ATLaS: A Framework for Traceability Links Recovery Combining Information Retrieval and Semi-supervised Techniques

Emma Effa Bella, Stephen Creff, Marie-Pierre Gervais, Reda Bendraou

ATLaS: A Framework for Traceability Links Recovery Combining Information Retrieval and Semi-supervised Techniques

Emma Effa Bella*‡, Stephen Creff*, Marie-Pierre Gervais†, Reda Bendraou†
*IRT SystemX, Paris-Saclay, France
firstname.lastname@irt-systemx.fr
† Université Paris Nanterre, CNRS- LIP6, Paris, France
firstname.lastname@lip6.fr
‡ Sorbonne Université, CNRS- LIP6, Paris, France
firstname.lastname@lip6.fr

***Abstract*—Current Model-Based Systems Engineering (MBSE) practices to design and implement complex systems require modeling and analysis based on many representations: structure, dynamics, safety, security, etc. This induces a large volume of overlapping heterogeneous artefacts which are subject to frequent changes during the project life cycle. In order to verify and validate systems requirements and ensure that models meet user's needs, MBSE techniques shall rely on consistent traceability management.**

In this paper, we investigate the benefits of Information Retrieval (IR) techniques and the latest advances in Natural Language Processing (NLP) approaches to suggest stakeholders with candidate semantic links generated from the processing of structured and unstructured contents.

We illustrate our approach called ATLaS (Aggregation Trace Links Support) through an application on the design and analysis of a mobility service gathering several industrial partners. We provide an empirical evaluation regarding its limitations as part of an industrial MBSE process. Most importantly, we highlight how our method drastically reduces the false positive links generated compared to current IR techniques. The results obtained suggest a good synergy between the presented approach and MBSE techniques.

***Index Terms*—Model-Based Systems Engineering, Requirements, Traceability, Information Retrieval, Natural Language Processing, Semi-supervised techniques**

I. Introduction

System complexity is a very pragmatic reality that project stakeholders must face and overcome throughout the life cycle of an industrial system. In the case of Systems of Systems (SoS), another level of complexity is added as the development of such systems implies collaborations between different companies.

Within such collaborations, each company has to share knowledge and know-how (tooled methodology) as well as requirements and models. The design and implementation of the SoS under study require productions of many artefacts correlated with each other and that evolve most of the time independently. Therefore, this collaboration requires a permanent and accurate synchronization of information.

Considering the large number of models that evolve independently in silos, it is very difficult to measure the impact of a local decision over all the requirements and models of the SoS. This issue requires then almost full traceability, including the linking of heterogeneous engineering artefacts encapsulated in black boxes [1]. Spatial and temporal decomposition of organizations and activities hamper traceability and thus assurance of the overall consistency of requirements [2].

In such a context, it is important to provide a framework for improving awareness regarding the impact of decisions beyond technological or organizational silos [3]. Such a framework shall provide facilities to recover traceability links to avoid a snowball effect due to not shared hypotheses.

From the perspective of Requirements Management (RM), tracking is generally performed manually by system analysts using specific-purpose RM tools. Analysts visually examine each pair of artefacts documented in the RM tools to build a traceability matrix. Most existing RM tools like Rational DOORS or Rational RequisitePro support traceability analysis. Besides, dedicated traceability management tools exist. Among them, we can cite the Eclipse Capra (open-source) or Reqtify (commercial).

Requirements are most commonly expressed as informal text. Then, the elicitation of traceability links requires a human intervention to understand and determine the validity of links. Existing rule-based approaches can be used to provide automation for the elicitation process, but with significant effort to obtain a precise and complete result. Moreover, although IR techniques are promising, their main drawback is the important amount of false positive links generated.

Following the trend of IR methods, promising machine learning approaches are emerging to improve their accuracy [4].

1) On Word *and* Sentence Embeddings*:* Unlike "conventional" IR techniques like Latent Semantic Indexing (LSI) [5] or Vector Space Model (VSM) [6], which are based on word frequency and treat each word as a unique symbol, *Word Embeddings* approaches do not consider words independently

from each other. It can be actually considered as one of the primary reasons justifying the success of recent machine learning models for NLP tasks [7]. Therefore, it has been successfully used for recovering traceability links [8]–[10]. Some *Word Embeddings* techniques use neural networks to process words as d-dimensional vectors of real numbers capturing their contextual semantic meaning. These techniques, based on Harris' distributional hypothesis [11], consider that words with the same context tend to have similar meanings, thus similar vector representations. Therefore, words can be characterized by other words surrounding them and syntactic and semantic relationships between words can be encoded as linear relationships between them [8].

To go further, the success of *Word Embeddings* has motivated new methods for generating semantic embeddings using longer pieces of text, such as sentences and paragraphs [12]–[14]. While *Word Embeddings* represents words as vectors, *Sentence Embeddings* captures the meaning of a chunk or a full sentence in a single vector.

Sentence Embeddings can be used to improve the accuracy of traceability links recovery between requirements and models. For instance, Arora methods [12], also called *Smooth Inverse Frequency (SIF)*, achieve significantly better performance than the unweighted average on a variety of textual similarity tasks.

2) Learning and semi-supervised techniques: Machine learning techniques have been widely used to improve information retrieval approaches [4]. In the literature, machine learning techniques are usually classified into three groups: the unsupervised, the supervised and the semi-supervised ones. Semi-supervised techniques make use of a small amount of - labeled - validated data (or considered validated) and a large number of invalidated data. Many researchers have found that unlabeled data, when used in conjunction with a small amount of labeled data, can produce a considerable improvement in learning accuracy [15].

Labeling often requires expensive human effort while unlabeled data is far easier to obtain. As such, semi-supervised learning has shown its relevance in many real-world problems [16].

Semi-supervised learning algorithms depend on the assumption that nearby points are likely to have the same label or points on the same cluster are likely to have the same label. This hypothesis has been verified on multiple occasions in communities of machine learning [17], [18], information retrieval [19], [20] and traceability management [6], [21]. This assumption referred to as the ***cluster hypothesis***, states that closely associated documents tend to be relevant to the same requests [22] in terms of traceability, i.e. that true links tend to be more similar to each other than false ones.

