Natural Language Requirements Processing: a 4D Vision

Alessio Ferrari, Felice Dell'Orletta, Andrea Esuli, Vincenzo Gervasi, Stefania Gnesi

Abstract

Natural language processing (NLP) and requirements engineering (RE) have a long-lasting relation, yet not well-established in industrial practice. We discuss how NLP technologies are applied in RE, and envision their upcoming, intertwined evolution according to four dimensions we propose: Discipline, Dynamism, Domain Knowledge and Datasets (our 4Ds).

Index Terms

D.2.18 Software Engineering Process, I.2.7 Natural Language Processing, D.2.1 Requirements/Specifications

I. Introduction

Requirements are generally expressed with the most human of the communication codes, which is natural language. In requirements engineering (RE), natural language processing (NLP) techniques can be applied to a wide range of tasks [1]. In this paper, we discuss the relevant research applications of NLP to RE, and we propose a 4-(D)imensions framework to provide the conceptual lenses to understand the interplay between the two areas. A first D is discipline, with NLP being used for defect detection. A second D is dynamism, with NLP employed for traceability and categorization. A third one is domain knowledge, since NLP can help to identify domain experts and surf knowledge sources. A fourth, crucial, challenging and cross-cutting D, is datasets, since modern NLP techniques are data hungry, and datasets are still scarce in RE. According to these four dimensions, we shape a vision of the state of the practice when the research in the 4Ds comes to fruition.

II. Discipline

Requirements are an abstract conceptualisation of the system needs that is inherently open to different interpretations. This openness is emphasised by the intrinsically ambiguous means that is commonly used to express requirements, which is natural language (NL). On the other hand, as the requirements process progresses, requirements are expected to be unequivocal enough to be interpreted in the same way by the stakeholders who deal with them. The key for writing unequivocal requirements is **Discipline**.

Several software development standards, such as CENELEC-50128 for railway software, the DO-178C for avionic, and also the IEEE Std 830™-1998(R2009) touch the issue of a *writing* discipline in the development of requirements. Looking at different aspects, they ask requirements to be unequivocal, but none of the standards provides language guidelines to facilitate an agreement in the interpretation of the requirements.

This lack of guidance allows requirements editors to put their individual vision of writing discipline into practice. The adherence to a self-defined rigorous style that reflects the mindset of the editor may lead to a sort of personal jargon, with acrobatic circumlocutions. These apparently enforce precision, but, given their poor readability, they often encourage freedom of interpretation instead.

In this scenario, how to hope for a common writing standard, or set of standards, for writing requirements? A scheme suggested by the RE research community is to employ NLP tools that let editors be aware of the *ambiguity* in their requirements. Ambiguities might cause inconsistencies between the expectation of the customer and the product developed, and possibly lead to undesirable reworks on the artifacts.

A large body of literature is available in RE that presents technical solutions to identify ambiguous expressions ("as possible", "taking into account", etc.) [2] and anaphoric constructions [3], where readers might disagree in the interpretation of pronouns – as in this example requirement: *"When the* system *sends a message to the* receiver*,* it *shall provide an acknowledgement"* (it = "system" or "receiver"?). More recently, solutions for detecting ambiguities have also been proposed that deal with the so called *pragmatic* ambiguities [4], which depend on the background knowledge of the reader, and are oriented to answer questions such as: will a customer and a developer interpret the requirement in the same way? In the previous example, a customer without a technical background might be oriented to interpret "receiver" as a human subject, while a developer is likely to consider "receiver" as a software component.

Less explored in RE, but still relevant, is the concept of *readability* of the requirements. Readability is the property of a text to be understood with limited effort. It is associated to the complexity of the terminology used and to the complexity of the syntax. Research in NLP for readability and text simplification is still ongoing [5], and proper tailoring shall be planned for the RE domain.

