Natural Language Requirements Processing

A 4D Vision

Alessio Ferrari, CNR-ISTI

Felice Dell'Orletta, CNR-ILC

Andrea Esuli, CNR-ISTI

Vincenzo Gervasi, University of Pisa

Stefania Gnesi, CNR-ISTI

// The future evolution of the application of natural language processing technologies in requirements engineering can be viewed from four dimensions: discipline, dynamism, domain knowledge, and datasets. //

REQUIREMENTS ARE GENERALLY expressed with the most human of the communication codes: natural language (NL). In requirements engineering (RE), natural language processing (NLP) techniques can be applied to a wide range of tasks.[1] Here, we discuss the relevant research applications of NLP to RE and propose a 4D framework that provides the conceptual lenses to understand the interplay between the two areas.

The first dimension is *discipline*, with NLP being used for defect detection. The second is *dynamism*, with NLP employed for traceability and categorization. The third is *domain knowledge*, because NLP can help users identify domain experts and surf knowledge sources. The fourth, crucial, challenging, and cross-cutting dimension is *datasets*, because modern NLP techniques are data hungry and datasets are still scarce in RE.

Discipline

Requirements are an abstract conceptualization of system needs that's inherently open to interpretation. This openness is emphasized by the intrinsic ambiguity of the NL used to express requirements. On the other hand, as the requirements process progresses, requirements are expected to be unequivocal enough to be interpreted in the same way by all 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, DO-178C for avionics, and IEEE Standard 830TM-1998 (R2009), touch on the issue of a writing discipline in requirements development. Looking at different aspects, they ask requirements to be unequivocal. However, none of them provide language guidelines to facilitate agreement on the requirements' interpretation.

This lack of guidance lets requirements editors put into practice their individual vision of a writing discipline. The adherence to a self-defined rigorous style reflecting the editor's mind-set could lead to a somewhat

0740-7459/17/$33.00 © 2017 IEEE

personal jargon, with acrobatic circumlocutions. This might appear to enforce precision but, given the poor readability, often encourages freedom of interpretation instead.

In this scenario, how can we hope for a common standard, or set of standards, for writing requirements? The RE research community has suggested employing NLP tools that make editors aware of their requirements' ambiguity. This ambiguity might cause inconsistencies between customer expectations and the developed product, possibly leading to undesirable rework on the product.

A large body of RE literature presents technical solutions to identify ambiguous expressions ("as possible," "taking into account," and so on)[2] and anaphoric constructions,[3] for which readers might interpret pronouns differently. For example, in "when the system sends a message to the receiver, it will provide an acknowledgment," does "it" mean the system or the receiver? More recently, researchers have proposed solutions for detecting *pragmatic ambiguities*,[4] which depend on the reader's background knowledge. These solutions aim to answer questions such as, will a customer and developer interpret the requirement the same way? In our previous example, a customer with no technical background might interpret "receiver" as a human subject, whereas a developer will likely consider "receiver" as a software component.

Less explored in RE, but still relevant, is requirements readability—the text's ability to be easily understood. This is associated with the complexity of the text's terminology and syntax. NLP research on readability and text simplification is ongoing,[5] and scientists plan to tailor this research for RE.

We foresee that leading companies' wider use of ambiguity checks and readability solutions will cause writing standards to emerge. This, in turn, will mitigate common editing mistakes. We also predict that these technologies' use will find a balance between the degree of abstraction and a requirement's expected readers. Indeed, user- and high-level technical requirements might exhibit some uncertainty, and overly strict ambiguity checks might be undesirable. Conversely, lower-level technical requirements should be ambiguity free, in some cases exhibiting poor readability but high precision. So, during software development, NL requirements will evolve into different forms that might need different treatments. This evolutionary characteristic leads us to the next dimension: dynamism.

> Modern NLP techniques are data hungry, and datasets are still scarce in requirements engineering.

