JETIR.ORG ISSN: 2349-5162 | ESTD Year : 2014 | Monthly Issue
JOURNAL OF EMERGING TECHNOLOGIES AND INNOVATIVE RESEARCH (JETIR)
An International Scholarly Open Access, Peer-reviewed, Refereed Journal

Natural Language Processing Techniques Using Deep learning ANN

Gadipally Prashanth[1],
Research Scholar, Dept of ECE, Email
Dr.Seetharam Khetavath[2],
Asst.Professor, Dept of ECE, Email
 # Chaitanya Deemed to be University, Warangal, TS, INDIA

Abstract:

Research in applying natural language processing (NLP) techniques to requirements engineering (RE) tasks spans more than 40 years, from initial efforts carried out in the 1980s to more recent attempts with machine learning (ML) and deep learning (DL) techniques. However, in spite of the progress, our recent survey shows that there is still a lack of systematic understanding and organization of commonly used NLP techniques in RE. We believe one hurdle facing the industry is the lack of shared knowledge of NLP techniques and their usage in RE tasks. In this paper, we present our effort to synthesize and organize the 57 most frequently used NLP techniques in RE. We classify these NLP techniques in two ways: first, by their NLP tasks in typical pipelines and second, by their linguist analysis levels. We believe these two ways of classification are complementary, contributing to a better understanding of the NLP techniques in RE and such understanding is crucial to the development of better NLP tools for RE.

Keywords: Requirements Engineering (RE), Natural Language Processing (NLP), NLP Tasks, NLP Techniques

I. INTRODUCTION:

Research in developing natural language processing (NLP) support for requirements engineering (RE), or NLP4RE for short, dates back to the early 1980s and has seen a continuous flow of contributions in the past 40 years [1]–[4]. However, in spite of huge improvements and advances in NLP in the last 20 years [5], [6], and great progress in NLP4RE research in the last 10 years, the uptake of NLP technologies in RE, and their industrial penetration, is still limited and fragmented [7], Thus large gaps remain between NLP4RE research and its practical application [8].

A recent survey [8] cites insufficient industrial evaluation of NLP4RE research, the lack of shared RE-specific language resources, and the lack of technical know-how in NLP among the reasons for these gaps. As a first step to closing these gaps, this paper aims to classify the NLP techniques commonly used in RE so that they are easy to understand. We believe that a better understanding of the NLP techniques in RE is not only crucial to the development of better NLP tools for RE, but also to their industrial adoption. In particular, the paper lays foundations for establishing a common terminology and vocabulary of the NLP techniques through the following contributions:

We extract and synthesize 57 commonly used NLP techniques in RE for NLP4RE research and practice.

We systematically classify these techniques in two ways: by their tasks typically performed in NLP pipelines and then by their linguistic analysis capability.

The paper is organized as follows. Sect. II provides a brief history of NLP for RE, as the background and motivation for this paper. Sect. III describes how we extract and synthesize the common set of NLP techniques for RE. Sect. IV and Sect. V presents our classification of these techniques. Sect. VI concludes the paper.

II. BACKGROUND: 40 YEARS OF NLP4RE

As a background to this paper, we provide a brief history of NLP4RE. We first point to some notable contributions in RE that use traditional NLP techniques, and then outline the recent application of machine learning (ML) and deep learning (DL) in RE. However, the focus of this paper is on NLP techniques, not ML and DL techniques.

A. Traditional NLP for RE

The relationship between NLP and RE is well-established and widely discussed, with supporters and detractors [1], [7], [9], [10]. Pioneering researchers in the field are Chen [11] and Abbott [12], who, in the early 1980s, proposed using syntactic features of English sentences for database modeling and program design. Abbott's approach was subsequently adapted to a program design tool by Berry et al. [13]. These works were mostly based on extracting relevant entities from the requirements text through simple syntactic rules, assuming that NL requirements were expressed in some constrained, predictable format, which, however, is rarely the case in practice [14].

After these pioneering works, the beginning of 1990s saw some serious attempts to develop NLP4RE tools, introducing techniques to account for the complexity and variety of NL. Two well-known NLP tools, find phrases by Aguilera and Berry [15] and OICSI by Rolland and Prix [16], were the results of these efforts. Both tools were still oriented to the extraction task [8], also referred to as abstraction [10], and used lexical affinity and semantic cases, respectively, two techniques that are far more sophisticated than those previously used.

