

**AN EBD-ENABLED DESIGN KNOWLEDGE
ACQUISITION FRAMEWORK**

Cheligeer Cheligeer

A Thesis
in
The Concordia Institute
for
Information System Engineering

Presented in Partial Fulfillment of the Requirements
for the Degree of
Doctor of Philosophy (Information and Systems Engineering) at
Concordia University
Montréal, Québec, Canada

August 2022

© Cheligeer Cheligeer, 2022

CONCORDIA UNIVERSITY

School of Graduate Studies

This is to certify that the thesis prepared

By: Cheligeer Cheligeer

Entitled: AN EBD-ENABLED DESIGN KNOWLEDGE ACQUISITION FRAMEWORK

and submitted in partial fulfillment of the requirements for the degree of

Doctor of Philosophy (Information and Systems Engineering)

complies with the regulations of the University and meets the accepted standards with respect to originality and quality.

Signed by the Final Examining Committee:

_____	Chair
<i>Dr. Fuzhan Nasiri</i>	
_____	External Examiner
<i>Dr. FangXiang Wu</i>	
_____	External to Program
<i>Dr. Mazdak Nik-Bakht</i>	
_____	Examiner
<i>Dr. Jun Yan</i>	
_____	Examiner
<i>Dr. Chun Wang</i>	
_____	Thesis Co-supervisor
<i>Dr. Yuan Xu</i>	
_____	Thesis Co-supervisor
<i>Dr. Nadia Bhuiyan</i>	
_____	Thesis Co-supervisor
<i>Dr. Yong Zeng</i>	

Approved by

Dr. Zachary Patterson, Graduate Program Director

8/16/2022

Dr. Mourad Debbabi, Dean
Gina Cody School of Engineering and Computer Science

Abstract

An EBD-Enabled Design knowledge acquisition framework

Cheligeer Cheligeer, Ph.D.

Concordia University, 2022

Having enough knowledge and keeping it up to date enables designers to execute the design assignment effectively and gives them a competitive advantage in the design profession. Knowledge elicitation or acquisition is a crucial component of system design, particularly for tasks requiring transdisciplinary or multidisciplinary cooperation. In system design, extracting domain-specific information is exceedingly tricky for designers. This thesis presents three works that attempt to bridge the gap between designers and domain expertise. First, a systematic literature review on data-driven demand elicitation is given using the Environment-based Design (EBD) approach. This review address two research objectives: (i) to investigate the present state of computer-aided requirement knowledge elicitation in the domains of engineering; (ii) to integrate EBD methodology into the conventional literature review framework by providing a well-structured research question generation methodology. The second study describes a data-driven interview transcript analysis strategy that employs EBD environment analysis, unsupervised machine learning, and a range of natural language processing (NLP) approaches to assist designers and qualitative researchers in extracting needs when domain expertise is lacking. The second research proposes a transfer-learning method-based qualitative text analysis framework that aids researchers in extracting valuable knowledge from interview data for healthcare promotion decision-making. The third work is an EBD-enabled design lexical knowledge acquisition framework that automatically constructs a semantic network -- RomNet from an extensive

collection of abstracts from engineering publications. Applying RomNet can improve the design information retrieval quality and communication between each party involved in a design project.

To conclude, this thesis integrates artificial intelligence techniques, such as Natural Language Processing (NLP) methods, Machine Learning techniques, and rule-based systems to build a knowledge acquisition framework that supports manual, semi-automatic, and automatic extraction of design knowledge from different types of the textual data source.

Acknowledgments

First, my sincere gratitude goes to my thesis supervisors, Dr. Yuan Xu, Dr. Nadia Bhuiyan, and Dr. Yong Zeng. Your insightful feedback elevated my work to another level. You have played a significant role in helping me to reconceptualize myself and become who I am today. All of my academic achievements stem from my supervisors. No words express my gratitude for your help in completing this journey. A special thanks to my supervisor Prof. Yong Zeng. Thank you for being so patient in guiding me during my most ignorant and stubborn moments.

I am grateful to Dr. Chun Wang, Dr. Jun Yan, and Dr. Ali Akgunduz for dedicating their time and effort to attending my Comprehensive Exam, Doctoral Proposal, and Seminar. I appreciate your valuable feedback and suggestions. In addition, I would like to thank my external examiner, Dr. Fangxiang Wu from the University of Saskatchewan, and my internal-external examiner, Dr. Mazdak Nik-Bakht, from BCEE at Concordia University. It is a great honor for me to have you on my Examination Committee.

Studying at Design Lab will be an unforgettable experience for me. I am honored and lucky to be a part of the Design Lab. I gained much knowledge from the lab and received much assistance from my colleagues. I will always be grateful to the people who helped me, and my deepest gratitude goes to my colleagues. Special thanks go to Daocheng Yang, Mengting Zhao, Hongyi Cao, Wenjun Jia, and Jie Pan for helping me to prepare for my final oral defense.

Working with my colleagues from the Centre for Health Informatics, University of Calgary, has been an excellent experience for me during my Ph.D. study. Working in the industry, attending

knowledge-sharing events, receiving professional training, and attending meetings also helped me achieve my Ph.D.

I would also like to thank Concordia University staff, CIISE staff, Graduate Program Coordinator, and Concordia student service for handling my concerns and matters during my study.

I express my deepest thanks to my parents; my parents are mine forever home. Your unfailing care and concern for me from my faraway hometown mean so much. Let me conclude by expressing my appreciation to my wife, Nandi, who has been an integral part of my success, and I am very appreciative of all her support, patience, and love. Without you, I would never achieve such a goal.

Contribution of Authors

Chapter 4, Chapter 5, and Chapter 6 are three manuscripts that we have already submitted to Artificial Intelligence for Engineering Design, Analysis, and Manufacturing, the Journal of Integrated Design & Process Science, and the Journal of Mechanical Design. The author of this thesis was responsible for developing, testing, and applying the methods discussed in this thesis, along with preparing manuscripts submitted to peer-reviewed journals. Dr. Yuan Xu, Professor, Community Health Science and Surgery, University of Calgary, Dr. Nadia Bhuiyan, Professor, Mechanical, Industrial and Aerospace Engineering, Concordia University, and Dr. Yong Zeng, Professor, Concordia Institute for Information Systems Engineering, Concordia University, have supervised these manuscripts and provided valuable guidance and advice on various aspects of the research.

Individuals listed below provided advice and assistance in all aspects of this thesis, as well as reviewed and edited all journal manuscripts: Dr. Hude Quan, Professor, Centre for Health Informatics, University of Calgary; Dr. Jingwei Huang, Professor, Department of Engineering Management and Systems Engineering; Dr. Lin Yang, Professor, Department of Community Health Science, University of Calgary; Dr. Guosong Wu, Data Analyst, Alberta Health Services; Dr. Lan Lin, Professor, Building, Civil, and Environmental Engineering, Concordia University; Dr. Tanistha Nandi, Data Scientist, University of Calgary; Dr. Shaminder Singh, Professor, Faculty of Health, Mount Royal University; Dr. Stéphane Dufresne, Advanced Design, Bombardier Inc; MEng. Alexandra Miklin, Concordia Institute for Information Systems

Engineering, Concordia University; MSc. Chelsea Doktorchik, Epidemiologist, Centre for Health Informatics, University of Calgary.

List of publications related to the thesis:

Cheligeer, C., Huang, J., Wu, G., Bhuiyan, N., Xu, Y., Zeng, Y. (2022). Machine Learning in Requirements Elicitation: A Literature Review. *Artificial Intelligence for Engineering Design, Analysis, and Manufacturing*, 1–34. (Under review)

Cheligeer, C., Yang, L., Nandi, T., Doktorchik, Chelsea., Quan, H., Zeng, Y., Singh, S. (2022). Natural Language Processing (NLP) Aided Qualitative Method in Healthcare Research. *Journal of Integrated Design & Process Science*. (Under review).

Cheligeer, C., Mikilin, A., Dufresne., S., Lin, L., Bhuiyan, N., Zeng, Y. (2022). ROM-based Semantic Networks Construction Framework: A Case Study of the Early Stages of an Aircraft Braking System Design Project. *Journal of Mechanical Design*. (Under review).

Contents

List of Figures	xii
List of Tables	xv
List of Abbreviations	xvii
Chapter 1. Introduction	1
1.1. Motivation	4
1.2. Objective.....	5
1.3. Outline	6
Chapter 2. Theoretical Framework	7
2.1. Environment-based design methodology and Recursive object modeling.....	7
2.2. Atomic design.....	10
2.3. User requirements.....	11
2.4. Natural language processing.....	12
2.5. Machine learning	15
Chapter 3. Literature Review	18
3.1. Systematic literature review	18
3.2. Automated requirement elicitation	24
3.3. Computer-aided qualitative data analysis.....	26
3.4. Semantic similarity algorithms.....	26

3.5.	Text vectorization and contextualized embeddings.....	29
3.6.	Transfer learning.....	30
3.7.	Design knowledge	31
3.8.	Knowledge acquisition	32
3.9.	Engineering semantic networks.....	34
Chapter 4. An EBD-based Systematic Literature Review: Machine Learning in Requirement Elicitation 39		
4.1.	Introduction	39
4.2.	Research methodology	41
4.3.	Results	57
4.4.	Research findings and discussion	73
4.5.	The limitations, open issues, and future works	87
4.6.	Limitations of the literature review	92
4.7.	Related works	94
4.8.	Conclusions	96
Chapter 5. A data-driven qualitative data analysis framework..... 103		
5.1.	Introduction	103
5.2.	Study design	104
5.3.	Experiment result.....	109
5.4.	Discussion.....	115

5.5. Limitations and future works.....	116
5.6. Conclusion	118
Chapter 6. A Design Science Enabled Project-Specific Semantic Networks Construction Framework	120
6.1. Introduction	120
6.2. Related works	122
6.3. Method.....	126
6.4. Discussion, limitations, and future works	140
6.5. Conclusion.....	143
Chapter 7. Conclusion.....	145
References.....	149

List of Figures

Figure 1. Design knowledge taxonomy in this thesis.	3
Figure 2. A conceptual framework for a general EBD-based design knowledge acquisition.	5
Figure 3. ROM basic components (Zeng, 2008, 2020).....	9
Figure 4. Question generation procedure (Wang & Zeng, 2009; Zeng, 2020).....	9
Figure 5. Question generation rules.	10
Figure 6. Design requirements, design knowledge, and environment (Zeng, 2020).	11
Figure 7. A machine learning taxonomy.....	17
Figure 8. The level of evidence in scientific research (Ahn & Kang, 2018).	19
Figure 9. Process of SLR by (Xiao & Watson, 2019).	20
Figure 10. The demonstration of Wohlin's snowballing SLR	21
Figure 11. SLR guideline by Brereton et al. (2007).	22
Figure 12. SLR method by (Torres-Carrión et al., 2018).	22
Figure 13. The demonstration of Mentefacto Conceptual (Torres-Carrión et al., 2018).	23
Figure 14. SLR methodology by (Okoli, 2015).	23
Figure 15. SLR process by (Ridley, 2012)	23
Figure 16. Semantic relationships, Storey. V (1993).	27
Figure 17. The illustration of the common super concept.	28

Figure 18. General knowledge engineering process.	34
Figure 19. Semantic network without taxonomy	35
Figure 20. An illustration for WordNet (L. Meng et al., 2013).	36
Figure 21. Illustration of the eRQ process for generating research questions.	44
Figure 22. The ROM diagram for the initial statement	44
Figure 23. ROM diagram from first-round question answering	52
Figure 24. PRISMA flow chart.	57
Figure 25. The number of included papers by year.	58
Figure 26. An illustration of the categorization schema of the collected studies	74
Figure 27. ML-based requirement elicitation tasks.	76
Figure 28. The data source for building ML-based requirement elicitation solutions.	79
Figure 29. Technologies and algorithms.	83
Figure 30. The proposed framework for qualitative coding.	106
Figure 31. Sample of the interview transcript.	107
Figure 32. The frequencies of words count.	111
Figure 33. The inputs and outputs of BERT architecture.	112
Figure 34. Relation between design, design requirement, design knowledge, and product.	125
Figure 35. The RomNet framework.	127
Figure 36. The ROM diagram for seed statement, generated with Graphviz.	129

Figure 37. The distribution of top-10 filed of studies.....	130
Figure 38. An example to illustrate styles in scientific paper abstract	131
Figure 39. Example of hyphen in the abstract.	132
Figure 40. The word count distribution of sentence dataset.	133
Figure 41. An example of constraint relations in ROM.....	134
Figure 42. A simple Skip-gram algorithm illustration.....	136
Figure 43. Vector visualization for words and phrases related to 'helicopter'.	138
Figure 44. New generated ROM diagram for seed statement.....	138
Figure 45. Semantic network for seed statement.....	139
Figure 46. A zoomed part from the semantic network.	139
Figure 47. The related terms to the given analogy question.	142

List of Tables

Table 1. NLP techniques – descriptions	14
Table 2. SLR methods and guidelines.	24
Table 3. Design knowledge taxonomies from different works.....	32
Table 4. Knowledge graphs and semantic networks are introduced in engineering design.	38
Table 5. Initial questions (IQs).	45
Table 6. The description of each cluster and its main entities.....	51
Table 7. The research scope under Cooper's literature review taxonomy.	53
Table 8. Table of search terms and operators (op).....	54
Table 9. Search engines and queries.	55
Table 10. Search engines and queries.	56
Table 11. Elements of data extraction.....	56
Table 12. The data source categorization.	60
Table 13. Preprocessing techniques.....	62
Table 14. classification of textual features in the selected literature.	66
Table 15. Learning algorithms from reviewed studies.	69
Table 16. Tools mentioned by included works.....	85
Table 17. Related works.	95

Table 18. The experiment's results.	114
Table 19. The experiment's results (Cont).	114
Table 20. The experiment's results (Cont).	114
Table 21. The comparison between research interests and extracted themes.	115
Table 22. Pseudocode Algorithm for extracting keywords and key phrases	128
Table 23. Hyperparameters for the skip-gram model selection.	137
Table 24. The extracted concepts are related to the project statement.	139

List of Abbreviations

Abbreviation	Definition
AI	Artificial Intelligence
ASUM	Aspect and Sentiment Unification Model
ATDM	Axiomatic Design modeling
AUC	Area Under Curve
BERT	Bidirectional Encoder Representations from Transformers
BFS	Breadth-first Search
BOW	Bag of Word
BTM	Bi-term Topic Model
CHREB	Conjoint Health Research Ethics Board
CNN	Convolutional Neural Networks
DbA	Design-by-Analogy
DL	Deep Learning
EA	Environment Analysis
EBD	Environment-Based Design
EM	Expectation-maximization
eRQ	Environment-based Research Question
FNN	Feedforward Neural Networks
FR	Functional Requirements
GRU	Gated Recurrent Unit
GT	Grounded Theory
IC	Information Content
ICT	Information and Communications Technology
IR	Information Retrieval
KBS	Knowledge-based System
KNN	K-Nearest Neighbors
KR	Knowledge Representation
LDA	Latent Dirichlet Allocation
LM	Language Model
LR	Literature Review
LSTM	Long Short-term Memory
MAE	Mean Absolute Error
ML	Machine Learning
MT	Machine Translating

NB	Naïve Bayes
NER	Named-entity Recognition
NFR	Non-functional Requirement
NLP	Natural Language Processing
NNLM	Neural Network Language Model
NPLM	Neural Probabilistic Language Model
NTRS	NASA Technical Reports Server
PCA	Principal Component Analysis
POS	Part-of-Speech
QA	Question Answering
RE	Regular Expression
RMSE	Root Mean Square Error
RNN	Recurrent Neural Network
ROM	Recursive Object Modeling
RS	Requirements Specification
SEO	Search Engine Optimization
SG	Skip Gram Language Model
SLR	Systematic Literature Review
SRL	Semantic Role Labeling
SRS	Software Requirements Specification
SVM	Support Vector Machine
TF-IDF	Term Frequency-Inverse Document Frequency
TL	Transfer Learning
TM	Text Mining
TSA	Tag Sentiment Aspect
UGC	User-generated Content
UPM	Unsmoothed Probability Measure
WSD	Word Sense Disambiguation
XAI	Explainable AI

Chapter 1. Introduction

Knowledge is an essential component of design activity. For a designer, team, or organization, knowledge represents the design capability of an individual or a group (Nguyen and Zeng 2017). Knowledge provides the design team with critical competitive advantages over its competitors and influences the team's ability to explore opportunities in new industries. As a consequence, if a design team loses its knowledge, it loses its competitiveness; however, if a team can continually acquire and retain knowledge, the team has a limitless amount of potential for developing a variety of goods. The field of knowledge management focuses on activities that contribute to organizational knowledge management, such as knowledge acquiring, modeling, sharing, using, and updating knowledge. To address knowledge management-related difficulties, the disciplines of knowledge management and design science have fused to develop a new interdisciplinary area called design knowledge management (Fu et al., 2006; Zhang et al., 2012). Researchers have proposed several frameworks to manage design knowledge; however, effectively acquiring sufficient knowledge, modeling design knowledge, and design information retrieval are still an open challenge.

The component of design knowledge is complex; unlike monodisciplinary tasks, design knowledge is diverse and heterogeneous. Due to its complexity, different researchers hold different explanations and classifications of design knowledge. The design knowledge can be classified as

tacit design knowledge and explicit design knowledge (Wong & Radcliffe, 2000); learning skills, social skills, product knowledge, and environment knowledge (Friedman 2000); general knowledge, domain-specific knowledge, procedural knowledge, and process knowledge (Tiwana and Ramesh 2001); natural environment, built environment and human environment knowledge (Zeng 2004); generic knowledge and domain-specific knowledge (Zeng 2008); process knowledge, product knowledge, function knowledge and issues (Ahmed 2005); process knowledge, product knowledge and task knowledge (Baxter et al. 2007). In this study, we integrated the aforementioned design knowledge classification into a three-dimensional cubic taxonomy shown in Figure 1, which includes (1) generic and domain-specific knowledge, (2) product and procedural design knowledge, and (3) natural, built, and human-related knowledge. The thesis focuses on product knowledge, represented by the lighter block (left part of the cubic) in Figure 1. Procedural design knowledge will not be involved since procedural design knowledge is usually taught by design schools and textbooks, which is the designer's domain knowledge and is usually readily accessed by the designer. This study aims to design and propose methodologies to support and guide designers in acquiring knowledge from a target domain or to minimize designers' reliance on target domain experts in design projects.

The management of knowledge and its modeling are popular research topics in many fields, such as the healthcare domain (Guptill 2005), engineering design (McMahon, Lowe, and Culley 2004), business (Gao, Meng, and Clarke 2008), and education (García et al. 2020). Numerous frameworks and methodologies aim to deliver an effective and efficient mechanism for general or specific knowledge management; however, only a few succeed in practice due to their complexity and immature information technologies.

As computing power and Internet technology have developed, data science has become an increasingly popular science that can solve various problems in various fields (Provost and Fawcett 2013). Data-driven methodologies have gradually become one of the effective solution patterns to complex issues.

With the rapid development of information technology, such as Natural Language Processing (NLP), Text Mining (TM), Information Retrieval (IR), Machine Learning (ML), and the recent rise of the Deep Learning (DL) frameworks started to provide feasible solutions to technical problems that troubled knowledge acquisition and modeling research before. Techniques such as automated ontology construction from the raw text (Wong et al., 2012) and automated knowledge graph construction (Rotmensch et al. 2017) are examples of modeling knowledge from available textual resources.

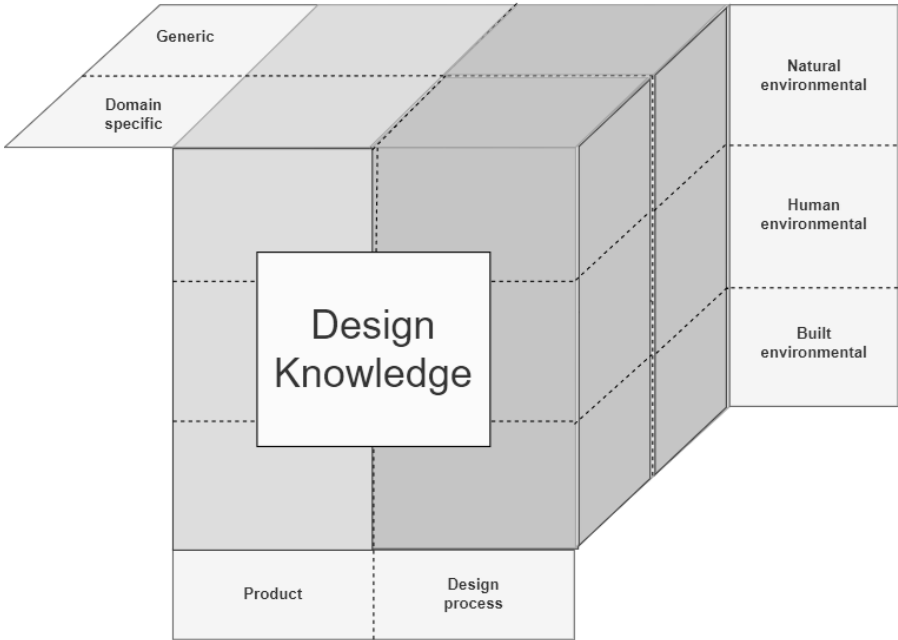


Figure 1. Design knowledge taxonomy in this thesis.

1.1. Motivation

The designer's role is to collect sufficient and necessary fragmental knowledge from different domains to solve an open-ended design problem (Yoshioka, Sekiya, and Tomiyama 1998). However, it is unrealistic for designers to be all-rounders of all disciplines. Because of this, designers need to learn by doing to solve many design challenges. As an essential skill, knowledge acquisition is the process of extracting, collecting, and eliciting knowledge from one or more sources. However, there are uncertainties regarding nearly all parts of design knowledge acquisition. The uncertain nature of the design makes knowledge acquisition more challenging. In addition, design knowledge is a compound knowledge closely connected with engineering, business, aesthetics, healthcare, and many other domains (Tatlisu and Kaya 2017). The compound characteristic of the knowledge causes various terms and jargon from different fields, which challenges communication, collaboration, and knowledge sharing among the design team. In addition, most design knowledge comes from practices and experience, which results in tacit design knowledge (Wong & Radcliffe, 2000). Research shows that 20% of a designer's time is spent searching and learning information (Baxter et al. 2007). As a result, gaining knowledge is one of the critical tasks for any design activity.

Information can be accessed in various media, such as audio, video, images, textbooks, blog posts, and research publications. According to Rahman & Harding (2012), 80% of industrial or business knowledge is stored in textual format. However, text-related activities such as collecting, filtering, querying, matching, interpreting, and analyzing are time-consuming and error-prone. Especially when working on multidisciplinary projects, domain-specific terms and phrases often have different meanings than those commonly used in their daily lives. The definition and explanation of these terms become one of the barriers to engineers while dealing with textual data.

1.2. Objective

Hence, motivated by the complexity of the unstructured nature of the textual document, the complex nature of design knowledge, and the ambiguous nature of domain-specific terms, this study focuses on design knowledge acquisition, synthesis, and modeling from available textual resources. Hence, the main objective of this thesis can be represented by a single statement:

A data-driven methodology for design knowledge acquisition from textual data.

By conducting an environment analysis with Environment-Based Design (EBD) and Recursive Object Modeling (ROM) on this statement, we can develop a three-layered framework for knowledge acquisition in the design shown in Figure 2.

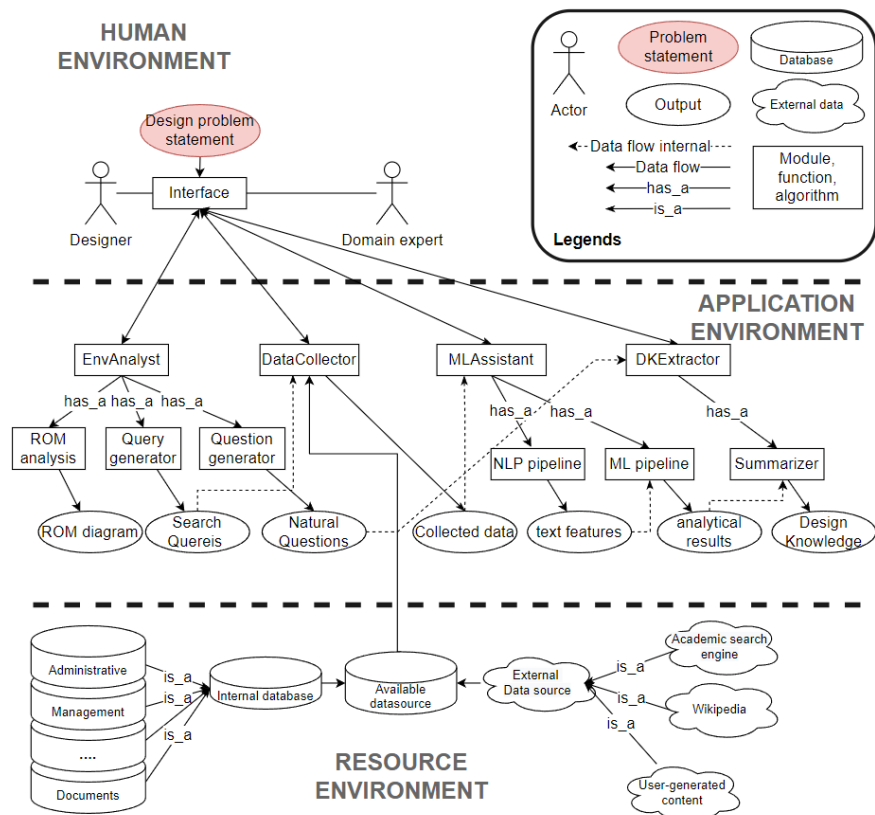


Figure 2. A conceptual framework for a general EBD-based design knowledge acquisition.

Different design knowledge sources are identified through the environment analysis, including research publications and interview transcripts. To efficiently utilize existing textual data in design, this thesis will focus on three tasks that support designers in closing the gap caused by insufficient design knowledge.

The objective can be achieved by following three research works:

- 1) To design a model-based systematic literature review on engineering requirement acquisition and synthesis.
- 2) To design an AI-aided qualitative data analysis methodology to support knowledge acquisition in qualitative research.
- 3) To design an automated lexical design knowledge modeling framework for engineering design.

1.3. Outline

The remainder of the thesis is structured as follows: Chapter 2 introduces the related concepts, theories, and research works to this thesis. Chapter 3 reviews the state of the art of methods and techniques associated with three proposed works (1) systematic literature review methodology, (2) AI-aided qualitative coding supporting methodology, and (3) automated design semantic networks construction framework. Chapter 4, Chapter 5, and Chapter 6 are three proposed research works. Last, in Chapter 7, the discussion about potential future research direction and conclusion are presented.

Chapter 2. Theoretical Framework

In this chapter, we mainly introduce our theoretical basis Environment-Based Design (EBD) and the technological basis of Recursive Object Modeling (ROM), Natural Language Processing (NLP), and Machine learning (ML).

2.1. Environment-based design methodology and Recursive object modeling

The EBD method is a systematic design theory that considers different environmental factors of the target product and generates design solutions aimed at solving the problems caused by the production environment to maintain a balanced state of the environment and the development and achieve true design harmony (Zeng 2011). The EBD methodology is fully supported by recursive design logic that fits human problem-solving logic and the evolutionary pattern of all things (Zeng 2020). The theoretical backbone of the Recursive Logic of Design and the EBD design methodology is Axiomatic Theory of Design Modeling (ATDM), an early design modeling work in design science that studied the design activities and proposed five theorems derived from two groups of axioms (Zeng 2002).

EBD is a methodology that provides step-by-step instructions for guiding a designer toward designing a product with the minimum amount of knowledge. EBD method significantly lowers the barriers to system design, allowing novice designers to generate design solutions without

sufficient domain knowledge. As an approach to interdisciplinary technique, the EBD methodology has been applied successfully in many different fields, including aviation, geometry, education, neurology, medical, and traditional product design (Jia and Zeng 2021; Pan et al. 2021; Tan, Zeng, and Montazami 2011; Yang et al. 2021).

The EBD methodology aims to decompose problem statements into several basic units (primitive objects) and relations between them with ROM techniques and apply a recursive procedure to retrieve sufficient knowledge for the design activities. The ROM is a modeling language for engineering design, which is also based on ADTM, that decomposes a given text length into individual text and the relationships between each word (Zeng 2008). The ROM model has three types of relationships: predicate, constraint, and connection (as shown in Figure 4). These three types of relations reflect the semantic dependencies between each word and contain implicit information that is hard to be identified by the designers without decomposition.

For the early design phase, such as conceptual design, the EBD method can decompose complex design statements into primitive, simple design questions to help designers with knowledge retrieval directions. The questioning technique (Figure 4) based on the ROM diagram is designed to enumerate all uncertain primitive objects using predefined questioning templates to ensure that the produced questions cover the scope of the entire product development project (Wang and Zeng 2009).

The EBD method for conceptional design contains five major procedures: problem statement modeling with ROM, question object identification, and question generation (Zeng 2020). The ROM analysis takes the problem statement as input and generates a relational graph diagram representing each item and its relationships. Object and relations are two types of primitive

components of ROM. A ROM diagram can be generated for each design statement with the fundamental part and the construction rules. The next step of the EBD analysis is to decide on the questioning objects from the ROM diagram. Either observation or ROM matrix can aid the identification of the questioning objects. The rationale behind questioning object identification is to count the number of informative relationships it has.