Dynamism

During development, requirements are discussed, justified, renegotiated, and refined, gradually evolving into executable artifacts. This evolution's trace must be identified and controlled, to ensure that the product implements each high-level requirement the customer agreed on. Traceability is the discipline of cross-linking requirements with other requirements, possibly at a different abstraction level, and with other artifacts of the software process—models, software components, and tests. In a sense, traceability links form the network controlling the inherent dynamism of requirements and software-related artifacts.

Researchers have employed NLP to help define traceability links from scratch[6] and update them[7] when novel requirements enter into play. Moreover, NLP has been used to ensure regulatory compliance by automatically tracing requirements to multiple NL regulations.[8]

Another prominent task that relates to dynamism is requirements categorization, which helps manage large numbers of requirements and drives the apportionment of requirements to specific software components. Moreover, categorizing requirements also supports their reuse in different projects.

Researchers have developed automated tools to incrementally categorize types of nonfunctional NL requirements and partition them into nonfunctional categories (security, availability, maintainability, usability, and so on).[9] Such technologies can help identify nonfunctional requirements that might be hidden in large numbers of functional ones. Moreover, approaches have been tested to partition functional NL requirements into functional categories (communication protocols, user interfaces, and so on).[10] These

TEXT-TO-KNOWLEDGE

Text-to-Knowledge (T2K; t2k.italianlp.it) is a web app for extracting domain-specific information from unstructured text in English or Italian.[1] T2K renders relevant domain entities in a knowledge graph, an intuitive visual format that eases identifying relations among entities. When applied to requirements, the app supports glossary definition and domain scoping and allows browsing of complex requirements documents in a topic-based, personalized way.

Figure A shows the results of applying T2K to requirements from the railway domain.[2] First, T2K automatically generates a structured glossary from the original document (see Figure A1). The user sees the term "Push-to-Talk" and searches for it in Wikipedia (see Figure A2). Once the user gains an overview of the concept's meaning, he or she goes back to T2K. After browsing the entities that include the string "Push-to-Talk," the user selects "push-to-talk button," which he or she feels might have something to do with the "momentary button" mentioned in Wikipedia. The user visualizes the relations of that entity and others extracted from the original document. Considering the "thick gloves" entity interesting, the user asks T2K to show the requirements in which "push-to-talk button" occurs with "thick gloves" (see Figure A3). Finally, the user expands the graph on the "one-handed operation" node and continues surfing the document, following the additional relations discovered (see Figure A4).

