even if filtered, the number of available requirement pairs is extremely high; therefore, it is important that NLP support human in focus their attention on the requirement pairs that are more likely to be equivalent. The *Lag* metric, as introduced by Hayes et al. in [26], is a measure of ranking effectiveness and it represents, in the average, the number of actual non-equivalent requirement pairs that have a higher (measured) similarity than actual equivalent requirement pairs. In other words, it represents the number of non-equivalent requirement pairs that, in the average, a similarity-based tool would propose for analysis before any actual equivalent pair.
- *Suggesting:* When the human analyst needs to decide about the equivalence or not of a given requirement pair, the similarity measure provides a practical support in suggesting such a classification (equivalent vs. non-equivalent). The underlying principle is that when the (measured) similarity is low, then 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 showed similarity measure has a significant impact on quality and speed of human 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 the technique is credible the more it reduces the time to classify a given requirement pair and improves the quality of the classification. In order to compare the different NLP techniques we computed the credibility of each NLP on each requirement pair of the industrial dataset; afterwards, we computed the credibility of a given NLP as the average among the requirement pairs in the industrial dataset.

Table 5: Formulas of performance metrics

Type of support	Metric	Formula
Filtering	Precision	*(true positive) / (false positive + true positive)*
	Recall	*(true positive) / (false negative + true positive)*
	ROC area	*True postive rate vs. False Positive rate*
Ranking	Lag	*(Rank of the last positive - Number of positive) / Number of positive*
Suggesting	Cedibility	*Average of 1- |similarity measure – human link|*

4.3 Data Collection

Figure 2 describes the flow of data in the case study, from top-left, clockwise. The data collection was performed at SELEX SI and automated by means of a recent open source tool called PROUD[3] (Proactive Reuse OpportUnity Discovery tool) which automatically collects data; PROUD incorporates linguistic engineering techniques for supporting equivalent requirements identification [2]. In order to characterize the NLP, we developed a tool named ANTARCTICA[4] (Automatic characterizatioN of naTural lAnguage pRoCessing meThods for estimatIng semantiC similarity). ANTARCTICA works as a post mortem simulator; it takes as input the log file as produced by PROUD and provides as output a benchmark, i.e., the similarity measure, for each requirement pair in the dataset, for each of the NLP described in Section 3. Therefore, ANTARCTICA does not directly support the human analyst; however, it provides a useful means to understand which NLP would better support him. In order to pre-process the similarity measurements that ANTARCTICA produced, we adopted Weka[3]. The data were analyzed by means of a statistical commercial tool (JMP™). Finally, the case study results were packaged and provided to SELEX SI with the aim of improving the process of detecting equivalent requirements. The dataset adopted in this study coincides with [2]. In particular, Table 6 reports the main characteristics of our case study at SELEX SI regarding five different projects on systems of systems. In order to make the study manageable, we sampled a subset of 983 requirement pairs to study out of a population of nearly three million possible pairs. Among these 983 requirements pairs, 138 were actually equivalent. For further details about the dataset please see [2].

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

4.4 Measurement and Analysis Procedure

The common procedure to compare a set of NLP consists of a post mortem analysis according to a given dataset(s) where the underlying principle is that a pair of artifacts should be traced upon each other in case the measured similarity is higher than a given threshold. The dataset(s) contains a large set of pairs of artifacts; for each pair, the dataset contains two text fragments, one text per artifact (upon which the similarity is measured), and the "actual" value (i.e., the oracle). The latter describes if such a pair should be traced or not. Each NLP is applied to measure the similarity of each pair. Then, the "estimated" value is computed, for each NLP, by applying a threshold on the measured similarity of each pair. Finally, the NLP performance is computed, for each NLP, by comparing the actual and estimated values. The less the difference between the actual and estimated values, the higher the NLP performance; a given performance metric (e.g. Precision, Recall, etc.) measures a given aspect of difference between the actual and the estimated values.

[3] http://eseg.uniroma2.it/tools/PROUD/index.htm.