Type		Graphic Representation	Definition			
Object	Primitive environments	E_i^a	Everything in the universe is an object.			
	Environment structure	$\oplus E$	It is an object that includes at least two other objects in it.			
Relations	Connection	$E_i \dashrightarrow I \dashrightarrow E_j$	It is to connect two objects that do not constrain each other.			
	Constraint	$A \bullet \rightarrow E_i$	It is a descriptive, limiting, or particularizing relation of one object to another.			
	Predicate	<table border="0"> <tr> <td>Subject-verb</td> <td>$S_i \blacksquare \rightarrow v$</td> <td rowspan="2">It describes an act of an object on another or that describes the states of an object.</td> </tr> <tr> <td>Verb/proposition-object</td> <td>$v/p \rightarrow O_j$</td> </tr> </table>	Subject-verb	$S_i \blacksquare \rightarrow v$	It describes an act of an object on another or that describes the states of an object.	Verb/proposition-object
Subject-verb	$S_i \blacksquare \rightarrow v$	It describes an act of an object on another or that describes the states of an object.				
Verb/proposition-object	$v/p \rightarrow O_j$					

Figure 3. ROM components (Zeng, 2008, 2020).

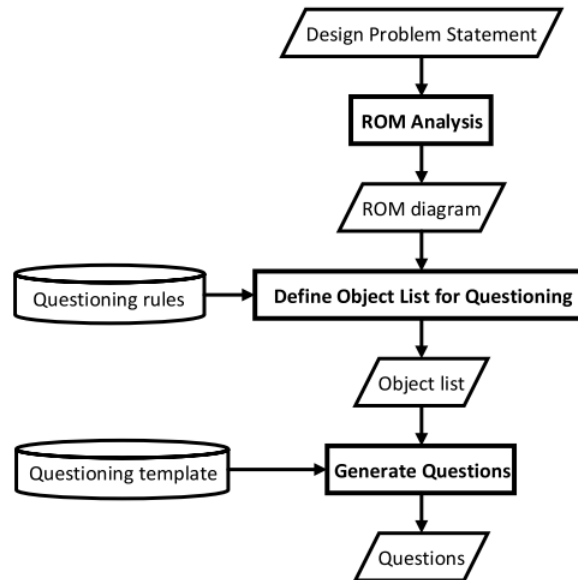


Figure 4. Question generation procedure (Wang & Zeng, 2009; Zeng, 2020)

The informative relation usually refers to predicate and constraints. After the questioning objects are identified, the next step is to generate simple questions according to the part-of-speech of each object. The detailed rules introduced by the original author are shown in Figure 5. After the questions are developed, designers can answer the questions according to the knowledge they possess and the experience they have. However, when the designers cannot answer the questions, designers may browse internal or external knowledge sources to retrieve relevant information to answer the questions.

#	Conditions	Question template
T1	For a concrete, proper, or abstract noun object N without any constraint	What/Who is N?
T2	For a concrete, proper, or abstract noun N with an adjective constraint A	What is A N?
T3	For a noun object A constraining an noun object N	What is A? What is/are A N?
T4	For a verb V with its subject N1 and object N2	What do you mean by V in the statement "N1 V N2"? How do/does N1 V N2? Why do/does N1 V N2? When do/does N1 V N2? Where do/does N1 V N2?
T5	For a verb object V constrained by an adverb A with its subject N1 and object N2	What do you mean by V A? Why do/does N1 V A N2? When do/does N1 V A N2? Where do/does N1 V A N2?
T6	For a verb V with an object N, but missing its subject	What/Who V N?

Figure 5. Question generation rules.

With these questions, designers can learn generic or domain-specific knowledge by retrieving relevant answers to each question. The procedure not only helps designers to acquire relevant design knowledge but also helps the designer to generate appropriate design solutions to uncertain problems.

2.2. Atomic design

The EBD methodology explained the nature of design and the relationships between design requirements, knowledge, and solution. The procedure is initiated with a valid design requirement composed of multiple atomic design requirements. The atomic design requirements represent the

user’s expected individual product facts, which include feature, function, and aspect. These product facts may or may not be the same as the final product because it only stands from the user’s point of view. On the other hand, the design requires designers to understand the design requirements and decompose the requirement into several valid design questions. The design team can decompose, distribute, track, and validate the design tasks with the design questions. The nature of design solution generation is to collect and integrate design knowledge to satisfy the design questions, and the nature of design knowledge is a combination of facts about the target product. Hence, the nature of design is to collect and synthesize atomic design knowledge to satisfy the design requirements (Figure 6).

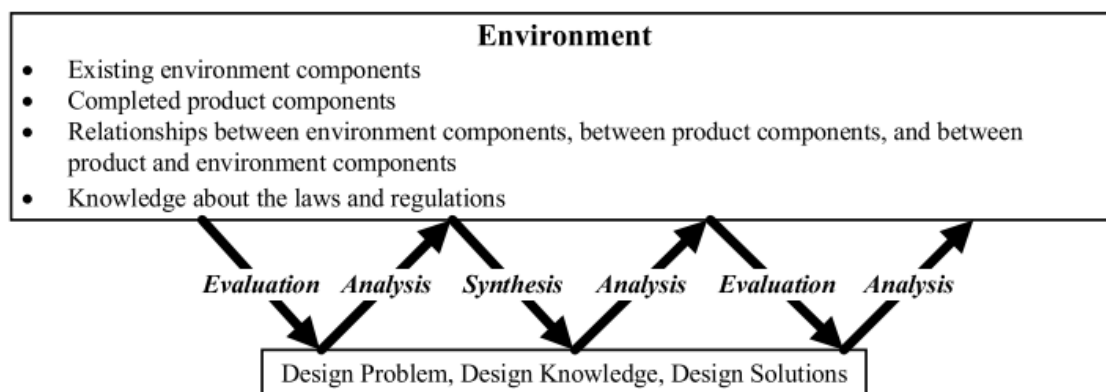


Figure 6. Design requirements, design knowledge, and environment (Zeng, 2020).

2.3. User requirements

According to the Cambridge dictionary, the requirement means “*something you must do or something you need.*” When a student meets the requirements of an academic degree, the student can graduate. Similarly, when a product meets customers' requirements, the product is successful.

The definition of requirements is slightly different in different domains. In the engineering design, a requirement is a physical product or system that “must be” created and delivered to satisfy the customers' needs (Brace and Cheutet 2012). In software engineering, requirements represent the

“must-have” functionalities and features of the software to solve problems in the real world (Bourque and Fairley 2014). For service design, the requirements are the service delivery and service development request from the customer (Patrício et al. 2011). The design requirements describe the target product's structure and performance (Cascini, Fantoni, and Montagna 2013; Zeng and Gu 1999). For product design, requirement means the written description of a product required by customers, which is usually in the form of documentation that lists “product description”, and “product performance”(Zeng and Gu 1999). In combination, a requirement means a written description of functions, attributes, or the quality of the target product or service that stakeholders expect.

2.4. Natural language processing

The design knowledge elicitation from a textual document is a challenging task for designers to complete. With the development of artificial intelligence and machine learning, natural language processing (NLP) techniques are becoming more and more mature enough to handle structured, monotonous, repetitive tasks related to the text. Designers can benefit significantly from using NLP techniques to ease knowledge elicitation tasks. Natural language processing is a broad topic of research and application that uses computer technology to understand and manipulate natural language to perform downstream tasks (Chowdhury 2003). Natural language processing has many different approaches, and it is a multi-disciplinary area that is a subfield of computer science, mathematics, linguistics, and artificial intelligence (Chowdhury 2003).

Lensu (2002) introduced the taxonomy of traditional NLP, where NLP is divided into three categories: parsing, semantic interpretation, and pragmatics. Under each category, there are many specific NLP tasks. A parsing task refers to a task that uses a parser to analyze input text according

to specific rules, such as formal grammar or dependency grammar, to generate a relational data structure that may contain grammatical or semantic information (Jurafsky and Martin 2018; Lensu 2002). The category semantic interpretation studies the meanings of words and language to analyze what reality each word (or text) stands for (Lensu 2002).

The tasks under semantic interpretation include semantic analysis, word-sense disambiguation (WSD), and lexical semantics. The NLP tasks related to context analysis are classified into the pragmatics category. Ganegedara (2018) put parsing, semantic, and pragmatics tasks under analysis tasks and extended the hierarchical taxonomy with the generation tasks category (Table 1). The generation tasks can produce new text based on the given inputs, and applications such as machine translation and question answering can be divided into this category. Based on the above explanation, the following conclude the popular tasks that NLP techniques could solve and the input/output of the specific traditional NLP task.

The statistical approach raised the revolution in the NLP field from the 1980s to the 1990s (Johnson 2009). The initial goal of the probabilistic model for natural language was to compute the probability of a sequence of words or to assign a class to the given sequence of the text. Unlike a human being, a machine cannot naturally read and understand a piece of natural language. To convert raw text into machine-readable content, there is a need to transform the representation of input text. The method for using data to represent input text is called text/word representation (Jurafsky and Martin 2018). There are many approaches for representing text, such as one-hot word encoding, co-occurrence vector (Lund and Burgess 1996), Glove (Pennington, Socher, and Manning 2014), and Word2Vec (Goldberg and Levy 2014). The model is called the language model that is learned from the data and produces predictions according to new data (Goldberg 2017).

Task name	Input	Output
<i>Regular expressions</i>	Matching pattern, text	Target text
<i>Text searching</i>	Keywords, phrase	Retrieved information
<i>Text normalization</i>	Word	Normalized text
<i>Part-of-Speech (POS) tagging</i>	Sentence	Text with POS labeled
<i>Semantic role labeling/ slot-filling (SRL)</i>	Sentence	Text with semantic role indicated
<i>Sentiment analysis</i>	Sentence	The emotional status of the text writer
<i>Word sense disambiguation (WSD)</i>	Word	The exact sense of input word indicating
<i>Named-entity recognition (NER)</i>	Text	Assign each word a pre-defined category
<i>Co-reference resolution</i>	Text	The relation for the same mention
<i>Similarity</i>	Two text documents	The similarity metric
<i>Topic modeling</i>	Text documents	Topics and topic words
<i>Question answering</i>	Question	Answer
<i>Machine translation</i>	Text in language A	Text in language B
<i>Text summarizing</i>	Text documents	Summarized document
<i>Word encoding</i>	Word	Numbers
<i>Feature extraction</i>	Text	Vector representation
<i>Word embedding</i>	Text	Dense word vectors
<i>Dependency parsing</i>	Sentence	Dependency relation of text
<i>Document clustering</i>	A set of text	Categorized groups
<i>Document classifying</i>	A set of text	Labeled categories
<i>Keyword extraction</i>	Text document	Keywords, key phrase

Table 1. NLP techniques – descriptions

Statistically, the language model assigns a probability distribution over given tokens and provides the probability of a given token following the distribution (Goldberg 2017; Manning and Hinrich Schütze 1999). The Neural Network Language Model (NNLM) is not new in NLP. In 2003, the first neural network language model was introduced by Bengio et al. (2001), named Neural Probabilistic Language Model (NPLM) is a simple feedforward neural network with three layers. Inspired by NPLM, there are different neural network-based language models introduced successively, such as the Recurrent Neural Network (RNN) based language modal (Mikolov et al. 2010) and transformer-based language model (Devlin et al. 2018).

NLP techniques are closely related to machine learning, and both methods have much in common. First, as introduced, most of the state-of-the-art NLP techniques are built with neural network architecture. Second, when doing text-based machine learning, some NLP techniques are used as the data preparation part of the machine learning pipeline, such as text data cleaning, preprocessing, and textual feature extraction.

2.5. Machine learning

Machine learning is a subcategory of artificial intelligence, and it refers to a set of automatic methods that take a large amount of information as input and generate patterns from it (Murphy 2012). Generally, there are three types of machine learning: supervised machine learning, unsupervised machine learning, and reinforcement learning (Figure 7). The primary objectives of various machine learning algorithms vary, resulting in these machine learning algorithms performing various jobs. Supervised machine learning uses labeled data to train the machine learning model to predict previously unknown data. (Murphy 2012). In supervised learning, the data and output are already known before training, and the supervised algorithms can evaluate iteratively on labeled data to discover possible data for any given input. In supervised machine

learning, two types of tasks can be performed: classification and regression. Classification challenges are designed to learn patterns that correspond to predetermined categories automatically. When there are just two classes, this is termed binary classification; when there are more than two classes, this is called multiclass classification (Murphy 2012). Another supervised machine learning task is regression. Unlike the classification task, the input/output continues (Murphy 2012). The real-world regression problems include but are not limited to the following tasks: stock market prediction, predicting the house price according to given statistics, and customer satisfaction prediction with the price and quality of a product.

Unsupervised machine learning does not have labeled data as input, and there is no prior knowledge about the relation between input data. Unsupervised learning algorithms can find helpful information about input data without knowledge. There are two main types of unsupervised learning algorithms one is called clustering algorithm, and the other one is dimensionality reduction (Murphy 2012). The clustering task can be viewed as “*knowledge discovery*” that takes unlabeled data as input and group them. Each group is called a cluster in clustering algorithms, and in (Raschka and Mirjalili 2019) introduced clustering as unsupervised classification. The real-world application of clustering includes cluster user groups, spam email filtering, and document clustering. The second necessary type of unsupervised machine learning is dimensionality reduction (Raschka and Mirjalili 2019). The primary task of the dimensional reduction algorithm is to keep essential data factors and omit irrelevant factors to project the multi-dimensional data into low-dimensional space. The most widely used algorithm in this family is Principal Component Analysis (PCA) which is described as the unsupervised version of linear regression (Murphy 2012). In practice, PCA or other dimensionality reduction algorithms are applied to reduce the dimensionality for data visualization, image compression, and computer vision.

Reinforcement learning uses an agent or a system that iteratively optimizes its performance of action to a specific task based on the interaction with the environment of the agent/system (Sutton and Barto 2018). Unlike supervised learning, there is no truth or false result from each iteration; instead, a measure of each action to indicate how well/bad each action performs. There are many real-world applications with reinforcement learning, such as robotics automatic control, game intelligence (Vinyals et al. 2019), recommender systems (Zheng et al. 2018), and Computer-aided automated design (Jie et al. 2021)

Machine learning can provide different types of aid according to the task type and the data. From the design knowledge elicitation perspective, several questions need to consider. How do we transfer design knowledge elicitation problems into machine learning problems? Which machine learning algorithm fits well with the target domain's design knowledge? How do we prepare the data to feed the machine learning algorithm?

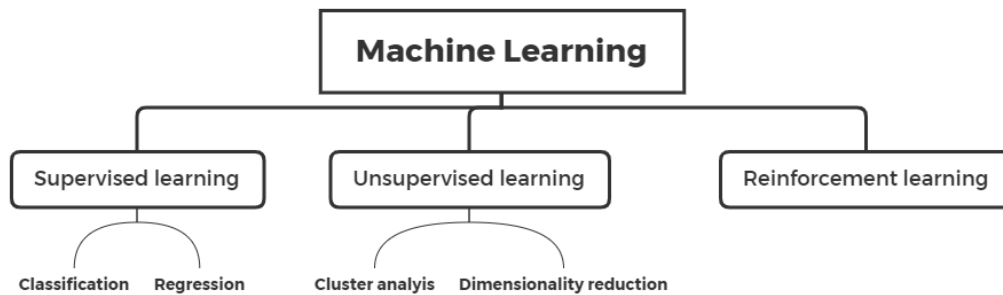


Figure 7. A machine learning taxonomy.

Thus, introducing machine learning into the problem of design knowledge extraction is not a plug-and-play technique; instead, it requires a carefully designed approach to ensure that each pipeline component works as envisioned.

Chapter 3. Literature Review

Our research includes three works to tackle design knowledge collection, synthesis, and modeling. In particular, as the first contribution, we proposed a method to support literature review in the engineering field. We conduct a systematic review of the literature on machine learning-based automatic requirement elicitation as part of a case study and thesis review. The literature review serves two significant roles in this thesis: 1) it is an application of the proposed literature review methodology; 2) it systematically reviews the current state of data-driven requirement knowledge extraction techniques and strategies to serve as a methodological review to our following two studies. Hence, the first studies involve the following three domains: a systematic literature review methodology (section 3.1), automated requirement elicitation (section 3.2), and design knowledge (section 3.4). The second work is proposed to support textual data analysis in qualitative research for gathering knowledge for stakeholder analysis, and the related research works are computer-aided qualitative data analysis (section 3.3) and semantic similarity (section 3.4). The last work is a ROM-based semantic network, which introduces a methodology that automatically constructs engineering lexical knowledge graphs from a given seed sentence by collecting and analyzing massive scientific research publication abstracts with NLP algorithms. The related research filed to this work is semantic similarity (section 3.4), engineering semantic network (section 3.9), design knowledge (section 3.4), and knowledge acquisition (section 3.8).

3.1. Systematic literature review

A systematic literature review (SLR) aims to identify, select, analyze, and criticize the included research for answering one or multiple specific research questions (Ahn and Kang 2018). There

are two ways to conduct research: design and develop theories from scratch or learn from the existing works (Gough, Oliver, and Thomas 2012). Reviewing existing works can help us answer questions such as the current state-of-the-art, the existing research limitations, and the potential future research directions (Xiao and Watson 2019). When conducting a literature review, two distinct objectives are pursued: one is to serve as background information for a practical study, and the other is to treat it as a standalone study (Xiao and Watson 2019).

In the term Systematic Literature Review (SLR), **systematic** (S) means the review is controlled by a fixed, well-defined, planned, and expected; **literature** (L) means books, papers, and writing publications on a particular subject; **review** (R) means to examine or assess literature critically. SLR is usually considered the highest level of evidence in scientific research (Ahn and Kang 2018). The level of evidence of scientific research is illustrated in Figure 8.

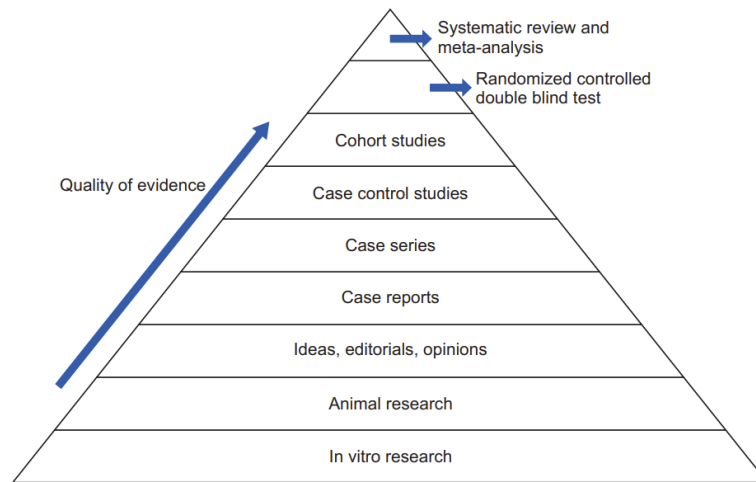


Figure 8. The level of evidence in scientific research (Ahn and Kang 2018).

Xiao & Watson (2019) summarized and categorized the existing literature review into four different categories by its objectives and research focus: the categories including *Describe*, *Test*, *Extend*, and *Critique*. By summarizing different types of literature review methods, they suggested an SLR method including three major phases: 1) planning, 2) conducting the review, and 3)

reporting the review (Figure 9). As a result of repeating steps two and one, the methodology will continue to harvest papers that meet the criteria.

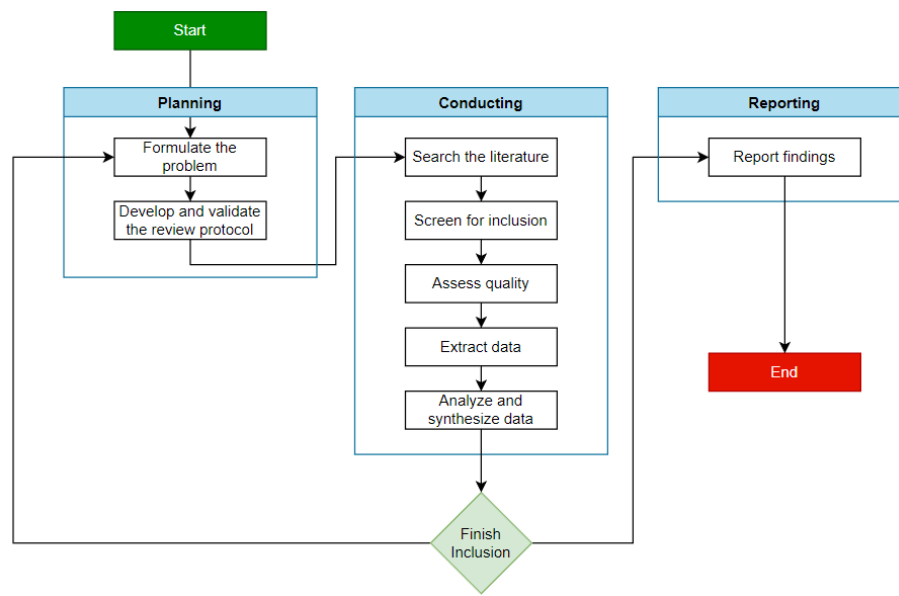


Figure 9. Process of SLR by (Xiao and Watson 2019).

Evidence-based Software Engineering (EBSE) is one of the early works advocating SLR in software engineering (Kitchenham et al., 2004). EBSE attempted to introduce a systematic software engineering methodology from evidence-based medical research using an “*analogy with medical practice*” (Kitchenham et al., 2004). At that time, the EBSE method was used only for software development and not for scientific research in the engineering field; however, in presenting the EBSE, there was much SLR published in the field of software engineering (Kitchenham et al., 2009). Similarly, in engineering design, researchers also advocated and discussed the rationale for adopting SLR in engineering and design studies (Lame 2019). The SLR has become a routinely used research methodology in the engineering field. There are several SLR methodologies and guidance proposed in the engineering domain. Among these methods, the snowballing SLR by Wohlin (2014) is one of the most widely applied SLR methods in the software engineering community (as shown in Figure 10). It is required that the user identify an initial set

of papers as a "tentative start set" and conduct a backward and forward snowballing process based on these papers. The backward method means looking at the selected papers' reference list to include more publications, and the forward method means looking at the papers that cite the included papers (Wohlin 2014).

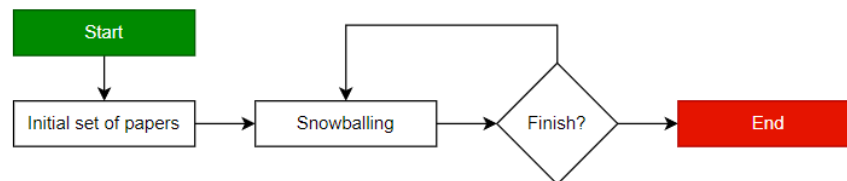


Figure 10. The demonstration of Wohlin's snowballing SLR

Brereton et al. (2007) categorized many subtasks in SLR into three main phases: planning, conducting the review, and reporting (Figure 11). The planning phase specifies research questions, LR protocol development, and validation. The conduct phase involves five activities: identifying relevant research, selecting primary studies, assessing study quality, extracting required data, and synthesizing data. The last phase is reporting review, which contains review report generation and LR reports validation.

SLR's increasing interest in engineering results in a specification of the proposed methods. Torres-Carrión et al. (2018) introduced a three-step literature review method, including planning, conducting, and reporting (as shown in Figure 12), which are similar to the phases described by Brereton et al. (2007) and Xiao & Watson (2019).

However, they focused more on the planning and conducting phase to specify the subtasks involved in these steps. They adopted a method called *Mentefactor Conceptual*, which uses conceptual ontology to support reading and learning (Torres-Carrión et al. 2018). The *Mentefactor Conceptual* is a simple ontology to represent a concept, that includes "Supraordination",

“Infraordiantion”, “Isoordiantion” and “Exclusion” of a given concept (Figure 13). The SLR reviewer can analyze the research question and generate search keywords through this framework.

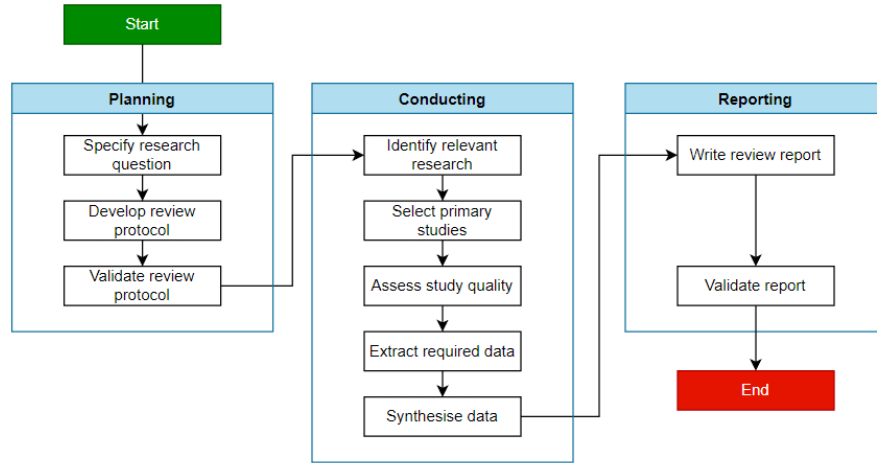


Figure 11. SLR guideline by Brereton et al. (2007).

Okoli (2015) introduced an eight-step guide to conducting SLR in information system research. Eight steps include identifying the research objective, training reviewers, practical screening, literature searching, data extraction, quality assessment, studies synthesis, and writing the report. According to the nature of the tasks in SLR, they also categorized these methods into four different phases (as shown in Figure 14): planning, selection, extraction, and execution (Okoli 2015).

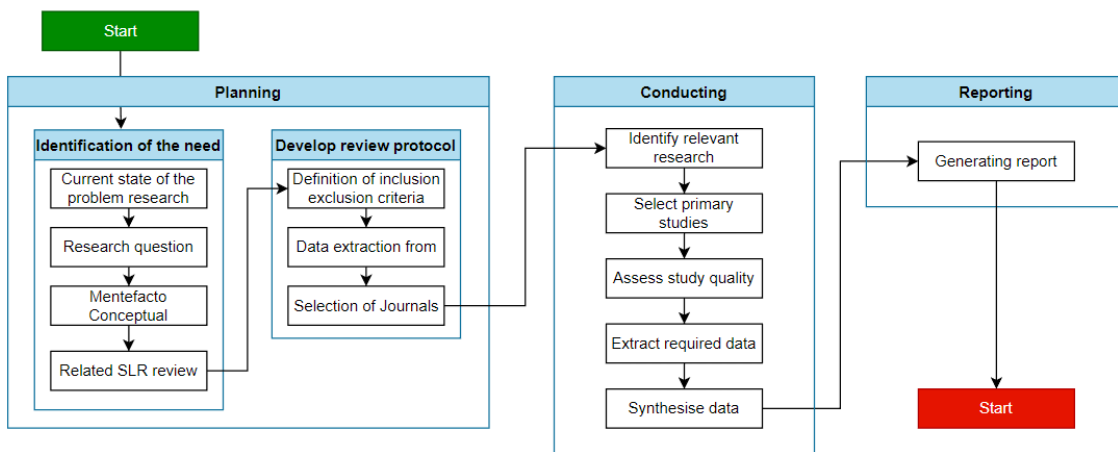


Figure 12. SLR method by (Torres-Carrion et al. 2018).

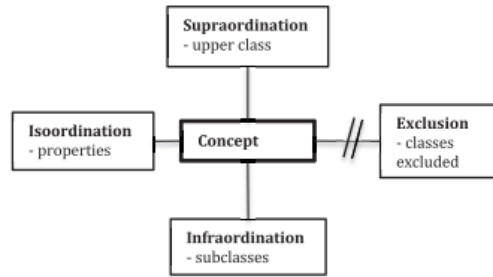


Figure 13. The demonstration of *Mentefacto Conceptual* (Torres-Carrión et al. 2018).

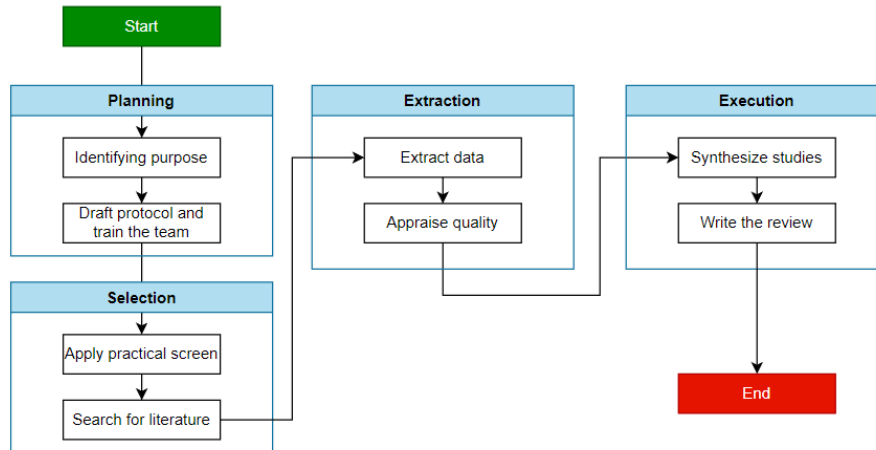


Figure 14. *SLR methodology* by (Okoli 2015).

The author of a book titled "The literature review: A step-by-step guide for students", Ridley (2012) gives a list of steps to be followed when conducting a literature review. The procedure contains 11 steps (Figure 15), and the author introduces each step generally in the book. While several other methods have primarily focused on planning and conducting phases, this book has also elaborated on the aspect of writing the report.