We foresee that a more widespread usage of ambiguity checks and readability solutions within leading companies will let writing standards emerge from the practice. This, in turn, will mitigate common editing mistakes. We also argue that the usage of these technologies shall be balanced on the degree of *abstraction* and on the expected reader of a requirement. Indeed, user-level and high-level technical requirements might be characterised by a certain level of uncertainty, and too strict ambiguity checks could not be desirable. Conversely, lower level technical requirements shall be ambiguity free, while one could tolerate poor readability to favor precision. Therefore, along the software process, NL requirements evolve into different forms, that might need different treatments. The evolutionary facet of requirements leads us to discuss another dimension of the problem: **Dynamism**.

III. Dynamism

Requirements are concepts that, as the software process goes on, are justified, re-negotiated, discussed, refined, and gradually evolve into executable artifacts. The trace that this evolution leaves has to be identified and controlled, to ensure that each high-level requirement agreed with the customer is implemented in the product. *Traceability* is the discipline of cross-linking requirements with other requirements, possibly at a different level of abstraction, and with other artifacts – models, software components and tests – of the software process. In a sense, traceability links form the network to control the inherent **Dynamism** of requirements and software-related artifacts.

NLP technologies have been experimented to support the definition of the traceability links from scratch [6], and for updating the links [7], when novel requirements enter into play. Moreover, NLP has also been used to ensure regulatory compliance [8], by automatically tracing requirements to multiple NL regulations.

Traceability is not the only means to enforce control within the dynamism of the requirements along the process. Another prominent task in this sense is requirements *categorisation*. Categorisation helps to manage large amounts of requirements, and drives the apportionment of requirements to specific software components. Moreover, sorting requirements into categories also supports re-using the requirements in different projects.

Automated tools have been developed to incrementally categorise different types of non-functional NL requirements [9], and partition them into different non-functional categories (security, availability, maintainability, usability, etc.). Such technologies can be employed to identify non-functional requirements that might be hidden within large amounts of functional ones. Moreover, approaches have been tested to partition functional NL requirements into functional categories [10] (communication protocol, user interface, etc.). These techniques are particularly useful during the transition from requirements to architectural design, when different requirements have to be apportioned to functional components.

The assessment of requirements traces and categories, which implies the verification that an automatically retrieved trace or category is correct, implies a deep understanding of the project context, and a clear

knowledge of the meaning of the NL content of the requirements. Hence, all the automated mechanisms discussed necessarily need a domain expert in the loop. This observation opens another dimension that we wish to discuss: **Domain Knowledge**.

IV. Domain Knowledge

Requirements belong to different domains with technical or domain-specific jargons. When requirements are shown to a requirements/software analyst or developer who is not confident with the domain, s/he will be likely to stand up and knock at the door of the requirements editor, to ask for explanations. Sometimes the door might be physical – when the editor is in the same workplace –, sometimes might be virtual – when s/he is reachable through phone calls or e-mails –, and sometimes such door might be behind far too many doors to be reachable, and you do not really know who holds the right information. As a result, analysts and developers have to acquire the **Domain Knowledge** needed to understand the requirements by browsing through Internet or by reading internal documentation, when available.

Cleland-Huang [11] suggests applying NLP and data-mining techniques that work by extracting domain specific terms, clustering them by common topics, and manually labelling the topics, to define a sort of personalised conceptual map to scope the domain of interest. Advanced techniques are available in NLP, which allow not only to find topic clusters, but also to reveal fine-grained relationships among relevant terms. To leave appropriate space to discuss such techniques, we will describe a reference example in Box I.

Domain knowledge is needed to interpret requirements already written down in NL, but is also paramount in requirements elicitation, when the requirements for a system are still concealed in their *sources*, such as the system stakeholders, other potentially reusable requirements available to the company, or the public documentation of competing products. NLP can provide help in all these cases. *Recommender systems* have been developed that are geared towards exploiting the domain knowledge of the stakeholders, and associate relevant stakeholders to appropriate requirements discussion forums [12]. Approaches have also been developed that address the problem of reusing internal NL requirements [13], and hence leverage internal domain knowledge encoded in the requirements. Finally, also the exploitation of domain knowledge embedded in marketing material has been explored, with recommender systems that, given a NL description of a novel product to develop, can suggest additional features based on online descriptions of competing products [14].