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

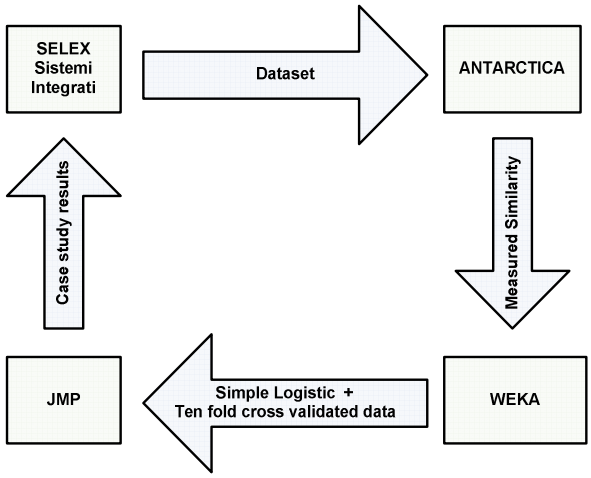

Figure 2. Flow of data from the data sampling to the results; clockwise from top-left.

4.4.1 Adoption of an Optimal Threshold

When comparing NLP techniques, a dataset is adopted, similarity measures are computed, and several thresholds are applied to define which artifacts are of interest. Comparing NLP according to their performance on an arbitrary set of thresholds may entail several problems including that the optimal threshold for a given NLP is not considered and hence the NLP performance is underestimated. For instance, in [1], the variants Dice and Cosine outperform Jaccard regarding "True positive rate", and vice versa regarding "False positive rate". Since such result is based on a limited set of thresholds (9 thresholds), it is possible that for a threshold not taken into account the Jaccard variant would outperform both Dice and Cosine, or vice versa.

We claim that it is important to compare the different NLP based on their performance according to their optimal threshold. One possible option to define the objective function is to assign a given weight to false positive and false negative errors; another option is to fix the desired amount of Recall. Once the objective function(s) is defined, computing the related optimal threshold, for each NLP, is a trivial task. Simple Logistic Regression [27] is a procedure that computes the linear equation that better predicts the dependent variable according to the independent variable. We can see the simple logistic regression as a procedure to find the optimal threshold (i.e. the coefficient in the resulting linear equation) among all the possible thresholds, for a given NLP, for a given objective function. Weka[5] is an example of open source tool that can run on most operative systems; in few seconds it can easily apply simple logistic regression even on large set of data, on standard computers.

4.4.2 Cross-validation of the NLP performance

If the optimal threshold is computed on a given dataset and hence the NLP performance is computed on the same dataset, thus the results tend to be optimistic [28]. The adoption of cross-validation is a standard approach for assessing prediction models in order to obtain realistic and general results. The underlying principle of cross-validation is to modifying the dataset on which the optimal threshold is computed by re-sampling randomly the whole

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

dataset. "The standard way of predicting the error rate of a learning technique given a single, fixed sample of data is to use stratified 10-fold cross-validation."[27]. According to our best understandings, cross-validation has not been adopted in any past comparative study on NPL techniques. Weka[5] has been adopted in this study to preprocess the data; in particular it has been used to detect the optimal threshold for each NLP by means of simple logistic regression and to cross validate the NLP performances by means of ten-fold cross validation (see Section 4.4).

5 RESULTS AND INTERPRETATION

5.1 R.Q. 1 Which is the best NLP? What is the rank of the available NLP techniques?

5.1.1 Results

The analysis procedure consists of computing, for each NLP, the following performance metrics: Precision, Recall, ROC, Lag, and Credibility. Since a table format was not suitable to rank the NLP given the type and the high number of data to display (242 NLP x 5 metrics) then our approach is to rank the variants. Since the same variant is adopted by several NLP (in combination with other variants) we computed the best (maximum) score achieved by a given variant in all the different combinations with the others. The performances of variants are then ranked/described by means of a radar chart; the greater the area of a given variant, the better the performance. We hence adopted a radar chart for variation point and a dimension for each of the above-mentioned metrics. Since a radar chart requires metrics ranging from 0 to 1 according to their goodness, then we normalized the Lag metric: NormLag is computed as *1 – Lag/worstLag*, where *worstLag* is the maximum (worst) value of Lag among NLP. As a results, NormLag measures the same concept of Lag but ranging from 0 (worst NLP) to 1 (best NLP).