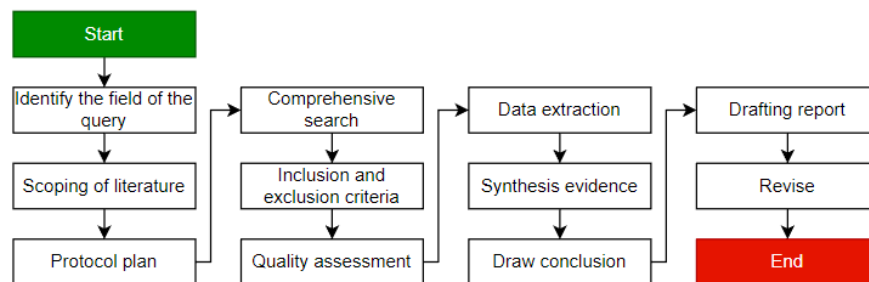


Figure 15. *SLR process* by (Ridley 2012)

In Table 2, we summarize the SLR methodologies that we have reviewed.

ID	Title	Author	Published year	Citation (until 2022-5-25)	Domain
SLR1	Procedures for Performing Systematic Reviews, Version 1.0	Barbara Kitchenham	2004	6667	Software engineering
SLR2	Guidelines for snowballing in systematic literature studies and a replication in software engineering	Claes Wohlin	2014	2603	Software engineering
SLR3	Lessons from applying the systematic literature review process within the software engineering domain	Brereton et al.	2007	2354	Software engineering
SLR4	An introduction to systematic reviews	Gough et al.	2012	2140	General
SLR5	The literature review: A step-by-step guide for students	Diana Ridley	2012	1280	General
SLR6	Guidance on Conducting a Systematic Literature Review	Xiao and Watson	2019	780	General
SLR7	A guide to conducting a standalone systematic literature review	Chitu Okoli	2015	609	Information system
SLR8	Methodology for Systematic Literature Review applied to Engineering and Education	Torres-Carrion et al.	2018	111	Engineering Education
SLR9	The systematic literature review as a research genre	Ramey and Rao	2011	33	General
SLR10	Systematic literature reviews: An introduction	Guillaume Lame	2019	31	Design science

Table 2. SLR methods and guidelines.

3.2. Automated requirement elicitation

During the past 15 to 20 years, the field of mechanical requirement engineering has been continuously explored and researched. Automated requirement engineering is an attractive topic, given the importance and challenges associated with tackling requirements within a project. According to research by (Verner, Sampson, and Cerpa 2008), over 70 failed software projects, of which 73% had unclear or inadequate requirements at the core of their failure. Most requirements-related project failures will be discovered in the latter stages of the project; however, requirements activities usually occur at the beginning of the project. The development team must restart the entire process if it fails to meet the requirements. Hence, understanding the stakeholder requirements is extremely important for the development team. Although flexible project

management frameworks such as Scrum and XP allow some degree of requirements changes and vague requirements, each development iteration still requires a particular well-defined requirement. As a result, considerable workforce, time, and budget were required in the early stage of the project. Due to such high demand, research on reducing labor and budget with computer systems or tools has become one of the most critical topics for academia and industry.

Automated requirement elicitation has grown in popularity as a study area during the last few years. As the name indicates, automated requirement elicitation is a broad term that refers to the process of extracting requirements or assisting in requirement acquisition operations using computer programs. Thus, to address this issue, two critical factors must be considered. To begin, defining the requirements for elicitation activities and subtasks is required. Secondly, it is necessary to determine how to implement an automated solution to assist the activities.

The requirement elicitation aims to determine stakeholders' needs for the target product. It takes intensive communication between various experts from different fields to fully understand stakeholders' requirements (Coughlan and Macredie 2002). The output from this process should be in a well-structured format that can be shared, systematically reviewed, evaluated, and approved by different parties. Typically, requirements are written in natural language to address this need. However, due to the nature of the requirement, it is difficult to restrict the input format. The needs of stakeholders can be both explicit and implicit. The purpose of requirement elicitation is to make as many implicit needs as possible and to document as many explicit needs as possible in formal written requirements.

3.3. Computer-aided qualitative data analysis

Qualitative research provides an in-depth description of the ideas and experiences of research subjects; however, conducting qualitative research is time-consuming, laborious, and expensive, especially in the medical and healthcare field (Leeson et al. 2019). The grounded theory approach is primarily used in qualitative research conducted by social scientists to develop an explanatory theory (Starks and Trinidad 2007). A general grounded theory method contains six major steps: research question generation, data collecting/sampling, open coding, merge codes, theme generation, and theory building. Each step from the grounded theory method requires domain experts' involvement. Recent advancements in artificial intelligence, particularly machine learning and deep learning methodologies, have brought potential benefits to the qualitative research community (Chen et al., 2018). Muller et al. (2016) identified the connection between machine learning and the grounded theory method in inductive reasoning. The collaboration between machine learning and grounded theory is still in its infancy (Singh et al., 2020). Nelson (2020) introduced a computer-aided three-step grounded theory framework fully supported by topic modeling techniques. Apart from a topic model, other machine learning methods can be applied to qualitative codings, such as supervised text classification (Smith and Tissing 2018) and the text clustering method (Henry et al. 2015).

3.4. Semantic similarity algorithms

The thesaurus-based approach, also known as the topological or knowledge-based approach, uses lexical information from the knowledge base to perform calculus. Multiple types of semantic relations are identified (as Figure 16), including inclusion, possession, attachment, attribution, antonyms, synonyms, and case (Storey 1993).

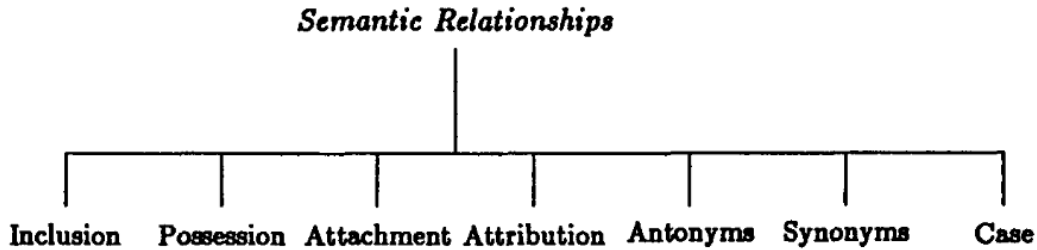


Figure 16. Semantic relationships, Storey. V (1993)

Semantic relations measuring is an essential subfield of NLP and semantic relatedness, and it is dedicated to several critical NLP applications, such as Question Answering (QA), Machine Translating (MT), Text Summarization, Word Correction, and Complete and Text Paraphrase.

WordNet is a well-known open-source lexical knowledge base containing semantic relations between words in multiple languages. In addition, there is a brief definition called “gloss” for each word. Synonyms are grouped into “*synsets*” without order, and these “*synsets*” and the relationship between them build up the whole lexical knowledge graph. There are four different types of relations in the WordNet, which are hypernymy (also known as “is-A”, hyponymy or super-subordinate), meronym (part-of) relation, holonymy (contain) relation, and antonyms relation. The field of semantic relatedness study considers all of the relationships in the WordNet; however, semantic similarity only studies the relation “is-A”.

With the information provided by the thesaurus like WordNet, there are three strategies to calculate the similarity/relatedness between two words. These strategies are described as path-based measures, Information Content (IC) based measures, and gloss-based measures. Here are definitions for related terms and concepts.

Let W be the set of words in the WordNet, and we want to calculate the similarity between two words $w_i, w_j \in W$, the number of links in the shortest path between w_i, w_j are described as $len(w_i, w_j)$.

The first family of similarity algorithms is called shortest path-based algorithms. This type of algorithm counts the edges between two terms and uses that measurement to determine the

closeness of two words. In thesaurus like WordNet, words and relations are organized into a graph structure, and the relationships between words are edges in the graph.

The most common path-based similarity measure is:

$$sim(w_i, w_j) = \frac{1}{(1 + len(w_i, w_j))} \quad (1)$$

The equation takes the reciprocal of the $len(w_i, w_j)$ to indicate the connection between two words. As the shortest path-based algorithm improved, the least common super concept was introduced to measure semantic relatedness (Wu & Palmer, 1994). In Figure 17, shown below, to compute the similarity between w_1, w_2 , the depth of least common super concept w_3 is also considered.

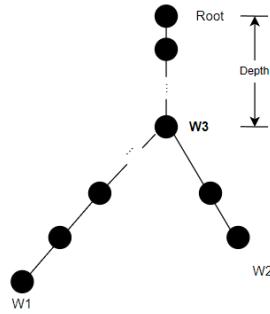


Figure 17. The illustration of the common super concept.

$$sim(w_i, w_j)_{Wu\&Palmer} = \frac{2 * len(w_i, root)}{(len(w_i, root) + len(w_i, w_j))} \quad (2)$$

Leacock and Chodorow measure semantic relatedness by considering the maximum depth of the graph. The equation is shown below, where $MaxDepth$ refers to the maximum depth of the given structure.

$$sim(w_i, w_j)_{Leacock\&Chodorow} = -\log \left(\frac{len(w_i, w_j)}{2 * MaxDepth} \right) \quad (3)$$

Resnik (1995) first introduced the notion of Information Content (IC). The IC is defined as the negative log-likelihood of the frequency of a given the word in a semantic network, and the detailed equation is shown in equations (4), (5), and (6), where $p(w)$ is probability of encountering

and instance of word w ; $freq(w)$ represent the frequency of the instance; and N denote the total number of words in the graph.

$$IC = -\log p(w) \quad (4)$$

$$p(w) = \frac{freq(w)}{N} \quad (5)$$

$$freq(w) = \sum_{w \in subset(W)} count(w) \quad (6)$$

With the explanation above, the Resnik algorithm can be described as equation (7), where $lcs(w_i, w_j)$ is the least common super concept in the graph.

$$sim(w_i, w_j)_{Resnik} = IC(lcs(w_i, w_j)) \quad (7)$$

Based on the IC concept, Lin proposed an algorithm related to IC of both the least super concept and individual words (as equation 8).

$$sim(w_1, w_2)_{Lin} = \frac{2 * IC(lcs(w_i, w_j))}{IC(w_i) + IC(w_j)} \quad (8)$$

3.5. Text vectorization and contextualized embeddings

Text vectorization to transforming text into numbers that machines can calculate. Machine learning and deep learning algorithms work on numeric vector space, meaning the learning model's input needs to be represented by numbers. Once the text is represented with the numbers, it can make both arithmetic and logical operations on the text.

Term Frequency-Inverse Document Frequency (TF-IDF) is a popular feature weighting method among the included works. TF-IDF neutralizes the term frequency with the number of documents containing that word to draw a weighted feature value, reflecting how important a feature (term or

n-gram) is to a document in a corpus. Contextualized word embeddings technique is an effective solution to represent text with numbers. It can learn to represent each token from the entire input sentence, dramatically boosting performance on tasks with more ambiguous, challenging, and unseen text data (Arora et al. 2020; Peters et al. 2018). Although the contextualized word embeddings reached a state-of-the-art performance with multiple downstream tasks, it is considered an expensive embedding method for both training and inference, and optimization of this type of method is still a research problem (Liu et al. 2019). A recent study tried to find a balance between contextualized word embedding and non-contextualized word embedding methods and concluded that when to use which type of method is determined by the dataset volume, complexity of the text structure, ambiguity of the word usage, and presence of unseen words (Arora et al. 2020). In addition, the performance of a text clustering algorithm relies on the quality of the text feature representations. Hence, we decided to use contextualized word embeddings for the unsupervised learning tasks.

3.6. Transfer learning

Applying a pre-trained language model or checkpoints is a revolutionary milestone of NLP and becoming a standard process for a wide range of NLP tasks (Edunov, Baevski, and Auli 2019; Gururangan et al. 2020; Rothe, Narayan, and Severyn 2020). A pre-trained language model is a general-purpose language model trained from a massive text source. A checkpoint is the exact value of all learned parameters without any information about the model architecture. Increasing numbers of checkpoints and pre-trained models are uploaded to publicly accessible communities such as GitHub, Hugging face transformers, and TensorFlow hub. Introducing pre-trained models into a custom framework (or “warm-start”) would reduce the local pretraining time and improve performance significantly (Rothe et al. 2020).

3.7. Design knowledge

We introduced the EBD analysis and the design question generation procedure in the former section. Answering the generated design questions usually require sufficient design knowledge and domain knowledge. EBD closes the gap to design question generation and insufficient design knowledge by providing a systematic question generation methodology. However, answering the design question requires designers to have sufficient knowledge. This section will introduce design knowledge with a design knowledge taxonomy that is a simple integration of existing design research.

Design knowledge in design science is classified into multiple categories based on different perspectives. Based on the knowledge representation clarity, the design knowledge can be categorized as explicit and implicit (Wong & Radcliffe, 2000). The design knowledge can be further categorized into learning skills, social skills, product knowledge, and environmental knowledge from the designers' capability perspective (Friedman 2000). Ahmed (2005) categorized design knowledge into design process knowledge, product knowledge, function knowledge, and knowledge about issues based on a hypothesis developed from the designers' perspective. According to design knowledge reusability, design knowledge can be categorized into process knowledge, product knowledge, and task knowledge (Baxter et al. 2007). Based on the EBD product environment classification, the design knowledge can further be categorized into natural, built, and human-environmental knowledge. The summary of the design knowledge taxonomy is in Table 3. Based on the explanation from existing works and EBD methodology, the definition of the design knowledge can be extracted. Design knowledge is a combination of knowledge about the design process, techniques, and the product to be designed.

Authors	Categorization of design knowledge
Wong & Radcliffe (2000)	Explicit design knowledge, tacit design knowledge
Friedman (2000)	Learning skills, social skills, product knowledge, environment knowledge
Ahmed (2005)	Design process, product knowledge, functions, issues
Baxter et al. (2007)	Process knowledge, product knowledge, task knowledge
Yong Zeng (2020)	Natural environmental knowledge, built environmental knowledge, human environmental knowledge

Table 3. Design knowledge taxonomies from different works.

The design process and techniques are fundamental knowledge for a designer, usually learned from standard training systems or past project experiences. The design process and techniques include multiple beneficial design activities, such as questionnaire design, interview question preparation, prototype development, and brainstorming. Product knowledge is everything about the product to be designed, such as the product's dimension, function, cost, component and assembly methods, and similar alternative artifacts. To system design, the system is the product. The system design knowledge can be further classified into three environmental knowledge as EBD categorized.

3.8. Knowledge acquisition

With the EBD methodology, design problems can be formalized and decomposed into primitive, atomic design problems represented by multiple simple questions. To answer these questions, knowledge, and experience are inevitable. Knowledge exists in many sources with various formats. Knowledge can be stored in the mind of an individual, such as common-sense -- *water can turn into steam* that is known to everyone. Knowledge can also be stored in forms outside of the human brain, such as books, documents, databases, messages, voice recordings, and scientific literature. Without readily available tools, guidelines, and procedures, acquiring knowledge for design activities is time- and labor-intensive. Systematically leveraging knowledge from these sources can boost design efficiency.

Knowledge acquisition is the process of uncovering specific knowledge that contributes to human performance and the cognition issues associated with this process. The concept is initially coined by the field of knowledge engineering, which is majorly used for constructing knowledge-based systems or expert systems. Knowledge-based System (KBS) development was considered a process of transferring human knowledge into an implemented knowledge base in the early 1980s (Studer, Benjamins, and Fensel 1998). Knowledge engineering is a long-standing branch of artificial intelligence that aims to create a system capable of imitating expert knowledge (Hahn and Schnattinger 1998). The knowledge engineering process involves five major steps (Figure 18): knowledge acquisition, knowledge representation, knowledge base construction, knowledge inference, and justification (Jung et al. 2020; Sharda et al. 2014). The knowledge engineering approaches provide a systematic guideline for building an information system. Knowledge elicitation is a broad term that covers concepts from early knowledge engineering steps: knowledge acquisition, knowledge representation, and knowledge base construction. The three steps involve knowledge engineers extracting knowledge from experts or the knowledge source in the same way system analysts, and designers elicit user requirements (Gavrilova and Andreeva 2012).

The importance of knowledge acquisition to product development is reflected in three folds. First, knowledge management is vital for a development team, and extracting knowledge to document existing knowledge with an integrated, centralized, accessible knowledge hub (or wiki) is a good team knowledge management practice (Sousa, Aparicio, and Costa 2010). Second, for cross-domain product design and development, extracting knowledge of the target domain is essential to build a product that can be easily applied and accepted by users in that domain. Third, it is essential to elicit knowledge about stakeholders, understand how stakeholders interacted with the product, what functions or features most stakeholders requested, and which part of the product is

most liked/disliked by the users. Knowing stakeholders bring more targeted strategies for the product design.

Hence, eliciting knowledge from various sources could help the design questions answering. However, manual knowledge extraction from different sources is inefficient, bringing many potential problems. Manual knowledge learning or acquisition is a time-consuming and error-prone process. Especially when there is no senior expert in providing timely feedback, it could be a disaster. For some domains, such as healthcare and aerospace design, the experts are expensive to consult.

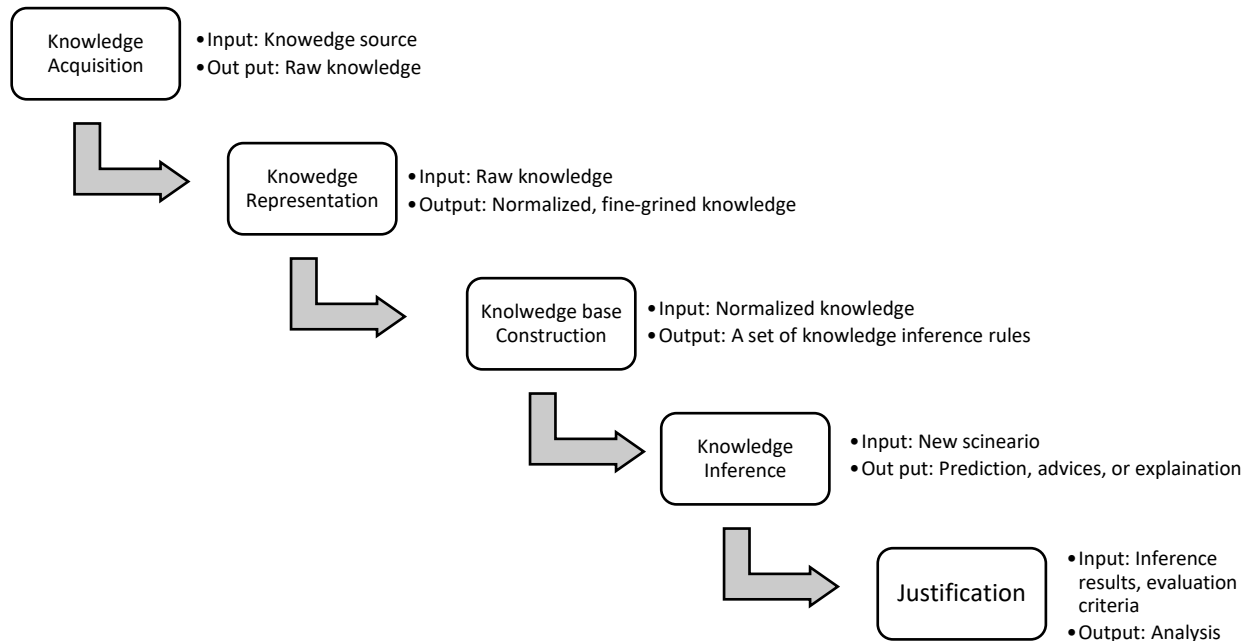


Figure 18. General knowledge engineering process.

3.9. Engineering semantic networks

Researchers have invented hundreds of knowledge representation techniques to map knowledge into computers so far (Davis, Howard, and Peter 1993). One of the goals of using Knowledge Representation (KR) is to represent the knowledge and store it in computer systems to process it efficiently by both human and computer systems.

Language is a system designed and used by language users to deliver the meaning they want to express. Understanding the meanings of the language is essential in a conversation. Until today, it is still difficult for machines to perform Natural Language Processing (NLP) and Information Retrieval (IR) tasks as an average normal human being do. The study of semantics is to reveal the meaning of any form of human expressions, such as language semantics and image semantics, and using a graph dataset to represent the lexical knowledge is defined as the lexical knowledge graph or semantic network in this thesis.

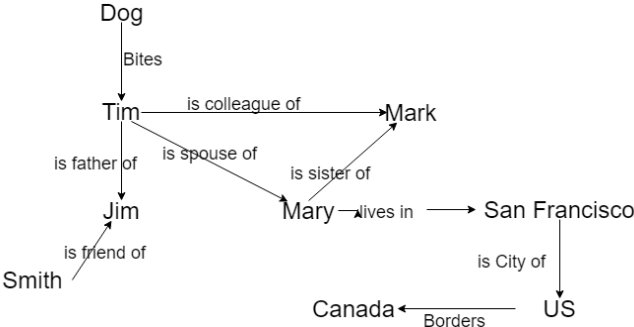


Figure 19. Semantic network without taxonomy

A Semantic Network uses graph notation, which consists of interconnected nodes, to represent knowledge by representing relationships and dependencies between concepts (Sowa 1987). Semantic networks can be easily understood by humans and machines. For example, from the graph listed in Figure 19, we can directly transfer the graph into formalized logical format *is_father_of(Tim, Jim)* or natural human language “*Tim is Jim’s father*”.

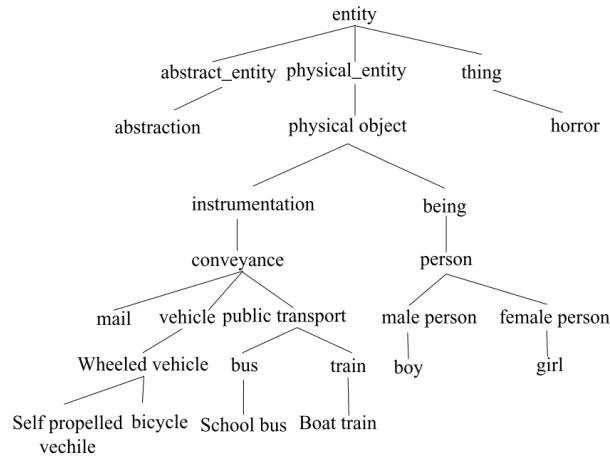


Figure 20. An illustration for WordNet (Meng, Huang, and Gu 2013).

Inspired by human lexical memory and semantic network, researchers from Princeton University introduced a lexical semantic network named WordNet (Miller, 1995). WordNet only includes nouns, verbs, and adjectives and divides them into four lexical categories: nouns, verbs, modifiers, and function words. WordNet is the most well-developed and widely used lexical database for English. Unlike other thesaurus or lexical knowledge bases, WordNet is handcrafted from scratch instead of mining from structured or semi-structured sources. As Figure 20 shown, terms are organized in hierarchy and taxonomy relationships.

Design knowledge formalization, modeling, and representation have been an ongoing research hotspot during the past few decades. For design knowledge modeling, ontologies are the most commonly applied frameworks, which conceptualize terms and the relationships among them with formal specifications (Gruber 1993; Noy and McGuinness 2001). The term “ontology” is a controversial concept, which was first proposed in the field of philosophy, and now it extends to various domains such as Natural Language Processing (NLP) (Miller et al., 1993), design knowledge management (Štorga, Andreasen, and Marjanović 2010), and healthcare (El-Sappagh et al. 2018). Lin et al. (1996) proposed a design knowledge model called requirement ontology to support requirement analysis by modeling the product knowledge into concepts and different types

of relationships. With the help of the requirement ontology, the engineer can refine, trace, validate, and change the customer requirement in the product design. Štorga et al. (2010) introduce an ontology framework based on a taxonomy of product concepts, including objects, processes, attributes, design attributes, propositions, quantities, and relations. Each category is further divided into subcategories of refinement. Some works proposed ontology to model the design itself, such as FBS ontology (Gero and Kannengiesser 2014), generic design activity ontology (Sim and Duffy 2003), TIPS ontology (Fernandes et al. 2007), PDO (Catalano et al. 2009), and DSO (Rockwell et al., 2009; Rockwell et al., 2010). Ontology and a variety of models of knowledge were used in the early works of knowledge-based design systems, and it is required that domain experts participate in most of the design activities, such as acquiring knowledge (Dixon et al. 1987), knowledge graph construction (Yoshioka et al. 1998), and knowledge validation (O’Leary 1991). A semantic network is an alternative approach for knowledge formalization to support engineering design decision-making. Unlike ontology, a semantic network usually does not require a pre-defined schema, instead, it uses node, relations to formalize narrative description (Sowa 1987). Usually, in an engineering design semantic network, a node represents a semantic entity, such as a concept, an idea, a physical or virtual location, or even a name of a person (Frisch 1982; Han et al. 2022; Sowa 1987). In addition to benefiting design simulation, knowledge sharing, and organizational best practices, semantic networks can also help in determining organizational best practices (Rogers, Priest, and Haddock 1995). A list of reviewed engineering semantic networks is summarized in Table 4.

Authors	Focus	Elements	Description	Downstream application
Rogers et al., (1995)	Knowledge model design	Node (term), Relationships (activity, interactions)	Proposed a conceptual design of a design semantic network that supports concurrent engineering in the design of semiconductor devices. The main application scenario of the design of semantic networks is for communication enhancement.	Conceptual design, communication support
Tiwana & Ramesh (2001)	Knowledge management system design	Node (term), Relationships (activity, interactions)	Proposed a knowledge management system that allows engineers to manually enter concepts and relationships between them.	Design knowledge acquisition and management
Hao et al., (2011)	Feature Semantic network design and automated construction.	Node (Features), Relationships (Operations)	The authors proposed a semantic network to support feature-based design. The	Formalized design description
Tawosi et al., (2015)	Product domain semantic networks	Node (entity) Relationships (6 defined relations)	A product domain semantic network provides the human engineer with a piece of sufficient background knowledge.	Building models for detailed design. ER diagram,
Shi et al., (2017)	B-link knowledge Graph construction, information retrieval	Node (term), Relationship (co-occurrence)	Mine concept (terms) from engineering papers (1 M) and connect the concept with the phrase co-occurrence. Introduced a threshold to reduce the graph size. In addition, they provide algorithms for information retrieval.	Design information retrieval
Sarica et al., (2019, 2021)	Automatic graph construction, TKG, TechNet	Node (term), relationship (distance)	Text mining from USPTO (Design patent documents) obtained in total 26.8 million sentences and extracted 4 million terms based on Tf-idf. The relation in the graph indicates semantic relatedness (cosine distance).	Semantic search, network analysis

Table 4. Knowledge graphs and semantic networks are introduced in engineering design.

Chapter 4. An EBD-based Systematic Literature Review: Machine Learning in Requirement Elicitation

4.1. Introduction

Requirement elicitation is an early-stage product development process to gather stakeholders' detailed descriptions of the target system. Several traditional elicitation techniques, such as interviews, meetings, and brainstorming sessions, can be used to collect precise and individualized requirements. However, due to growing user demands and the rapid pace of product iterations, these conventional methods are often insufficient.

The fourth industrial revolution is triggering a pervasive digital transformation in many fields of human activities. Particularly, engineering is being transformed into “Digital Engineering” (Huang et al. 2020; US DoD 2018; Zimmerman 2017). In digital engineering, digital data and models will be shared in the engineering life cycle (US DoD 2018); engineering artifacts and processes will be digitalized with standardized digital representation, unique identifier, and augmented metadata about their attributes, including provenance, thus making those digital artifacts machine-processible, uniquely identifiable, traceable, and accountable (Huang et al. 2020). The digital engineering transformation brings both opportunities and challenges for requirement elicitation.

The evolution of digital transformation has led to improved productivity, quality, and customer satisfaction through agile and robust big data collection, analysis, learning, and decision-making processes. Success stories, advancing technologies, and growing customer demands are why digitalization has become necessary for various fields. For example, in recent years, there has been a growing number of studies involving a digital transformation in requirement engineering, such as identifying requirements from documents (Wang et al., 2019), and automatically classifying the requirements (Casamayor, Godoy, and Campo 2012), and prioritizing the requirements (Maiti and Mitropoulos 2017). By applying advanced technologies and shifting the existing process to a new digitized paradigm, it may be possible to solve the problem.

Traditionally, expert experience or intuition has directed requirements activities. Each decision is based on a combination of implicit and explicit domain expertise. Developing a computer model that mimics expert rules is expensive to construct and maintain, if not impossible. A data-driven strategy, unlike rule-based systems, does not codify the rules and knowledge for decision-making. The term *data-driven* refers to a decision-making strategy based on data analytics, interpretation, and prediction rather than pure intuition (Provost and Fawcett 2013). Over the past fifteen years, several studies have been published on the application of machine learning to requirements engineering and systems engineering, followed by reviews that summarize these studies (Lim et al., 2021; Meth et al., 2013; Wong et al., 2017). Different from those existing studies, this current literature review, covering 86 studies, provides a roadmap for building an ML-based requirement elicitation pipeline, including data collection, data preprocessing, feature extraction, training, and evaluation of the model and the open-source tools.

The rest of this chapter is structured as follows. In Section 4.2, literature reviews related to the proposed review are summarized; and in Section 4.3, the scope and methodology of the literature

review, as well as search strategies, criteria for inclusion and exclusion, and the data extraction template, are presented. Section 4.4 shows the primary results of the literature review. Section 4.5 summarizes the major findings from the review by analyzing the included works and categorizing them into various categories from seven different research concerns. In Section 4.6, the current role of ML in requirement elicitation and its limitations are discussed. In addition, the open issues and potential future works in this field are discussed. In Section 4.7, we discuss the potential threat to the validity of the review and the measures we took to address these limitations. Finally, Section 4.8 concludes the paper.