Besides, the *label propagation* algorithm and its variant the *label spreading* are semi-supervised machine learning techniques that are used for the detection of common features in large complex networks. Among these existing techniques, the label spreading algorithm is the most robust to reduce noise in text classification [23].

Semi-supervised approaches need a first set of representative links to start learning which can be provided by rule-based approaches. Regarding the improvement of IR techniques, one major challenge addressed in this paper is the accuracy of identification of semantics similarities, between words and sentences used with the same meaning. Doing so, we can be able to tackle the well-known drawback of usual IR techniques, i.e. the generation of a large number of false positives.

In this paper, we investigate the benefits of the latest advances in semi-supervised approaches and NLP approaches in order to improve the performances of IR techniques. We aim to enhance candidate traceability links generation and suggestion to the analysts. Thus, we propose an approach, called ATLaS (Aggregation Trace Links Support). It is based on the clustering hypothesis that combines different strategies of IR and NLP techniques (i.e. word-embeddings and sentence embeddings) to improve the accuracy of IR techniques. Following an industrial case study from the automotive domain, we provide an empirical evaluation and discussion on our approach.

The remainder of the paper is organized as follows. Section II presents a concrete industrial case study. Then, Section III formalizes our approach for the particular issue illustrated in the motivating example. Section IV presents the implementation and evaluation performed. Next, Section VI provides some related works. Finally, the last Section draws our conclusion and narrows down possible future works.

II. The automotive case study

The experiments for the validation of our approach have been conducted over an industrial case study build upon the publicly available datasets ARC-IT (Architecture Reference for Cooperative and Intelligent Transportation) in version 8.2. This reference architecture is provided by the U.S. Department of Transportation and is freely available on the ARC-IT website[1]. It provides a common framework for designing intelligent transportation systems (ITS). It includes a set of interconnected engineering artefacts organized into four views focusing on different architecture perspectives: Enterprise, Functional, Physical and Communications.

From a tooling perspective, requirements are managed in ProR[2] (open-source), system models (operational, functional, physical or product) have been captured in the Capella tool[3].

Additional models have been used at the system level. For instance, behaviours of operational actors have been captured using the BPMN2 tool, while their interactions have been captured using Colored Petri Nets tools[4]; finally, their modes and states have been captured using the UPPAAL tool[5]. The choice of those tools has been driven by the necessity to assist engineers with more formal tools to make rational decisions.

[1]https://local.iteris.com/arc-it/index.html
[2]https://www.eclipse.org/rmf/pror/
[3]https://www.polarsys.org/capella/
[4]http://cpntools.org/
[5]http://www.uppaal.org/

For the purpose of our demonstration, we limit the scope of our experiment only on the traceability elicitation between requirements (2395 in total) and system functions models (802 in total). System functions cover among others the following concerns: Manage ITS, Manage Traffic, Manage Commercial Vehicles, Provide Vehicle Monitoring and Control, Manage Transit, Manage Emergency Services and Support Secure Transportation Services.

Sharing and linking (manually) engineering artefacts are performed thanks to the use of collaborative spaces, based on semantic web technologies [3]. The Tracelink support (ATLaS) described in Section III generates candidate traceability links to the collaborative space for validation.

III. ATLaS: Aggregation Trace Links Support

In this section, we provide a description of our approach to identify traceability links between requirements and models. ATLaS takes as input requirements documents described in natural language and models in XML Metadata Interchange (XMI) format and generates a list of traceability links with their confidence measures, see Figure 1. To this aim, the framework performs three main steps.

In step 1 "*Pre-processing artefacts*", requirements and models are taken as inputs and transformed into bags of words and sentences. In step 2 "*Compute similarities*", these bags are used to compute syntactic and semantic similarities between them (i.e. requirements and models). These similarities are used to build the descriptor matrix. In step 3 "*confidence metrics computation*", we defined a heuristic to identify the most likely true and false links by using the syntactic similarities, and we build the labeled links set. This labeled links set associated with the descriptor matrix are used as inputs to compute a predictive model which will provide a confidence measure on whether or not the potential links are true or false. Figure 1 and the following sections describe in details the 3 steps of our approach.

1) ***Pre-processing artefacts****:* The aim of this step is to convert data in natural language for further processing by IR and NLP techniques. It consists in extracting text from requirements and models and performing some basic textual pre-processing actions to divide them into bags of words and sentences.

a) Artefacts extraction: A first step is to extract text from artefacts. Two kinds of artefacts are considered: requirements and models.

- Requirements are by nature text-oriented and their extraction in the ProR tool is quite trivial. Basically, *id* and description of each requirement are extracted and concatenated into a text. For example, a requirement with id equals to "*R255*" and description equals to "*The center shall provide the collected border activities statistics data to archived data and planning systems.*" will be represented as "*R255 The center shall provide the collected border activities statistics data to archived data and planning systems.*";
- Models and more precisely model elements are a little bit more complex to process since it is important to define the appropriate granularity which will capture the maximum amount of textual information and its semantics. For instance, in a UML class diagram, the ideal granularity would be the class because it would gather enough information through the attributes and operations. Actually for any given type of models, one needs to define the appropriate granularity (model elements) to be taken into account by ATLaS.
  In our case study, the functional architecture model of the system was used as input and for each function, the *id*, the name, the type and the description were extracted and concatenated into a text and given as output. For example, a function with id equals to "*F474*", name equals to "*TransportationInformationCenter*", type equals to "*system function*" and the description equals to "*Traffic Regulation Dissemination disseminates rules, regulations, and statutes that govern motor vehicle operation.*" will be represented as "*F474 TransportationInformationCenter system function Traffic Regulation Dissemination disseminates rules, regulations, and statutes that govern motor vehicle operation.*"

b) Sentence Splitting: consists in splitting each artefact into sentences. To perform this operation, NLP techniques require as inputs, symbols that indicate the beginning and end of a sentence. A sentence must begin with a capital letter and ends with a punctuation mark.

c) "Stopword" removal: "Stopwords" [22] are words that appear so frequently in the artefacts that they become irrelevant for link recovery, for example, article "*a*" or "*the*". They are therefore removed and the result is sent to the tokenization and text chunking actions.

d) Tokenization: Tokens are built by splitting the sentences into words. Then, punctuations and numbers are removed.