5.1.2 Discussion

According to Figure 3, the best single technique is the combination of the following variants: VSM, Raw, Cosine, and Stanford (verbs and nouns). Moreover, again from Figure 3, each of such variants, outperform all the other variants in all the performance metrics. Since the Raw and VSM variants are pretty simple in respect to other variants like TF-IDF and LSA respectively, we can conclude that simple measures resulted as more precise than complex ones. Such a result may be explained by observing that the involved retrieving task is related with duplicates, i.e., with artifacts at the same abstraction level. The most reasonable justification is that, in a mature context, humans tend to adopt the same terms when they express the same technical concept; therefore, techniques that are able to detect similarity even when the terms are not identical results in worst performances by providing more noise rather than positive contribution. In other words, results suggest that in case the terms are different, it is better to judge their semantic as different rather than observing any possible semantic similarity. This conjecture is supported by the fact that simplest NLP techniques are usually adopted to retrieve duplicates [1, 6, 17, 29]. Conversely, in case humans need to trace artifacts of different types or belonging to different abstraction levels, then there is a need to discover similarity even when the words do not coincide. As a matter of fact, complex NLP techniques performed better than simpler in case they are adopted to retrieve (trace) artifacts between different abstraction levels [10-13, 15, 16, 30-34]. The fact that VSM outperforms LSA could be also be explained by the very consistent vocabulary due to the fact that requirements come from the same company (thus sharing culture) and from systems belonging to a well-defined domain. LSA may perform better than VSM in case the equivalent requirements come from different projects/companies; in this case, the use of polysemy and synonymy may be valuable.

5.2 R.Q. 2 How does the best NLP perform? Is there space for improvement? How far are we from the ideal performances?

5.2.1 Results

Our analysis procedure starts by reporting the performance of the best technique (see Table 7). Then, we computed the scores of the ideal best, ideal worst, and random (i.e. ideal non-intelligent) NLP. The performance of the non-intelligent NLP has been computed by generating the random scores 1000 times and checking related low standard deviation. Figure 4 shows the ranking effectiveness by relating the percentage of requirements pairs that needs to be analyzed to retrieve a given percentage of equivalent pairs, according to the provided ranking. The more the curve is vertical the better the ranking provided by the given NLP. In particular, the red line shows the performances of the "Ideal best" ranking, the violet line shows the performance of the "Real best" NLP, the blue line shows the performances of a "Non intelligent" ranking. Therefore, the more an NLP is close to the red line and far away from the blue line, the better the ranking provided by the given NLP. The green line shows the performances of the "Ideal worst" ranking; it represents the lower bound in ranking effectiveness. The sky blue line represents the performance in average among NLP ("Real average").

5.2.2 Discussion

Regarding filtering, according to Table 7, the best NLP performed "near" the ideal best performance and much better than a non-intelligent (NLP) approach. In particular, the best NLP seems to be better in terms of ROC area rather than Precision or Recall. Let us note that Precision and Recall resulted equal, however, a Non-intelligent (random) NLP have a high Recall on the adopted dataset in respect to Precision. In other words, the best NLP performed worst in Recall over Precision and ROC area. Therefore, results suggest that future research would achieve major improvements if aimed at improving Recall.

As far as suggesting is concerned, according to the credibility reported in Table 7 and the results reported in [2], showing the similarity measure provided by the best NLP is expected to improve both speed and quality of humans classifications. In particular, it is supposed to decrease of 1.6 seconds the amount of time required to classify a given requirement pair and the correctness is supposed to increase 5%.