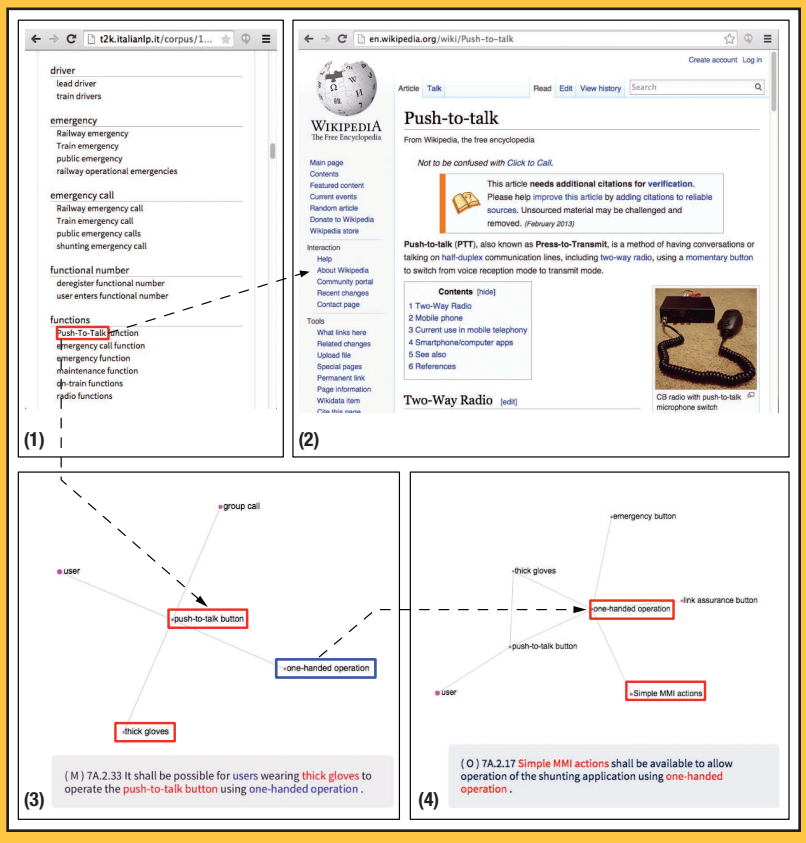

FIGURE A. Applying the Text-to-Knowledge (T2K) web app to a railway requirements standard. (1) The app creates a glossary of domain-specific entities for the document. (2) The user employs the glossary entities as pointers to surf the web or internal repositories, to identify associated domain-specific information. (3) The user selects an entity on which to focus and visualizes its relations with other entities in the requirements. From this representation, the user can go back to the original document, to browse the entities' textual context. (4) The user expands the graph on interesting nodes, to continue surfing the requirements.

References

1. F. Dell'Orletta et al., "T2K: A System for Automatically Extracting and Organizing Knowledge from Texts," *Proc. 9th Int'l Conf. Language Resources and Evaluation* (LREC 14), 2014, pp. 2062–2070.
2. *EIRENE Functional Requirements Specification Version 7.4.0*, GSM-R Functional Group, 2014; www.uic.org/IMG/pdf/p0028d003.4r0.4-7.4.0.pdf.

techniques are particularly useful during the transition from requirements to architectural design, when different requirements must be apportioned to functional components.

The assessment of requirements traces and categories implies verification that an automatically retrieved trace or category is correct. It also implies a deep understanding of the project context and a clear knowledge of the meaning of the requirements' NL content. So, the automated mechanisms we discussed require a domain expert in the loop. This observation opens up the next dimension: domain knowledge.

Domain Knowledge

Requirements belong to different domains with technical or domain-specific jargons. When a requirements analyst, or a developer, who isn't confident with the domain has to interpret domain-specific requirements, he or she will likely go knock on the requirements editor's door to ask for explanations. The door might be physical (for example, when the editor is in the same workplace) or virtual (for example, when he or she is reachable through phone calls or email). Sometimes, the door might be behind so many other doors that it's unreachable, and the analyst or developer doesn't really know who holds the right information. In that case, the analyst or developer must acquire the necessary domain knowledge by browsing the Internet or reading the available internal documentation.

Jane Cleland-Huang suggested applying NLP and data-mining techniques that extract domain-specific terms, clustering them by common topics, and then manually labeling them, to define a personalized conceptual map to scope the domain of interest.[11] Some advanced NLP techniques let users not only find topic clusters but also discover fine-grained relationships among relevant terms. For a discussion of one such technique, see the sidebar, "Text-to-Knowledge."

Domain knowledge is also needed for requirements elicitation. In this phase, a system's requirements are still concealed in their sources, such as the system stakeholders, other potentially reusable requirements available to the company, or competing products' public documentation.

NLP can help in these cases. Some recommender systems help exploit stakeholders' domain knowledge and direct them to appropriate requirements discussion forums.[12] Some approaches support the reuse of internal NL requirements,[13] thus leveraging the domain knowledge encoded in them. Finally, recommender systems exist that exploit domain knowledge embedded in marketing material.[14] Given an NL description of a novel product to develop, these systems suggest additional features derived from online descriptions of competing products.

All these NLP approaches need large amounts of NL data to work properly. So, let's go to our final dimension: datasets.