Note that tokenization is done differently for models and requirements. In the case of models, compound words are split by upper camel case rule. For example, "VehicleIntersectionWarning" is split into "Vehicle", "Intersection" and "Warning".

e) Text chunking: is a natural language technique that decomposes a sentence into non-overlapping segments [24]. We use text chunking to detect phrases in the sentences. We only consider the noun and verbal phrases because they are the main elements that give the meaning of sentences. For example, the text chunking of the requirement "*R255*" with the description "*The center shall provide the collected border activities statistics data to archived data and planning systems.*" will be "*The center*, "*shall provide*", "*the collected border activities statistics data*", "*to archived*", "*data and planning systems.*".

Finally, all words, noun and verbal phrases of pre-processed artefacts, namely requirements and models are converted to lower case and are sent to the dictionary builder (see Figure 1).

2) ***Compute similarities****:* takes as input bags of words, nouns and verbal phrases and computes syntactic and se-

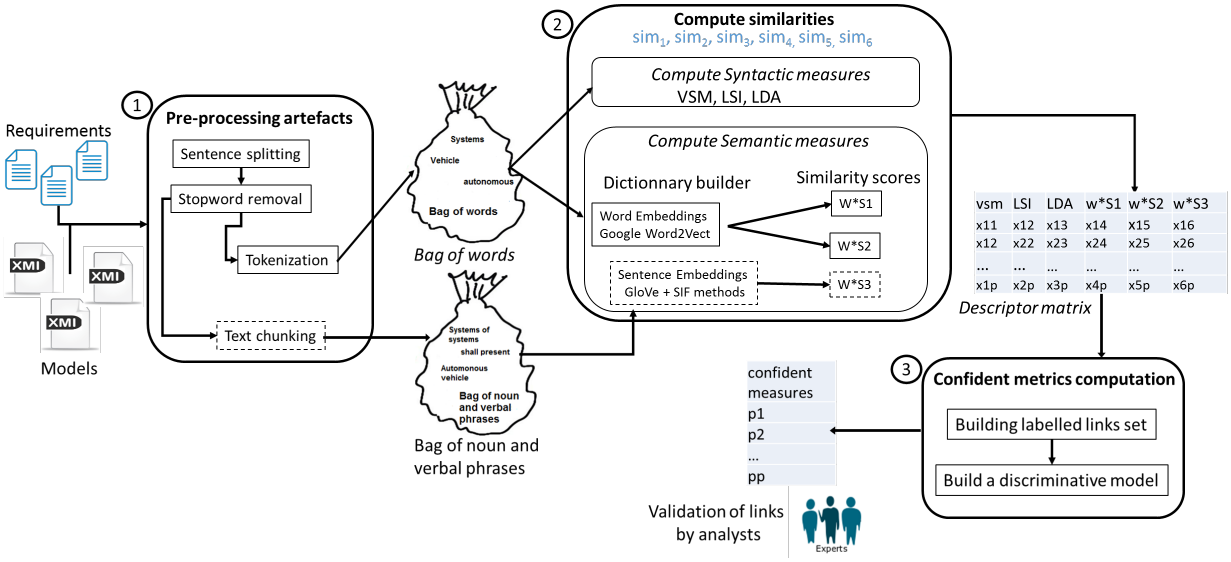

Fig. 1. Overview of ATLaS (Aggregation Trace Links Support)

mantic similarities. A vector containing similarity scores is constructed for each pair of artefacts. Each vector constitutes a row of the descriptor matrix, which is the output of this step.

a) syntactic measures computation: For each pair of artefacts, we quantify LSI [5], Latent Dirichlet Allocation - LDA [25] and VSM [6] scores. These scores are used to build a labeled links set for the semi-supervised technique. They are also inserted into the descriptor matrix.

b) Semantic measures computation: In a previous work [26], we proposed an approach that combines syntactic and semantic measures. Results reached were better than conventional IR techniques but were still not satisfactory. In order to improve these results, we investigate word and sentence embeddings to build contextual dictionaries to capture more semantics. These dictionaries have been used to define three similarity scores that will be defined a little bit further. The construction of the contextual dictionaries and computation of the similarity scores are described below.

Dictionary builder. This activity is used to determine the vector representations of words and phrases contained in requirements and models according to their contextual meaning. We use bags of words and phrases as inputs. The output of this activity is a list of synonymous words and phrases.

Word Embeddings learning refers to finding vector representations of words such that words with similar meaning are associated with similar vector embeddings [27]. Many words embeddings pre-trained models are available publicly. Among these *Word Embeddings* pre-trained models, google *Word2Vec* model [7] is popular for its simplicity and efficiency. It was trained on about 100 billion words from Google News. The training was performed using the continuous bag of words architecture and its vocabulary size is 3 million entities.

Besides Google *Word2Vec*, we have also used *GloVe* [28] pre-trained *Word Embeddings*. Like Google *Word2Vec* model, *GloVe* model is significantly efficient to word analogy and similarity tasks. *GloVe* stands for "Global Vectors for Word Representation". It is an embedding technique, which tabulates how frequently words co-occur with one another in a given corpus. The *GloVe* is used in the *Smooth Inverse Frequency (SIF)* methods of Arora et al. [12] to compute noun and verbal phrases embeddings.

Both pre-trained models are used to determine the contextual words, noun and verbal phrases dictionaries to identify synonymous words, noun and verbal phrases.

Similarity score computation. With the list of synonymous words and phrases, we defined three similarity scores. The first similarity score S_1 is one of the most popular similarity scores in the traceability community, the Naive Satisfaction Method [29], [30]. Like most similarity scores used in text classification and traceability, S_1 focuses on common terms shared by related pairs of artefacts. The second similarity score S_2 is similar to the first one (S_1) but at the phrase level. It computes the ratio of common noun and verbal phrases between related artefacts, including synonyms. Finally, the last one, S_3, is also computed at the word level like S_1, but after filtering out some words that have been considered irrelevant.

For example, in systems engineering, words like "system" or "shall" are recurrent in requirements and yet, irrelevant with respect to requirements traceability. These words can, therefore, yield numerous false links if not taken into account. We make the assumption that these words are more frequent than the ones that are actually useful for identifying true links.