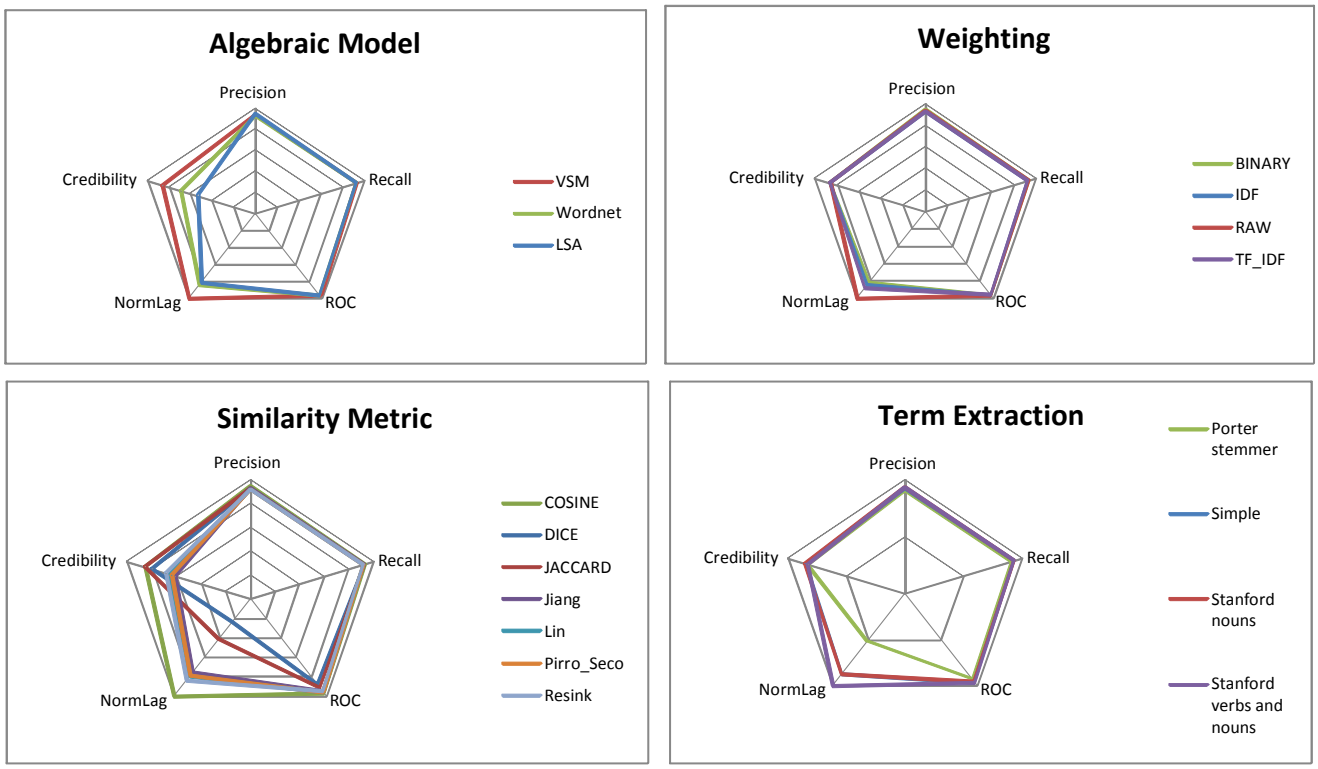

Figure 3 Radar chart of the max performance of each NLP variants.