For the remaining 1990s right up to the beginning of 2000s, a succession of NL tools had been proposed, among which were AbstFinder by Goldin and Berry [17], NL-OOPS by Mich [18], Circe by Ambriola and Gervasi [19], CM-Builder by Harmain and Gaizauskas [20]. These works normally use traditional rule-based NLP techniques, and are oriented to term extraction and model generation. Other tools, such as QuARS by Fabbrini et al. [21], and ARM by Wilson et al. [22], focus on defect detection and mostly use dictionary-based techniques.

The early 2000s appeared to be a period of experimentation of new NLP techniques and new ideas addressing other tasks and phases of the RE process. Information retrieval (IR) techniques were used to improve requirements tracing [23], statistical NLP techniques were applied to identify "shallow knowledge" from requirements text [2], and to trace relationships between requirements [24].

Since the late 2000s, NLP4RE has become a full-fledged research area, attracting researchers from the wider RE community. A large number of tools have since been developed, among which are SREE (Tjong and Berry [25]) for ambiguity detection and a toucan (Yue et al. [26]) for model generation. Further developments include tools detection of defects [27], smells [28] and equivalent requirements [29].

Given the increasing need to make software systems trustworthy, accountable, and legally compliant, as well as security-and privacy-aware, NLP has been largely applied also to legal documents [30] and privacy policies [31], in the field of RE and Law. Finally, to support agile software development, requirements expressed in the form of user stories have been identified as an interesting area of application for NLP [32].

B. Machine Learning and Deep Learning for RE

Following the development of successful statistical NLP methods based on ML in the 1990s [5], [33], ML techniques have become increasingly important to NLP. The advantages of the ML-based approaches over the traditional, rule-based NLP approaches are effectiveness, considerable savings in terms of expert manpower, and straightforward portability to different domains [34].

In RE, the earliest adoption of ML to NLP can be traced to a study by Cleland-Huangetal. [35], published in 2007, in which the authors presented an approach for automatically detecting and classifying non-functional requirements (NFRs) from requirements documents. The approach uses a set of weighted indicator terms to classify requirements; a probability value of each indicator term is computed by a probability function similar to Na¨ıve Bayes [36], to estimate the likelihood of an input requirement being classified into a certain NFR category.

The development of this approach thus marked the beginning of the work on ML-based approaches for RE and, as a seminal work in this area, this approach has been frequently used as the baseline to assess the performance of new techniques [7], [37].

With the recent widespread availability of NL content relevant to RE, such as feedback from users in app stores and social media, and developers' comments in discussion forums and bug-tracking systems, we have observed a rising interest in using ML techniques to support data-driven RE [38] and crowd-based RE [39].

These areas aim to leverage the information available from stakeholders' implicit and explicit feedback, including diverse sources such as app reviews [40], issue tracking systems [41], Twitter [42], or user fora [43], to improve RE activities such as requirements elicitation and prioritization. Most of the works use ML techniques, as these can be effectively exploited when the task can be reduced to a classification problem, and a large amount of data is available. The analysis of different forms of feedback can be regarded as the main trend of the last years in NLP4RE research [8].

However, several other RE tasks have profited from ML and even DL techniques, for example : glossary extraction, with the usage of unsupervised learning [44] and convolutional neural networks [45]; requirements classification with the early works from Casamayor et al. [46] and developments from Kurtanovic and Maalej [37]; requirements tracing [47], [48], which can be regarded as the field where ML/DL have been more widely experimented for traditional requirements, especially due to the inherent nature of the problem, which entails finding a relevant relationship (i.e., trace links) within a large amount of potential ones.

With the advent of DL and transfer learning, in particular, initial experiments have been carried out in RE with promising results. In particular, DL-based approaches have been proposed to classify software requirements into FR or NFR [49], to discover requirements from open-source issue reports [50] and to extract and classify requirements from software project contracts [51]. We predict that research in developing DL-based approaches for RE tasks will grow rapidly in the coming years, overtaking the work on ML-based approaches.

III. METHOD

The main source of the literature used for our data collection is the set of 404 NLP4RE studies identified in our systematic review [8][1], which covers the studies up to 2019. We then performed a complementary targeted review to identify recent publications, to find more recent techniques emerging in the last 2 years. This complimentary review focused on the major RE and software engineering conferences (i.e., RE, REFSQ and ICSE) and journals (i.e., REJ, JSS, ASE, DKE, IST, and TSE). Based on this updated literature, we extract NLP techniques.

To help us identify and extract NLP techniques from each paper, we followed this definition: ”An NLP technique is a practical method, approach, process, or procedure for performing a particular NLP task, such as POS tagging, parsing or tokenizing [8].” Our data extraction resulted in a large collection of diverse terms and phrases. To

synthesize different terms and phrases into a coherent set of standard terms, we consulted many books written by NLP experts (e.g., [52]–[54]). This process gave rise to a total of 57 different NLP techniques.