Datasets

Traditional NLP approaches relied 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, categorize documents, or find relations among sentences. That assumption was wrong. Instead, machine learning (ML) approaches emerged that 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. In our context, a dataset is a collection of requirements with annotations providing task-dependent semantic information about them. For example, in a categorization task, each requirement is annotated with its category; for ambiguity detection, annotations mark the text's ambiguous parts. A human normally performs the annotation. An ML algorithm aims to predict the annotation, either on the basis of a subset of the annotated data, as in the case of supervised learning, or without relying on the existing annotations, as in the case of unsupervised learning.

ML approaches have been used in NL requirements for tasks such as requirements classification, identification of equivalent requirements, ambiguity detection, and traceability. Nevertheless, most of these results aren't generalizable because each study focused on a limited set of requirements in a specific domain. Indeed, not many requirements datasets covering multiple domains are publicly available, and researchers must work with a lack of resources. Generalization is a key issue because a technique might not work well in different domains, given the different terminologies and processes and the absence of a common discipline.

To address this dataset scarcity, the Center of Excellence for Software Traceability (CoEST; coest.org) provides a page containing requirements annotated for traceability. We encourage similar initiatives for other NL-related requirements tasks. Still, it's worth keeping in mind three reasons why requirements datasets are scarce.

First, requirements annotation requires domain expertise. Because

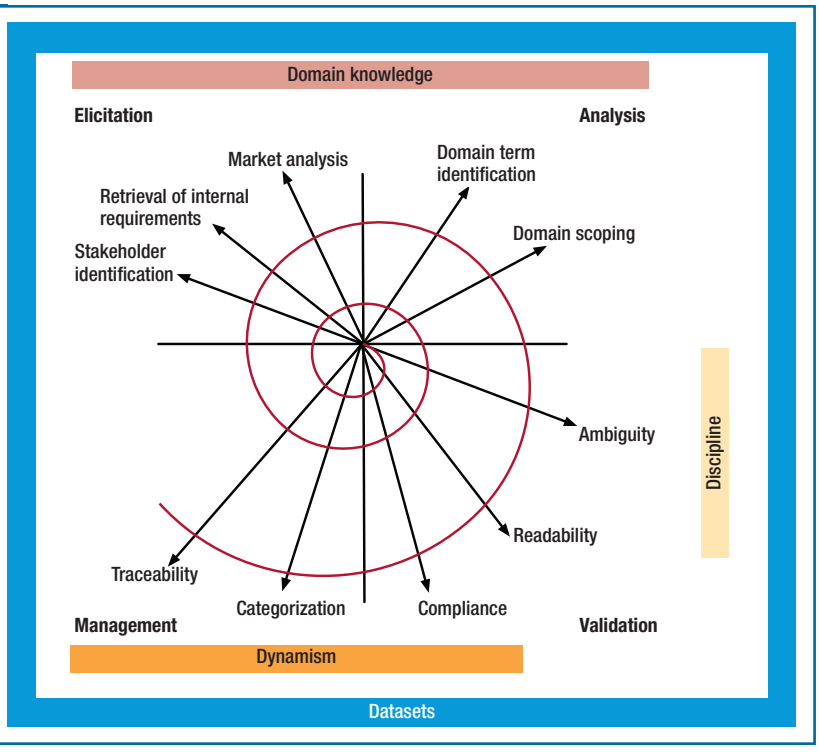

FIGURE 1. How the four dimensions (discipline, dynamism, domain knowledge, and datasets) fit a software requirements process supported by natural language processing (NLP). During requirements elicitation, NLP can help identify stakeholders, internal requirements, and existing product features. During requirements analysis, automated glossary definition and support for domain scoping can be provided. During validation, NLP can identify requirements defects and regulatory compliance. As the process continues, NLP can also make traceability and classification easier.

domain knowledge is a problem in companies, it's even a harsher problem for researchers who are NLP experts but have limited expertise in application domains. This problem requires these researchers to provide annotation tools that domain experts in the companies can use productively. For example, Mercedes Benz employs a categorization tool that suggests categories to users while learning how to better categorize.[15]

Second, requirements often belong to companies and are confidential. Stand-off markup technologies (wiki.tei-c.org/index.php/Stand-off_markup), which separate annotations from the text, can help solve this problem. With these technologies, companies can retain the requirements' text but share the text's annotated features (such as sentence structure, relevant terms, and the requirement's class), which researchers can use in their experiments.