4.2. Research methodology

4.2.1. Literature review and design

There is a paradox in conducting literature review research: before completing the literature review, the researcher does not have a holistic overview of the study; before the researcher has a holistic view of the study, the researcher does not know what to search for; before the researcher search, the researcher does not have any knowledge about the topic. The literature review procedure is described as linear rather than recursive. Hence, there is a contradiction existing between the research statement and the researcher's knowledge. The research statement helps the researcher to retrieve knowledge from correct research works, and the knowledge helps the researcher to complete the research statement. However, most textbooks and guidelines on conducting a literature review do not address this contradiction. Instead, they assume that the audience has a well-designed research question and is capable of writing sufficient and appropriate research questions.

In design science, a similar logical paradox is described by Zeng (2020) as the recursive nature of the design process. The recursive design process describes a recursively evolutionary process for design. Before the designer has the design knowledge and design solution, the designer cannot fully capture the requirements from the design statement; before the design statement is fully captured, the designer is unable to get sufficient design knowledge, which results in an incomplete design solution. Further, through over 30 years of studies, scholars have proven that the design process is recursive rather than linear (Jia and Zeng 2021; Thanh An and Zeng 2012; Zeng and Cheng 1991). The recursion comes from the designer's enhanced understanding of the design problem.

Comparatively, literature review research shares the same logic and paradox with design. We can relate the literature review in the design to three activities, namely, the research statement (design statement), the reviewed literature, and the study results. The initial research statement provides the researcher (or designer) with a vague direction and scope regarding the study. The researcher further analyzes the research statement and searches related literature. With the initial set of research papers, the researcher can extract initial knowledge, which helps the researcher to make the research statement complete and valid. Thus, the EBD methodology is applied to this literature review to generate relevant research questions.

This study contains two parts, which are the data preparation part, and the paper writing part. For data preparation, we customized the RomNet framework for the literature review to help the researchers to extract useful information from research articles.

4.2.2. EBD-based research question generation

The literature review methodology in this chapter contains research question generation, literature retrieval, and literature review. This literature review's research questions are generated with Environment Based Design (EBD) methodology (Zeng 2020).

EBD is a design methodology that was initially a design guideline for designers with limited knowledge (Zeng 2011). Because of its question-driven and structured nature, EBD acts as a literature review method for secure collaboration (Zeng et al. 2012). EBD provides an adaptive question-driven information retrieval method to ensure the objectivity of design activity. Researchers from different fields applied EBD for various problems due to its supportability to complex and uncertain problems, such as education science, management science, computational geometry, and neuroscience (Jia & Zeng, 2021; Jie et al., 2021; Meng et al., 2021; Milhim & Schiffauerova, 2013; Wang & Zeng, 2017). EBD can solve various uncertain problems because of its ability to represent problem statements and generate questions in a logical order.

The Environment-based Research Question (eRQ) generation used in this review starts with an initial statement that describes the central theme of this chapter. Specifically, there are seven steps to generate research questions: 1) to generate a Recursive Object Modeling (ROM) diagram to represent the research statement in a structured formal format; 2) to extract key terms from the ROM diagram; 3) to generate lists of questions according to the question templates provided in EBD methodology; 4) to seek answers for the questions following an EBD answer template; 5) to merge the topic sentence of each answer into an integrated final statement; 6) to update the ROM diagram based on the final statement; 7) conduct environmental analysis on the updated ROM diagram, and generate final research questions. The entire procedure is shown in Figure 21.

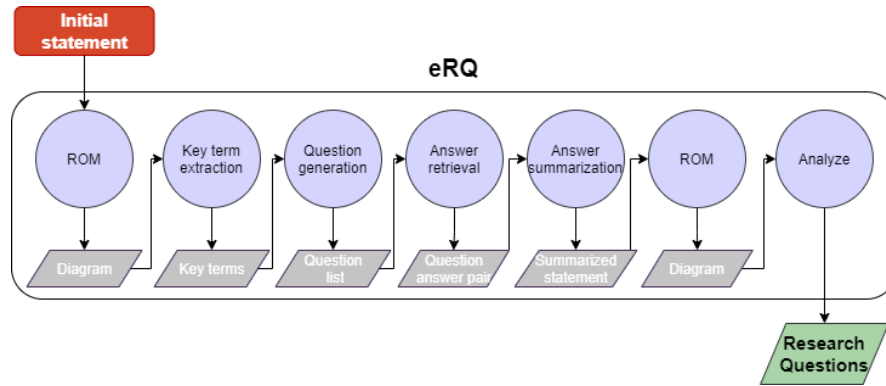


Figure 21. Illustration of the eRQ process for generating research questions.

The objective of the first three steps is to analyze the initial statement and generate a list of initial questions that provide direction and scope for information searching. During these steps, the core function is ROM which generates structured questions for a single statement (Zeng 2008). The ROM is a semantic language model representing text in a structured dependency graph. The initial statement of this chapter is:

Data-driven Approach for Intelligent Requirements elicitation.

According to ROM construction rules (Zeng 2008), we generate a ROM diagram, as is shown in Figure 22. The next step is to extract keywords for question generation with the generated ROM diagram. The key terms and phrases are “*requirement*”, “*requirements elicitation*”, “*data*”, “*data-driven*”, and “*intelligence*”. The third step is generating questions about key terms. The key-term extraction and the question generation details are introduced in Section 2.1.

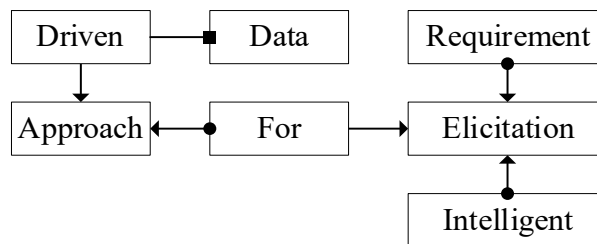


Figure 22. The ROM diagram for the initial statement

According to EBD question generation rules, we generated 11 initial questions and answered the questions with knowledge retrieved from the existing works of literature (as Table 5).

What is a requirement? The definition of requirements is slightly different in different domains. In the engineering design field, a requirement is a physical product or system that “must be” created and delivered to satisfy the customers' needs (Brace and Cheutet 2012). In software engineering, requirements represent the “must-have” functionalities and features of the software to solve problems in the real world (Bourque and Fairley 2014). For service design, the requirements are the service delivery and service development request from the customer (Patrício et al. 2011). The design requirements describe the target product's structure and performance (Cascini et al. 2013; Zeng and Gu 1999). In combination, a requirement means a written description of functions, attributes, or the quality of the target product that stakeholders expect.

Question-id	Question	Key term
IQ1	What is a requirement?	Requirement
IQ2	What is the lifecycle of the requirement?	Lifecycle, requirement
IQ3	What is requirements elicitation?	Requirement elicitation
IQ4	Why elicits requirements?	Requirement elicitation
IQ5	How to elicit requirements?	Requirement elicitation
IQ6	Who elicits requirements?	Requirement elicitation
IQ7	When to elicit requirements?	Requirement elicitation
IQ8	Where to elicit requirements?	Requirement elicitation
IQ9	What is data?	Data, information, knowledge
IQ10	Why driven by data?	Data-driven
IQ11	What is “data-driven”?	Data-driven

Table 5. Initial questions (IQs).

What is the lifecycle of the requirement? According to Brace & Cheutet (2012), the requirement development process comprises customer requirements identification, defining specific requirements, requirement refinement, requirement exploration, and requirement specification. In ISO/IEC/IEEE 29148:2011 standard, the requirement process contains extracting requirements, defining requirements, analyzing the requirement, and maintaining requirements (García et al.

2020). Almfelt et al. (2006) indicated that the requirement lifecycle should stick together with the entire product life cycle. The requirement lifecycle includes business requirement analysis, product requirement extraction, system requirements elicitation, specific requirements generation, and requirement validation. In SWEBOK V3.0, the requirement lifecycle is generalized into four steps, which are requirements elicitation, requirement analysis, requirement specification, and requirement validation (Bourque and Fairley 2014). Researchers in the requirement engineering community also believe that the requirement process is an ongoing process that can continuously change through user feedback and online review (Jha and Mahmoud 2019; Nayebi, Cho, and Ruhe 2018; Oriol et al. 2018). Hence, by combining these different definitions, requirement lifecycles can be generalized as 1) initial requirements elicitation, 2) requirement analysis, 3) requirement documentation, 4) requirement validation and 5) requirement maintenance.

What is requirements elicitation? Requirements elicitation is a process that contains a set of activities in the initial phase of engineering, which aim to gather, identify or discover the application domain, the required performance, the task-related stakeholders, and business rules (Bourque and Fairley 2014; Sommerville and Sawyer 2003). Requirement elicitation is the first phase in the requirement lifecycle, which identifies initial requirements for the project and the continuous incremental requirements from various sources such as user feedback and reviews. The requirements elicitation's main tasks include understanding the application domain, identifying requirements sources, analyzing related stakeholders, selecting proper requirements elicitation techniques, and eliciting requirements (Zowghi and Coulin 2005).

Why elicits requirements? Product definition is one of the most critical factors for product design and development, and product definition comes from customers' requirements (Yan, Chen, and Khoo 2002). It is also essential for an organization to decide on various detailed activities such as

budget estimation and project scheduling (Azadegan, Papamichail, and Sampaio 2013). Requirements elicitation is vital to define clear project scope and build a communication channel to minimize the potential risks (Bourque and Fairley 2014).

How to elicit requirements? Hundreds of requirements elicitation techniques were developed and applied by academia and industries to elicit requirements from stakeholders (Zowghi and Coulin 2005). To answer *how to elicit requirements*, we only illustrate a few requirements elicitation techniques as examples commonly applied by multiple domains: interview, questionnaire, prototyping, domain analysis, brainstorming, and observation.

Who elicits requirements? Requirement engineers and users are two main actors for requirements elicitation, and the degree of their involvement depends on the specific technology for requirements elicitation. Requirement engineers in this chapter refer explicitly to designers and engineers in the requirement team. For example, an interview requires both requirement engineers and target stakeholders to attend together, whereas observation does not depend on stakeholders.

When to elicit requirements? Requirements elicitation is the first phase of engineering design and helps designers understand the problems to be solved (Bourque and Fairley 2014; Yan et al. 2002). However, the design process is not always sequential. The design process is a recursive process that consists of several stages (Zeng and Cheng 1991). The current stage's design solution becomes part of the design requirements in the subsequent design stages (Wang and Zeng 2009). Hence, design requirements and design solutions span through the entire product life cycle. The recursive nature of customer requirements determines that requirements elicitation is an ongoing activity that spans throughout the entire product development life cycle.

Where to elicit requirements? Designers can extract requirements through communication with stakeholders, such as interviews, questionnaires, observation, brainstorming, and formal or informal meetings. Requirements can also be found in many types of documents, e.g., tenders documents (Fantoni et al. 2021), crowdsourcing project requests (Li et al., 2018), product-relevant multimedia documents (Dadzie et al. 2009), domain documents (Wu & Srihari, 1996), technical documents (Garzoli et al. 2013) and problem record document (Xu et al. 2020). Besides, customers usually express their demands and expectation directly or indirectly through feedback and reviews (Birch, Simondetti, and Guo 2018; Maalej, Nayebi, and Ruhe 2019). Moreover, researchers suggested other sources such as data from existing systems (Rusu et al. 2012), social networks (Lim & Finkelstein, 2012), and gamified platforms (Kolpondinos and Glinz 2020).

What is data? Oxford dictionary defines the term data as *facts and statistics collected for reference or analysis*¹. More specifically, data are a numerical or symbolic representation of the real world (Targowski 2005). Data is the statistical observations and computer recordings in the most basic form (Zins 2007). Data can be categorized, interpreted, analyzed, and extracted to produce meaningful output by human experts and artificial intelligence (O’Leary 2013).

Why driven by data? Although conventional requirements elicitation techniques are robust and have many mature procedures and guidelines to direct requirement engineers and stakeholders to gather requirements, these existing techniques are labor-intensive, time-consuming, and error-prone (Cleland-Huang et al. 2007; Reinhartz-Berger and Kemelman 2020). For example, there are about 40,000 documents generated during a single-engine project in an aerospace company, and most of them are written in plain text (Li and Ramani 2007). Handling massive documents multiply

¹ https://www.oxfordlearnersdictionaries.com/definition/american_english/data

the workload of requirement engineers, and a heavy workload might lead to substandard performance (Nguyen and Zeng 2012). Also, requirements tend to change during the entire project lifecycle, which requires additional attention from engineers to cope with the changing requirements (Morkos, Mathieson, and Summers 2014). Moreover, the traditional requirements elicitation methods require multiple skills and knowledge, such as domain knowledge and interpersonal skills. Using data-driven methods can provide automatic tools in the target field, which can reduce designers' workload on a particular task or replace the process with an automated procedure that requires minimal human intervention.

What is “data-driven”? Data-driven is an adjective that means *based on or decided by collecting and analyzing data*². The noun form data-drivenness method uses technologies, tools, and processes that act on data (Turner 2015). Unlike conventional decision-making strategies based on human experience, the data-driven represents a strategy that requires knowledge learned from data (Provost and Fawcett 2013).

By summarizing the answers in the question-answer pairs, we generated a brief description of the target domain, and the answers are organized as follows.

The requirement means the description of functions, attributes, or the quality of the target product expected by stakeholders. Requirement lifecycle contains, 1) initial/continues requirements elicitation 2) requirement analysis, 3) requirement documentation, 4) requirement validation and 5) requirement maintenance. Requirement elicitation is a process that contains a set of activities in the initial phase of engineering, which aim to gather, identify, or discover the application domain, the required performance, the task-related stakeholders, and business rules. Requirement elicitation is essential to define clear project scope and build a communication channel to minimize the risks. Multiple methods

² <https://www.oxfordlearnersdictionaries.com/us/definition/english/data-driven?q=data-driven>

elicit requirements, such as interviews, questionnaires, prototyping, domain analysis, brainstorming, and observation. Requirement engineers and users are the two main actors for requirements elicitation. Requirement elicitation is an ongoing activity that spans the entire product development lifecycle. Various requirement techniques can extract requirements from end-users. Data is facts and statistics collected for reference or analysis. Data-driven is an adjective that means based on or decided by collecting and analyzing data. The noun form data-drivenness is a method that uses technologies, tools, and processes that act on data. Using data-driven methods can automate several requirements elicitation tasks that reduce requirement engineers' workload.

A ROM diagram (Figure 23) is generated based on the answer summary, where related answers are merged and connected. The ROM diagram can be divided into seven parts, which are represented by different colors. The summary of these different clusters is presented in Table 6. Among these clusters, the red cluster represents the “requirement” and “requirements elicitation” that works as a central hub in this model. The yellow part represents the conventional requirements elicitation techniques, and the navy-blue part shows the motivation and lifecycle of requirements elicitation. The purple part lists the main actors in requirements elicitation. The grey zone is the lifecycle of a requirement that is not the primary focus of this study. The white cluster lists the objectives of requirements elicitation, and the green part is the study scope of this chapter, which is the data-driven requirements elicitation. We can generate effective research questions by analyzing the relationship between the green cluster and other clusters.

The relation between the green cluster and the red cluster indicates the relation between data-driven methods and requirements elicitation. In addition, the relationship between the green cluster and the blue cluster refers to the motivation of data-driven requirements elicitation. Third, understanding how the green cluster is related to the white cluster helps us find the objectives of data-driven requirements elicitation. Fourth, the relation between the green cluster and the yellow

cluster represents the data source of data-driven requirements elicitation. Besides, the connection between the green and the purple parts raises the question about who attends to the data-driven requirements elicitation.

Based on the ROM analysis, six research questions can be generated as follows.

RQ1. Which specific requirement elicitation activities are supported with ML methods?

RQ2. Which types of data sources are being used in the current works?

RQ3. What technologies are used for building an ML-based requirements elicitation methodology?

RQ4. What criteria are used to assess data-driven solutions?

RQ5. What are the available tools to support ML-based requirements elicitation methodology?

Based on the merged ROM diagram we can identify the main entities and the related keywords accordingly.

Cluster-id	Cluster	Color	Main entities
Clu-1	Data-driven	Green	Data-driven, data, techniques, tool, process, collecting, analyzing
Clu-2	Requirement elicitation	White	Process, gather, identify, discover, application domain, stakeholders, required performance, business rule
Clu-3	Definition of requirements	Red	Description, function, attribute, quality, target product, customer's expectation
Clu-4	Motivation and lifecycle of requirements elicitation	Blue	Minimize risks, communication channel, project scope, ongoing, entire product lifecycle
Clu-5	Conventional requirements elicitation	Yellow	Interview, questionnaire, prototype, domain analysis, brainstorming, observation
Clu-6	Actors in requirements elicitation	Purple	Engineers, users
Clu-7	Requirement lifecycle	Grey	Elicitation, analysis, documentation, validation, maintenance

Table 6. The description of each cluster and its main entities.

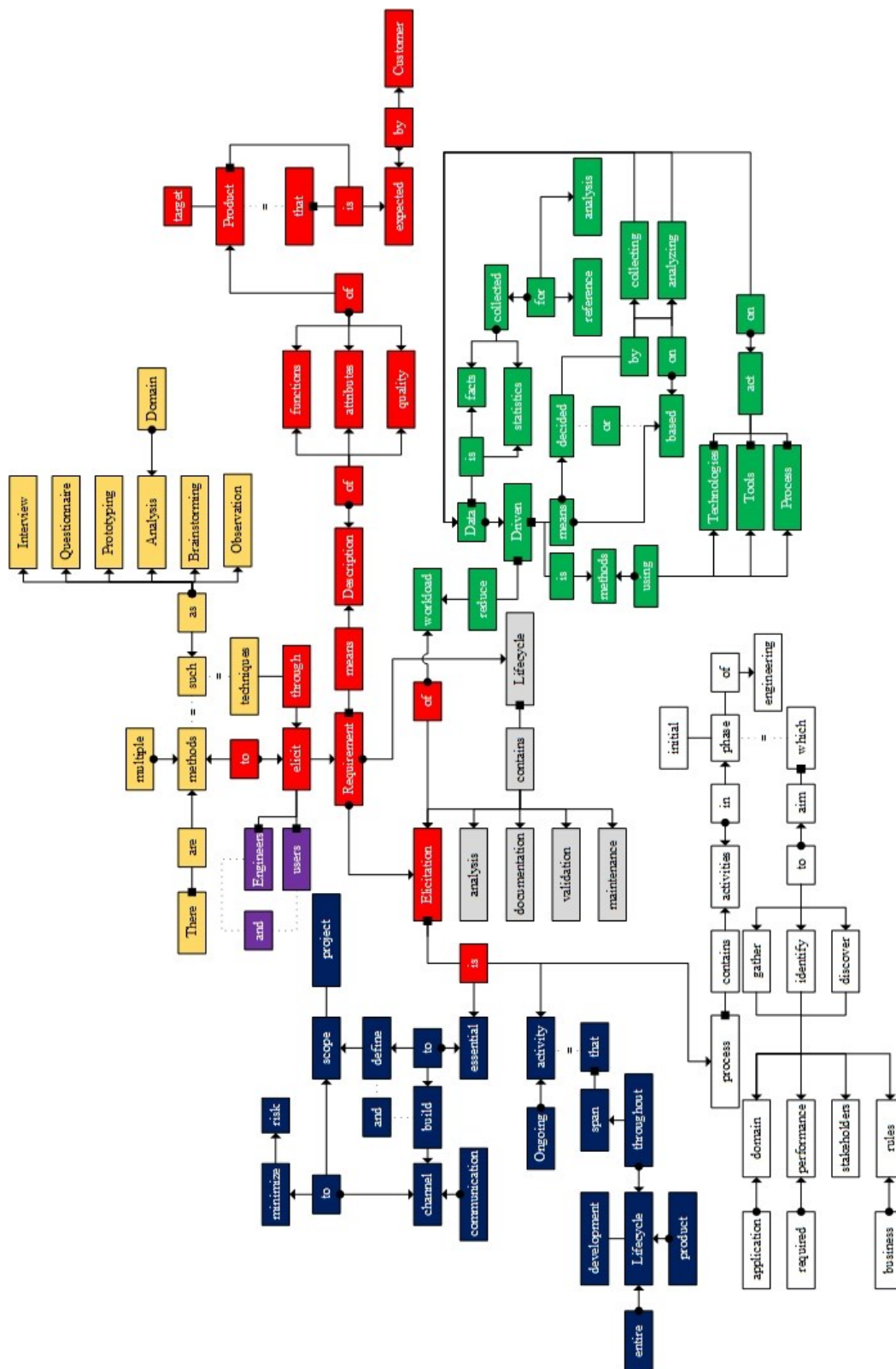


Figure 23. ROM diagram from first-round question answering

4.2.3. Review scope

Characteristic	Categories	The focus of this work
Focus	Research findings	✓
	Research methods	✓
	Practices of applications	✓
Goal	Theories	×
	Integration	✓
	Identification of the central issue	✓
Perspective	Criticism	×
	Neutral representation	✓
Coverage	Espousal of position	×
	Exhaustive with selective citation	✓
	Exhaustive	×
Organization	Representative	×
	Central or pivotal	×
	Methodological	✓
Audience	Conceptual	×
	Historical	×
	Specialized scholars	✓
	General scholars	✓
	Practitioners or policymakers	✓
	General public	×

Table 7. The research scope under Cooper's literature review taxonomy.

The review scope is defined in Table 7 according to Cooper's taxonomy for literature review, including focus, goal perspective, coverage, organization, and audience (Cooper 1988; Cooper, Hedges, and Valentine 2019). First, in this work, the emphasis is on practical solutions that can be applied; therefore, theoretical works are not our *focus*. Second, this study aims to synthesize and integrate existing studies to identify the tasks supported by ML-based requirements elicitation; thus, criticism of the field or related works is not a *goal* of this article. Third, this chapter does not take an espousal *perspective* to advocate for or against ML-based requirement elicitation. Instead, it demonstrates how the existing work would convert requirement elicitation challenges into ML problems. Fourth, the *coverage* of the literature is determined by its inclusion and exclusion criteria. Fifth, the work applies a methodological *organization* that group and organize similar methodologies or tools from the same step together, presenting a modular organization to the target

audience. Finally, the targeted *audiences* are mainly requirements analysts, engineers, and scholars to implement ML-based requirement elicitation solutions.

4.2.4. From the research question to the search string

According to the central theme of this literature review, we have key terms: requirements elicitation, data-driven, and intelligence. The search terms are expanded by introducing the synonyms of key terms. As a result, the following search terms are created to retrieve documents from electronic databases (Table 8).

Data-driven	OP	Requirement	OP	Elicitation	OP	Methods (techniques, process, tools)
Automated	AND	Requirement(s)	AND	Elicitation	AND	Machine learning
OR		OR		OR		OR
Automatic		Need(s)		Gathering		Deep learning
OR		OR		OR		OR
Data-driven		Demand(s)		Collecting		Natural language processing
OR		OR		OR		OR
Intelligent		Request(s)		Extraction		Artificial intelligence
				OR		OR
				Discovery		Neural network
						OR
						Data mining
						OR
						Text mining
						OR
						Data science

Table 8. Table of search terms and operators (op).

Seven bibliographic databases, including Scopus, Web of Science, Google Scholar, IEEE Xplore, Springer Link, ACM digital library, and ASME digital library, are adopted to guarantee the coverage of the review. Three search strategies are adopted: 1) the query expanding is used to add synonyms, inflectional, and derivational morphological forms to the original term; 2) a wildcard character is used to capture multiple forms of a keyword by replacing one or more characters with a star symbol (*) or question marks (?); and a 3) query scoping strategy is applied when the search

term is too general to retrieve a related result. The search queries and the search engines can be found in Table 9.

Search engine	Query
Web of Science	(TI=((requirement* OR demand* OR need*) NEAR (elicit* OR collect* OR gather* OR detect* OR identif* OR classif*))) AND TS=(data-driven OR automat* OR NLP OR ML OR deep learning)
Scopus	TITLE-ABS-KEY (("data-driven" OR "ML" OR "deep learning" OR "neural network" OR "text mining" OR "data mining" OR "NLP") AND ("requirement*" OR "need*" OR "demand*") AND ("elicit*" OR "collect*" OR "gather*" OR "detect*" OR "identif*" OR "classif*"))
Google Scholar	("data-driven" OR "automated" OR "automatic" OR "NLP" OR "text mining" OR ("machine" OR "deep") AND learning) AND ("requirement elicitation" OR "requirement engineering" OR "requirement identification" OR "requirement detection" OR "requirement collection" OR "requirement
Springer Link	Requirement AND (elicit* OR collect* OR gather* OR extract* OR detect* OR identif* OR classif*) NEAR (automat* OR "ML" OR "deep learning" OR "neural network" OR "text mining" OR "natural language processing" OR "nlp")
IEEE Xplore	("Publication Title":requirement) AND ("All Metadata":extract* OR "All Metadata":collect* OR "All Metadata":identifi* OR "All Metadata":detect*) AND ("All Metadata": "ML" OR "All Metadata": "deep learning" OR "All Metadata": "data-driven" OR "All Metadata": "neural network" OR "All Metadata": "automat*" OR "All Metadata": "natural language processing" OR "All Metadata": "nlp" OR "All Metadata": "text mining" OR "All Metadata": "supervise*" OR "All Metadata": "unsupervise*")
ACM digital library	[Title: requirement*] AND [[Title: elicit*] OR [Title: collect*] OR [Title: gather*] OR [Title: extract*] OR [Title: detect*] OR [Title: identif*] OR [Title: classif*]] AND [[Abstract: automat*] OR [Abstract: "ML"] OR [Abstract: "deep learning"] OR [Abstract: "neural network"] OR [Abstract: "text mining"] OR [Abstract: "natural language processing"] OR [Abstract: "nlp"]]
ASME digital library	"Data-driven requirement elicitation", "need elicitation."

Table 9. Search engines and queries.

4.2.5. Inclusion/exclusion criteria and data extraction table

The next step of the literature review is selecting studies by screening the title, abstract, and full text of the works found in the previous steps. We applied the inclusion/exclusion criteria in Table 10. The inclusion-exclusion criteria list pre-defined rules that work as filters to decide whether to keep or delete papers from the selected article list.

Research information was collected from each included article with a data extraction form. Characteristics of the study (author, title, year of publication, etc.) and measures of research interests (data source, preprocess, feature extraction, etc.) were collected. This includes 14 data

elements described in Table 11. A commercial literature review tool *Covidence*³ is used to design and manage the data extraction activities.

Exclusion	Inclusion
The work is a survey paper or literature review.	The work should either capture requirements in an automated way from text data or support the requirement elicitation process with an automated approach trained from big data.
The work is an editorial, conference abstract, or introductory popular science article.	The article presents the details of the datasets and data processing.
The work only generally describes the problem and the solution without providing an experiment result or convincing evidence.	The article explains in detail the ML algorithm that they employed, such as the learning algorithms they used and the validation strategy they employed.
If there are multiple similar works from the same authors, only one of the earliest works would be retained.	The full text of the article should be accessible.
The work is written in a non-English language.	The article is written in English.

Table 10. Search engines and queries.

#	Data	Description
1	<i>Author(s)</i>	Author(s) of the included work in the literature review.
2	<i>Title</i>	Title of the included work in the literature review.
3	<i>Country</i>	The country of the corresponding author is the primary country, and the rest of the countries follow the order of the author list.
4	<i>Year</i>	The published year of the work.
5	<i>Citation count</i>	The number of citations of included work. (From Google Scholar, until 2021-10-20)
6	<i>Venue</i>	The publication type of the work includes conference, journal, book chapter, and workshop.
7	<i>Venue title</i>	The name of the journal or conference contains the article.
8	<i>Requirement elicitation task(s)</i>	Identify which requirement elicitation subtask is supported by the paper.
9	<i>Data source</i>	The data source is used for training or validation purposes.
10	<i>Preprocess</i>	The specific data preprocessing techniques from the selected paper.
11	<i>Feature extraction</i>	The specific features applied by the selected work in the literature review.
12	<i>Learning algorithm</i>	The learning algorithm was adopted by the included study in the literature review.
13	<i>Evaluation method</i>	The evolution method was introduced by the selected papers in the literature review.
14	<i>Tools</i>	List the tools mentioned by the authors in the paper as aids to their work.

Table 11. Elements of data extraction.

³ <https://app.covidence.org/reviews>

4.3. Results

4.3.1. Results overview

A total of 975 papers were retrieved with the search queries from the seven included scientific search engines. Upon initial screening and title screening, 915 works were forwarded to title-abstract screening. A subset of 774 was irrelevant and thus discarded. As a result, 129 papers are retained for the full-text screening. According to the inclusion-exclusion criteria, 43 works were included in the literature review after 22 were excluded. The complete process of study selection is shown in Figure 24.

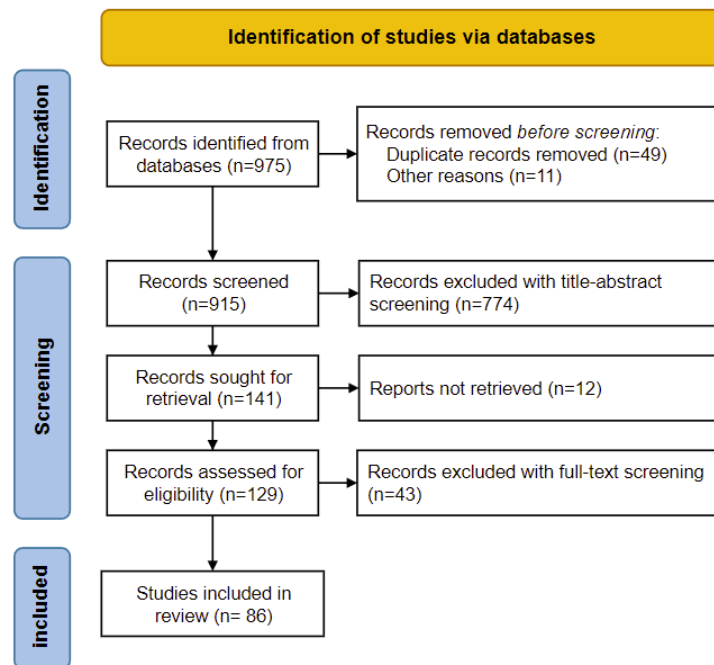


Figure 24. PRISMA flow chart.