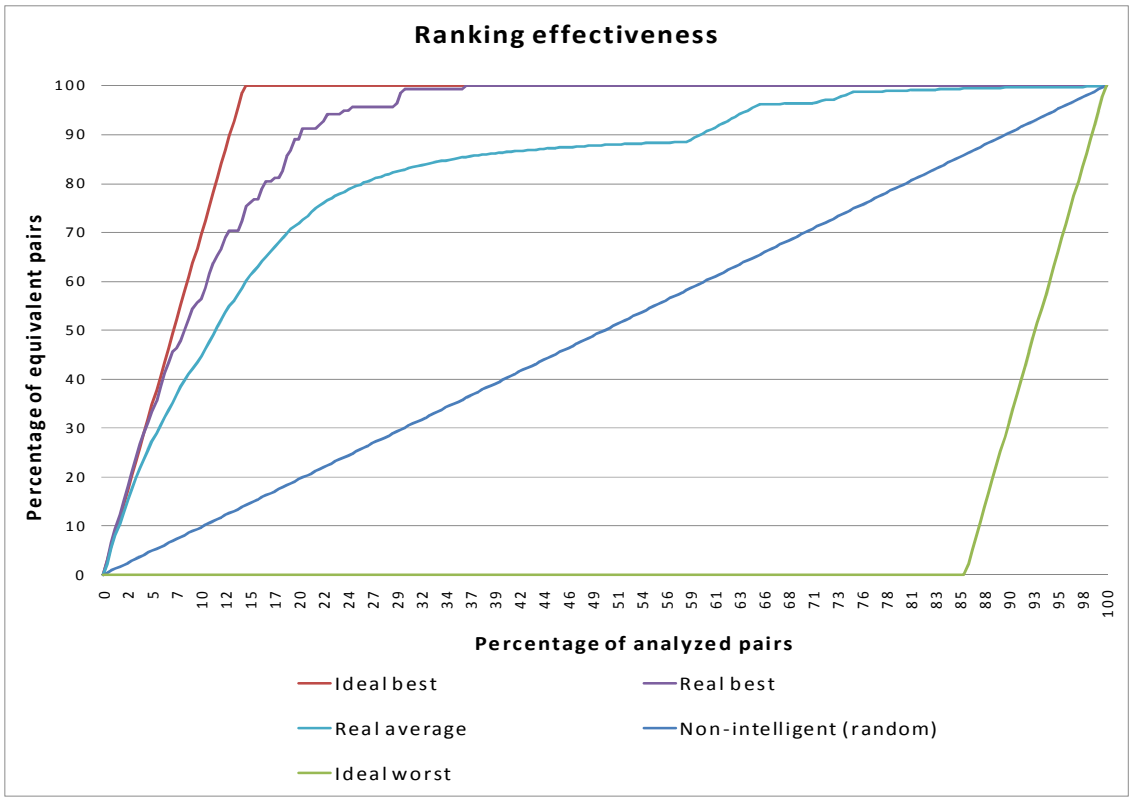

Figure 4 Ranking effectiveness of NLP techniques.

Regarding ranking effectiveness, according to Figure 4, the best NLP technique resulted quite close to the best ideal performance (see the proximity between the red and the violet lines). Moreover, according to Table 7, the Lag value of the best NLP is significantly better than a random technique. In particular, a random ranking has a Lag value fifteen times higher the best NLP technique. Hence, ranking the requirements pairs according to the best NLP technique drastically reduces the effort required to retrieve equivalent requirements when compared to a random ranking. In conclusion, the best NLP performed well in ranking equivalent requirements pairs.

Table 7: Performance of the real and ideal NLP techniques.

NLP	Precision	Recall	Roc Area	Lag	Credibility
VSMRAWCOSINE0Stanford (nouns and verbs)	0.935	0.936	0.967	0.754	0.850
Ideal best	1.000	1.000	1.000	0.000	1.000
Ideal worst	0.000	0.000	0.000	4.370	0.000
Ideal non-intelligent	0.739	0.860	0.500	4.340	0.500

6 VALIDITY DISCUSSION

The present section describes the major threats to validity as Wohlin et al. synthesized and structured in [35].

Conclusion validity. Regarding *Reliability of Measures*, the adopted performance metrics are the state of the practice.

Internal validity. Due to the characteristic of the present case study, threats to internal validity are pretty insignificant; in fact, the following threats don't apply at all*: History, Maturation, Testing, Instrumentation, Statistical regression, Selection of subjects, Mortality,* and *Casual influence*.

Construct validity. Regarding *Mono-operation bias*, the adopted dataset contains a large sample from real industrial requirements; additionally, the sampling strategy aimed to tradeoff the time constraints and the generalizability of both the requirements population, and the equivalent requirements (see [2]). 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 4.4.1. The use of such an objective function allowed use to use an optimal threshold for each NLP; this excluded the effect of the *experimenter expectancies* on results.

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 threats on *interaction of setting and treatment*. Moreover, the enacted cross-validation enhanced this kind of validity. Last, neither the *Interaction of history and treatments* nor the *Interaction of selection and treatment* apply to the case study. Finally, though industrial data, because the analyzed text belong to a single company, the vocabulary is likely to be clear and concise; this assumption may not hold in case the NLP is applied across several companies/countries.