Finally, the use of requirements—especially in a disciplined, common form that can ease interdomain comparison of NLP technologies—isn't that common in software development practice. The solution involves a hidden dimension of the problem, which is dissemination. As researchers, we're well aware that the proper dissemination of the role of requirements within companies is paramount to realize our vision, which we describe next.

Our Vision

What will be the state of the practice if the research in the four dimensions we described comes to fruition? We can formulate an initial answer by charting the technologies available in an NLP-supported iterative requirements process (see Figure 1). NLP can be applied from requirements elicitation to analysis, in which domain knowledge has more impact, and from validation to management, in which discipline and dynamism are more relevant. But how long will it be before we see that process, and what forces will shape this vision? Let's look at a 10-year perspective.

Discipline

A rigorous writing standard will gradually surface in specific domains. This will be driven by not only NLP tools for discipline—researchers have recently developed automated approaches that combine template conformance and ambiguity checking[16]—but also the current state of the market. Applications tend to emerge as the integration of different services and components, developed by different companies. To match multiple providers' functionalities and to support requirements communication among globally distributed stakeholders, a common requirements language, possibly complemented with graphical languages similar to AUTOSAR (*Aut*omotive Open *S*ystem *Ar*chitecture; www.autosar.org), is inevitable.

Discipline is also already emerging for specific domain-independent requirements formats. User stories adopted in agile methods will conform

to well-defined templates, and researchers are developing promising quality evaluation approaches.[17]

Dynamism

Requirements management tools will implement NLP to handle dynamism. Traceability and categorization techniques are based mainly on computing NL similarity between requirements. They are among the most mature in research and will likely become industry-ready in short term.[15] The driving force here is the recent widespread diffusion of word embeddings in NLP (see the sidebar "Distributional Semantic Models and Word Embeddings"). These are semantic-laden word representations that are modeled through observation of a word's linguistic context; they can improve the computation of NL similarity by boosting context awareness. Although these techniques haven't been applied yet in RE, we conjecture that their introduction in tasks that are already mature will provide a breakthrough in dynamism.

Domain Knowledge

More domain knowledge will be accessible through human-centered tools. NL-intensive and knowledge-rich collaborative tools, such as Slack, Trello, or Workplace by Facebook, 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's Watson, which in 2011 beat human players on the *Jeopardy!* quiz show. Watson's services were recently made available to the large public within

DISTRIBUTIONAL SEMANTIC MODELS AND WORD EMBEDDINGS

Distributional semantic models (DSMs) follow Zellig Harris's distributional hypothesis that "a word is characterized by the company it keeps."[1] DSMs focus on acquiring the semantics of words through statistical analysis of their use in language. DSM research dates back to the 1990s but has recently gained renewed interest, owing especially to the successful application of neural-network models to the task.

One of the most popular results is word2vec,[2] which creates word embeddings (WEs)—numeric vectors that capture words' semantic and syntactic properties. 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 (for example, Wikipedia), without requiring 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 radimrehurek.com/2014/02/word2vec-tutorial). For example, the sum of the vectors for "Germany" and "capital" is a vector close to the vector of "Berlin." [2]

WE vectors can also be an effective substitute for the typical feature-engineering process, in which many natural language processing (NLP) tools enrich the representation of text (performing part-of-speech tagging, lemmatization, and disambiguation). When working in specific domains, which occurs frequently in requirements engineering, NLP tools might perform poorly because they're usually trained on non-domain-specific examples. Training an NLP tool on a new domain requires a human-labeled training set, whereas word2vec needs only plain text from that domain to capture the domain-specific use of language.