Among the included 86 articles, an increasing trend was observed in Figure 25. The 86 studies came from 30 different countries, and 17 of them were conducted by more than one country. With 3.8 being the average of the included publications per country, seven countries published more: the United States (n=28, 25.0%), China (n=14, 12.5%), Germany (n=13, 11.6%), Canada (n=8, 7.1%), Singapore (n=5, 4.5%), South Korea (n=5, 4.5%), and the United Kingdom (n=5, 4.5%).

Thirty-nine of the studies are conference papers (n=39, 45.3%) and 31 are journal papers (n=31, 36.0%). In addition, eight workshop papers (n=8, 9.3%) and eight book sections (n=8, 9.3%) are included. The included conference papers are collected in 23 unique conference proceedings, with 16 appearing in the Proceedings of the IEEE International Conference on Requirements Engineering. The majority of the journal publications in this collection are from the Journal of Mechanical Design, Information and Software Technology, and Requirement Engineering.

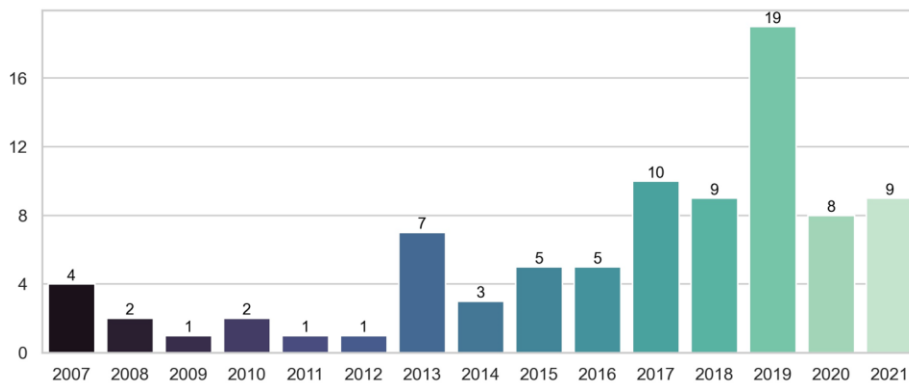


Figure 25. The number of included papers by year.

4.3.2. Resources and methods supporting ML-based requirement elicitation

ML-based requirements elicitation mainly includes requirement categorization, user preference analysis, and review helpfulness analysis. First, requirement categorization is constituted by classification, clustering, and topic modeling methods to support requirements elicitation. Second, user preference analysis involves mining user opinions, sentiments, complaints, and expectations from massive online reviews to provide designers with a more comprehensive and detailed understanding of customer needs. Finally, user review helpfulness analysis supports requirement elicitation, which awards a score to a given user review that quantifies its usefulness to product development. The general pipeline of ML-based requirement elicitation tasks contains data collection and preparation, data preprocessing, feature extraction, model training, and evaluation. According to this pipeline, we will summarize the review results in the following subsections.

4.3.3. The data source for building an ML-based requirement elicitation system

This review identifies two types of data sources: requirements specification data and user-generated content data. The requirement specification is a document that systematically stores system definition and system requirements (Bourque and Fairley 2014). Most of the requirement specifications applied by the reviewed works are written in English, and only two are bilingual (Ko et al. 2007; Lyutov, Uygun, and Hütt 2019), and three are written in a non-English language (Falessi, Cantone, and Canfora 2010; Gulle et al. 2020; Ott 2013).

A total of 15 included studies built the ML models with DePaul’s NFRs corpus provided by Cleland-Huang et al. (2006). Two works expanded DePaul’s NFR corpus by including other SRS for constructing a larger corpus (Baker et al. 2019; Sabir, Banissi, and Child 2021). Along with DePaul’s corpus, SecReq (Knauss et al. 2011) and PURE (Ferrari, Spagnolo, and Gnesi 2017) are two open requirement datasets adopted by several selected articles. The SecReq dataset was the second task introduced by the RE’17 data challenge, which aims to identify quality (security) requirements from given text. SecReq is a tagged corpus containing cleaned sentences and their associated labels, which contains the tags *sec* and *nonsec* for security-related and non-security-related requirements, respectively. Among the included articles, two works trained and tested their ML framework with SecReq corpus (Dekhtyar and Fong 2017; Kobilica, Ayub, and Hassine 2020). The PURE corpus has 79 requirement specifications, including about 35,000 sentences with an average length of fifteen words (Ferrari et al. 2017). Unlike DePaul’s NFRs and SecReq, the PURE dataset is not labeled and produced for a particular goal; instead, the authors made it open for various applications. Deshpande et al. (2019) studied requirement dependencies with the PURE corpus, and EzzatiKarami & Madhavji (2021) utilized both DePaul’s and PRUE corpora for constructing a more extensive training set.

Types of data	Data type	Sub-categories	Included works	Count	
Document data	Requirement documents	Pseudo requirement specification	(Cleland-Huang et al. 2007; Ormandjieva, Hussain, and Kosseim 2007)	2	
		Industrial requirement specification	(Abualhaija et al. 2019; Falesi et al. 2010; Lyutov et al. 2019; Mahmoud 2015; Ott 2013; Winkler and Vogelsang 2017)	6	
		Publicly available requirement specification	(Gulle et al. 2020; Halim and Siahaan 2019; Jeon, Lee, and Jeong 2021; Li et al. 2018; Myllynen et al. 2021; Rahimi, Eassa, and Elrefaei 2020; Riaz et al. 2014a)	7	
		Documents without a specified source	(Ko et al. 2007; Polpinij and Namee 2021)	2	
		Other documents	(Barbosa et al. 2015; Gulle et al. 2020; Massey et al. 2013; Rodeghero et al. 2017)	4	
	Existing requirement corpus	DePaul’s NFRs corpus		(Abad et al. 2017; Asif et al. 2019; Canedo and Mendes 2020; Casamayor, Godoy, and Campo 2009, 2010; Dalpiaz et al. 2019; Dekhtyar and Fong 2017; EzzatiKarami and Madhavji 2021; Gnanasekaran et al. 2021; Haque, Rahman, and Siddik 2019; Hussain, Kosseim, and Ormandjieva 2008; Khelifa, Haoues, and Sellami 2018; Navarro-Almanza, Juarez-Ramirez, and Licea 2018; Rahman et al. 2019; Rashwan, Ormandjieva, and Witte 2013; Sabir et al. 2021; Tóth and Vidács 2018)	20
			SecReq corpus	(Dekhtyar and Fong 2017; Kobilica et al. 2020; Li 2018)	3
			PURE corpus	(Deshpande et al. 2019; EzzatiKarami and Madhavji 2021)	2
			Others	(Liu, Lu, and Loh 2007; Parra et al. 2015)	2
			User-generated content	Produce review	e-commerce
App review	(Carreno and Winbladh 2013; Chen et al. 2014; Dhinakaran et al. 2018; Fu et al. 2013; Guzman and Maalej 2014; Jha and Mahmoud 2019; Joung and Kim 2021; Lu and Liang 2017; Maalej et al. 2016; Martens and Maalej 2019; Noei, Zhang, and Zou 2021; Panichella et al. 2015; C. Wang et al. 2018)	17			
Social media	Microblog	(Guzman, Ibrahim, and Glinz 2017; Kengphanphanit and Muenchaisri 2020; Prasetyo et al. 2012; Stanik, Haering, and Maalej 2019; Stone and Choi 2013)		5	
	Other	(Jones and Kim 2015; Lange 2008; Nyamawe et al. 2019)		3	

Table 12. The data source categorization.

User-generated Content (UGC) is another important source for building an ML-based requirement elicitation model. Research shows that requirements from system users are hidden in rich user-generated content, such as user feedback, social networks, online software markets review, and

product discussion forums (Lu and Liang 2017; Maalej et al. 2015, 2016; Perini 2018). UGC contains any forms of data generated by platform users, like numerical ratings, textual product reviews, and videos. In recent years, a growing number of studies have focused on the extraction of requirement-related information from textual UGC. The majority of the selected studies favored three platforms: mobile app market, microblog, and e-commerce websites. The included collection contains 37 articles that aid requirement elicitation via textual UGC. The detailed distribution is shown in Table 12.

4.3.4. Requirement data preprocessing

Among various preprocessing activities found in reviewed studies, tokenization and punctuation removal are the most frequently applied strategies in the collected works. Other approaches include language filtering (e.g., excluding review in non-English language), special character removal, and many other methods applied by the selected studies. On average, most papers described at least one preprocessing methodology, with only ten papers omitting such information. Table 13 shows the preprocessing technologies utilized in the included investigations.

Each row of the table includes a maximum of five articles because some techniques such as stemming (26 articles), case folding (31 articles), and removal of stop words (50 articles) are widely used in the included studies. We categorized these text preprocessing techniques into three groups: text chunking, text filtering, and text normalization. Text chunking is to segment input raw text down into smaller components, and there are different granularities of text chunks, such as paragraphs, sentences, and words. Most of the reviewed studies applied sentence segmentation in the preprocessing phase, splitting a given paragraph into several sentences.

Category	Preprocessing tasks	Included works
Text chunking	Sentence tokenization	(Chen et al. 2014; Li et al. 2020; Petcuşin, Stănescu, and Bădică 2020; Rashwan et al. 2013; Zhan et al. 2009)
	Word tokenization	(Barbosa et al. 2015; Jeon et al. 2021; Jindal, Malhotra, and Jain 2016; Liu et al. 2007; Wang 2016)
Text filtering	Stop words removal	(Maalej et al. 2016; Mahmoud 2015; Qi et al. 2016; Rahimi et al. 2020; Tóth and Vidács 2018), and other 45 works.
	Rare word filtering	(Chen et al. 2014, 2016; Fu et al. 2013; Joung and Kim 2021; W. Wang et al. 2018)
	Non-English word removing	(Al-Subaihin et al. 2016; Fu et al. 2013; Noei et al. 2021)
	Punctuation removal	(Joung and Kim 2021; Khelifa et al. 2018; Lyutov et al. 2019; Martens and Maalej 2019; Noei et al. 2021) and other 14 works.
	URL removal	(Jo and Oh 2011; Lyutov et al. 2019; Mahmoud 2015; Prasetyo et al. 2012; Stone and Choi 2013)
	Special symbol removal	(Ferrari et al. 2018; Fu et al. 2013; Gnanasekaran et al. 2021; Gulle et al. 2020; Stone and Choi 2013)
	Empty value handling	(Chen et al. 2014)
	Emoticon handling	(Kengphanphanit and Muenchaisri 2020; Khelifa et al. 2018; Zhou et al. 2020)
	Filtering out non-informative/ irrelevant	(Chen et al. 2014; Guzman et al. 2017; Timoshenko and Hauser 2019; Zhou et al. 2020)
	Filtering inconsistency	(Fu et al. 2013; Noei et al. 2021)
Text normalization	Stemming	(Barbosa et al. 2015; Ferrari et al. 2018; Li et al. 2018; Mahmoud 2015; Rahimi et al. 2020)
	Case folding	(Chen et al. 2014, 2016; Joung and Kim 2021; Timoshenko and Hauser 2019; Zhou et al. 2020)
	Lemmatization	(Guzman and Maalej 2014; Kurtanovic and Maalej 2017b; Li et al. 2018; Lyutov et al. 2019; Maalej et al. 2016)
	Slang translation	(Khelifa et al. 2018; Lu and Liang 2017; Zhou et al. 2020)
	Abbreviation replacement	(Khelifa et al. 2018; Lu and Liang 2017; Noei et al. 2021)
	Typo correction	(Kengphanphanit and Muenchaisri 2020; Noei et al. 2021)
	Acronym replacement	(Lyutov et al. 2019)

Table 13. Preprocessing techniques.

Text filtering is a group of preprocessing methods, which aim to eliminate as much redundant, erroneous, non-representative, inconsistent, and ineligible text data as possible. Text normalization is another text preprocessing category that aims to transform a text sequence into its standard form to reduce its randomness and feature size. Stemming and lemmatization are the most common text normalization methods. In a document, a word has various forms, and some of these forms can be

converted to one another by adding or removing the prefix or suffix (Manning, Raghavan, and Schütze 2008). Stemming is a crude heuristic procedure that removes the tails from words to get word stems, which are the fundamental word units, e.g. for word *requirements*, the word stem is “*require*”. In comparison, lemmatization yields a basic dictionary form of a word. For example, the lemmatization of “*requirements*” will yield “*requirement*”. Case folding is another popular text normalization approach that changes all letters in a word into lower cases (Manning et al. 2008). Slang translations, abbreviation translations, typo corrections, and acronym substitutes can also be considered text normalization procedures since they convert text into a more generic form.

Text pre-processing is a collection of techniques for transforming unstructured text input into a machine-readable format. This section introduces and summarizes the preprocessing techniques applied by the reviewed literature and categorizes the related techniques like chunking, filtering, and normalization. According to the selected studies, most preprocessing techniques positively impact the downstream tasks. Despite that, there is no conclusive evidence that the more preprocessing methods applied, the better the downstream results will be. So, such techniques should be chosen based on the dataset, downstream learning algorithms, and the results they will provide.

4.3.5. Approaches of feature extraction from requirement text

Text features utilized in the included papers are classified as follows: lexical features, rule-based features, embedding features, and pre-trained features. The lexical feature is built upon the Bag of Word (BOW) model, which uses a single (unigram) or several words (n-gram) as a fundamental feature to represent a document. The raw count (or frequency) is a common approach to quantifying the lexical feature. Term Frequency-Inverse Document Frequency (TF-IDF) is an advanced feature weighting method among the included works (Manning and Schütze 1999). TF-

IDF neutralizes the term frequency with the number of documents containing that word to draw a weighted feature value, reflecting how important a feature (term or n-gram) is to a document in a corpus.

The rule-based features are the second category in this section, usually tailored for specific tasks with sufficient domain knowledge and text mining experience. In the included articles, seventeen papers adopted this type of hand-craft feature. Linguistic features, sentiment-based features, meta-data features, domain-specific features, quality features, and temporal features are commonly used by the rule-based features. Linguistic features are usually based on part-of-speech (POS) information in each sentence or other syntactic structural information. The related features such as POS n-gram (Kurtanovic and Maalej 2017b), number of Noun/Verb/Adj/Adv/Modal (Hussain et al., 2008; Kurtanovic & Maalej, 2017a; Liu et al., 2013), frequency of POS of the keywords (Halim and Siahaan 2019), and number of syntax sub-tree (Dalpiaz et al. 2019; Kurtanovic and Maalej 2017b, 2017a). Textual descriptive statistics from the reviewed articles include the number of characters (Abualhaija et al. 2019), number of words (Kurtanovic and Maalej 2017b), number of sentences (Qi et al., 2016), number of paragraphs (Parra et al. 2015), and number of words per sentence (Ormandjieva et al. 2007). The Meta-data features include descriptive product data, review text, and user data. The product and review meta-data is descriptive information about the product, such as the average star ratings (Maalej et al. 2016) and the total number of reviews (Martens and Maalej 2019). The user meta-data includes the total number of reviews/ratings of the user performed (Martens and Maalej 2019), the account usage information, and the grade of the user (Qi et al., 2016). Temporal features include verb tense (Stanik et al. 2019), number of elapsed days (Liu et al., 2013), and temporal tags (Abad et al. 2017). Document quality features include the number of subjective/objective sentences in a review (Liu et al., 2013), the number of

ambiguous expressions in a requirement (Ormandjieva et al. 2007; Parra et al. 2015), and the number of the sentence referring product feature appeared in a user review (Liu et al., 2013; Qi et al., 2016). Domain features are based on domain knowledge and are usually based on domain terms and terminology, such as the number of design terms (Parra et al. 2015) and the number of keywords from the input text (Hussain et al. 2008; Stanik et al. 2019). Other features are not widely applied through the included works, such as smoothed probability measure (SPM), Unsmoothed Probability Measure (UPM) (Hussain et al. 2008), and named entities (Abad et al. 2017).

The third type of text feature is the embedding feature, which effectively represents texts by iteratively learning through neural networks rather than calculating weighted frequencies (Mikolov et al. 2013). In recent years, word embedding has gained popularity in a range of natural language processing applications. The selected articles used a range of embedding techniques, including Word2vec, FastText, and Glove models, to represent words. Utilizing embedding features entails three strategies: training the embedding from scratch, using a pre-trained embedding, and fine-tuning the previously trained embedding. The weights are initially set to random and continuously updated by backpropagation by training the embeddings from scratch. Pre-trained embeddings are usually trained earlier, and weights are saved in a particular file that may be used for other tasks without training it from scratch. Fine-tuning a model uses a task-specific dataset to retrain a pre-trained model, adjusting it to fit the target task better. Sixteen works mentioned the embedding features, and eight were applied to the pre-trained word vectors. Word2vec is the most popular pre-trained model among the included studies, where six studies applied Word2vec models (Skip-

Gram or CBOW model), and four of them (Dekhtyar and Fong 2017; Gnanasekaran et al. 2021; Gulle et al. 2020; Rahman et al. 2019) applied Google pre-trained Word2Vec model⁴.

Feature Group	Feature name	Included studies
Lexical feature	N-gram model with raw count weighting	(Carreno and Winbladh 2013; Dalpiaz et al. 2019; El Dehaibi et al. 2019; Deshpande et al. 2019; Dhinakaran et al. 2018; EzzatiKarami and Madhavji 2021; Fu et al. 2013; Gulle et al. 2020; Guzman and Maalej 2014; Han and Moghaddam 2021; Haque et al. 2019; Jo and Oh 2011; Jones and Kim 2015; Joung and Kim 2021; Kengphanphanit and Muenchaisri 2020; Khelifa et al. 2018; Kurtanovic and Maalej 2017b; Liang et al. 2017; Liu et al. 2007; Lyutov et al. 2019; Maalej et al. 2016; Noei et al. 2021; Polpinij and Namee 2021; Prasetyo et al. 2012; Rashwan et al. 2013; Singh and Tucker 2017; Stone and Choi 2013; W. Wang et al. 2018; Zhou et al. 2020)
	N-gram model with tf-idf weighting	(Al-Subaihini et al. 2016; Asif et al. 2019; Bakiu and Guzman 2017; Barbosa et al. 2015; Canedo and Mendes 2020; Casamayor et al. 2009, 2010; Chen et al. 2016; Cleland-Huang et al. 2007; Deocadez, Harrison, and Rodriguez 2017; EzzatiKarami and Madhavji 2021; Falessi et al. 2010; Guzman et al. 2017; Haque et al. 2019; Jha and Mahmoud 2019; Jindal et al. 2016; Jones and Kim 2015; Ko et al. 2007; Li et al. 2018, 2020; Lu and Liang 2017; Massey et al. 2013; Nyamawe et al. 2019; Petcuşin et al. 2020; Rahimi et al. 2020; Stanik et al. 2019; Tóth and Vidács 2018)
Rule-based features	Linguistic feature	(Abad et al. 2017; Abualhaija et al. 2019; Dalpiaz et al. 2019; Halim and Siahaan 2019; Hussain et al. 2008; Kurtanovic and Maalej 2017b, 2017a; Li 2018; Liu et al. 2013; Ormandjieva et al. 2007; Parra et al. 2015; Stanik et al. 2019)
	Frequency-based statistic	(Dalpiaz et al. 2019; Kurtanovic and Maalej 2017b, 2017a; Liu et al. 2013; Ormandjieva et al. 2007; Parra et al. 2015; Qi et al. 2016; Rodeghero et al. 2017; Stanik et al. 2019)
	Sentiment-based features	(Dalpiaz et al. 2019; Kurtanovic and Maalej 2017b; Liu et al. 2013; Stanik et al. 2019)
	Meta-data features	(Dalpiaz et al. 2019; Kurtanovic and Maalej 2017b; Maalej et al. 2016; Martens and Maalej 2019; Qi et al. 2016)
	Domain Features	(Abualhaija et al. 2019; Hussain et al. 2008; Li 2018; Parra et al. 2015; Stanik et al. 2019)
	Qualitative features	(Liu et al. 2013; Ormandjieva et al. 2007; Parra et al. 2015; Qi et al. 2016)
	Temporal features	(Abad et al. 2017; Liu et al. 2013; Stanik et al. 2019)
	Other features	(Abad et al. 2017; Hussain et al. 2008)
Embedding feature	Pretrained embeddings	(Dekhtyar and Fong 2017; Gnanasekaran et al. 2021; Gulle et al. 2020; Jeon et al. 2021; Petcuşin et al. 2020; Rahman et al. 2019; Stanik et al. 2019; Zhou et al. 2020)
	Embeddings from scratch	(Baker et al. 2019; Chen et al. 2016; Jeon et al. 2021; Navarro-Almanza et al. 2018; Sabir et al. 2021; Winkler and Vogelsang 2017)
	Fine-tuned pre-trained embeddings	(Han and Moghaddam 2021; Sabir et al. 2021)

Table 14. classification of textual features in the selected literature.

⁴ <https://code.google.com/archive/p/word2vec/>

Additionally, two works applied Glove vectors (Jeon et al. 2021; Petcuşin et al. 2020), and two reviewed papers mentioned FastText (Stanik et al., 2019; Zhou et al., 2020), and one study utilized three different pre-trained transformer models for a comparison experiment (Myllynen et al. 2021). Five works completely trained the word embedding model from beginning with UGC data (Chen et al., 2016; Timoshenko & Hauser, 2019), NFR corpus (Sabir et al. 2021), and requirement documents (Winkler and Vogelsang 2017). Moreover, a newer model such as BERT is fine-tuned by another selected study (Han and Moghaddam 2021).

The detailed categorization is shown in Table 14. In the same way as text preprocessing techniques, the choice of features will differ from project to project, depending on various factors, such as the corpus, downstream algorithms, and research concerns.

4.3.6. ML methods for requirement elicitation

All of the selected papers employed supervised or unsupervised learning methods to tackle various kinds of requirements elicitation problems. Regression models were applied in three reviewed papers (Chen et al., 2016; Liu et al., 2013; Qi et al., 2016) to predict a helpfulness score based on different scenarios. For example, Liu et al. (2013) used different regression methods to evaluate the relativeness of Amazon reviews from the designer's perspective, and the score ranged from -2 (very helpless) to 2 (very helpful).

Most of the articles used classification algorithms for classifying requirement text. Out of the various classification algorithms, the Naïve Bayes algorithm was the most popular model in the included studies. Thirty-three articles discussed and applied the Naïve Bayesian technique to support the requirement elicitation activities. Among them, a total of four works adopted EMNB algorithms to improve the performance of the NB classifier with massive unlabeled data

(Casamayor et al. 2009, 2010; Chen et al. 2014; Yang et al. 2019). The EMNB algorithm was first introduced by Nigam et al. (2000) as a semi-supervised learning algorithm combining Expectation-Maximization (EM) and Naïve Bayes (NB). Support Vector Machine (SVM) is another popular conventional ML model included in 30 articles.

Neural networks (NN), particularly Deep Learning, have been broadly used in many fields. Among the 20 reviewed articles that utilized neural networks, 12 studies applied Convolutional Neural Networks (CNN), nine works applied Feedforward Neural Networks (FNN), and five papers introduced Recurrent Neural Networks (RNN). Gated Recurrent Unit (GRU) (Cho et al. 2014) and Long Short-Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) are two popular RNN variations. GRU was introduced in two of the included studies (Rahman et al. 2019; Sabir et al. 2021), and Long Short-term Memory (LSTM) was adopted in four studies (Gnanasekaran et al. 2021; Kobilica et al. 2020; Rahman et al. 2019; Sabir et al. 2021). The reviewed articles also apply a variety of supervised ML methods, including decision tree algorithms (e.g., C4.5, J48, CART, LMT), logistic regression, K-Nearest Neighbors, and random forests.

Relatively few selected studies utilized unsupervised learning algorithms compared to supervised learning. In contrast to supervised learning, the input to unsupervised learning is not labeled. Text categorization is a family of algorithms that aim to divide input data into groups so that instances in the same groups are closer or more relevant than instances from other groups (James et al. 2013). The unsupervised algorithms applied by the included studies can be further subdivided into text clustering and topic modeling. Five studies used clustering algorithms, and 14 studies employed topic modeling to uncover the hidden information in user-generated content.

Types	Learning algorithms	Included papers	Total count	
Supervised learning	<i>Naïve Bayes</i>	(Asif et al. 2019; Canedo and Mendes 2020; Casamayor et al. 2009, 2010; Chen et al. 2014; Dhinakaran et al. 2018; EzzatiKarami and Madhavji 2021; Guzman et al. 2017; Halim and Siahaan 2019; Haque et al. 2019; Jha and Mahmoud 2019; Jones and Kim 2015; Kengphanphanit and Muenchaisri 2020; Ko et al. 2007; Kurtanovic and Maalej 2017b; Lange 2008; Li 2018; Li et al. 2018; Lu and Liang 2017; Lyutov et al. 2019; Maalej et al. 2016; Martens and Maalej 2019; Nyamawe et al. 2019; Ott 2013; Panichella et al. 2015; Polpinij and Namee 2021; Rahimi et al. 2020; Riaz et al. 2014a; Singh and Tucker 2017; Taj et al. 2019; Tóth and Vidács 2018; C. Wang et al. 2018; Yang et al. 2019)	33	
	<i>Support Machine</i>	<i>Vector</i>	(Abualhaja et al. 2019; Bakiu and Guzman 2017; Canedo and Mendes 2020; Dalpiaz et al. 2019; Deshpande et al. 2019; Dhinakaran et al. 2018; EzzatiKarami and Madhavji 2021; Haque et al. 2019; Jha and Mahmoud 2019; Jones and Kim 2015; Khelifa et al. 2018; Kobilica et al. 2020; Kurtanovic and Maalej 2017b, 2017a; Li 2018; Li et al. 2018; Liu et al. 2007; Lyutov et al. 2019; Martens and Maalej 2019; Nyamawe et al. 2019; Ott 2013; Panichella et al. 2015; Prasetyo et al. 2012; Rahimi et al. 2020; Rashwan et al. 2013; Riaz et al. 2014a; Rodeghero et al. 2017; Singh and Tucker 2017; Stone and Choi 2013; Tóth and Vidács 2018)	30
	<i>Decision Tree-based</i>		(Abad et al. 2017; Abualhaja et al. 2019; Asif et al. 2019; EzzatiKarami and Madhavji 2021; Halim and Siahaan 2019; Haque et al. 2019; Hussain et al. 2008; Jindal et al. 2016; Kobilica et al. 2020; Li 2018; Lu and Liang 2017; Lyutov et al. 2019; Maalej et al. 2016; Martens and Maalej 2019; Ormandjieva et al. 2007; Panichella et al. 2015; Parra et al. 2015; Rahimi et al. 2020; Singh and Tucker 2017; Taj et al. 2019; Tóth and Vidács 2018; C. Wang et al. 2018)	22
	<i>Logistic Regression</i>		(Abualhaja et al. 2019; Asif et al. 2019; Canedo and Mendes 2020; El Dehaibi et al. 2019; Dhinakaran et al. 2018; EzzatiKarami and Madhavji 2021; Falessi et al. 2010; Kurtanovic and Maalej 2017b; Li 2018; Nyamawe et al. 2019; Panichella et al. 2015; Rahimi et al. 2020; Rodeghero et al. 2017; Tóth and Vidács 2018)	14
	<i>Convolutional Neural Network</i>	<i>Neural</i>	(Baker et al. 2019; Dekhtyar and Fong 2017; Han and Moghaddam 2021; Jeon et al. 2021; Kobilica et al. 2020; Navarro-Almanza et al. 2018; Rahman et al. 2019; Sabir et al. 2021; Stanik et al. 2019; Tamai and Anzai 2018; Timoshenko and Hauser 2019; Winkler and Vogelsang 2017)	12
	<i>Feedforward Neural Network (Multi-layer Perceptron)</i>	<i>Neural</i>	(Jeon et al. 2021; Liu et al. 2013; Lyutov et al. 2019; Martens and Maalej 2019; Myllynen et al. 2021; Petcuşin et al. 2020; Sabir et al. 2021; Suryadi and Kim 2019; Tóth and Vidács 2018)	9
	<i>K Nearest Neighbor</i>		(Canedo and Mendes 2020; Haque et al. 2019; Kobilica et al. 2020; Nyamawe et al. 2019; Riaz et al. 2014a; Singh and Tucker 2017; Tóth and Vidács 2018; C. Wang et al. 2018)	8
	<i>Random Forest</i>		(Abualhaja et al. 2019; EzzatiKarami and Madhavji 2021; Martens and Maalej 2019; Singh and Tucker 2017; Tóth and Vidács 2018)	5
	<i>Recurrent Neural Network</i>	<i>Neural</i>	(Gnanasekaran et al. 2021; Jeon et al. 2021; Kobilica et al. 2020; Rahman et al. 2019; Sabir et al. 2021)	5
	<i>Regression</i>		(Chen et al. 2016; Liu et al. 2013; Qi et al. 2016)	3
Unsupervised learning	<i>Topic modeling</i>	(Carreno and Winbladh 2013; Chen et al. 2014; Fu et al. 2013; Gulle et al. 2020; Guzman et al. 2017; Guzman and Maalej 2014; Jo and Oh 2011; Joung and Kim 2021; Li et al. 2020; Massey et al. 2013; Noei et al. 2021; Tang et al. 2019; W. Wang et al. 2018; Zhou et al. 2020)	13	
	<i>Clustering</i>	(Al-Subaihini et al. 2016; Barbosa et al. 2015; Mahmoud 2015; Suryadi and Kim 2019; Zhan et al. 2009)	5	

Table 15. Learning algorithms from reviewed studies.