All the NLP approaches that we have briefly discussed along the different dimensions have something in common. They all need large amounts of NL data to properly work. Therefore, let us go to our fourth dimension: **Datasets**.

V. Datasets

Traditional NLP approaches were relying on the assumption that language was dominated by regularities, which could be listed in the form of *rules* that could extract relevant information from a text, categorise documents, or find relations among sentences. That assumption was wrong. Instead, machine learning (ML) approaches have emerged, which extract statistical information from large amounts of documents, and, in a sense, learn the inherent rules of language without explicitly listing them.

ML approaches need **Datasets** to work. In this context, a dataset is a collection of requirements with annotations that provide task-dependent semantic information about the requirements. For example, in a categorisation task, each requirement is annotated with its category; for ambiguity detection, annotations mark those parts of the text that are ambiguous. The annotation is normally performed by a human operator. The goal of a ML algorithm is to predict the annotation, either based on a subset of the annotated data – in case of *supervised* learning – or without relying on the existing annotations – in case of *unsupervised* learning.

ML approaches have been used in NL requirements for a large variety of tasks: from requirements classification, to the identification of equivalent requirements, to ambiguity detection and to traceability.

Nevertheless, most of these results are not generalisable, since each study focuses on a limited set of requirements in a specific domain. Indeed, not so many requirements datasets exist that are publicly available and that cover multiple domains, and researchers have to work in a context with an evident lack of resources. Generalisation is a key issue, since a technique might not work well in different domains, given the different terminologies, processes, and the mentioned absence of a common discipline. To address this scarcity of datasets, the Center of Excellence for Software Traceability – CoEST (`http://coest.org/`) has recently provided a page where requirements annotated for traceability are available. We encourage similar initiatives also for the other NL-related requirements tasks. Still, it is worth keeping in mind the two main reasons why datasets are scarce in requirements:

1) Domain expertise is needed to annotate the requirements: as domain knowledge is a problem within companies, it is even a harsher problem for researchers who are expert in NLP, but have limited expertise on application domains. This problem requires researchers to provide annotation tools that can be directly used by the domain experts in the companies, and that have a positive impact in their daily work. The example of the categorisation tool employed at Mercedes Benz [15], which suggests categories to the operator and, at the same time, learns how to better categorise, is a reference case in this sense.
2) Requirements often belong to companies and are usually confidential. Solutions to this issue shall be searched within Stand-off markup technologies (`http://wiki.tei-c.org/index.php/Stand-off_markup`), which enable annotations to be separated from the text. In practice, the company can retain the text of the requirements, but can share the annotated features of the text (structure of sentences, relevant terms, class of the requirement), to be employed by researchers to perform their experiments.

A third reason why requirements datasets are rare, is that the usage of requirements – especially in a disciplined, common form that can ease inter-domain comparison of NLP technologies – is not so frequent in the software development practice. The solution involves a hidden dimension of the problem, which is **Dissemination**. As researchers, we are well aware that a proper dissemination of the role of requirements within companies is paramount to realise the **Vision** that we foresee in the following section.