With this assumption, we compute the frequency of each word in the corpus and we filter out words with a frequency above a chosen threshold. We can then compute a new similarity score S_3 as the ratio of semantically related important words (i.e. words that have not been filtered out at the previews step) between two artefacts.

These similarity scores are computed using the following equation:

$$S_1 = \frac{N_{common}}{N_{TotalWords}}. \quad (1)$$

$$S_2 = \frac{N_{commonNounAndVerbalPhrases}}{N_{TotalPhrases}}. \quad (2)$$

$$S_3 = \frac{N_{commonImportantWords}}{N_{TotalWords}}. \quad (3)$$

Where :

- N_{common} is the number of common terms including synonyms,
- $N_{TotalWords}$ is the total number of terms of the two artefacts,
- $N_{commonNounAndVerbalPhrases}$ is the number of identical noun and verbal phrases and their synonyms between two artefacts,
- $N_{TotalPhrases}$is the total number of noun and verbal phrases of the two artefacts,
- and $N_{commonImportantWords}$ is the number of common important words between two artefacts including synonyms.

*3) **Confidence metrics computation**:* We use the descriptor matrix as input. All similarity scores (S_1, S_2, S_3) described above along with IR measures (LSI, LDA, VSM) provide similarity measures for each pair of artefacts in the form of a vector. This vector is used to decide whether a link exists or not between two given artefacts.

In order to overcome the low performance of syntactic measures, we associate a weight to each similarity score. Each pair of artefacts is described by a descriptor vector. Thus, for a pair of artefacts (x, y) with (m_1, m_2, m_3, m_{LSI}, m_{LDA}, m_{VSM}), the similarity measures assigned by the 3 similarity scores and the 3 IR techniques, the associated descriptor vector would be equal to ($weight * m_1$(x, y), $weight * m_2$(x, y), $weight * m_3$(x, y), m_{LSI}(x, y), m_{LDA}(x, y) m_{VSM}(x, y)).

As mentioned earlier, semi-supervised algorithms need a small number of validated links. Unfortunately, training dataset (i.e. a set of artefacts with validated links) was not available for our experiments. The following paragraphs describe how we coped with this issue.

a) Building labeled links set: We have implemented a heuristic to label some pairs of artefacts as related or non-related. This heuristic is motivated by the statistical ranking of IR techniques [31], which sets good links on top and bad links (false positives) to the bottom. Thus, for each similarity measure, we select a small percentage of links with the highest and lowest similarity scores.

b) Build a discriminative model: We use the weighted similarity scores ($weight * S_1$, $weight * S_2$, $weight * S_3$), the IR measures (LSI, LDA, VSM) and the labeled of true and false links defined by the heuristic as inputs to build the discriminative model. Based on these inputs, the discriminative model finds the community structures in order to group them in true and false links. In our approach, the training set is built, i.e. it is not manually validated. In order to build our discriminative model, we use label spreading [32] among other semi-supervised learning methods. This choice is justified by the ability of the technique to modify initially provided labels.

As output, the resulting predictive model provides a probability for any link that belongs to the class of true links. We refer to this probability as a *confidence measure*.

The ATLaS generates this way candidate traceability links to the semantic web collaborative space for validation; each link being coupled with a confidence measure. From a practical point of view, it provides a prioritization criterion to the analyst during the phase of links validation.

IV. Implementation and Experimentations

The implementation relies on existing tools and libraries. First, VSM has been implemented via the Tracelab tool [33]. Then, LSI and LDA have been implemented with Gensim[6]. As mentioned in the previous section, words and phrases dictionaries have been build using word embeddings pre-trained models available publicly: Google *Word2Vec* and *GloVe* vectors; Both sets are 300-dimensional. The *GloVe* model is used inside the SIF method which source code is available on GitHub[7]. The size of the dataset, which is the case for industrial-size case studies, requires a good computation power along with a significant amount of working memory. For this purpose, we used a computer with 176 physical CPU cores and 2 TB of RAM.

Table I shows the total number of requirements and model elements, the corpus and vocabulary size of the dataset, as well as the number of validated links (golden standard).

	ARC-IT Req- Functions
Nb Requirements	2395
Nb models elements	802
Corpus size	76643
Vocabulary size	2731
Golden Standard	2395 links

TABLE I
DESCRIPTIVE STATISTICS OF DATASETS.

The current implementation of label-spreading in scikit-learn[8] is not scalable. Thus, failing to find a label-spreading implementation solution that scales up, we opted to learn from a subset of the dataset from which we can extrapolate for the whole dataset. We have defined two algorithms to build the subset for learning.

Algorithm 1 consists of building the subset by selecting the reference links of the heuristic and randomly adding to these links, a defined number of links that have not been chosen by the heuristic. This algorithm is very simple to implement. The result of this algorithm is represented by *aggreg0_V1* (see Figure 4).

Algorithm 2 aims to ensure that the subset of links selected is representative of the complete links set. The core idea

[6]https://radimrehurek.com/gensim/index.html
[7]https://github.com/PrincetonML/SIF/blob/master/README.md
[8]https://scikit-learn.org/stable/modules/generated/sklearn.semi_supervised.LabelSpreading.html

is the following: when a link is retained for the subset, it is not necessary to have the nearest neighborhood of this link within the subset because they are likely to belong to the same class. The neighborhood is defined in terms of Euclidean distance between the links descriptor vectors. The subset is built as follow: first, we compute the neighborhood graph of the complete dataset. Then, we divide this graph in multiple neighborhoods. The number of neighborhoods is equal to a defined down-sampling factor. When a link is randomly selected, its nearest neighbors are discarded. This operation is repeated until there is no link left. This algorithm is more suitable for label-spreading technique since its goal is to let every link iteratively spread its label information to its neighbors until a global stable state is achieved [23]. The result of this algorithm is represented by *aggreg0_V2* (see Figure 4).

In the following, we present the experimental results obtained on the ARC-IT dataset.