For clustering algorithms, Hierarchical Agglomerative Clustering (Al-Subaihini et al. 2016; Mahmoud 2015), K-medoids (Barbosa et al. 2015), X-means (Suryadi and Kim 2019), and Frequent word sequence-based Custom clustering method (Zhan et al. 2009) is applied by the

included studies. Topic modeling is a probabilistic approach for discovering hidden or abstract topics in a collection of documents and mapping a document into topics with a probability distribution (Blei, Ng, and Jordan 2003). Ten studies included in this collection used the Latent Dirichlet Allocation (LDA) algorithm (B. Fu et al., 2013; Gulle et al., 2020; Guzman et al., 2017; Guzman & Maalej, 2014; Joung & Kim, 2021; J. Li et al., 2020; Massey et al., 2013; Noei et al., 2021; W. Wang et al., 2018; Zhou et al., 2020). Four studies proposed a custom topic model based on LDA, such as the Sentence-LDA model, the Aspect and Sentiment Unification Model (ASUM) (Jo and Oh 2011), and Tag Sentiment Aspect (TSA) model (Tang et al. 2019).

Two included works studied the UGC with the ASUM model (Carreno and Winbladh 2013; Chen et al. 2014). The Bi-term Topic Model (BTM) is another LDA-like model designed explicitly for short text topic modeling. A study compared BTM and LDA and found that BTM produced more informative topics than LDA on requirement extraction tasks with a short text corpus (Guzman et al. 2017). The detailed categorization is shown in Table 15.

4.3.7. Model evaluation methods

The quality of models can be reflected in the evaluation metrics, which are a set of formulas and units of measurement that reflect how well the learning algorithm could perform (Hossin & Sulaiman, 2015). The evaluation metrics are employed differently in supervised and unsupervised approaches due to the differences between their learning methods. Manually annotated data corpus for supervised ML algorithms could be utilized for training and validation purposes. Hence, comparing machine predictions with actual values is a simple, straightforward way to evaluate a supervised learning algorithm. For regression models, metrics *Mean Absolute Error* (MAE) and *Root Mean Square Error* (RMSE) are common error functions to reflect the accuracy of regression methods (Chai and Draxler 2014). Both MAE and RMSE are negatively oriented, which means

the better the model, the lower the errors. Common metrics for classification tasks are precision, recall, accuracy, f-score, and Area-Under-Curve (AUC) (Chai and Draxler 2014). The precision (n=58), recall (n=56), and F1 score (n=49) are the most applied metrics to evaluate a supervised classifier by the included works. Due to the differences in data and research questions, it is difficult to compare the included works.

Observing the performance of a clustering method is more complex than determining the accuracy, recall, or number of mistakes in a supervised classification process. There are two strategies for evaluating an unsupervised learning method: internal and external evaluation (Palacio-Niño and Berzal 2019). Internal evaluation measures the quality of an unsupervised algorithm based on its input data. In internal evaluation methods, the cohesion and separation of the clusters will be determined by the efficacy of the clustering results, as cohesion shows how closely the elements cluster within the same group, and separation shows how far the clusters are separated from one another (Palacio-Niño and Berzal 2019). Zhan et al. (2009) applied cosine similarity-based intra-cluster and inter-cluster similarity to measure the separation and cohesiveness. In addition, another measure of clustering cohesion, the Silhouette score, was used by the four included studies to assess clustering outcomes (Abad et al. 2017; Al-Subaihin et al. 2016; Barbosa et al. 2015; Mahmoud 2015). Furthermore, five papers introduced perplexity, a prevalent method to evaluate a language model (N. Chen et al., 2014; Joung & Kim, 2021; Massey et al., 2013; W. Wang et al., 2018; Zhou et al., 2020).

External evaluation either compares the clustering results to a manually built truth set or evaluates the results manually (Palacio-Niño and Berzal 2019). The procedure is similar to the one used to evaluate supervised learning algorithms. The clustering results are evaluated with precision, recall, and F1-score (Abad et al. 2017; Carreno and Winbladh 2013; Chen et al. 2014; Guzman and Maalej

2014). In addition, several reviewed articles examine the performance with manual inspection (Al-Subaihini et al. 2016; Gulle et al. 2020; Guzman et al. 2017; Massey et al. 2013).

4.3.8. Tools

Most of the included works are built upon existing open-access tools and libraries. *Scikit-learn*⁵ and *Waikato Environment for Knowledge Analysis*⁶ (*Weka*) are the two most popular ML tools mentioned in the included articles. The primary focus of *Scikit-learn* is to bring ML to non-specialists with easily encapsulated libraries (Pedregosa et al. 2011). Seventeen works applied *Scikit-learn* to build different kinds of algorithms such as Naïve Bayes, Support Vector Machine, and Random Forest. Another popular tool in the reviewed articles is *Weka*, with 19 articles reporting that they applied *Weka* for building their solutions. *Weka* is a stand-alone ML tool that integrates a variety of cutting-edge ML algorithms (Hall et al. 2009). Both *Scikit-learn* and *Weka* provide ready-to-use learning algorithms and have numerous tricks for preprocessing and feature extraction. For example, C. Wang et al. (2018) used the *StringToWordVector* package from *Weka* to produce TF-IDF word vectors. In contrast, Dekhtyar & Fong (2017) applied *TfidfVectorizer* from the *Scikit-learn* library for the same purpose.

For Natural Language Processing (NLP), the most popular tool is the *Natural Language Toolkit* (NLTK)⁷, a Python library designed specifically for human language processing (Loper and Bird 2002). The NLTK library is applied in selected papers for numerous preprocessing and feature extraction tasks, such as tokenization (Rahman et al. 2019), sentiment analysis (Noei et al. 2021), Part-of-speech tagging (Halim and Siahaan 2019), lemmatization (Guzman and Maalej 2014), and

⁵ <https://scikit-learn.org/stable/index.html>

⁶ <https://www.cs.waikato.ac.nz/ml/weka/>

⁷ <https://www.nltk.org/>

stemming (Jha and Mahmoud 2019). For POS and dependency parsing tasks, tools from the Stanford NLP group are mentioned, such as *Stanford parser*, *CoreNLP*, and *POS tagger*. *TensorFlow* and its high-level wrapper *Keras* are the most often used neural network libraries in the listed studies. Furthermore, SentiStrength is the most widely applied sentiment analysis tool to measure sentiment information quantitatively.

4.4. Research findings and discussion

This review extracted the main activities in the ML-based requirements elicitation process from 86 included articles. First, the included articles are analyzed and categorized according to the requirement elicitation subtasks. Subsequently, the review demonstrates the current state of the ML-based requirement elicitation studies through seven primary aspects: tasks, data sources, data preprocessing techniques, text representation techniques, learning algorithms, evaluation methods, and tools. Our findings from the literature review will be discussed in this section. This section is organized according to the order of our research questions (section 4.2.2). The articles included in this review are categorized according to the different perspectives on the research questions. The summarization of our categorization is illustrated in Figure 26.

4.4.1. Main tasks of ML-based requirement elicitation

RQ1. In requirement elicitation, which specific activities are supported by ML algorithms?

After analyzing the selected 86 papers in-depth, fifteen different ML-based requirement elicitation tasks are identified (as Figure 27). The identified tasks can be categorized into four main categories, which are *Preparation*, *Collection*, *Inspection*, and *Negotiation*.

Preparation refers to a set of activities that engineers must undertake before the elicitation of requirements to ensure that the process is supported by sufficient knowledge. A total of five articles

are proposed to extract knowledge about the design from textual documents. For example, Liu et al. (2007) proposed an SVM-based design knowledge acquisition framework that can collect research articles according to organizational design knowledge taxonomy.

In addition, extracting user preferences, requests, and complaints from massive UGC is also considered a *Preparation* task. The ML-based text mining algorithms would be used to extract useful information from UGC, providing engineers with insights and knowledge about the target product. For example, Maalej et al. (2016) proposed a supervised method to automatically classify user app reviews into four predefined categories: user experience, bug report, feature quest, and ratings.

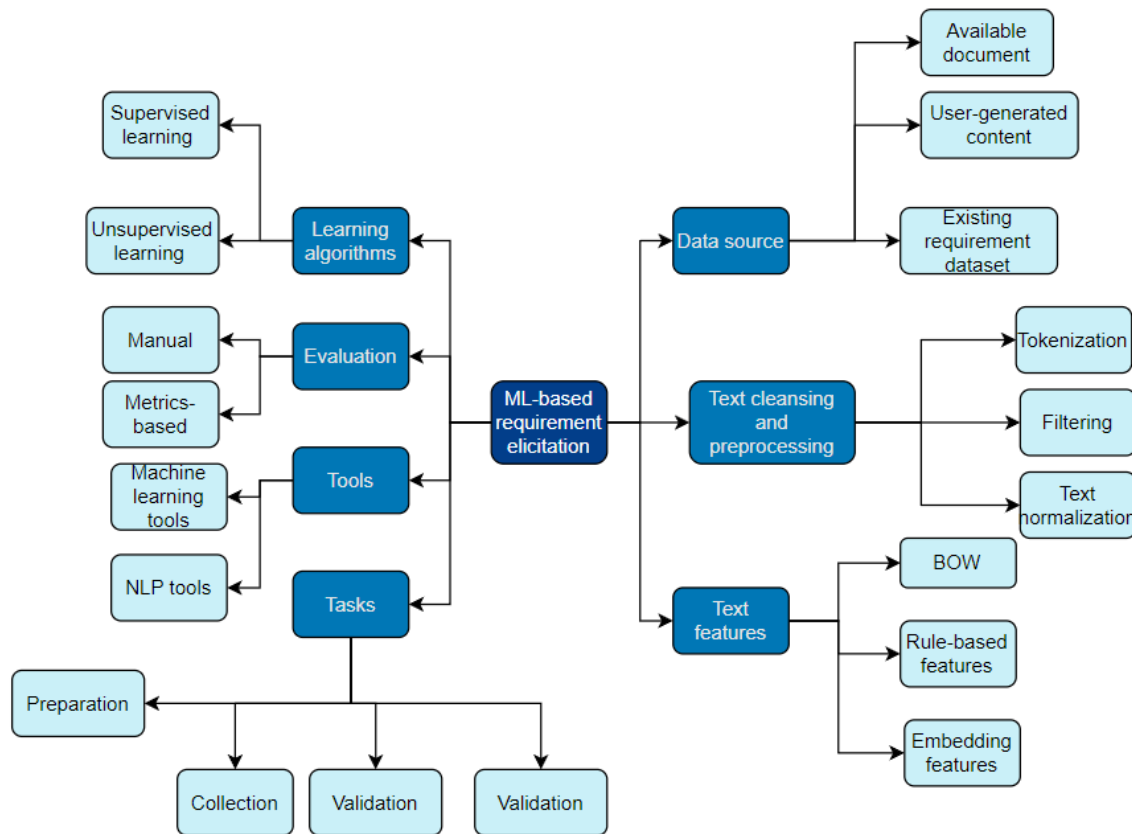


Figure 26. An illustration of the categorization schema of the collected studies

Liu et al. (2013) present a regression model which enables engineers to estimate the usefulness of customer reviews. UGC helpfulness analysis helps determine whether users' feedback is constructive. However, evaluating usefulness is a subjective activity that often entails a viewpoint. In a data-driven approach, the annotators represent the viewpoint. This review identifies two perspectives including the designer-perspective (Liu et al., 2013; Qi et al., 2016) and the consumer perspective (Chen et al., 2016).

Stakeholder preference (or tendency, rationale) is another activity categorized as *Preparation*. Since UGC is the cumulative contribution of users over some time, it incorporates their preferences and emotions about the product, product functions, and product features. For example, combining the LDA and sentiment analysis techniques can help engineers to explain which features of the product are loved by users (Guzman & Maalej, 2014; Zhou et al., 2020), and which are the most dissatisfied product characteristics (Fu et al. 2013).

The second group of tasks is *Collection*, which includes tasks related to directly extracting requirements or identifying specific types of requirements from a given collection of documents. In selected articles, all ML-based solutions in this category are supervised methods. The first type of collection task is requirement identification, which refers to the activity to determine whether a given sentence or paragraph is a user requirement. For example, Kengphanphanit & Muenchaisri (2020) proposed a requirement identification framework named ARESM, which can distinguish whether a given text is a requirement or non-requirements.

Requirement classification is another task in the *Collection* category. The objective of this task is to categorize the given requirements based on a certain concern. For example, Hussain et al. (2008) proposed a decision tree algorithm that can classify natural language requirements into functional

requirements (FRs) and non-functional requirements (NFRs). The NFRs/FRs classification task takes NFRs or FRs as input and classifies them further into fine-grained sub-categories. Cleland-Huang et al. (2007) proposed a TF-IDF-based classification algorithm that is capable of classifying textual requirements into predefined NFR subcategories. For this purpose, Cleland-Huang et al. (2007) established a manually labeled dataset for NFR classification, which we will discuss in-depth in the next section.

Preparation	Collection	Inspection	Negotiation
Product knowledge acquisition (n=6, 6.98%) e.g. (Liu et al., 2007)	Requirement identification (n=6, 6.98%) e.g. (Winkler & Vogelsang, 2017)	Equivalent detection (Falessi et al., 2010)	Requirement change support (Khelifa et al., 2018)
UGC helpfulness prediction (n=3, 3.49%) e.g. (Qi et al., 2016)	Requirement classification (n=17, 19.77%) e.g. (Myllynen et al., 2021)	Fake review extraction (Martens & Maalej, 2019)	
UGC Categorization (n=11, 12.79%) e.g. (Maalej et al., 2016)	NFR/FR classification (n=15, 17.44%) e.g. (Cleland-Huang et al. 2007)	Requirement dependency extraction (Deshpande et al., 2019)	Requirement distribution (Lyutov et al., 2019)
User preference extraction (n=16, 18.60%) e.g. (Fu et al., 2013)	Security requirement identification (n=4, 16.28%) e.g. (Riaz et al., 2014)	Requirement quality assessment (Parra et al., 2015) & (Ormandjieva et al., 2007)	Refactoring decision support (Nyamawe et al., 2019)

Figure 27. ML-based requirement elicitation tasks.

The last type of task identified in the *Collection* is security requirement identification. Riaz et al. (2014) trained a K-NN classifier that can automatically detect six predefined security requirement types from natural text documents. Two articles introduce binary classifiers for identifying security requirements from written requirements (Kobilica et al., 2020; Li, 2018). Jindal et al. (2016)

trained a decision tree to further categorize security requirements into four specific categories, which are authentication-authorization, access control, cryptograph-encryption, and data integrity.

The *Inspection* and *Negotiation* could happen at any stage during a requirement engineering process. *Inspection* refers to the ML-based methods applied to inspect and assure the quality and validity of the requirements. The *Inspection* category includes equivalent requirement detection (Falessi et al. 2010), requirement quality support (Ormandjieva et al. 2007; Parra et al. 2015), requirement dependency analysis (Deshpande et al. 2019), and fake review detection (Martens and Maalej 2019). The *Negotiation* category includes activities to support resolving requirement-related conflicts, and there are three types of tasks were identified under this category. An SVM classifier was used by Khelifa et al. (2018) to automatically classify users' change requests into functional change and technical change, thereby assisting project managers to negotiate requirements and make appropriate decisions. In a recent paper, Lyutov et al. (2019) presented a supervised learning-enabled workflow that facilitates the automatic transmission of customer requirements to the corresponding department to facilitate the process of requirement negotiation. Moreover, a machine-learning-based software refactoring recommendation method is proposed to assist decision-makers in deciding which major update should be applied according to customers' requests (Nyamawe et al. 2019).

4.4.2. Data sources

RQ2. Which types of data sources are being used in current works?

Based on an in-depth analysis of included studies, we found that current studies heavily rely on three types of data sources: *Textual Documents*, *User-Generated Content (UGC)*, and *Existing Requirement Datasets*. The detailed categorization is listed in Figure 28. The category *Textual*

Documents includes product requirement specification (RS) from actual projects (n=9), RS from open access online resources (n=8), user stories (Barbosa et al. 2015; Rodeghero et al. 2017), policy documents (Massey et al. 2013), and research publications (Liu et al., 2007).

DePaul's NFRs corpus is the most extensively used (n=21) *Existing Requirement Datasets*, which was originally introduced by Cleland-Huang et al. (2006). The dataset is manually labeled by graduate students from DePaul University into 10 NFR sub-categories and one functional requirements category including *Availability, Look and Feel, Legal, Maintainability, Operational, Performance, Scalability, Security, Usability*, and *FRs*. In total, the dataset contains 358 FRs and 326 NFRs from 15 different RS. Follow-up studies apply the DePaul NFR dataset to build binary classifiers to distinguish between FR and NFR (Canedo and Mendes 2020; Hussain et al. 2008), or multi-class classifiers to assign requirements to finer categories (Abad et al. 2017; Rahman et al. 2019).

SecReq is another publicly available requirement dataset, which was created to assist in the early stages of security requirement elicitation (Houmb et al. 2010). The dataset contains 3 projects, which are Electronic Purse, Customer Premises Network, and Global Platform Specification. Three projects contain 511 requirements that are tagged as security-related requirements (*sec*) and non-security-related requirements (*non-sec*). Three works trained and tested their data-driven requirement elicitation methods with SecReq corpus (Dekhtyar & Fong, 2017; Kobilica et al., 2020; Li, 2018).

The PURE dataset has 79 requirement specifications including about 35,000 sentences with an average length of fifteen words (Ferrari et al. 2017). Unlike the previously described two datasets, the PURE is not labeled; rather, the authors made it open for a variety of applications. Deshpande

et al. (2019) studied requirement dependencies with the PURE corpus, and EzzatiKarami & Madhavji (2021) merged both DePaul NFR and PRUE datasets for constructing a bigger training set for their study.

User-generated data (UGC) is another important source for data-driven requirements elicitation. Research shows that the needs of system users are hidden in rich user-generated content, such as user feedback, social networks, online software markets review, and product discussion forums (Lu and Liang 2017; Maalej et al. 2015, 2016; Perini 2018). UGC contains any forms of data generated by users, like numerical ratings, textual product reviews, and videos. In total, half of the included studies (n=43) applied UGC to build their ML-based solutions. The UGC source, includes mobile application platform user reviews (Apple App Store and Google Play Store), e-commerce user reviews (Amazon and other online retailers), social media (Twitter and Facebook), and crowdsourcing platforms.

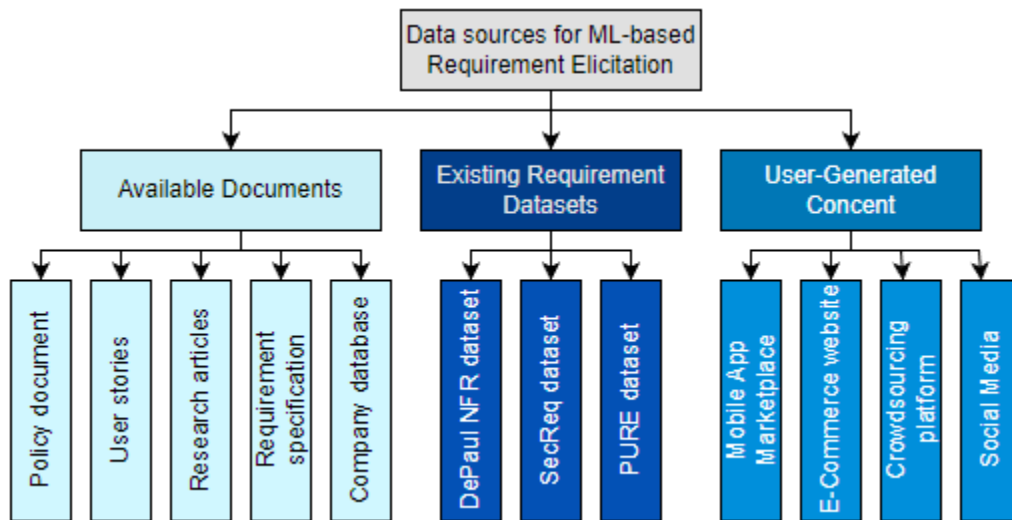


Figure 28. The data source for building ML-based requirement elicitation solutions.

4.4.3. Technologies and algorithms

This subsection answers the following three research questions.

RQ3. What technologies are used by selected studies?

RQ4. What criteria are used to assess data-driven solutions?

RQ5. What are the available tools to support ML-based requirements elicitation methodology?

Our study identified the technical approaches and algorithms used by the included studies and divided them into three categories: *Textual Data Cleansing and Preprocessing*, *Textual Features Extraction*, and *Machine Learning (ML)* (as Figure 29 shows). The ML models are evaluated by two strategies, which are *Manual evaluation* and *Metrics-based evaluation*. In addition, we categorized several open-source tools identified from the reviewed articles into two categories: *ML tools*, and *NLP tools*.

Textual Data Cleansing and Preprocessing

Twenty different techniques were identified from the included papers specifically for cleaning and preparing data, which we categorized under the *Textual Data Cleansing and Preprocessing* category. In addition, due to the functional features of these techniques, we further group these techniques into three parts: *tokenization*, *text chunking*, and *text normalization*.

Tokenization is a procedure to break a given sequence of text down into smaller parts, such as breaking a document into sentences (sentence tokenization) or breaking a sentence into individual words (word tokenization). *Text filtering* is a group of preprocessing methods, which aim to eliminate redundant, erroneous, non-representative, inconsistent, and ineligible data from a text document. In the reviewed articles techniques include stopword removal, rare word filtering, non-English word removing, URL removing, special character handling, empty value handling,

punctuation removal, emoticon handling, non-informative/irrelevant word removal, and inconsistent information removal are considered under this classification.

Text normalization aims to transform a text sequence into a standard form to reduce its randomness. Stemming and lemmatization are the most common text normalization methods. In a document, a word has various forms, and some of these forms can be converted to one another by adding or removing the prefix or suffix (Manning et al. 2008). Stemming is a crude heuristic procedure that removes the tails from words to get word stems, which are the fundamental word units, such as for word requirements, the word stem is required (Manning et al. 2008). In comparison, lemmatization yields a basic dictionary form of a word. For example, the lemmatization of requirements will yield requirements. Case folding is another popular text normalization approach that changes all letters in a word into lower cases (Manning et al. 2008). In addition, slang translations, abbreviation translations, typo corrections, and acronym substitutes are considered text normalization procedures since they convert text into a more generic form.

Textual Features Extraction

Textual Features Extraction includes a set of techniques to convert natural text into numbers. We found three major textual data representation strategies from the reviewed articles: *Bag-of-word*, *Rule-based*, and *Embedding* features. The Bag-of-word considers a sequence of text as a set (or multi-set) of the word regardless of word order and grammar (Manning and Schütze 1999). Various BOW representation strategies can be found in the included works, such as using simple raw counts for words, a bag of bigram or trigram (Kurtanovic and Maalej 2017a), and BOW with TF-IDF weighting (Li et al., 2018).

In addition to BOW features, studies included in this review also applied rule-based handcraft features, such as POS n-gram (Kurtanovic and Maalej 2017b), the number of Noun/Verb/Adj/Adv/Modal (Hussain et al., 2008; Kurtanovic & Maalej, 2017a; Liu et al., 2013), frequency of POS of the keywords (Halim and Siahaan 2019), and the count of syntax sub-tree (Dalpiaz et al. 2019; Kurtanovic and Maalej 2017b, 2017a). In addition, textual descriptive statistics are also applied to represent requirements, including the number of characters (Abualhaija et al. 2019), word count (Kurtanovic and Maalej 2017b), sentence count (Qi et al., 2016), paragraphs count (Parra et al. 2015), and the number of words per sentence (Ormandjieva et al. 2007). Furthermore, temporal features including verb tense (Stanik et al. 2019), number of elapsed days (Liu et al., 2013), and temporal tags, such as time, duration, and time set (Abad et al. 2017) were used to represent the temporal information of the requirements. For UGC-based research, some platforms provide metadata that can be extracted to represent user comments. Metadata features include star ratings (Maalej et al. 2016), review count (Martens and Maalej 2019), and the number of links (Parra et al. 2015).

Moreover, some studies applied document quality features to represent textual requirements, including the number of subjective/objective sentences in a review (Liu et al., 2013), the number of ambiguous expressions in a requirement (Ormandjieva et al. 2007; Parra et al. 2015), and the number of the sentence referring product feature appeared in a user review (Liu et al., 2013; Qi et al., 2016). Additionally, some articles introduce domain-specific features, such as the number of design terms (Parra et al. 2015) and the number of keywords from the input text (Hussain et al. 2008; Stanik et al. 2019).

In recent years, word embedding has gained popularity in a range of natural language processing applications. The selected articles used a range of embedding techniques, including Word2vec

(Mikolov et al. 2013), FastText (Joulin et al. 2017), Glove (Pennington et al. 2014), and BERT (Devlin et al. 2019) to represent words. Three strategies associated with embedding features are identified in the included studies: training the embedding from scratch using a pre-trained embedding and fine-tuning the previously trained language models.

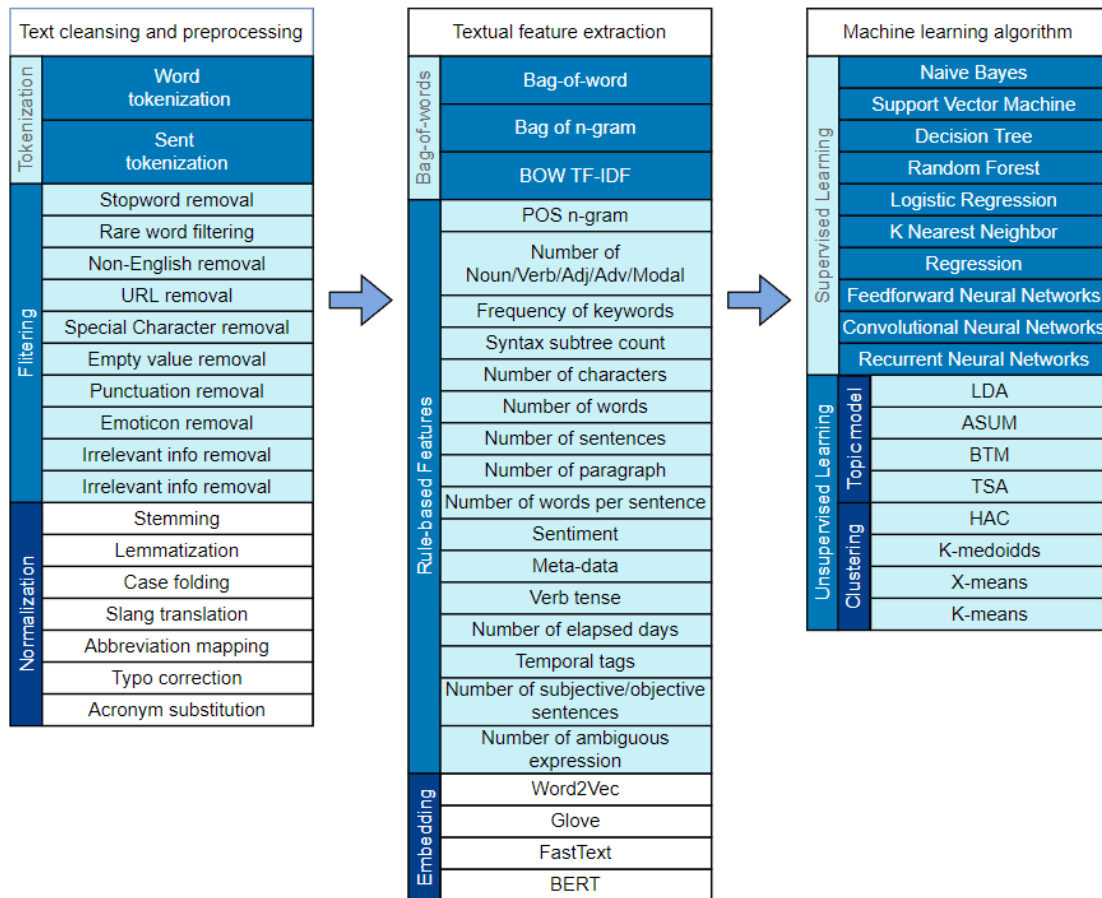


Figure 29. Technologies and algorithms.

Machine Learning

In this review, the learning algorithms applied by the included studies are categorized into two categories: *supervised* and *unsupervised learning*. Under *supervised learning* categories, only three studies have applied regression models (Chen et al., 2016; Liu et al., 2013; Qi et al., 2016). The regression methods can help engineers to predict a numerical value to reflect the helpfulness

of a given user review. The rest of the methods in *supervised learning* are all classification algorithms. *Topic modeling* and *clustering* techniques are two frequently applied *Unsupervised Learning* methods and the LDA is the most widely applied unsupervised method in the papers included.

Evaluation methods

The quality of models can be reflected in the evaluation metrics, which are a set of formulas and units of measurement that reflect how well the learning algorithm could perform (Hossin & Sulaiman, 2015). For different types of learning tasks, the evaluation methods are used differently. In the included studies, the Mean Absolute Error (MAE) and Root Mean Square Error (RMSE) are employed for regression models. Both MAE and RMSE are negatively oriented, which means the better the model, the lower the errors. Precision, Recall, and F1-score are most frequently applied for classification models. On the other hand, the unsupervised method is evaluated by two strategies: internal and external evaluation. The included works applied Intra and inter-cluster similarity (or distance), Silhouette score, and perplexity to assess the clustering results for internal evaluation. In the case of external evaluation, domain experts are asked to evaluate the models' results manually. Additionally, a truth set can be built to evaluate the clustering results, similar to a supervised classifier.