VI. Vision

What will be the state of the practice if the research in all dimensions that we have described comes to fruition? A first answer can be given by placing the different technologies available in an NLP-supported iterative requirements process, as in Fig. **??**. NLP can be applied from requirements elicitation to analysis, where the impact of the domain knowledge dimension is stronger, and from validation to management, where discipline and dynamism become more relevant. But how much should we wait to touch that process, and what are the main driving forces that will shape this vision? Let us look in a ten-year perspective.

a) Discipline: A rigorous writing standard will gradually surface in specific domains. This will be driven not only by NLP tools for discipline – automated approaches that combine template conformance and ambiguity checking have recently been developed [16] –, but also by the current state of the market. Applications tend to emerge as integration of different services and components, developed by different companies. To match functionalities of multiple providers and to support requirements communication among stakeholders that are distributed in a global setting, a common requirements language, possibly complemented with graphical languages similar to AUTOSAR in the automotive sector (`http://www.autosar.org`), will be inevitable. Discipline is also already emerging for specific requirements formats that are independent from the domain: user stories adopted in agile methods shall conform to well-defined templates, and promising quality evaluation approaches are under development [17].

b) Dynamism: Requirements management tools will be equipped with NLP capabilities to handle dynamism. Techniques for traceability and categorisation are mainly based on computing NL similarity between requirements. They are among the most mature in research, and, as shown, e.g., in [15], are

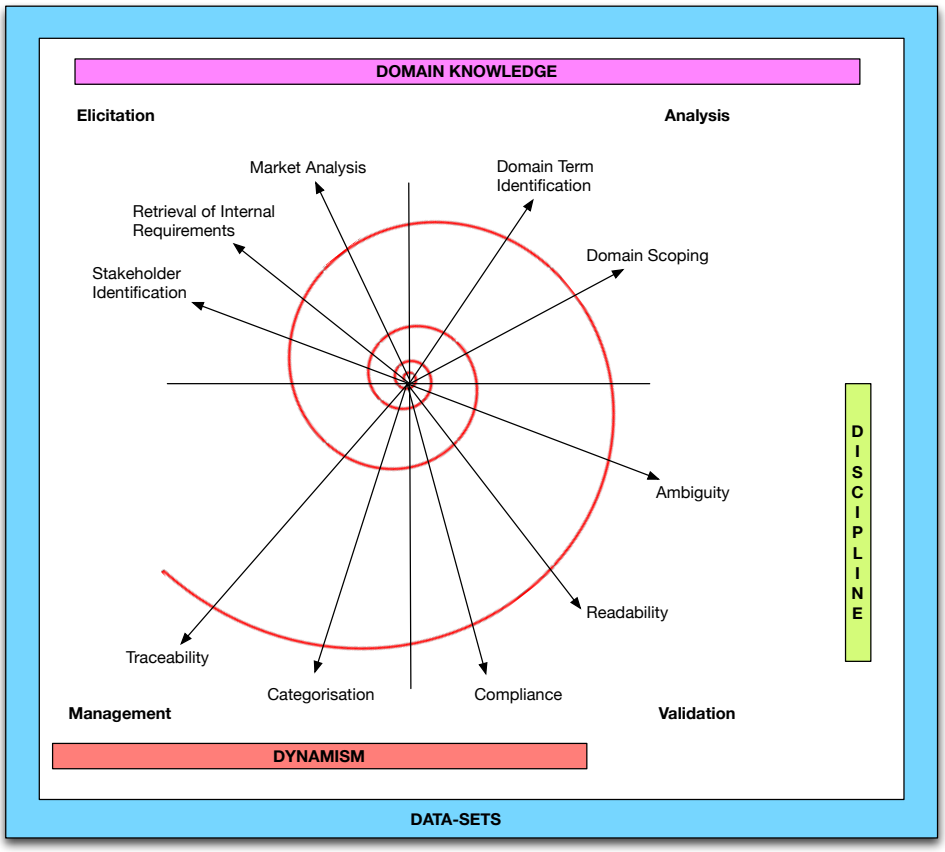

Fig. 1. During elicitation, NLP techniques can be applied to identify stakeholders, internal requirements, and existing product features. In requirements analysis, automated glossary definition and support for domain scoping can also be provided. In the validation phase, requirements defects can be identified with NLP, as well as compliance with regulations. As the process goes on, also traceability and classification can be made easier with the support of NLP.