7 CONCLUSION AND FUTURE WORK

Linking artifacts, of the same type (clone detection) or of different types (traceability recovery), is a common practice in software engineering. Unfortunately, it is also extremely tedious, error-prone and time-consuming. Thus, past research attempted to provide analysts with Natural Processing Language (NLP) support to effectively retrieve the artifacts to trace. In the paper, we characterized 242 NLP techniques for identifying equivalent requirements in the context of an Italian company in the defense and aerospace domain. Major results from the case study are:

- Simple measures resulted more precise than complex ones. The most reasonable justification is that, in the industrial context, humans tend to adopt the same terms when they have to express the same technical concept; therefore, techniques that are able to detect similarity even when the terms are not identical resulted in worst performances by providing more noise rather than positive contributions.
- According to [2],showing the similarity measure computed by the best NLP is expected to decrease of 1.6 seconds the amount of time required to classify a given requirement pair and the correctness is supposed to increase 5%.
- Ranking the requirements pairs according to the best NLP technique drastically reduces the effort required to retrieve equivalent requirements when compared to a random ranking.
- The best NLP performed worst in Recall over Precision and ROC area. Therefore, results suggest that future research would achieve major improvements if aimed at improving Recall.

By observing the results of the NLP characterization, SELEX SI knows which NLP to adopt and is satisfied about its performance. The main limitation of the proposed results is their relation to one industrial source only; it is therefore unclear whether a sample taken from another source would lead to different results. In order to alleviate this problem, we are currently working on a novel analysis procedure for comparing NLP techniques which is aimed at minimizing the influence of the adopted dataset on the NLP comparison results.

8 ACKNOWLEDGEMENT

We would like to thank Lionel Briand for useful suggestions in early phase of this work. The authors thank the Engineering Department of SELEX SI for insightful interactions and support provided; in particular, we thank Emanuela Barbi and Vincenzo Sabatino. We thank Carlo Ieva for ideating and developing ANTARCTICA. This work was been partially supported by SELEX SI under the research grant: SSI-DISP/07/08. Davide Falessi is partially supported by Det Norske Veritas in the ModelME! project.

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

[4] D. Falessi, G. Cantone, S. A. Sarcià, G. Calavaro, P. Subiaco, and C. D'Amore, "Peaceful Coexistence: Agile

Developer Perspectives on Software Architecture," *IEEE Software,* vol. 27, 2010.

[5] D. Falessi, G. Cantone, R. Kazman, and P. Kruchten, "Decision-making Techniques for Software Architecture Design: A Comparative Survey " *ACM Computing Surveys,* vol. to appear, 2010

[6] 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.

[7] 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.

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

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

[10] 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.

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

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

[13] M. Di Penta, S. Gradara, and G. Antoniol, "Traceability Recovery in RAD Software Systems," in *Proceedings of the 10th International Workshop on Program Comprehension*: IEEE Computer Society, 2002.

[14] 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.

[15] 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.

[16] 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.

[17] P. Runeson, M. Alexandersson, and O. Nyholm, "Detection of Duplicate Defect Reports Using Natural Language Processing," in *Proceedings of the 29th international conference on Software Engineering*: IEEE Computer Society, 2007.

[18] G. Antoniol, G. Canfora, G. Casazza, and A. De Lucia, "Identifying the Starting Impact Set of a Maintenance Request: A Case Study," in *Proceedings of the Conference on Software Maintenance and Reengineering*: IEEE Computer Society, 2000.

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

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

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

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

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

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

[25] X.-Y. Liu, Y.-M. Zhou, and R.-S. Zheng, "Measuring Semantic Similarity in Wordnet," in *Machine Learning and Cybernetics, 2007 International Conference on*. vol. 6, 2007, pp. 3431-3435.

[26] 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.

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

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

[29] 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.

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

[31] M. Lormans and A. van Deursen, "Can LSI help Reconstructing Requirements Traceability in Design and Test?," in *Proceedings of the Conference on Software Maintenance and Reengineering*: IEEE Computer Society, 2006.

[32] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing Supporting Evidence to Improve Dynamic Requirements Traceability," in *Proceedings of the 13th IEEE International Conference on Requirements Engineering*: IEEE Computer Society, 2005.

[33] A. Marcus and J. Maletic, "Recovering documentation-to-source-code traceability links using latent semantic indexing," in *Proceedings of the 25th International Conference on Software Engineering* Portland, Oregon: IEEE Computer Society, 2003.

[34] G. Antoniol, A. Cimitile, and G. Casazza, "Traceability Recovery by Modeling Programmer Behavior," in *Proceedings of the Seventh Working Conference on Reverse Engineering (WCRE'00)*: IEEE Computer Society, 2000.

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