A. Evaluation of the capture of the semantic

The evaluation analyses the contribution of semantic techniques versus syntactic ones. At the optimum threshold, syntactic methods missed some links compared to our approach. For instance, the link between requirement "***R2539*** *The vehicle shall present information to the driver in audible or visual forms without impairing the driver s ability to control the vehicle in a safe manner.*" and the function "***F405 Vehicle Control Warning.*** *Monitors areas around the vehicle and provides warnings to a driver so the driver can take action to recover and maintain safe control of the vehicle.*" was missed by the syntactic techniques. Nonetheless, some links identified by the syntactic methods were missed by our approach.

Syntactic methods identify those links because they are based on common terms between artefacts. For instance, the link between the requirement "***R6396*** *The* ***center*** *shall aggregate updates to rules,* ***regulations****, and statutes in order to define updates to be sent to vehicles and other mobile devices.*" and "***F474 Transportation Information Center.*** *Traffic* ***Regulation*** *Dissemination disseminates rules,* ***regulations****, and statutes that govern motor vehicle operation.*" was identified by syntactic techniques.

Some links were missed by both kinds of methods due to the fact that semantic captured lacks precision: The link between the requirement "***R255*** *The center shall provide the collected border activities statistics data to archived data and planning systems.*" and the function "***F188 Border Inspection Administration*** *performs administrative functions relating to the inspection of goods and vehicles at the border*" illustrates such a situation.

B. Discrimination with LDA-LSI-VSM

Pairs of artefacts are represented with the similarity measures LDA, LSI, and VSM. Links of the golden standard (referred as "true links") and others links (referred as "false links") are represented with different colors in order to observe resulting clusters in Figure 2. Clusters of "true links" and "false links" appear very distinctly and this observation reinforces the clustering hypothesis at the heart of our approach. It appears that in the ARC-IT dataset, similarity measures are strongly discriminating against each other.

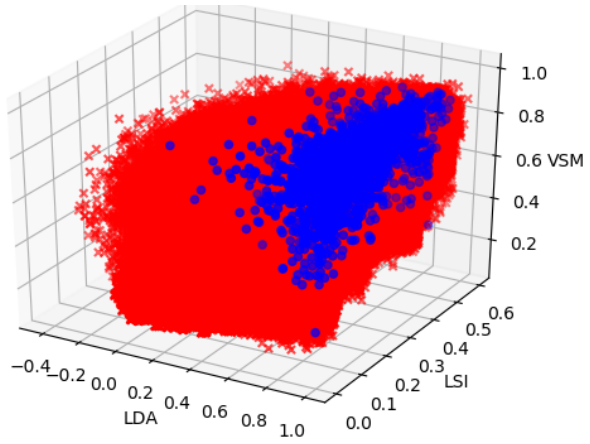

Fig. 2. Scatter plot representing requirements-models pairs described by the similarity measures of VSM, LSI, LDA techniques; blue points correspond to true links and red points to false links.

C. Evaluation of the build training set

The reference links used to build the predictive model were automatically created by our heuristic. To evaluate the relevance of this model, we have compared the reference true links and false links to the actual true links and false links. Table II shows the percentage of coverage; *aggreg0_V1* represents the result obtained by the heuristic described in section III and computed with Algorithm 1. It shows that the percentage of coverage of false links is above 99%, while the percentage of coverage of true links is under 5% for the 10% lowest and the 10% highest similarity scores respectively. One conclusion we can draw is that the proposed heuristic is very good to identify false links but it requires significant improvement for the identification of true links.

This can be explained by the well-known *curse of dimensionality problem* [34], or in our case *"curse of cardinality problem"*. The curse of dimensionality refers to various phenomena that arise when analyzing data in high-dimensional spaces. When the dimensionality increases, the volume of the space increases so fast that available data becomes sparse. The ARC-IT dataset contains 2395 high-level requirements and 802 logical functions; so the number of candidate links will be 2395 x 802 = 1920790 links for 2395 true links.

Indeed, the number of false links increases drastically when the limit of the number of true links barely reaches the number of requirements. In this Cartesian product of artefacts, true links become negligible regarding the number of false links. It is more likely to find false links than true ones. This raises one question: *how to increase the percentage of coverage of true links?*

To answer this question, we have defined three strategies. Table II shows the percentage of coverage of true links for

each of these strategies. Note that the heuristic to obtain false links remains unchanged. *aggreg1*, *aggreg2*, and *aggreg3* in Table II represent the results obtained for *strategy 1, strategy 2, and strategy 3* respectively.

In **strategy 1**, either the analyst identifies a number of true links equal to 5% of the number of requirements (randomly or with a distribution on the concerns of the systems), or a rule is defined that allows random identification of the same number of true links. Example of such rule could be a linguistic pattern based on syntactic analysis and grammatical heuristics like the detection of the modal *shall* in requirements in order to link them to existing system functions (note that these rules may require human validation).

Strategy 2 and strategy 3 use the confidence measure obtained with the proposed heuristic (*aggreg0_V1*). With this strategy, 20% of the links with the highest confidence measures are considered as true links.

In **strategy 3**, the analyst has to examine the 20% of the links with the highest confident measures. The false links identified are added to the computed reference false links. These links are then used as inputs for the predictive model.

For each of these strategies, the percentage defined is based on the number of requirements and the time for an analyst to validate suggested links. Strategies 1 and 3 involve the expertise of an analyst, thus the percentage of coverage of true links is equal to 100%. However, the number of true links is still low compared to the number of false links. With strategy 2, the percentage of coverage of true links is improved by approximately 3% compared to the initial heuristic. Still, the resulting percentage of true links remains low compared to the percentage of false links.

Datasets	True links correctness	False links correctness
aggreg0_V1	4.81%	99.7%
aggreg1	100%	99.7%
aggreg2	7.5%	99.7%
aggreg3	100%	99.9%

TABLE II
TRUE AND FALSE LINKS CORRECTNESS IN THE BUILD TRAINING SET

D. Evaluation of the correlation of the similarity measures

The complementarity of similarity measures constitutes another important hypothesis that stands at the heart of our approach. Our intuition is that the more similarity measures are diverse and complementary, the more the descriptor matrix will contain enough information to discriminate true and false links. One way to evaluate the complementarity of similarity measures is to identify how they are correlated with each other. Correlation refers to the degree to which similarity measures are linearly related. For instance, Figure 3 shows the dependence between each similarity measure. We can see that syntactic and semantic measures are weakly correlated (under 0.25). However, LDA-VSM, S_1-S_3 have an average dependence (between 0.25 and 0.5). One conclusion we can draw from this correlation matrix is the following: although IR methods capture almost the same information, the differences between captured information add value to the collected information for each pair of artefacts.