likely to become industry-ready in short term. The driving force here is the recent widespread diffusion of word embeddings in NLP (Box II). These are semantic-laden word representations that are modelled by observing the linguistic context of a word, and, in turn, can improve the computation of NL similarity with the aid of a context-aware boost. Though these techniques are not applied yet in RE, we conjecture that their introduction in tasks that are already mature, will provide a breakthrough in the dynamism dimension.

c) Domain Knowledge: More domain knowledge will be accessible via human-centered tools. NL-intensive and knowledge-rich collaborative tools, such as SLACK, TRELLO, or FACEBOOK AT WORK, produce vast repositories of NL knowledge. NLP tools will mine these and other internal sources of diffused knowledge, to identify key expertise, and will provide structured access to such knowledge. Intelligent question-answering systems will leverage available knowledge to reply to domain-specific questions. A tangible reference in this sense is IBM WATSON, which, in 2011, won the *Jeopardy!* quiz show against human players. Services of IBM WATSON were recently made available to the large public within the BLUEMIX project (`http://www.ibm.com/cloud-computing/bluemix/watson/`). Another key role will be played by the development of domain-specific ontologies, which are formal representations of concepts and relations in a domain. Such representations are expected to be a formal basis to support traceability and categorisation, and hence positively impact also the dynamism dimension.

d) Datasets: Providing datasets is the most relevant challenge that the RE community needs to address. The growth of publicly available datasets is slow and an inter-domain generalisation of current NLP techniques is hard to envision in a ten-year horizon. Nevertheless, the tight collaboration between researchers and industries, and the availability of NLP tools that can be trained on-the-job with limited

costs, will help to tailor current NLP techniques to specific contexts and domains. Naturally, this requires more dialogue between NLP and RE, and we suggest software companies to include people with NLP background in their personnel, and to train their personnel in NLP. A first, simple step would be asking their requirements engineers to download and start playing with GATE (`https://gate.ac.uk`), an engineer-friendly tool for text analysis with extensive documentation, or go for software libraries like PYTHON NLTK (`http://www.nltk.org`) or OPENNLP (`https://opennlp.apache.org`), for those more inclined to coding.

VII. CONCLUSION

Each company has its own requirements process, adapted to the dimension of the company, the safety integrity level of the products, and to the degree of specialisation of the company in its domain. Therefore, we foresee that NLP techniques for RE shall be supported by a *process-cognisant*, component-based infrastructure, which can be tailored according to dominant needs of the company – e.g., dynamism and discipline are essential for safety critical-systems, while domain knowledge technologies are crucial for consumer products. In this sense, the torch passes on to software engineers, who are called to design such a customisable platform, picking the technologies from those that we have outlined, and enabling their combination based on the process in Fig. **??**.

Of course, NLP will provide support only for those aspects of RE that are dominated by NL. Tacit knowledge, socio-cultural issues, design decisions, user interface requirements and hybrid control systems requirements are just a small set of RE problems and artifacts in which NL plays a collateral role. To account for these aspects, other means are needed that go far beyond NLP capabilities. Nevertheless, when planning their investments in model-based techniques, or in agile methods, companies should keep in mind that, within few years, NLP will be able to extract much more information from prescriptive statements, user stories, textual use cases and all the different forms that NL requirements can take.

REFERENCES

[1] A. Casamayor, D. Godoy, and M. Campo, "Mining textual requirements to assist architectural software design: a state of the art review," *AIR*, vol. 38, no. 3, pp. 173–191, 2012.

[2] B. Gleich, O. Creighton, and L. Kof, "Ambiguity detection: Towards a tool explaining ambiguity sources," in *REFSQ*. Springer, 2010, pp. 218–232.

[3] H. Yang, A. De Roeck, V. Gervasi, A. Willis, and B. Nuseibeh, "Analysing anaphoric ambiguity in natural language requirements," *RE*, vol. 16, no. 3, pp. 163–189, 2011.