Available tools

The included studies widely mentioned two types of tools: ML tools and NLP tools. The NLP tools such as NLTK and CoreNLP are applied to preprocess and extract the features from the textual data. The most widely mentioned ML tools are Weka and Scikit-learn, which integrate multiple ML algorithms and quickly build a data-driven solution. Keras is a popular deep learning library

among the included studies, which contains the most popular neural network architectures with compact and straightforward APIs.

Table 16 lists the tools mentioned in the reviewed articles, arranged by their uses. In total, seven types of tools are extracted: ML tools (conventional), deep learning tools, language, and topic model tools, multi-purpose NLP tools, sentiment analysis tools, parsing tools, and single-purpose tools.

Category	Sub-categories	Tool name
<i>ML tools</i>	ML	WEKA, Scikit-learn, MATLAB, Classifier4J (Java)
	Deep learning	TensorFlow, Keras
	Language/topic model tools	JLDADMM (Java), Stanford topic model toolbox (Java), Spacy (Python), Genism (Python), tm text mining package (R)
<i>NLP tools</i>	Multi-purpose NLP tools	NLTK (Python), LingPipe (Java), koRups package (R), CoreNLP (Java), Stanford NLP toolkit (Java/Python), WordNet, koRpus statistical readability package, IBM Watson NLU
	Sentiment analysis tools	SEMAFOR (Java), SentiStrength (Java), VaderSentiment (Python), Textblob
	Parsing tools	MaltParser (Java), Stanford POS tagger, Stanford parser, Berkeley parser, Stanford temporal tagger
	Single-purpose tools	Jazzy (spell-checker), Jieba (Chinese character tokenizer)

Table 16. Tools mentioned by included works.

4.4.4. The process of building ML-based requirement elicitation

Building an ML-based requirement elicitation method contains four major steps: study design, data preparation, model construction, and model implementation. The first step is to design the ML-based requirement elicitation study by considering two fundamental elements: identifying the requirement elicitation subtasks treatable with ML and analyzing available datasets related to the subtasks. This review identifies three major tasks that support requirement elicitation, which various ML algorithms can facilitate. The performance of machine learning depends on the data set, which should be structured following the structure of the problem. Therefore, domain

knowledge is crucial to structure the requirement elicitation task into simpler, repetitive, and well-defined problems solvable with ML methods. In achieving the objective, several questions must be answered, such as 1) what are the inputs and outputs of the concerned ML model? and 2) what is the operational environment of the ML model? The answers to these questions lead to different study designs. For example, both requirement documents and UGC data are studied in the selected research. The requirement documents are stored in plain text format, and the tasks usually focus on requirement text classification. On the other hand, the UGC data contain additional structured meta-data, which describes the UGC in many aspects, such as ratings and timestamps. With these additional data, studies such as the prediction of usefulness associated with user reviews(Liu et al., 2013), and analysis of user preferences on a timely basis are possible (Fu et al. 2013). In addition, sentiment is another reason that causes differences in requirement document analysis and UGC analysis. Documents describing the requirements are usually written with neutral language; therefore, analyzing the sentiment of each requirement may not be as significant as analyzing the sentiment of UGC. As a result, sentiment analysis is not applied to requirement document analysis tasks but is commonly used in UGC-based research.

The second step to build an ML-based requirement elicitation method is data preparation. In this step, the relevant data should be sampled from available sources based on the study design, such as databases, social networks, app markets, and e-commerce websites. Annotations and labels are required for supervised learning, where domain expertise is essential. Domain experts must assess the quality of the datasets and guide the development of rules for annotation and labeling.

The third step is to construct a model with the help of the conventional ML pipeline, which relies heavily on the understanding of machine learning and related techniques. Data cleansing, data preprocessing, feature extraction, model training, and model evaluation are part of this

phase. Though the model construction pipeline can be independent of domain knowledge for unsupervised learning, domain expertise is still necessary to validate and evaluate the models.

Finally, the model implementation is an important final step to build an ML-based requirement elicitation. Multiple aspects must be considered, such as culture, management, security, development, and operation. Hence, the major challenge comes from the gap between research and practice.

In summary, machine learning algorithms can assist designers in analyzing and extracting needs from a variety of documents provided by traditional methodologies, such as interview transcripts and meeting minutes. Building an ML solution for requirement engineering is a transdisciplinary effort involving design, computer, and implementation sciences. At this stage, the machine learning-based requirement elicitation can only take on a supportive role that complements conventional methods. Conventional requirement elicitation cannot yet be replaced entirely by machine learning.

4.5. The limitations, open issues, and future works

4.5.1. The role of ML and its limitations in the requirement elicitation

It is important to note that eliciting requirements is not one single activity, rather it comprises multiple sessions and operations that work together as a whole. However, there is no very detailed definition or uniform approach to this stage in academia and industry. For example, Young (2004) suggested a twenty-eight-step requirement gathering activities checklist including planning, managing, collecting, reviewing, tracing, etc. Wiegers & Beatty (2013) summarizes 21 best practices for requirements elicitation, including defining scope, identifying stakeholders, reusing existing requirements, modeling the application environment, and analyzing requirements

feasibility. Using a single ML model cannot accomplish so many different tasks. Therefore, ML techniques are only able to accomplish partial tasks involved in requirement elicitation. Furthermore, most of the included studies are all focusing on resolving a particular task with ML, rather than designing a complete system that supports requirement elicitation. In this regard, most of the ML-based methods developed so far have a supporting or complementary role to traditional methods. For example, in an ML-aided requirement elicitation system, conventional methods, such as interviews, questionnaires, and brainstorming, are responsible for producing and collecting requirement-related data. ML algorithms, however, are responsible for analyzing data or supporting follow-up data-related activities.

In section 4.1, we summarized 15 ML-based requirements elicitation sub-tasks from included studies and categorize them into four groups. However, most works were classified as Preparation (n=37), and Collection (n=41) tasks. Only eight articles were identified as Validation (n=5) and Negotiation (n=3). One of the reasons for this is that the validation and negotiation are hard to articulate due to the high complexity of the tasks. For example, tasks from Negotiations require collaboration, discussion, and trade-offs between stakeholders from many aspects. Therefore, most of the challenges related to these tasks are related to background knowledge, communication, budgets, or other limitations imposed by the real world. As a result, it is difficult to train a machine learning algorithm without a well-developed dataset that incorporates all required criteria and information.

4.5.2. Open issues and future works

It is still challenging to build an ML-based solution to fully automate requirement elicitation. First, since requirement elicitation is a comprehensive process composed of a variety of tasks and goals, it is difficult to develop an end-to-end ML model to fully automate the requirement elicitation

process. Second, requirements could come from a large variety of sources, particularly in the big data era. In terms of data type and format, the datasets included in the study were highly heterogeneous. For example, sentiment analysis may be useful when analyzing UGC data, but it is not valuable when analyzing neutral document data. Hence, using the model specifically designed for UGC, such as ASUM (Jo & Oh, 2011), cannot perform as expected on document data, and vice versa. Third, the ML-based requirement elicitation approach is automatic but easily affected by errors and failure. Unlike rule-based systems that can be debugged and fixed locally in the coded knowledge body, it is difficult to directly tune the ML model when dealing with known errors. In addition, the interpretation of ML models is still an open challenge in academia and industry. For example, deep neural networks learn features automatically, which makes it more challenging to analyze the reasons behind ML-based solutions. Furthermore, only a few research considered the changing nature of the requirements. Due to the dynamic nature of the requirements, in practice, requirement elicitation requires engineers to identify and modify requirements based on the unpredictable nature of user needs (Xie et al., 2017). Besides, in terms of both content and type of task, the current research is monotonous. The vast majority of studies still focus on categorical clustering.

To tackle these challenges, the following future research directions are suggested by the authors. First, although, there are growing interests and works in building ML-based requirement elicitation methods, there is still a vacancy for a systematic guide on how to integrate the ML-based components into the requirement elicitation framework. Multiple aspects of the integrated system should be considered, such as how humans and machines interact in requirements elicitation, what is the input-output of the system and each sub-system, and which specific tasks should be performed by machines when expert involvement is required, among others. Hence, a systematic

study and guidance of AI system design, engineering, implementation, and management are required.

Second, there is a lack of in-depth analysis of ML-based requirement elicitation failure and errors. For example, research papers and projects typically rely on statistical metrics for ML model validation and evaluation. This type of evaluation can tell us how good or bad a model is, but neglects to address the question of what leads a model to perform unexpectedly. Future studies should address this issue by introducing methods and techniques to explore the factors that affect the performance of ML-based requirement elicitation.

Third, the ML-based methods, especially deep learning models are lacking transparency. Because deep neural networks derive their features not from experience and knowledge, but from data, which is more effective but less intuitive. Since requirement elicitation is knowledge-intensive human-involved activity, the engineers not only expect models to solve the problems but also to explain them. The significance of Explainable AI (XAI) is increased along with the widespread adoption of deep learning methods in recent years (Xu et al., 2019). In the future, research in ML-based engineering of requirements will also need to leverage XAI techniques and methods to investigate the nature of decision-related requirements.

Forth, a broad range of NLP tasks could be incorporated into the requirements elicitation. Apart from text classification, many other NLP techniques can be utilized to support requirements elicitation, such as neural summarization, text generation, neural conversational bots, question asking, question answering, and text-to-speech. Due to its wide range of tasks, requirements elicitation provides an excellent opportunity to practice cutting-edge NLP methods. Future research works should try more to incorporate these methods into requirement elicitation. As an

example, neural text generation technologies such as Seq2Seq(Sutskever et al., 2014), GAN (Goodfellow et al., 2014), and T5 Text-to-Text transformers (Matena et al., 2019) have the potential to produce new mock requirements based on a particular context, which may provide innovative data-driven ideas from a new perspective.

Fifth, aside from natural text, use requirements also can be mined from other data formats. E-commerce platforms, for instance, allow individuals to upload videos and pictures to share usage experiences, complaints, and feedback. Although techniques such as neural image description (Karpathy & Li, 2017; Vinyals et al., 2015), and neural video description (Yao et al., 2015) are not as mature as text classification techniques, they are also of great research value and can play a major role in requirement engineering as well.

Sixth, due to the data-intensive nature of ML methods, more requirements related to high-quality text data should also be introduced. However, some interest-related requirements are requested to be kept confidential by the relevant stakeholders. Hence, sharing high-quality requirement data with the requirement engineering community is challenging. Masking sensitive data or substituting entities can be effective means of modifying sensitive requirements, which can facilitate the sharing of information within the requirement engineering community. Another strategy to address insufficient training data is to develop a language model specifically for requirements engineering. Research shows that transfer learning techniques can overcome the limitations of insufficient data (Howard & Ruder, 2018). Future works could also consider building neural language models that are specifically trained with requirement specifications.

Seventh, since user-requirement elicitation is a human-centric activity, analyzing user behavior may provide valuable insight into understanding and eliciting requirements. As the study of

representation learning, such as user embedding, is being applied to a variety of different domains, including recommendation and healthcare systems (Miotto et al., 2016; Pan & Ding, 2019). Analyzing user behavior can help to predict user preference and explore potential requirements change.

Last, future work should address the issues caused by the dynamic nature of user requirements. In practice, stakeholder requirements are not always static; however, in the studies reviewed, ML algorithms were used to read the static text to identify requirements. Further research on ML-based methods should be focusing on changing requirements and reducing their impact is urgently needed.

4.6. Limitations of the literature review

We used PRISMA as the research framework to identify the primary research studies in this review. Unlike other popular methods, such as snowballing approach, in this study, we did not exhaustively identify further relevant studies by iterating through the reference lists. This review chose to use minimum evidence to reflect the current state of ML-based requirements elicitation rather than providing an exhaustive result. Thus, some relevant studies may have been omitted from this review. In addition, there is a paradox between literature review and search query generation. Before a literature review is completed, it is not easy to define a set of exact keywords to represent the topic. Simultaneously, the absence of good search queries and keywords could defy the effort to retrieve relevant papers effectively. Hence, it is challenging to develop a perfect set of search queries at the initial stage that covers all of the aspects related to the field. To deal with these issues, we dynamically adjusted the search queries for seven academic databases to reduce bias and loss in the search results.

Furthermore, the inclusion and exclusion criteria may exclude work related to the topic due to its qualitative and subjective nature. This review provides a synthesis of recent research by highlighting the techniques used to develop machine learning-based solutions for requirements elicitation. The purpose of this review is to provide an index of success stories of ML applications in the field of requirements elicitation, which supports the audiences in replicating existing solutions or building their own. Numerous publications are excluded due to a lack of technical details; this does not imply that those articles are unimportant to this field. Various ideas and concepts may still be derived from these works. Moreover, only one of the similar works by the same author has been retained in the study; however, it is difficult to define a clear boundary to decide which work to keep. As a precaution to minimize the risks associated with inclusion-exclusion criteria, the authors discussed and evaluated the articles through meetings in cases wherever it was challenging to decide individually.

Additionally, human errors could not be avoided in the data extraction phase due to its nature of subjectivity (section 2.4). As Table 4 illustrated, the reviewer needs to enter two types of data manually. The first type of data is the descriptive data of an article, which can be accessed from the article database and the journal website. However, the second type of data requires reviewers to assess and extract information based on personal understanding. Therefore, the data extraction process inevitably contains a certain amount of bias and subjectivity. In addition, since requirement elicitation is an interdisciplinary problem, many definitions may be disputed. For example, the definition of the requirement, requirement elicitation, and the lifecycle and the scope of requirement elicitation are all defined differently by various researchers. Thus, we cannot avoid a certain amount of subjectivity in some interpretations, citations, and descriptions. Besides, some information was not explicitly stated in the reviewed articles, which led to difficulties in

corresponding information retrieval. To overcome this limitation, the author team iterated and adjusted the data extraction table before reaching a final agreement

4.7. Related works

To our knowledge, eight existing review articles, as shown in Table 17, are relevant to our study. Meth et al. (2013) conducted a review mainly focused on the automated approach applied for requirement elicitation, mainly focusing on the degree of the automation of proposed approaches. Binkhonain & Zhao (2019) introduced ML algorithms in the requirement elicitation domain by dividing the 24 related articles into three sections: NLP techniques, ML algorithms, and evaluation. Perez-Verdejo et al. (2020) applied topic models and visualization techniques to analyze ML-based requirement classification articles. Wong et al. (2017) identified various software requirement elicitation methods, including manual, rule-based, and machine-learning-based approaches. Shabestari et al. (2019) proposed a systematic literature review that covers early product development phases, including various activities such as requirement elicitation, requirement identification, and requirement categorization. Similarly, Sampada et al. (2020) focus on the early requirement phases but are more concerned with requirement elicitation and documentation. Ahmad et al. (2020) reviewed a collection of articles for identifying requirements for Q&A platforms.

Among the existing studies, one existing work proposed by Lim et al. (2021) is the closest to our research, which was conducted almost concurrently with ours. Both works aim to introduce the current state of the works in data-driven requirement elicitation; however, the focuses of the two works are different. Lim et al. (2021) focus more on data sources, data types, learning techniques, and the degree of automation. In comparison, the present review focuses more on technical aspects.

Our work aims to provide a comprehensive overview of current work and include a more detailed investigation into the existing methods, algorithms, and tools. This review could provide a more practical guide to requirement elicitation researchers, data engineers, and designers to leverage the existing techniques in their projects.

Title	Works (#)	Structure	Included latest work	Focus
The state of the art in automated requirements elicitation	36	Conceptual	2010	The automated approach in requirement elicitation; degree of automation; knowledge reuse; evaluation; and the relationship between the included works.
A systematic literature review about software requirements elicitation	42	Conceptual	2014	Level of automation; knowledge reuse; the importance of human factors; collaborative approach; and types of projects.
A review of ML algorithms for identification and classification of non-functional requirements	24	Methodological	2017	ML algorithms for non-function requirements identification/classification
A survey on the applications of ML in the early phases of product development	40	Conceptual	2018	ML in requirements, modeling, and concept design.
A Systematic Literature Review on Using ML Algorithms for Software Requirements Identification on Stack Overflow	12	Methodological	2018	The ML and NLP techniques were applied for the requirement identification task on Stack overflow data.
A review on advanced techniques of requirement elicitation and specification in software development stages	13	Methodological	2019	Introduced the application of NLP techniques and ML algorithms in requirement elicitation and specification activities.
A systematic literature review on ML for automated requirements classification	13	Methodological	2019	The ML algorithms are used in requirement classification tasks.
Data-Driven Requirements Elicitation: A Systematic Literature Review	68	Methodological	2020	Data; ML techniques; evaluation methods; degree of automation in requirement elicitation.
Proposed study	86	Methodological	-	Data; ML-based requirement elicitation subtasks; textual requirements techniques; feature extraction techniques; learning algorithms, process; and tools in requirement elicitation.

Table 17. Related works.

4.8. Conclusions

The review provides an overview of the current research on machine learning-based requirements elicitation. First, we categorized the included studies into four ML-based requirement elicitation tasks: Preparation, Collection, Validation, and Negotiation. Second, we examined the data sources and corpora used by the included studies to develop machine learning models for requirements elicitation. As a result, we identified three types of data sources for building ML solutions, which are Textual Documents, User-Generated Content (UGC), and Existing Requirement Datasets. Third, in this review, general ML pipelines are extracted from the included studies: text cleansing and preprocessing, textual feature extraction, machine learning, and evaluation. Further, we identified nineteen tasks among the selected works and assigned them to three types of text cleaning and preprocessing groups: filtering, normalizing, and tokenizing. For the text feature extraction part, we classified the included works into three groups according to the technique used to extract the features. BOW language models and handcrafted features are frequently found in reviewed publications, but in recent years, an increasing trend towards using embedding features has been observed. In addition, we discovered the most popular algorithms, such as Naive Bayes, Support Vector Machines, Decision Trees, and Neural Networks in this review. Precision, Recall, and F1-score are the most prevalent evaluation metrics applied to assess model performance. Finally, we listed the most popular NLP tools, which are NLTK and CoreNLP, and the most commonly applied machine learning tools, Weka and Scikit-learn.

Apart from the main findings, one major observation is that most research focuses on requirements categorization tasks. There is a notable majority of papers in the collection that are focused on supervised text classification, followed by topic modeling and clustering techniques. Second, we noticed that the existing articles are more focused on using machine learning to solve specific and

fine-grained problems in requirements elicitation, such as classifying NFRs and extracting main topics from massive user reviews. It has, however, been relatively rare for research to examine how to integrate machine learning-based requirements acquisition methods into existing requirements elicitation workflows. Hence, the lack of expertise in designing, engineering, implementing, and configuring ML-based requirement elicitation systems calls for further research. Furthermore, most studies lack concrete evidence that machine learning can assist designers and engineers in reducing time and effort in requirement extraction. Last, although supervised learning is prevalent in this field, we have found only two publicly accessible labeled datasets from the 86 reviewed papers: DePaul's NFRs dataset (Cleland-Huang et al., 2006) and SecReq (Knauss et al., 2011).

Thus far, ML-based solutions have been monolithic in eliciting requirements; however, the publications in this field provide sufficient evidence that machine learning can support requirements activities both theoretically and practically. Some labor-intensive, error-prone activities from requirement engineering are waiting to be supported by ML. Despite what has already been accomplished, the best is yet to come.

Appendix. The list of included works

ID	Reference of included works
S01	*Abad, Z. S. H., Karras, O., Ghazi, P., Glinz, M., Ruhe, G., & Schneider, K. (2017). What Works Better? A Study of Classifying Requirements. <i>Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017</i> , 1, 496–501. https://doi.org/10.1109/RE.2017.36
S02	*Abualhaja, S., Arora, C., Sabetzadeh, M., Briand, L. C., & Vaz, E. (2019). A machine learning-based approach for demarcating requirements in textual specifications. <i>Proceedings of the IEEE International Conference on Requirements Engineering, 2019-Septe</i> , 51–62. https://doi.org/10.1109/RE.2019.00017
S03	*Al-Subaih, A. A., Sarro, F., Black, S., Capra, L., Harman, M., Jia, Y., & Zhang, Y. (2016). Clustering Mobile Apps Based on Mined Textual Features. <i>International Symposium on Empirical Software Engineering and Measurement, 08-09-Sept</i> . https://doi.org/10.1145/2961111.2962600
S04	*Asif, M., Ali, I., Malik, M. S. A., Chaudary, M. H., Tayyaba, S., & Mahmood, M. T. (2019). Annotation of Software Requirements Specification (SRS), Extractions of Nonfunctional Requirements, and Measurement of Their Tradeoff. <i>IEEE Access</i> , 7, 36164–36176. https://doi.org/10.1109/ACCESS.2019.2903133
S05	*Baker, C., Deng, L., Chakraborty, S., & Dehlinger, J. (2019). Automatic multi-class non-functional software requirements classification using neural networks. <i>Proceedings - International Computer Software and Applications Conference</i> , 2, 610–615. https://doi.org/10.1109/COMPSAC.2019.10275
S06	*Bakiu, E., & Guzman, E. (2017). Which feature is unusable? Detecting usability and user experience issues from user reviews. <i>Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017</i> , 182–187. https://doi.org/10.1109/REW.2017.76
S07	*Barbosa, R., Janeiro, D., Silva, A. E., Moraes, R., & Martins, P. (2015). An Approach to Clustering and Sequencing of Textual Requirements. <i>Proceedings - 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2015</i> , 39–44. https://doi.org/10.1109/DSN-W.2015.20
S08	*Canedo, E. D., & Mendes, B. C. (2020). Software requirements classification using machine learning algorithms. <i>Entropy</i> , 22(9). https://doi.org/10.3390/E22091057
S09	*Carreno, L. V. G., & Winbladh, K. (2013). Analysis of user comments: An approach for software requirements evolution. <i>Proceedings - International Conference on Software Engineering</i> , 582–591. https://doi.org/10.1109/ICSE.2013.6606604
S10	*Casamayor, A., Godoy, D., & Campo, M. (2009). Semi-supervised classification of non-functional requirements: An empirical analysis. <i>Inteligencia Artificial</i> , 13(44), 35–45. https://doi.org/10.4114/ia.v13i44.1044
S11	*Casamayor, A., Godoy, D., & Campo, M. (2010). Identification of non-functional requirements in textual specifications: A semi-supervised learning approach. <i>Information and Software Technology</i> , 52(4), 436–445. https://doi.org/10.1016/j.infsof.2009.10.010
S12	*Chen, J., Zhang, C., & Niu, Z. (2016). Identifying helpful online reviews with word embedding features. <i>Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)</i> , 9983 LNAI, 123–133. https://doi.org/10.1007/978-3-319-47650-6_10
S13	*Chen, N., Lin, J., Hoi, S. C. H., Xiao, X., & Zhang, B. (2014). AR-miner: Mining informative reviews for developers from mobile app marketplace. <i>Proceedings - International Conference on Software Engineering</i> , 1, 767–778. https://doi.org/10.1145/2568225.2568263
S14	*Cleland-Huang, J., Settini, R., Zou, X., & Solc, P. (2007). Automated classification of non-functional requirements. <i>Requirements Engineering</i> , 12(2), 103–120. https://doi.org/10.1007/s00766-007-0045-1
S15	*Dalpiaz, F., Dell'Anna, D., Aydemir, F. B., & Çevikol, S. (2019). Requirements classification with interpretable machine learning and dependency parsing. <i>Proceedings of the IEEE International Conference on Requirements Engineering, 2019-Septe</i> , 142–152. https://doi.org/10.1109/RE.2019.00025
S16	*Dekhtyar, A., & Fong, V. (2017). RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow. <i>Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017</i> , 484–489. https://doi.org/10.1109/RE.2017.26
S17	*Deshpande, G., Arora, C., & Ruhe, G. (2019). Data-driven elicitation and optimization of dependencies between requirements. <i>Proceedings of the IEEE International Conference on Requirements Engineering, 2019-Septe</i> , 416–421. https://doi.org/10.1109/RE.2019.00055
S18	*Dhinakaran, V. T., Pulle, R., Ajmeri, N., & Murukannaiah, P. K. (2018). App review analysis via active learning: Reducing supervision effort without compromising classification accuracy. <i>Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018</i> , 170–181. https://doi.org/10.1109/RE.2018.00026

S19	*El Dehaibi, N., Goodman, N. D., & MacDonald, E. F. (2019). Extracting customer perceptions of product sustainability from online reviews. <i>Journal of Mechanical Design, Transactions of the ASME</i> , 141(12). https://doi.org/10.1115/1.4044522
S20	*EzzatiKarami, M., & Madhavji, N. H. (2021). Automatically Classifying Non-functional Requirements with Feature Extraction and Supervised Machine Learning Techniques: A Research Preview. In <i>Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)</i> : Vol. 12685 LNCS. Springer International Publishing. https://doi.org/10.1007/978-3-030-73128-1_5
S21	*Falessi, D., Cantone, G., & Canfora, G. (2010). A comprehensive characterization of NLP techniques for identifying equivalent requirements. <i>ESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement</i> . https://doi.org/10.1145/1852786.1852810
S22	*Fu, B., Lin, J., Liy, L., Faloutsos, C., Hong, J., & Sadeh, N. (2013). Why people hate your App - Making sense of user feedback in a mobile app store. <i>Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Part F1288</i> , 1276–1284. https://doi.org/10.1145/2487575.2488202
S23	*Gnanasekaran, R. K., Chakraborty, S., Dehlinger, J., & Deng, L. (2021). Using recurrent neural networks for classification of natural language-based non-functional requirements. <i>CEUR Workshop Proceedings</i> , 2857.
S24	*Gulle, K. J., Ford, N., Ebel, P., Brokhausen, F., & Vogelsang, A. (2020). Topic Modeling on User Stories using Word Mover's Distance. <i>Proceedings - 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020</i> , 52–60. https://doi.org/10.1109/AIRE51212.2020.00015
S25	*Guzman, E., Ibrahim, M., & Glinz, M. (2017). A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution. <i>Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017</i> , 3, 11–20. https://doi.org/10.1109/RE.2017.88
S26	*Guzman, E., & Maalej, W. (2014). How do users like this feature? A fine grained sentiment analysis of App reviews. 2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings, 153–162. https://doi.org/10.1109/RE.2014.6912257
S27	*Halim, F., & Siahaan, D. (2019). Detecting Non-Atomic Requirements in Software Requirements Specifications Using Classification Methods. 2019 1st International Conference on Cybernetics and Intelligent System, ICORIS 2019, August, 269–273. https://doi.org/10.1109/ICORIS.2019.8874888
S28	*Han, Y., & Moghaddam, M. (2021). Eliciting Attribute-Level User Needs from Online Reviews with Deep Language Models and Information Extraction. <i>Journal of Mechanical Design, Transactions of the ASME</i> , 143(6). https://doi.org/10.1115/1.4048819
S29	*Haque, M. A., Rahman, M. A., & Siddik, M. S. (2019). Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study. 1st International Conference on Advances in Science, Engineering and Robotics Technology 2019, ICASERT 2019, 2019(Icasert). https://doi.org/10.1109/ICASERT.2019.8934499
S30	*Hussain, I., Kosseim, L., & Ormandjieva, O. (2008). Using linguistic knowledge to classify non-functional requirements in SRS documents. In <i>Lecture Notes in Computer Science: Vol. 5039 LNCS (pp. 287–298)</i> . https://doi.org/10.1007/978-3-540-69858-6_28
S31	*Jeon, K., Lee, G., & Jeong, H. D. (2021). Classification of the Requirement Sentences of the US DOT Standard Specification Using Deep Learning Algorithms. <i>Lecture Notes in Civil Engineering</i> , 98, 89–97. https://doi.org/10.1007/978-3-030-51295-8_8
S32	*Jha, N., & Mahmoud, A. (2019). Mining non-functional requirements from App store reviews. <i>Empirical Software Engineering</i> , 24(6), 3659–3695. https://doi.org/10.1007/s10664-019-09716-7
S33	*Jindal, R., Malhotra, R., & Jain, A. (2016). Automated classification of security requirements. 2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016, 2027–2033. https://doi.org/10.1109/ICACCI.2016.7732349
S34	*Jo, Y., & Oh, A. (2011). Aspect and sentiment unification model for online review analysis. <i>Proceedings of the 4th ACM International Conference on Web Search and Data Mining, WSDM 2011</i> , 815–824. https://doi.org/10.1145/1935826.1935932
S35	*Jones, I. A., & Kim, K. Y. (2015). Systematic Service Product Requirement Analysis with Online Customer Review Data. <i>Journal of Integrated Design and Process Science</i> , 19(2), 25–48. https://doi.org/10.3233/jid-2015-0011
S36	*Joung, J., & Kim, H. M. (2021). Automated keyword filtering in latent dirichlet allocation for identifying product attributes from online reviews. <i>Journal of Mechanical Design, Transactions of the ASME</i> , 143(8), 1–6. https://doi.org/10.1115/1.4048960
S37	*Kengphanphanit, N., & Muenchaisri, P. (2020). Automatic Requirements Elicitation from Social Media (ARESM). <i>PervasiveHealth: Pervasive Computing Technologies for Healthcare</i> , 57–62. https://doi.org/10.1145/3418994.3419004
S38	*Khelifa, A., Haoues, M., & Sellami, A. (2018). Towards a software requirements change classification using support vector machine. <i>CEUR Workshop Proceedings</i> , 2279, 1–10.
S39	*Ko, Y., Park, S., Seo, J., & Choi, S. (2007). Using classification techniques for informal requirements in the requirements analysis-supporting system. <i>Information and Software Technology</i> , 49(11–12), 1128–1140. https://doi.org/10.1016/j.infsof.2006.11.007