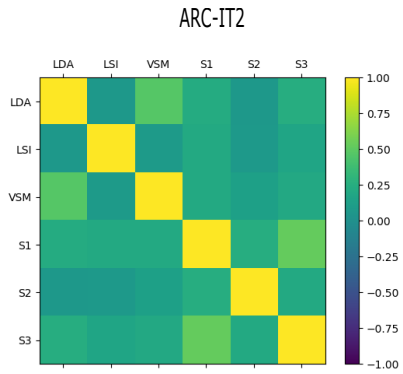

Fig. 3. Correlation between similarity scores and similarity of IR techniques

E. Evaluation with F-measure and recall-precision curves

The metrics most used for evaluating any IR techniques are: *recall*, *precision*, and *F-measure*. The recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances. The precision is the fraction of relevant instances among the retrieved instances. F-measure is the harmonic mean of the recall and precision. It shows the trade-offs between precision and recall. It can then be used to provide insights regarding the performance of a method.

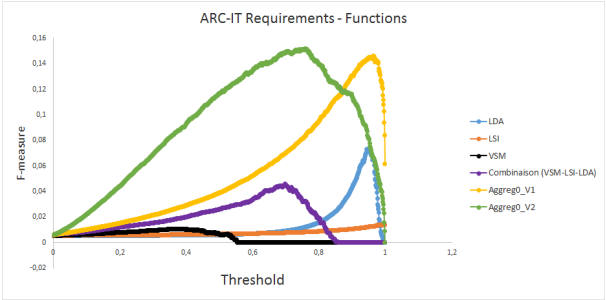

Fig. 4. F-measure curves of LSI, LDA, VSM and the two implementations of our approach (*aggreg0_V1* and *aggreg0_V2*) and the previous paper approach (combinaison of $VSM - LSI - LDA$)

As mentioned at the beginning of section IV, we use two algorithms to build the discriminative model. The result of Algorithm 1 is represented by *aggreg0_V1* and the result of Algorithm 2 is represented by *aggreg0_V2*.

Note that, it is not relevant to give strong weights to the semantics measures because this would make the information provided by the syntactic measures negligible. For this reason in this experimentation, we used empirically a weight equals to 2.

We have evaluated the results of the previous paper approach (combinaison of $VSM - LSI - LDA$), *aggreg0_V1*, *aggreg0_V2*, LSI, LDA and VSM for different thresholds. The

F-measure of all these techniques is illustrated in Figure 4. It shows that our approaches *aggreg0_V1* and *aggreg0_V2* are more effective than VSM, LSI, LDA for any threshold. We can see that precision and recall are significantly improved by these techniques. In particular, precision and recall of *aggreg0_V2* are much higher than all other results. In summary, *aggreg0_V1* achieves 94% of recall and 0.51% of precision at its optimum (threshold=0.1) and narrow the percentage of false links down to 50%, while *aggreg0_V2* achieves 76% of recall and 1.15% of precision at the same threshold and filters out 80% of false links.

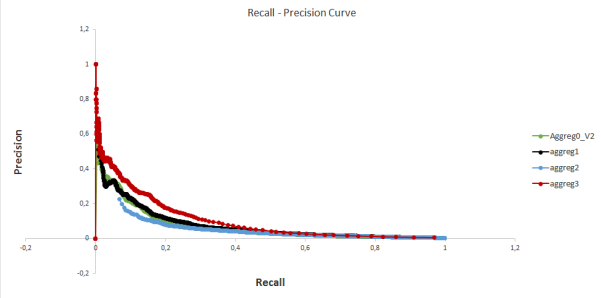

Fig. 5. recall-precision curves of the proposed heuristic and all the proposed strategies

In this sub-section, we have also evaluated the 3 proposed strategies in sub-section IV-C at different thresholds. Thus, we evaluated the results by plotting the best result F-measure curves *aggreg0_V2* and the results of all the three strategies with Algorithm 2 (*aggreg1*, *aggreg2*, *aggreg3*). The recall-precision curves in Figure 5 show that Strategy 1 has better results than all the other strategies. However, its recall is very low in front of the recall of *aggreg0_V1*. The other two strategies are victims of the curse of cardinality, explained in sub-section IV-C, and therefore do not produce satisfactory results.

In summary, ATLaS can significantly reduce the number of false links that developers would need to manually identify and can potentially reduce errors during the validation process. The study results show that ATLaS outperforms the conventional IR Techniques and our previous works, but still has room for improvement. In future work, we plan to explore more strategies to improve the low coverage of true links of the built training set. We also plan to implement an algorithm of labelSpreading for industrial-sized datasets.

V. Threats to validity

Two primary threats to validity potentially impact our work. First, due to the challenge of evaluating a large industrial dataset and the time needed to generating trace links. Indeed, given the size of the dataset and the severe imbalance in data (2395 true links and 1918395 false links) standard measures such as F-measure might be inappropriate for measuring the performance of trace links recovery. However, the metrics we used (i.e. Recall, Precision, and F-measure) are all accepted research standards for evaluating traceability results [35]. In future work, we plan to use more robust techniques to such imbalances and to compare our approach with other state-of-the-art approaches.

Second, the threat to validity comes from the dataset used in this experiment. Indeed, this work focused on a single dataset. As a result, we cannot generalize the result beyond the underlying experimental setting. However, this traceability dataset is of large and do not raise some scalability concerns.

VI. Related Works

There are few works that address the problem of traceability in the context of an MBSE approach. These approaches usually propose traceability information models to capture the semantic integration of artefacts. For instance, Taromirad et al. [4] propose an approach for building a multi-domain traceability framework. They define a Traceability Information Model (TIM) that represents artefacts from different domains and their relationships. This Traceability Model can be used to derive traceability information from sources, record the information in the model and perform traceability analyses based on traceability goals. As mentioned in section I, and even though this approach contributes to the identification of links through the use of rules, it requires a significant effort from experts.