[4] A. Ferrari and S. Gnesi, "Using collective intelligence to detect pragmatic ambiguities," in *RE*. IEEE, 2012, pp. 191–200.

[5] K. Collins-Thompson, "Computational assessment of text readability: A survey of current and future research," *ITL*, vol. 165, no. 2, pp. 97–135, 2014.

[6] H. Sultanov and J. H. Hayes, "Application of reinforcement learning to requirements engineering: requirements tracing," in *RE*. IEEE, 2013, pp. 52–61.

[7] V. Gervasi and D. Zowghi, "Supporting traceability through affinity mining," in *RE*. IEEE, 2014, pp. 143–152.

[8] J. Cleland-Huang, A. Czauderna, M. Gibiec, and J. Emenecker, "A machine learning approach for tracing regulatory codes to product specific requirements," in *ICSE*. ACM, 2010, pp. 155–164.

[9] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *IST*, vol. 52, no. 4, pp. 436–445, 2010.

[10] ——, "Functional grouping of natural language requirements for assistance in architectural software design," *KBS*, vol. 30, pp. 78–86, 2012.

[11] J. Cleland-Huang, "Mining domain knowledge," *IEEE Software*, vol. 32, pp. 16–19, 2015.

[12] C. Castro-Herrera, C. Duan, J. Cleland-Huang, and B. Mobasher, "A recommender system for requirements elicitation in large-scale software projects," in *SAC*. ACM, 2009, pp. 1419–1426.

[13] V. Alves, C. Schwanninger, L. Barbosa, A. Rashid, P. Sawyer, P. Rayson, C. Pohl, and A. Rummler, "An exploratory study of information retrieval techniques in domain analysis," in *SPLC*. IEEE, 2008, pp. 67–76.

[14] J.-M. Davril, E. Delfosse, N. Hariri, M. Acher, J. Cleland-Huang, and P. Heymans, "Feature model extraction from large collections of informal product descriptions," in *FSE*. ACM, 2013, pp. 290–300.

[15] E. Knauss and D. Ott, "(semi-) automatic categorization of natural language requirements," in *REFSQ*. Springer, 2014, pp. 39–54.

[16] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated checking of conformance to requirements templates using natural language processing," *TSE*, vol. 41, no. 10, pp. 944–968, 2015.

[17] G. Lucassen, F. Dalpiaz, J. M. E. van der Werf, and S. Brinkkemper, "Forging high-quality user stories: Towards a discipline for agile requirements," in *RE*. IEEE, 2015.

BOX I
TEXT TO KNOWLEDGE

Text-to-Knowledge (T2K - `http://t2k.italianlp.it`) [1] is a Web application that supports the extraction of domain-specific information from unstructured text (English or Italian). The tool renders relevant domain entities in a knowledge graph, an intuitive visual format that eases the identification of relations among entities. When applied on requirements, the tool supports glossary definition and domain scoping, and allows browsing of complex requirements documents in a topic-based, personalised way. In Fig. **??**, we show the results of applying the tool on requirements from the railway domain (EIRENE Functional Requirements Specification v.7.4.0 - `http://www.uic.org/IMG/pdf/p0028d003.4r0.4-7.4.0.pdf`). The tool creates a glossary of domain-specific entities for the document (1), which the user employs as pointers to surf the Web or internal repositories, to identify associated domain-specific information (2). Then, the user selects the entity on which s/he wishes to focus, and visualises its relations with other entities in the requirements (3). From this representation, the user can go back to the original document, to browse the textual context of the entities. Finally, the user can expand the graph on interesting nodes, to continue surfing the requirements (4).

[1] F. Dell'Orletta, G.Venturi, A.Cimino, and S.Montemagni, "T2K: a system for automatically extracting and organizing knowledge from texts," in LREC, 2014, pp. 2062-2070.