So, WEs are a low-cost tool to model the domain-specific relations between terms. For example, they can help spot potential sources of ambiguities, leveraging on the observed term similarities to converge on a unified lexicon. Similarly, comparing WEs generated for the same domain but from documents from different stakeholders (for example, developers or users) can support the identification of terms whose semantics differ among the stakeholders, thus avoiding misunderstandings between them.

References

1. Z.S. Harris, "Distributional Structure," *Word*, vol. 32, no. 3, 1954, pp. 146–162.
2. T. Mikolov et al., "Distributed Representations of Words and Phrases and Their Compositionality," *Proc. 2013 Conf. Advances in Neural Information Processing Systems* (NIPS 13), 2013, pp. 3111–3119.

the Bluemix project (www.ibm.com/cloud-computing/bluemix/watson).

Also crucial will be the development of domain-specific ontologies, which are formal representations of concepts and relations in a domain. Such representations will likely be a formal basis for supporting traceability and categorization and will thus also positively impact dynamism.

Datasets

Providing datasets is the most relevant challenge the RE community must address. The growth of publicly available datasets is slow, and it's hard to imagine the interdomain generalization of current NLP techniques in 10 years. Nevertheless, the tight collaboration between researchers and industries and the availability of NLP tools that can be inexpensively trained on the job will help tailor current NLP techniques to specific contexts and domains.

Naturally, this requires more dialogue between NLP and RE. Software companies should hire people with an NLP background and train their personnel in NLP. A first, simple step would be asking their requirements engineers to download and start playing with GATE (General Architecture for Test Engineering; gate.ac.uk), an engineer-friendly tool for text analysis with extensive documentation. Requirements engineers more inclined toward coding can go for software libraries such as the Python Natural Language Toolkit (NLTK; www.nltk.org) or Apache OpenNLP (opennlp.apache.org).

Each company has a requirements process adapted to its size, its products' safety integrity level, and its degree of specialization in its domain. So, we foresee that NLP techniques for RE will be supported by a process-cognizant, component-based infrastructure that can be tailored to a company's dominant needs. For example, dynamism and discipline are essential for safety-critical systems, whereas domain knowledge technologies are crucial for consumer products. In this sense, the torch passes to software engineers, who must design such a customizable platform, picking the technologies from those we discussed and enabling their combination on the basis of the process in Figure 1.

Of course, NLP will support only those RE aspects that are dominated by NL. Tacit knowledge, sociocultural issues, design decisions, UI 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 will require means that transcend NLP capabilities. Nevertheless, when planning investments in model-based techniques or agile methods, companies should keep in mind that, within a few years, NLP will be able to extract much more information from prescriptive statements, user stories, textual use cases, and all the forms 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," *Artificial Intelligence Rev.*, vol. 38, no. 3, 2012, pp. 173–191.
2. B. Gleich, O. Creighton, and L. Kof, "Ambiguity Detection: Towards a Tool Explaining Ambiguity Sources," *Requirements Engineering: Foundation for Software Quality*, LNCS 6182, Springer, 2010, pp. 218–232.
3. H. Yang et al., "Analysing Anaphoric Ambiguity in Natural Language Requirements," *Requirements Eng.*, vol. 16, no. 3, 2011, pp. 163–189.
4. A. Ferrari and S. Gnesi, "Using Collective Intelligence to Detect Pragmatic Ambiguities," *Proc. 20th IEEE Int'l Requirements Eng. Conf.* (RE 12), 2012, pp. 191–200.
5. K. Collins-Thompson, "Computational Assessment of Text Readability: A Survey of Current and Future Research," *Int'l J. Applied Linguistics*, vol. 165, no. 2, 2014, pp. 97–135.
6. H. Sultanov and J.H. Hayes, "Application of Reinforcement Learning to Requirements Engineering: Requirements Tracing," *Proc. 21st IEEE Int'l Requirements Eng. Conf.* (RE 13), 2013, pp. 52–61.
7. V. Gervasi and D. Zowghi, "Supporting Traceability through Affinity Mining," *Proc. 22nd IEEE Int'l Requirements Eng. Conf.* (RE 14), 2014, pp. 143–152.
8. J. Cleland-Huang et al., "A Machine Learning Approach for Tracing Regulatory Codes to Product Specific Requirements," *Proc. 32nd ACM/IEEE Int'l Conf. Software Eng.* (ICSE 10), 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," *Information and Software Technology*, vol. 52, no. 4, 2010, pp. 436–445.
10. A. Casamayor, D. Godoy, and M. Campo, "Functional Grouping of Natural Language Requirements for Assistance in Architectural Software Design," *Knowledge-Based Systems*, June 2012, pp. 78–86.
11. J. Cleland-Huang, "Mining Domain Knowledge," *IEEE Software*, vol. 32, no. 3, 2015, pp. 16–19.
12. C. Castro-Herrera et al., "A Recommender System for Requirements Elicitation in Large-Scale Software