In Maro et al. [36], authors have provided a framework for traceability management and an implementation in the Eclipse Modeling Framework (EMF). Provided tool supports basic services for managing traceability links: creation, removal, modification as well as typing. Besides the aforementioned limitations, the approach is constrained by technological choices, which does not reflect industrial practices. Indeed, we shall be able to manage traceability links beyond technological silos.

Some approaches have been proposed to address this limitation. In particular, approaches combining different IR techniques have been proposed in order to improve their effectiveness while compensating for their weaknesses. For instance, Cleland-Huang et al. [37] propose three enhancement strategies: hierarchical modeling, logical clustering of artefacts, and semi-automated pruning of the Probabilistic Network (PN). Results indicate that these strategies effectively improve trace retrieval performance.

In the same way, Wang et al. [5] present four strategies: source code clustering, identifier classifying, similarity thesaurus, and hierarchical structure enhancement. These strategies aim to improve LSI. Their approach has higher precision but lower recall. Le Tien-Duy et al. [38] propose an approach to predict the effectiveness of an IR technique on a bug localization tool for a given bug report. Their proposed approach extracts many features from the textual contents of the bug report and similarity scores outputted by the IR techniques.

Sannier and Baudry [39] proposed combining both MBSE approaches and IR techniques to improve traceability of requirements while addressing the ambiguity of textual content of requirements. The drawback of their proposition is that

it still requires a significant effort from experts to reach a relevant set of rules, that can lead to the generation of traceability links with a high level of confidence.

Indeed, IR techniques aim to match a set of pairs of artefacts and rank the retrieved pairs based on predefined similarity measures. These techniques automate a tedious task that does not require expertise in any specific domain.

IR techniques heavily investigate in recovering traceability links in the literature, including VSM [6], LSI [5], and unsupervised machine learning approach - LDA [25]. However, due to their limited accuracy, candidate links are systematically checked by an analyst that manually classifies them into two groups: the approved links called true-links and the rejected ones called false-links. Thereby, candidate links evaluation is really time-consuming [40] as the number of false links represents 90% of the candidate links at a low threshold.

To improve this situation, machine learning techniques have been widely used to improve information retrieval approaches [4]. They are used to capture knowledge about pairs of artefacts, to define strategies for link classification and to make predictive models. For instance, Mills et al. [41] use a predictive model with two features (text retrieval rankings and query quality metrics) to automatically classify links as true or false. Their approach achieves high accuracy on average using both types of features but there are still a high number of miss-classified links. Sultanov et al. [42] use reinforcement learning and improve the results compared to VSM. Niu and Mahmoud [6] use clustering to group links in high-quality and low-quality clusters respectively to improve accuracy.

Our proposed approach is most related to these approaches, which have been applied to traceability links recovery. Our work differs from the ones presented above in that it combines IR techniques with recent NLP techniques to capture more semantics in order to reduce the number of false links.

VII. Conclusion and future works

In this paper, we investigate the benefits of IR techniques and the latest advances in NLP approaches to suggest stakeholders with candidate semantics links generated from the processing of structured and unstructured documents. We aim to fill the gap between formal and informal contents of MBSE models to free engineers from an important workload by providing them with relevant assistance.

The empirical evaluation of our approach applied to the case study from the automotive industry shows good results as it drastically reduces the number of false positive, compared to usual IR techniques.

However, the interpretation of behavioral models was not considered in this study as we have considered only the static aspects of behavioral specifications. In practice, we note that these kind of models are usually less documented than structural ones and the rules concerning their interpretation are subject to several semantic variation points. Then, more effort is required to transform elements of models with the interpretations given by experts.

Considering encouraging results from the combinaision of techniques, future work will investigate the combination of different strategies with the aim of improving the coverage of true links of the built training set in order to get better accuracy for the generated candidate links.

Another direction will be to consider in the implementation of label spreading for big data. Building a predictive model for the whole dataset graph can probably increase the results making good use of all data in the process.

Acknowledgment

This research work has been carried out in the framework of the Technological Research Institute SystemX, and therefore granted with public funds within the scope of the French Program "Investissements d'Avenir".

References

[1] J. E. Hobbs, "Information asymmetry and the role of traceability systems," *Agribusiness*, vol. 20, no. 4, pp. 397–415, 2004.

[2] A. Sheth and M. Rusinkiewicz, "Management of interdependent data: Specifying dependency and consistency requirements," in *Management of Replicated Data, 1990. Proceedings., Workshop on the*. IEEE, 1990, pp. 133–136.

[3] L. Wouters, S. Creff, E. E. Bella, and A. Koudri, "Towards semantic-aware collaborations in systems engineering," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*, Dec 2017, pp. 719–724.

[4] M. Taromirad, N. D. Matragkas, and R. F. Paige, "Towards a multi-domain model-driven traceability approach." in *MPM@ MoDELS*, 2013, pp. 27–36.

[5] X. Wang, G. Lai, and C. Liu, "Recovering relationships between documentation and source code based on the characteristics of software engineering," *Electronic Notes in Theoretical Computer Science*, vol. 243, pp. 121–137, 2009.

[6] N. Niu and A. Mahmoud, "Enhancing candidate link generation for requirements tracing: the cluster hypothesis revisited," in *2012 20th IEEE International Requirements Engineering Conference (RE)*. IEEE, 2012, pp. 81–90.

[7] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in neural information processing systems*, 2013, pp. 3111–3119.

[8] J. Guo, J. Cheng, and J. Cleland-Huang, "Semantically enhanced software traceability using deep learning techniques," in *2017 IEEE/ACM 39th International Conference on Software Engineering (ICSE)*. IEEE, 2017, pp. 3–14.

[9] T. Zhao, Q. Cao, and Q. Sun, "An improved approach to traceability recovery based on word embeddings," in *2017 24th Asia-Pacific Software Engineering Conference (APSEC)*. IEEE, 2017, pp. 81–89.

[10] B. Xu, D. Ye, Z. Xing, X. Xia, G. Chen, and S. Li, "Predicting semantically linkable knowledge in developer online forums via convolutional neural network," in *Proceedings of the 31st IEEE/ACM International Conference on Automated Software Engineering*. ACM, 2016, pp. 51–62.

[11] Z. S. Harris, "Distributional structure," *Word*, vol. 10, no. 2-3, pp. 146–162, 1954.