IV. CLASSIFYING NLP TECHNIQUES BY TASKS

We first classify the NLP techniques based on their text-processing tasks. Figure 1 depicts the relationship between NLP techniques, NLP tasks, NLP resources, and tools. We define an NLP task as a piece of text processing work that can be done by means of one or more NLP techniques, supported by some NLP tools and resources. A list of frequently performed NLP tasks in RE is described below:

Part-of-Speech (POS) Tagging: To associate words with part-of-speech (POS) tags to distinguish between nouns, verbs, adjectives, adverbs, etc. The input unit is a sentence, as context words (i.e., neighboring ones) are normally used to infer the POS of a word.

Semantic Tagging: To extract useful bits of information (words, terms, relations, etc.) from the text.

Syntactic Analysis: To analyze the syntactic structure of a sentence to represent the relationship between its components. Different representation structures can be used, such as the parse tree, or the dependency parsing graph.

Semantic Analysis: To identify and label semantically relevant components and relations in the text. This entails identifying the meaning of a certain word or phrase in a context and the relationship between words or terms.

Frequency Analysis: To analyze the frequencies of words or terms in a certain context and to produce probabilistic data.

Similarity Analysis: To calculate the numerical estimates of similarity between text elements, for example, to identify semantic relatedness, synonyms, or to support topic

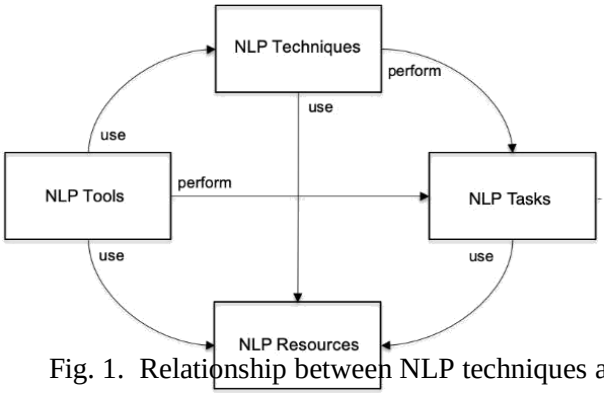

Fig. 1. Relationship between NLP techniques and NLP Tasks.

Modeling:

Rule-Based Analysis: To use grammar rules, semantic rules, or patterns to analyze the syntax of a text.

Text Normalization: To convert the words into their original form and remove unnecessary words or characters from the text.

Text Segmentation: To break down a text into a sequence of individual sentences or words.

Text Normalization: To reduce the words to a standardised format, with the removal of stop words, and reduction of typographical forms (e.g., upper case, camel case) to a unique form.

The classification results of the NLP techniques for RE are based on these tasks.

V. CLASSIFYING NLP TECHNIQUES BY LINGUISTIC ANALYSIS LEVELS

Here, we classify the NLP techniques by levels of linguistic analysis. According to Liddy [55], linguistic analysis can be performed at the following seven levels:

Phonology. This level deals with the interpretation of speech sounds within and across words.

Morphology. This is the lowest level of text analysis. At this level, an NLP technique analyzes the smallest parts of words that carry meaning, which is composed of morphemes, including prefixes, roots and suffixes of words.

Lexical. An NLP technique at this level can interpret the meaning of individual words to gain word-level understanding

[55]. Lexical analysis may require a lexicon or dictionary, which may be quite simple, with only the words and their POS tags, or may be increasingly complex and contain information on the semantic class of the word, its arguments, etc. [55].

Syntactic. An NLP technique at this level focuses on analyzing the words in a sentence through the grammatical structure of the sentence. This requires both a grammar and a parser [55]. There are two general types of parsers: dependency and constituency [52]. The dependency parser produces a syntactic representation of a sentence based on the dependencies between the words in the sentence, whereas the constituency parser represents a sentence as a parse tree of related constituents (i.e., sub-phrases). These representations (i.e., syntax) carry meaning in most languages, because the arrangement of words or sub-phrases in a sentence contributes to meaning [55].

Semantic. An NLP technique at this level may focus on the meanings of individual words (e.g., dictionary definitions of words and word-sense disambiguation), or compositional semantics, which looks at the interactions among word-level meanings in sentences (e.g., semantic role labeling). The semantic analysis thus can be divided into word-level semantics and sentence-level semantics (groups of words or sentence-level). Semantic role labeling [56] and Case Grammar [57], [58] are among the examples of semantic analysis techniques.