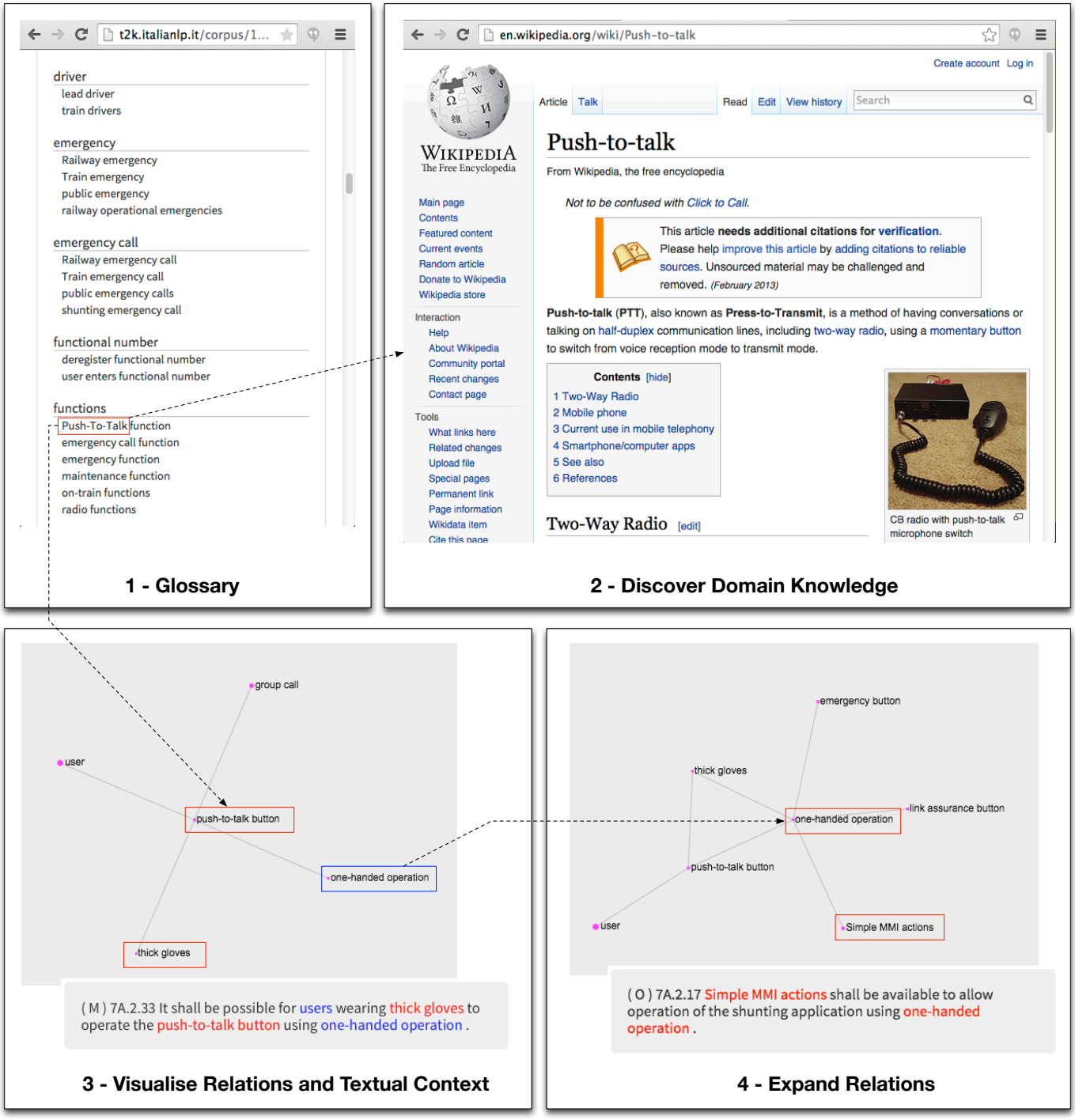

Fig. 2. Application of the T2K tool to a requirements standard of the railway domain. First, a structured glossary was automatically generated from the original document (1). The user saw the term “Push-to-Talk”, unknown to her/him, and searched it in Wikipedia (2). Once gained an overview of the meaning of the concept, s/he went back to T2K, and, by browsing the entities including the “Push-to-Talk” term, s/he selected one named “push-to-talk button” – which, in his/her view, could have had something to do with the “momentary button” mentioned in Wikipedia. S/he visualised the relations of such entity with others extracted from the original document, and, considering the “thick gloves” entity interesting, asked T2K to show the requirement(s) where “push-to-talk button” occurred together with “thick gloves” (3). Then, the user decided to expand the graph on the “one-handed operation” node, and continued surfing the document following the additional relations discovered.

BOX II
DISTRIBUTIONAL SEMANTIC MODELS & WORD EMBEDDINGS

Distributional semantic models (DSM) follow Harris distributional hypothesis [1] that "a word is characterized by the company it keeps". DSM focus on acquiring the semantic of words through the statistical analysis of their use in language. Research on DSM dates back to the nineties, but it has recently gained renewed interest, especially due to the successful application of neural network models to the task.

One of the most popular results is Mikolov's word2vec [2]. Word2vec creates word embeddings, WEs, that are numeric vectors that capture the semantic and syntactic properties of words. WEs are the by-product of the training of a neural network that predicts the most probable word given a set of context words. Training contexts are usually sampled from a large collection of text (e.g. Wikipedia), not requiring any kind of human annotation.

WE vectors find application in word-relatedness problems, such as finding the words similar to a given one by picking the words whose vectors are closer to the given one (see a demo at `http://radimrehurek.com/2014/02/word2vec-tutorial/`). A paradigmatic example from Mikolov's work is that the sum of vectors for words "Germany" and "capital" is a vector close to the vector of "Berlin".