[12] S. Arora, Y. Liang, and T. Ma, "A simple but tough-to-beat baseline for sentence embeddings," 2016.

[13] J. Wieting, M. Bansal, K. Gimpel, and K. Livescu, "Towards universal paraphrastic sentence embeddings," *arXiv preprint arXiv:1511.08198*, 2015.

[14] A. Conneau, D. Kiela, H. Schwenk, L. Barrault, and A. Bordes, "Supervised learning of universal sentence representations from natural language inference data," *arXiv preprint arXiv:1705.02364*, 2017.

[15] X. J. Zhu, "Semi-supervised learning literature survey," University of Wisconsin-Madison Department of Computer Sciences, Tech. Rep., 2005.

[16] M. Seeger, "Learning with labeled and unlabeled data," Tech. Rep., 2001.

[17] O. Chapelle, J. Weston, and B. Schölkopf, "Cluster kernels for semi-supervised learning," in *Advances in neural information processing systems*, 2003, pp. 601–608.

[18] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proceedings of the 20th International conference on Machine learning (ICML-03)*, 2003, pp. 912–919.

[19] A. Leuski, "Evaluating document clustering for interactive information retrieval," in *Proceedings of the tenth international conference on Information and knowledge management*. ACM, 2001, pp. 33–40.

[20] X. Chen and J. Grundy, "Improving automated documentation to code traceability by combining retrieval techniques," in *Proceedings of the 2011 26th IEEE/ACM International Conference on Automated Software Engineering*. IEEE Computer Society, 2011, pp. 223–232.

[21] 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*. ACM, 2007, pp. 244–253.

[22] C. Manning, P. Raghavan, and H. Schütze, "Introduction to information retrieval," *Natural Language Engineering*, vol. 16, no. 1, pp. 100–103, 2010.

[23] D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf, "Learning with local and global consistency," in *Advances in neural information processing systems*, 2004, pp. 321–328.

[24] D. Jurasky and J. H. Martin, "Speech and language processing: An introduction to natural language processing," *Computational Linguistics and Speech Recognition, Prentice Hall: San Francisco*, 2000.

[25] A. Panichella, B. Dit, R. Oliveto, M. Di Penta, D. Poshyvanyk, and A. De Lucia, "How to effectively use topic models for software engineering tasks? an approach based on genetic algorithms," in *Proceedings of the 2013 International Conference on Software Engineering*. IEEE Press, 2013, pp. 522–531.

[26] E. Effa Bella, M.-P. Gervais, R. Bendraou, L. Wouters, and K. Ali, "Semi-supervised approach for recovering traceability links in complex systems," in *Proceedings of the 23rd IEEE International Conference In the Engineering of Complex Computer Systems (ICECCS)*, 2018.

[27] X. Ye, H. Shen, X. Ma, R. Bunescu, and C. Liu, "From word embeddings to document similarities for improved information retrieval in software engineering," in *Proceedings of the 38th international conference on software engineering*. ACM, 2016, pp. 404–415.

[28] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *Empirical Methods in Natural Language Processing (EMNLP)*, 2014, pp. 1532–1543. [Online]. Available: http://www.aclweb.org/anthology/D14-1162

[29] E. A. Holbrook, J. H. Hayes, A. Dekhtyar, and W. Li, "A study of methods for textual satisfaction assessment," *Empirical Software Engineering*, vol. 18, no. 1, pp. 139–176, 2013.

[30] K. Divya, R. Subha, and S. Palaniswami, "Similar words identification using naive and tf-idf method," *International Journal of Information Technology and Computer Science (IJITCS)*, vol. 6, no. 11, p. 42, 2014.

[31] W. B. Frakes and R. Baeza-Yates, *Information retrieval: Data structures & algorithms*. prentice Hall Englewood Cliffs, NJ, 1992, vol. 331.

[32] O. Delalleau, Y. Bengio, and N. Le Roux, "Efficient non-parametric function induction in semi-supervised learning." in *AISTATS*, vol. 27, no. 28, 2005, p. 100.

[33] J. Cleland-Huang, A. Czauderna, A. Dekhtyar, O. Gotel, J. H. Hayes, E. Keenan, G. Leach, J. Maletic, D. Poshyvanyk, Y. Shin *et al.*, "Grand challenges, benchmarks, and tracelab: developing infrastructure for the software traceability research community," in *Proceedings of the 6th international workshop on traceability in emerging forms of software engineering*. ACM, 2011, pp. 17–23.

[34] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.

[35] 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, no. 1, p. 4, 2006.

[36] S. Maro and J.-P. Steghofer, "Capra: A configurable and extendable traceability management tool," in *2016 IEEE 24th International Requirements Engineering Conference (RE)*. IEEE, 2016, pp. 407–408.

[37] J. Cleland-Huang, R. Settimi, C. Duan, and X. Zou, "Utilizing supporting evidence to improve dynamic requirements traceability," in *Requirements Engineering, 2005. Proceedings. 13th IEEE International Conference on*. IEEE, 2005, pp. 135–144.

[38] T.-D. B. Le, D. Lo, and F. Thung, "Should i follow this fault localization tool's output?" *Empirical Software Engineering*, vol. 20, no. 5, pp. 1237–1274, 2015.

[39] N. Sannier and B. Baudry, "Toward multilevel textual requirements traceability using model-driven engineering and information retrieval," in *2012 Second IEEE International Workshop on Model-Driven Requirements Engineering (MoDRE)*. IEEE, 2012, pp. 29–38.

[40] D. Cuddeback, A. Dekhtyar, J. H. Hayes, J. Holden, and W.-K. Kong, "Towards overcoming human analyst fallibility in the requirements tracing process: Nier track," in *2011 33rd International Conference on Software Engineering (ICSE)*. IEEE, 2011, pp. 860–863.

[41] C. Mills and S. Haiduc, "A machine learning approach for determining the validity of traceability links," in *Proceedings of the 39th International Conference on Software Engineering Companion*. IEEE Press, 2017, pp. 121–123.

[42] H. Sultanov and J. H. Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing," in *2013 21st IEEE International Requirements Engineering Conference (RE)*. IEEE, 2013, pp. 52–61.