Discourse. An NLP technique at this level focuses on the properties of the text as a whole that convey meaning by making connections between component sentences. Several types of discourse processing can occur at this level, two of the most common being anaphora resolution and coreference resolution [55].

Pragmatic. This is the highest level of NLP. To reach this level, NLP techniques need to be able to achieve human-like language understanding, the ultimate goal of natural language understanding (NLU). This entails inferring extra meaning from texts that are not actually encoded in them [55] and understanding narratives according to different contexts and with respect to different actors and their intentions [33]. This requires NLP tools to have world knowledge and human intelligence, and the ability to project semantics and sentics dynamically [33]. The pragmatic analysis appears to be the most challenging NLP curve to jump [33].

It is assumed that humans normally produce or comprehend language by utilizing all of these levels [59]. These levels thus represent the competence of an NLP tool: The more levels of analysis the tool supports, the stronger or more capable the tool; the more higher levels of analysis the tool supports, the more advance the tool.

Classifies the NLP techniques based on these levels. As the table shows, we have not found any techniques for the phonetic level analysis, as NLP techniques have been largely using to deal with texts (including requirements documents [60], app reviews, tweets, social media posts and usage data [7], [9], [38], [61], [62], legal documents [30], and privacy policies [31]). In addition, we have not found any techniques for pragmatic analysis either. This is because NLP4RE research has so far focused on text processing of documents and has not reached the level of natural language understanding (NLU).

VI. CONCLUSION

This paper presents 57 commonly used NLP techniques in RE and organizes them in two different ways: by their NLP tasks and by their analysis levels. The organization provides a knowledge base for sharing these techniques. A user of this knowledge base can query each NLP technique progressively: Through Table I and Table II, the user can ask: What technique is it? Does it work at the word level or sentence level of text processing? Based on the user can ask: Which text processing task does this technique support? What are the alternative techniques for the same task? From Table IV, the user can ask: What level of language analysis does this technique provide? What are the techniques for performing other levels of analysis? The answers to these questions can help the user to decide if a specific technique is relevant to the task at hand.

Our future work will improve this knowledge base as follows:

To show the relationship between a given technique and other techniques. For example, for text normalization, what techniques can I use together? in what order? For text representation, which technique is better for my case?

To provide information on the available NLP tools that support each technique.

REFERENCES

[1] K. Ryan, "The role of natural language in requirements engineering," in [1993] Proceedings of the IEEE International Symposium on Requirements Engineering. IEEE, 1993, pp. 240–242.

[2] P. Sawyer, P. Rayson, and K. Cosh, "Shallow knowledge as an aid to deep understanding in early phase requirements engineering," IEEE Transactions on Software Engineering, vol. 31, no. 11, pp. 969–981, 2005.

[3] V. Berzins, C. Martell, P. Adams, et al., "Innovations in natural language document processing for requirements engineering," in Monterey Workshop. Springer, 2007, pp. 125–146.

[4] D. Berry, R. Garcia, P. Sawyer, and S. F. Tjong, "The case for dumb requirements engineering tools," in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2012, pp. 211–217.

[5] J. Hirschberg and C. D. Manning, “Advances in natural language processing,” Science, vol. 349, no. 6245, pp. 261–266, 2015.

[6] G. Goth, "Deep or shallow, NLP is breaking out," Communications of the ACM, vol. 59, no. 3, pp. 13–16, 2016.

[7] F. Dalpiaz, A. Ferrari, X. Franch, and C. Palomares, "Natural language processing for requirements engineering: The best is yet to come," IEEE Software, vol. 35, no. 5, pp. 115–119, 2018.

[8] L. Zhao, W. Alhoshan, A. Ferrari, K. J. Letsholo, M. A. Ajagbe, E.-V. Chioasca, and R. T. Batista-Navarro, "Natural language processing (NLP) for requirements engineering (re): A systematic mapping study," ACM Computing Surveys, in the press, 2021.

[9] A. Ferrari, F. Dell'Orletta, A. Esuli, V. Gervasi, and S. Gnesi, "Natural language requirements processing: a 4D vision," IEEE Software, vol. 34, no. 06, pp. 28–35, 2017.

[10] D. M. Berry, "Evaluation of tools for hairy requirements and software engineering tasks," in 2017 IEEE 25th International Requirements Engineering Conference Workshops (REW). IEEE, 2017, pp. 284– 291.

[11] P. P.-S. Chen, "English sentence structure and entity-relationship diagrams," Information Sciences, vol. 29, no. 2, pp. 127–149, 1983. [Online]. Available: https://www.sciencedirect.com/science/article/pii/0020025583900142

[12] R. J. Abbott, "Program design by informal English descriptions," Communications of the ACM, vol. 26, no. 11, pp. 882–894, 1983.