WE vectors can also be an effective substitute of the typical "feature engineering" process, in which many NLP tools enrich the representation of text (performing POS tagging, lemmatization, disambiguation). When working on specific domains, which is a common situation in RE, NLP tools may perform poorly, because they are usually trained on non-domain-specific examples. Training an NLP tool on a new domain requires a human labeled training set, while word2vec only needs plain text from that domain to capture the domain-specific use of language.

WEs are thus a low-cost tool to model the domain-specific relations between terms, e.g., to spot potential sources of ambiguities, leveraging on the observed term similarities in order to converge on a unified lexicon. Similarly, comparing WEs generated on the same domain but from documents produced by different stakeholders (e.g., developers, end users) can support the identification of terms whose semantic is different among the stakeholders, thus avoiding misunderstandings between them.

[1] Z. S. Harris, "Distributional structure." Word, vol. 32, no. 3, 1954, pp. 146-162.
[2] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in NIPS, 2013, pp. 3111-3119.

Alessio Ferrari is research fellow at CNR-ISTI (Consiglio Nazionale delle Ricerche – Istituto di Scienza e Tecnologia dellInformazione, Pisa). His research focuses on NLP applied to RE. Contact: alessio.ferrari@isti.cnr.it.

Felice Dell'Orletta is research scientist at CNR-ILC (Consiglio Nazionale delle Ricerche – Istituto di Linguistica Computazionale, Pisa) and head of the *ItaliaNLP Lab*. His research focuses on NLP and knowledge extraction. Contact: felice.dellorletta@ilc.cnr.it.

Andrea Esuli is research scientist at CNR-ISTI. His research focuses on ML technologies applied to NLP. Contact: andrea.esuli@isti.cnr.it.

Vincenzo Gervasi is associate professor at the CS Department of the University of Pisa. His research focuses on NLP applied to RE. Contact: gervasi@di.unipi.it.

Stefania Gnesi is director of research at CNR-ISTI, and head of the *FMT Lab*. Her research focuses on formal methods applied to RE. Contact: stefania.gnesi@isti.cnr.it.