ABOUT THE AUTHORS

ALESSIO FERRARI is a research fellow at CNR-ISTI (Consiglio Nazionale delle Ricerche—Istituto di Scienza e Tecnologia dell'Informazione). His research focuses on natural language processing applied to requirements engineering. Ferrari received a PhD in computer engineering from the University of Florence. Contact him at alessio.ferrari@isti.cnr.it.

FELICE DELL'ORLETTA is a research scientist at CNR-ILC (Consiglio Nazionale delle Ricerche—Istituto di Linguistica Computazionale) and head of the ItaliaNLP Lab. His research focuses on natural language processing and knowledge extraction. Dell'Orletta received a PhD in Informatics from the University of Pisa. Contact him at felice.dellorletta@ilc.cnr.it.

ANDREA ESULI is a research scientist at CNR-ISTI (Consiglio Nazionale delle Ricerche—Istituto di Scienza e Tecnologia dell'Informazione). His research focuses on machine-learning technologies applied to natural language processing. Esuli received a PhD in information engineering from the University of Pisa. Contact him at andrea.esuli@isti.cnr.it.

VINCENZO GERVASI is an associate professor in the University of Pisa's Computer Science Department. His research focuses on natural language processing applied to requirements engineering, formal specifications, and software architectures. Gervasi received a PhD in computer science from the University of Pisa. Contact him at gervasi@di.unipi.it.

STEFANIA GNESI is a director of research at CNR-ISTI (Consiglio Nazionale delle Ricerche—Istituto di Scienza e Tecnologia dell'Informazione) and the head of the Formal Methods && Tools Group. Her research focuses on formal methods applied to requirements engineering. Gnesi received a master's in computer science from the University of Pisa. Contact her at stefania.gnesi@isti.cnr.it

Projects," *Proc. 2009 ACM Symp. Applied Computing* (SAC 09), 2009, pp. 1419–1426.

13. V. Alves et al., "An Exploratory Study of Information Retrieval Techniques in Domain Analysis," *Proc. 12th Int'l Software Product Line Conf.* (SPLC 08), 2008, pp. 67–76.
14. J.-M. Davril et al., "Feature Model Extraction from Large Collections of Informal Product Descriptions," *Proc. 9th Joint Meeting Foundations of Software Eng.* (ESEC/FSE 13), 2013, pp. 290–300.
15. E. Knauss and D. Ott, "(Semi-) automatic Categorization of Natural Language Requirements," *Requirements Engineering: Foundation for Software Quality*, LNCS 8396, Springer, 2014, pp. 39–54.
16. C. Arora et al., "Automated Checking of Conformance to Requirements Templates Using Natural Language Processing," *IEEE Trans. Software Eng.*, vol. 41, no. 10, 2015, pp. 944–968.
17. G. Lucassen et al., "Forging High-Quality User Stories: Towards a Discipline for Agile Requirements," *Proc. 23rd IEEE Int'l Requirements Eng. Conf.* (RE 15), 2015, pp. 126–135.