[13] D. M. Berry, N. Yavne, and M. Yavne, "Application of program design language tools to Abbott's method of program design by informal natural language descriptions," Journal of Systems and Software, vol. 7, no. 3, pp. 221–247, 1987. [Online].Available: https://www.sciencedirect.com/science/article/pii/0164121287900446

[14] G. Booch, R. A. Maksimchuk, M. W. Engle, B. J. Young, J. Connallen, and K. A. Houston, "Object-oriented analysis and design with applications," ACM SIGSOFT software engineering notes, vol. 33, no. 5, pp. 29–29, 2008.

[15] C. Aguilera and D. M. Berry, "The use of a repeated phrase finder in requirements extraction," Journal of Systems and Software, vol. 13, no. 3, pp. 209–230, 1990. [Online]. Available: https: //www.sciencedirect.com/science/article/pii/0164121290900976

[16] C. Rolland and C. Prix, "A natural language approach for requirements engineering," in International Conference on Advanced Information Systems Engineering. Springer, 1992, pp. 257–277.

[17] L. Goldin and D. M. Berry, "Abstfinder, a prototype natural language text abstraction finder for use in requirements elicitation," Automated Software Engineering, vol. 4, no. 4, pp. 375–412, 1997.

[18] L. Mich, "Nl-oops: from natural language to object oriented requirements using the natural language processing system lolita," Natural language engineering, vol. 2, no. 2, pp. 161–187, 1996.

[19] V. Ambriola and V. Gervasi, "On the systematic analysis of natural language requirements with Circe," Automated Software Engineering, vol. 13, no. 1, pp. 107–167, 2006.

[20] H. M. Harmain and R. Gaizauskas, "Cm-builder: an automated NL-based case tool," in Proceedings ASE 2000. Fifteenth IEEE International Conference on Automated Software Engineering. IEEE, 2000, pp. 45–53.

[21] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, “The linguistic approach to the natural language requirements quality: benefit of the use of an automatic tool,” in Proceedings 26th Annual NASA Goddard Software Engineering Workshop. IEEE, 2001, pp. 97–105.

[22] W. M. Wilson, L. H. Rosenberg, and L. E. Hyatt, “Automated analysis of requirement specifications,” in Proceedings of the 19th international conference on Software engineering, 1997, pp. 161–171.

[23] J. H. Hayes, A. Dekhtyar, and J. Osborne, "Improving requirements tracing via information retrieval," in Proceedings. 11th IEEE International Requirements Engineering Conference, 2003. IEEE, 2003, pp. 138–147.

[24] J. Cleland-Huang, B. Berenbach, S. Clark, R. Settimi, and E. Romanova, "Best practices for automated traceability," Computer, vol. 40, no. 6, pp. 27–35, 2007.

[25] S. F. Tjong and D. M. Berry, “The design of sree—a prototype potential ambiguity finder for requirements specifications and lessons learned,” in International Working Conference on Requirements Engineering: Foundation for Software Quality. Springer, 2013, pp. 80–95.

[26] T. Yue, L. C. Briand, and Y. Labiche, "a toucan: an automated framework to derive UML analysis models from use case models," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 24, no. 3, pp. 1–52, 2015.

[27] A. Ferrari, G. Gori, B. Rosadini, I. Trotta, S. Bacherini, A. Fantechi, and S. Gnesi, "Detecting requirements defects with NLP patterns: an industrial experience in the railway domain," Empirical Software Engineering, vol. 23, no. 6, pp. 3684–3733, 2018.

[28] H. Femmer, D. M. Fernandez,´ S. Wagner, and S. Eder, “Rapid quality assurance with requirements smells,” Journal of Systems and Software, vol. 123, pp. 190–213, 2017.

[29] D. Falessi, G. Cantone, and G. Canfora, "Empirical principles and an industrial case study in retrieving equivalent requirements via natural language processing techniques," IEEE Transactions on Software Engineering, vol. 39, no. 1, pp. 18–44, 2011.

[30] A. Salimi, N. Sannier, M. Sabetzadeh, L. Briand, and J. Dann, "Automated extraction of semantic legal metadata using natural language processing," in 2018 IEEE 26th International Requirements Engineering Conference (RE). IEEE, 2018, pp. 124–135.

[31] J. Bhatia, T. D. Breaux, and F. Schaub, "Mining privacy goals from privacy policies using hybridized task recomposition," ACM Transactions on Software Engineering and Methodology (TOSEM), vol. 25, no. 3, pp. 1–24, 2016.