S40	*Kobilica, A., Ayub, M., & Hassine, J. (2020). Automated Identification of Security Requirements: A Machine Learning Approach. <i>PervasiveHealth: Pervasive Computing Technologies for Healthcare</i> , 1, 475–480. https://doi.org/10.1145/3383219.3383288
S41	*Kurtanovic, Z., & Maalej, W. (2017a). Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning. <i>Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017</i> , 490–495. https://doi.org/10.1109/RE.2017.82
S42	*Kurtanovic, Z., & Maalej, W. (2017b). Mining User Rationale from Software Reviews. <i>Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017</i> , 61–70. https://doi.org/10.1109/RE.2017.86
S43	*Lange, D. S. (2008). Text classification and machine learning support for requirements analysis using blogs. <i>Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)</i> , 5320 LNCS, 182–195. https://doi.org/10.1007/978-3-540-89778-1_14
S44	*Li, C., Huang, L., Ge, J., Luo, B., & Ng, V. (2018). Automatically classifying user requests in crowdsourcing requirements engineering. <i>Journal of Systems and Software</i> , 138, 108–123. https://doi.org/10.1016/j.jss.2017.12.028
S45	*Li, J., Zhang, X., Wang, K., Zheng, C., Tong, S., & Eynard, B. (2020). A personalized requirement identifying model for design improvement based on user profiling. <i>Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM</i> , 34(1), 55–67. https://doi.org/10.1017/S0890060419000301
S46	*Li, T. (2018). Identifying Security Requirements Based on Linguistic Analysis and Machine Learning. <i>Proceedings - Asia-Pacific Software Engineering Conference, APSEC, 2017-Decem</i> , 388–397. https://doi.org/10.1109/APSEC.2017.45
S47	*Liu, Y., Jin, J., Ji, P., Harding, J. A., & Fung, R. Y. K. (2013). Identifying helpful online reviews: A product designer's perspective. <i>CAD Computer Aided Design</i> , 45(2), 180–194. https://doi.org/10.1016/j.cad.2012.07.008
S48	*Liu, Y., Lu, W. F., & Loh, H. T. (2007). Knowledge discovery and management for product design through text mining - A case study of online information integration for designers. <i>Proceedings of ICED 2007, the 16th International Conference on Engineering Design, DS 42(August)</i> , 1–12.
S49	*Lu, M., & Liang, P. (2017). Automatic classification of non-functional requirements from augmented app user reviews. <i>Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering, Part F1286</i> , 344–353. https://doi.org/10.1145/3084226.3084241
S50	*Lyutov, A., Uygun, Y., & Hütt, M. T. (2019). Managing workflow of customer requirements using machine learning. <i>Computers in Industry</i> , 109, 215–225. https://doi.org/10.1016/j.compind.2019.04.010
S51	*Maalej, W., Kurtanović, Z., Nabil, H., & Stanik, C. (2016). On the automatic classification of app reviews. <i>Requirements Engineering</i> , 21(3), 311–331. https://doi.org/10.1007/s00766-016-0251-9
S52	*Mahmoud, A. (2015). An information theoretic approach for extracting and tracing non-functional requirements. <i>2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings</i> , 36–45. https://doi.org/10.1109/RE.2015.7320406
S53	*Martens, D., & Maalej, W. (2019). Towards understanding and detecting fake reviews in app stores. <i>Empirical Software Engineering</i> , 24(6), 3316–3355. https://doi.org/10.1007/s10664-019-09706-9
S54	*Massey, A. K., Eisenstein, J., Anton, A. I., & Swire, P. P. (2013). Automated text mining for requirements analysis of policy documents. <i>2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings</i> , 4–13. https://doi.org/10.1109/RE.2013.6636700
S55	*Myllynen, S., Suominen, I., Raunio, T., Karell, R., & Lahtinen, J. (2021). Developing and Implementing artificial intelligence-based classifier for requirements engineering. <i>Journal of Nuclear Engineering and Radiation Science</i> , 7(4), 1–9. https://doi.org/10.1115/1.4049722
S56	*Navarro-Almanza, R., Juarez-Ramirez, R., & Licea, G. (2018). Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification. <i>Proceedings - 2017 5th International Conference in Software Engineering Research and Innovation, CONISOFT 2017, 2018-Janua</i> , 116–120. https://doi.org/10.1109/CONISOFT.2017.00021
S57	*Noei, E., Zhang, F., & Zou, Y. (2021). Too Many User-Reviews! What Should App Developers Look at First? <i>IEEE Transactions on Software Engineering</i> , 47(2), 367–378. https://doi.org/10.1109/TSE.2019.2893171
S58	*Nyamawe, A. S., Liu, H., Niu, N., Umer, Q., & Niu, Z. (2019). Automated recommendation of software refactorings based on feature requests. <i>Proceedings of the IEEE International Conference on Requirements Engineering, 2019-Septe</i> , 187–198. https://doi.org/10.1109/RE.2019.00029
S59	*Ormandjieva, O., Hussain, I., & Kosseim, L. (2007). Toward a text classification system for the quality assessment of software requirements written in natural language. <i>SOQUA'07: Fourth International Workshop on Software Quality Assurance - In Conjunction with the 6th ESEC/FSE Joint Meeting</i> , 39–45. https://doi.org/10.1145/1295074.1295082

S60	*Ott, D. (2013). Automatic requirement categorization of large natural language specifications at Mercedes-Benz for review improvements. <i>Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)</i> , 7830 LNCS, 50–64. https://doi.org/10.1007/978-3-642-37422-7_4
S61	*Panichella, S., Di Sorbo, A., Guzman, E., Visaggio, C. A., Canfora, G., & Gall, H. C. (2015). How can i improve my app? Classifying user reviews for software maintenance and evolution. <i>2015 IEEE 31st International Conference on Software Maintenance and Evolution, ICSME 2015 - Proceedings, September</i> , 281–290. https://doi.org/10.1109/ICSM.2015.7332474
S62	*Parra, E., Dimou, C., Llorens, J., Moreno, V., & Fraga, A. (2015). A methodology for the classification of quality of requirements using machine learning techniques. <i>Information and Software Technology</i> , 67, 180–195. https://doi.org/10.1016/j.infsof.2015.07.006
S63	*Petcuşin, F., Stănescu, L., & Bădică, C. (2020). An Experiment on Automated Requirements Mapping Using Deep Learning Methods. <i>Studies in Computational Intelligence</i> , 868, 86–95. https://doi.org/10.1007/978-3-030-32258-8_10
S64	*Polpinij, J., & Namee, K. (2021). Automatically Retrieving of Software Specification Requirements Related to Each Actor. In <i>Lecture Notes in Networks and Systems (Vol. 251)</i> . Springer International Publishing. https://doi.org/10.1007/978-3-030-79757-7_12
S65	*Prasetyo, P. K., Lo, D., Achananuparp, P., Tian, Y., & Lim, E. P. (2012). Automatic classification of software related microblogs. <i>IEEE International Conference on Software Maintenance, ICSM</i> , 596–599. https://doi.org/10.1109/ICSM.2012.6405330
S66	*Qi, J., Zhang, Z., Jeon, S., & Zhou, Y. (2016). Mining customer requirements from online reviews: A product improvement perspective. <i>Information and Management</i> , 53(8), 951–963. https://doi.org/10.1016/j.im.2016.06.002
S67	*Rahimi, N., Eassa, F., & Elrefaei, L. (2020). An Ensemble Machine Learning Technique for Functional Requirement Classificatio. <i>Symmetry, MI</i> , 1–25.
S68	*Rahman, M. A., Haque, M. A., Tawhid, M. N. A., & Siddik, M. S. (2019). Classifying non-functional requirements using RNN variants for quality software development. <i>MaLTeSQuE 2019 - Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, Co-Located with ESEC/FSE 2019</i> , 25–30. https://doi.org/10.1145/3340482.3342745
S69	*Rashwan, A., Ormandjieva, O., & Witte, R. (2013). Ontology-based classification of non-functional requirements in software specifications: A new corpus and SVM-based classifier. <i>Proceedings - International Computer Software and Applications Conference</i> , ii, 381–386. https://doi.org/10.1109/COMPSAC.2013.64
S70	*Riaz, M., King, J., Slankas, J., & Williams, L. (2014). Hidden in plain sight: Automatically identifying security requirements from natural language artifacts. <i>2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings</i> , 183–192. https://doi.org/10.1109/RE.2014.6912260
S71	*Rodeghero, P., Jiang, S., Armaly, A., & McMillan, C. (2017). Detecting User Story Information in Developer-Client Conversations to Generate Extractive Summaries. <i>Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017</i> , 49–59. https://doi.org/10.1109/ICSE.2017.13
S72	*Sabir, M., Banissi, E., & Child, M. (2021). A Deep Learning-Based Framework for the Classification of Non-functional Requirements. In <i>Advances in Intelligent Systems and Computing: Vol. 1366 AISC</i> . Springer International Publishing. https://doi.org/10.1007/978-3-030-72651-5_56
S73	*Singh, A., & Tucker, C. S. (2017). A machine learning approach to product review disambiguation based on function, form and behavior classification. <i>Decision Support Systems</i> , 97, 81–91. https://doi.org/10.1016/j.dss.2017.03.007
S74	*Stanik, C., Haering, M., & Maalej, W. (2019). Classifying multilingual user feedback using traditional machine learning and deep learning. <i>Proceedings - 2019 IEEE 27th International Requirements Engineering Conference Workshops, REW 2019</i> , 220–226. https://doi.org/10.1109/REW.2019.00046
S75	*Stone, T., & Choi, S. K. (2013). Extracting consumer preference from user-generated content sources using classification. <i>Proceedings of the ASME Design Engineering Technical Conference</i> , 3 A, 1–9. https://doi.org/10.1115/DETC2013-13228
S76	*Suryadi, D., & Kim, H. M. (2019). A data-driven approach to product usage context identification from online customer reviews. <i>Journal of Mechanical Design, Transactions of the ASME</i> , 141(12), 1–13. https://doi.org/10.1115/1.4044523
S77	*Taj, S., Arain, Q., Memon, I., & Zubedi, A. (2019). To apply data mining for classification of crowd sourced software requirements. <i>PervasiveHealth: Pervasive Computing Technologies for Healthcare</i> , 42–46. https://doi.org/10.1145/3328833.3328837
S78	*Tamai, T., & Anzai, T. (2018). Quality requirements analysis with machine learning. <i>ENASE 2018 - Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering, 2018-March(Enase 2018)</i> , 241–248. https://doi.org/10.5220/0006694502410248
S79	*Tang, M., Jin, J., Liu, Y., Li, C., & Zhang, W. (2019). Integrating Topic, Sentiment, and Syntax for Modeling Online Reviews: A Topic Model Approach. <i>Journal of Computing and Information Science in Engineering</i> , 19(1), 1–12. https://doi.org/10.1115/1.4041475

S80	*Timoshenko, A., & Hauser, J. R. (2019). Identifying customer needs from user-generated content. <i>Marketing Science</i> , 38(1), 1–20. https://doi.org/10.1287/mksc.2018.1123
S81	*Tóth, L., & Vidács, L. (2018). Study of various classifiers for identification and classification of non-functional requirements. <i>Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)</i> , 10964 LNCS, 492–503. https://doi.org/10.1007/978-3-319-95174-4_39
S82	*Wang, C., Zhang, F., Liang, P., Daneva, M., & Van Sinderen, M. (2018). Can app changelogs improve requirements classification from app reviews?: An exploratory study. <i>International Symposium on Empirical Software Engineering and Measurement</i> . https://doi.org/10.1145/3239235.3267428
S83	*Wang, W., Feng, Y., & Dai, W. (2018). Topic analysis of online reviews for two competitive products using latent Dirichlet allocation. <i>Electronic Commerce Research and Applications</i> , 29(April), 142–156. https://doi.org/10.1016/j.eierap.2018.04.003
S84	*Winkler, J., & Vogelsang, A. (2017). Automatic classification of requirements based on convolutional neural networks. <i>Proceedings - 2016 IEEE 24th International Requirements Engineering Conference Workshops, REW 2016</i> , 39–45. https://doi.org/10.1109/REW.2016.16
S85	*Yang, B., Liu, Y., Liang, Y., & Tang, M. (2019). Exploiting user experience from online customer reviews for product design. <i>International Journal of Information Management</i> , 46(May 2018), 173–186. https://doi.org/10.1016/j.ijinfomgt.2018.12.006
S86	*Zhou, F., Ayoub, J., Xu, Q., & Yang, X. J. (2020). A machine learning approach to customer needs analysis for product ecosystems. <i>Journal of Mechanical Design, Transactions of the ASME</i> , 142(1), 1–14. https://doi.org/10.1115/1.4044435

Chapter 5. A data-driven qualitative data analysis framework

5.1. Introduction

Health data frequently consist of written statements, responses, and reports and require word-by-word or line-by-line analysis, which is a time-consuming and skilled process of sense-making. Text-based analysis of health data refers to the process of studying and analyzing qualitative text data (Guetterman et al. 2018), which includes written text derived from open-ended surveys, patient complaints, discussions, discourses, dialogues, and interviews (Lacity and Janson 1994). Technological advances such as emails, text messages, comments, and the increasing use of electronic records and reports present a plethora of text-based digital data that may remain unattended or under-utilized. Text-based digital data and its analysis provide an opportunity for health care professionals to engage with patients in real-time dialogue-based communication, which has the potential to expedite assessing health concerns and providing prompt health recommendations (Elbattah et al. 2021).

Traditionally, text-based data analysis is a time-consuming process carried forward manually by qualitative researchers. Artificial Intelligence (AI)-based text analysis, often referred to as natural language processing (NLP) and machine learning (ML), demonstrates capabilities of the automated process of textual data analysis by deriving meaningful inferences from patient interviews (Rustan and Hasriani 2020) and patient authored textual data (Dreisbach et al. 2019).

Applying AI-based methods to assist text-based health data analysis can reduce the analytic workload of qualitative researchers and produce instant results, which empower real-time dialogue-based communication between healthcare providers and patients (Chen et al. 2018).

This study explored the use of AI-based algorithms in automating qualitative text-based data analysis to reduce the workloads of analysts. The objective of the study is to 1) present a framework empowered by AI-based methods to analyze interview transcripts and 2) evaluate the usability of the presented framework and its methods by comparing the results of manually analyzed findings of the same text.

In this study, we aim to combine text mining technology with qualitative research methods to reduce qualitative researchers' workload by providing automated time-saving analytic support to various coding procedures, qualitative decision-making processes, and summative outputs. The purpose framework takes raw interview transcripts as input and extracts key themes from large amounts of text.

The rest of the chapter is structured as follows: 5.2 reports the research data and related study design; section 5.3 illustrate our case study and the final results; section 5.4 provides the discussion and lesson we learned from this study; section 5.5 points out the potential issue and limitation that we encountered, and the section 5.6 is the conclusion.

5.2. Study design

5.2.1. Research data

Seven shopping mall managers' in-depth interview transcripts from a previous study served as qualitative data for this study. The interviews were originally conducted in 2019-2020 to explore mall managers' barriers and facilitators in developing and implementing mall walk programs in

Alberta, Canada. All the interviews were conducted in the English language, recorded digitally, and transcribed verbatim after each participant provided written consent based on ethical approval from the Conjoint Health Research Ethics Board (CHREB) at the University of Calgary.

The interview questions were focused on understanding mall managers' perspectives on what would help or hinder them in leading or supporting mall walk programs. Yet, the questions and their responses were non-standardized and frequently changed during the conversations and between the interviews thereby, generating unstructured text-based data. In total, there are seventeen questions are prepared for the interview including six types of major research interest from the interviewers, which are *the mall manager's role, location of the mall, mall's business, the community of the mall, the mall manager's interests in developing an in-mall physical program, and sustainable long-term activity support from the mall manager.*

There were seven different mall managers interviewed, with recordings being further transcribed by professional transcriptionists into plain text. In the interview transcripts, there are an average of 31 rounds of conversation.

5.2.2. Study framework

The coding process involves inductive reasoning, where the qualitative researcher needs to read the data line by line and assign a code to each chunk of words. The nature of coding is to identify concepts, and similarities in the data (Chun Tie, Birks, and Francis 2019). However, working with large amounts of data can be challenging for qualitative researchers in terms of concept extraction, comparative and contrastive analysis, and coding. This article proposes a framework (as Figure 30) that will assist qualitative researchers in the initial coding stage of grounded theory in healthcare research. The input for the proposed framework consists of plain text transcripts of the interviews,

while the output is a summary of key themes derived from the interview transcripts. The framework contains five major steps, which are 1) data preprocessing; 2) feature extraction; 3) text clustering; 4) keyword extraction; and 5) interpreting. Raw interview transcripts provide the input into the framework, and the outputs are the main topics identified from the given qualitative data.

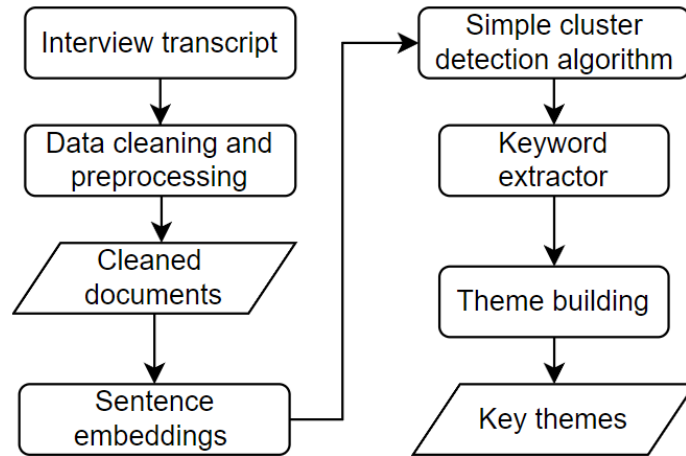


Figure 30. The proposed framework for qualitative coding.

i) Data cleaning and preprocessing

Data cleaning is a general approach that simply inspects samples from the given dataset and decides which information should be kept and which information should be omitted. The text data preprocessing is to prepare the text document into a more digestible form for the following tasks. The input of this step is multiple interview transcripts that are written in plain text, and the output is cleaned and normalized sentences. A raw interview transcript follows a standard interview flow that contains a set of questions from the interviewer and the corresponding answers from the interviewee (Figure 31). Each turn/answer is composed of one or multiple sentences. The study framework analysis the answers from the interviewee in sentence granularity. Hence the output of the preprocessing step needs to be a set of cleaned individual sentences. In our study, the

preprocessing step includes data cleaning, special character and marks removal, sentence segmentation, contraction mapping, case folding, and tokenization.

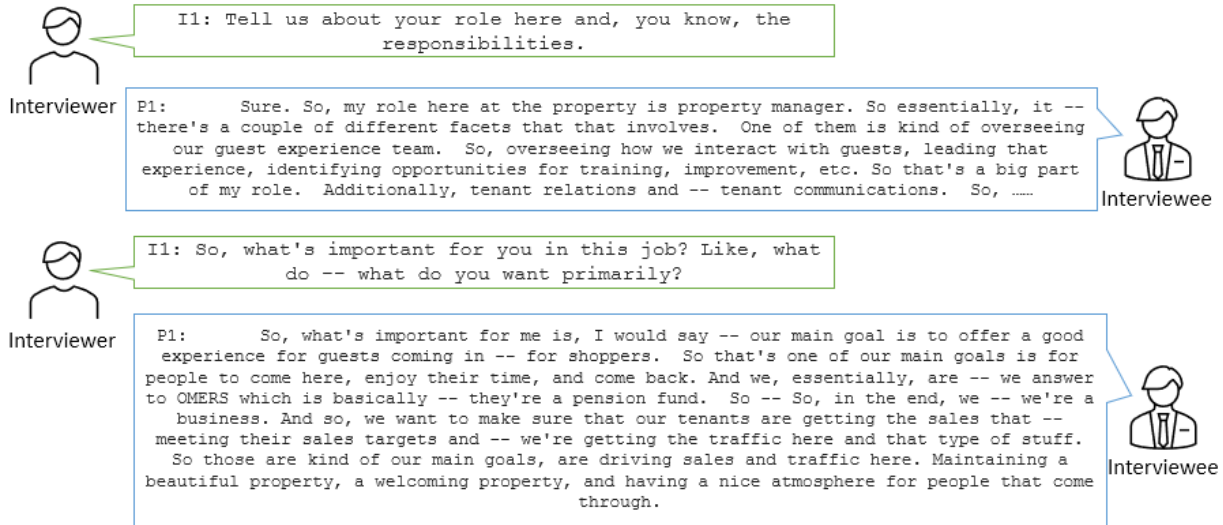


Figure 31. Sample of the interview transcript.

ii) Feature extraction

We extracted used both embedding features and TF-IDF features for text clustering and concept extraction algorithms. The pre-trained Sentence-BERT (SBERT) model (Reimers and Gurevych 2020) from Sentence-Transformer⁸ is a modification of the pre-trained-BERT model. With the pre-trained SBERT model, each sentence is represented with a 784-dimensional dense vector, and in this article, this feature will be referred to as sentence embeddings. The sentence embeddings will be input into the downstream clustering algorithms to extract semantically related sentence groups.

iii) Custom key concept extraction algorithms

⁸ <https://www.sbert.net/>

The ultimate goal of this study is to extract core concepts or themes from the given interview transcript. Hence, the goal of text clustering algorithms is to detect major clusters from the documents. A cosine similarity-based algorithm was built to extract major groups from the given files (Algorithm 1). Cosine similarity can reflect how close two given vectors are by measuring angles between two vectors. The cosine similarity between two vectors A and B can be illustrated as formula (1). The detailed algorithm's implementation is described as Algorithm 1 below. The algorithm takes sentence embeddings as input and extracts the most coherent groups according to the rank of the cosine scores. There is only one hyperparameter required, which is the group size to indicate how many sentences we want in each cluster.

$$\text{cosine similarity} = \frac{\sum_{i=1}^n A_i B_i}{\sqrt{\sum_{i=1}^n A_i^2} \sqrt{\sum_{i=1}^n B_i^2}} \quad (1)$$

For a large amount of qualitative data analysis, even if similar sentences are grouped, it is still a huge amount of work to read the sentences in each cluster line by line to understand the meaning of each cluster. To reduce the workload of experts reviewing the transcripts of each cluster, the proposed framework contains a keyword extraction algorithm that can extract keywords according to their occurrence. Using a list of keywords to represent each cluster can support experts to navigate the scope and domain of the cluster.

Using the sentence clustering algorithm, the semantically related sentences are grouped. The next concern is to extract several most representative words and phrases to represent each group. A TF-IDF-based key concept extractor has been developed which can simply extract words and phrases with a high TF-IDF score. The higher the TF-IDF score means the more representative the given

document (Jurafsky and Martin 2018). Top-n keywords are selected to represent each cluster, and in our experiment, we choose 5 keywords for each category

iv) Interpreting

From the previous step, there are several clusters, and keywords are extracted. With the help of sentence clusters and keywords, in this step, the qualitative researcher can manually assign topics to each cluster.

Input	$E \leftarrow$ Embeddings; $T \leftarrow$ threshold; $M \leftarrow$ group size; $N \leftarrow$ Number of sentences
Output	<i>unique_clusters</i> : Clustered sentences
Initialize	<p><i>Let similarity_scores</i> be an $N \times N$ Matrix. <i>Let desceding_scores</i>, and <i>desceding_index</i> be $N \times M$ Matrices. <i>Let clusters</i>, <i>unique_sentence</i>, and <i>unique_clusters</i> be empty lists;</p>
1	for $i = 1$ to N do
2	for $j = 1$ to N do
3	$similarity_scores[i][j] \leftarrow \text{cosine_similarity}(E[i], E[j])$
4	for $k = 1$ to N do
5	$desceding_scores[k], desceding_index[k] \leftarrow \text{rank}(similarity_scores[k]): M$
6	if last element of <i>desceding_scores</i> [k] $\geq T$
7	<i>clusters.add(desceding_index[k])</i>
8	for $c = 1$ to $\text{length}(clusters)$ do
9	$add \leftarrow \text{Ture}$
10	for $s = 1$ to $\text{length}(clusters[c])$ do
11	if s in <i>unique_sentence</i>
12	$add \leftarrow \text{False}$
13	if add is True:
14	for $s = 1$ to $\text{length}(clusters[c])$ do
15	<i>unique_sentence.add(clusters[c][s])</i>
16	<i>unique_clusters.add(clusters[c])</i>

Algorithm 1. Cluster detection algorithms

5.3. Experiment result

5.3.1. Data cleaning and preprocessing

a) Data cleaning

The interview data are stored in seven spreadsheet files (.xlsx format). First, after merging all the files into one document, the empty turns are removed from the transcript. In the interview transcript,

some turns only contain questions without any answers. In such a case we simply removed the turn from the transcript. The second step was to extract each interview turn from seven interview transcripts and store it as a CSV file. Each turn corresponds to one question and one answer. Next, we removed special characters from the dataset, such as escape characters and tags that are used to indicate the role of the speaker. In our case study, there are two types of special conditions required to handle. First, the transcriptionist added special tags starting with “I” or “P” before the interview question and answers, where “I” specifies the given content is from an interviewer, and “P” indicates the turn is from an interviewee. Second, due to the anonymity requirement, the identifiable details are masked in the interview transcript. For example, the name of a mall is masked as “[name of the mall]”. We applied a series of regular expression and dictionary-based algorithms to remove these special characters and mapped the identity mask to a regular format. Such as the text “*P1: Yeah. So I'm the marketing manager here at [name of the mall]*” would be mapped to “*Yeah. So I'm the marketing manager here at the mall*”.

5.3.2. Text preprocessing

Each turn is composed of one or multiple sentences. Based on our initial exploratory analysis, we decided to study at a sentence level. Sentence tokenization is a special case of text segmentation that refers to a technique to divide the written text into sentences (Manning and Hinrich Schütze 1999). After the sentence tokenization, there are a total of 2601 individual sentences in the transcript, with an average word count of 15, where the maximum word count is 84 and the minimum word count is only one, such as "sure", "yes" or "no". Word counts in most sentences are ranged from eight to thirteen words, and there are only 2.2% of sentences are containing more than forty-one words (Figure 32). In the initial analysis of the data, we decided to filter out

sentences with fewer than three words and more than forty words. After removal, there are in total of 2284 sentences left for the experiment.

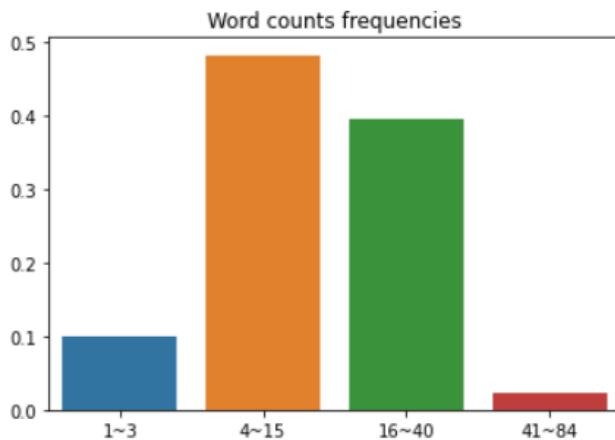


Figure 32. The frequencies of words count.

In addition, we applied several text normalization techniques to reduce the randomness, reduce the dictionary size, and increase the frequencies of certain terms. First, we built a dictionary-based algorithm to map most frequently appeared contractions into complete forms, such as “It’s” to “It is”, “I’d” to “I would”, and “we’ll” to “we will”. Second, we applied case folding a text-normalization task, that simply converts uppercases into lower cases. Third, we run WordNetLemmatizer from (Natural Language Toolkit) NLTK⁹ to lemmatize each word into its dictionary form. For instance, “walking”, “walked”, and “walks” become “walk” after lemmatization.

5.3.3. Feature extraction

The bidirectional transformers for language understanding (BERT) are regarded as one of the most significant milestones in the NLP community in recent years. BERT achieved the best results with eleven downstream tasks in NLP, and soon became the most widely used pre-trained language

⁹ <https://www.nltk.org/>

model in the field (Devlin et al. 2019). In this case study, we used a pre-trained language model as a feature extractor, which takes plain text as input and generates corresponding 768 embeddings as output. The CLS token appears at the beginning of each sentence and stands for sentence-level classification task (Devlin et al. 2019). The output related to the CLS token is H_{cls} embedding, which contains all the information from all the other words. Hence, the 768-dimensional H_{cls} embeddings can represent the whole meaning of the sentence, which can be used as sentence embedding (Devlin et al. 2019). However, according to Reimers & Gurevych (2020) directly using CLS from BERT output has a relatively lower performance compared to the average pooling of the BERT embeddings. Hence, instead of directly applying CLS embeddings, we decided to use the average value of each word embedding.

To extract sentence embeddings, we first applied the Tokenizer¹⁰ methods from the Hugging Face¹¹ library to complete multiple tasks to prepare the input for the following works. In addition, we adopted the pre-trained language model from Sentence-Transformers¹², which is specifically tuned for sentence-level text to get a more meaningful representation of each sentence (Song et al. 2020).

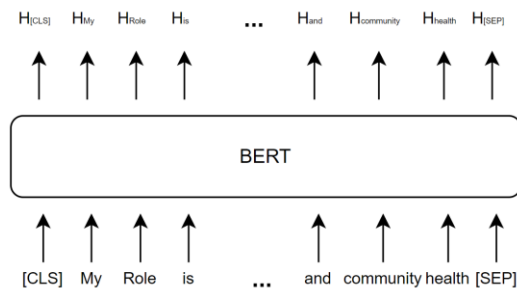


Figure 33. The inputs and outputs of BERT architecture.

¹⁰ https://huggingface.co/transformers/main_classes/tokenizer.html

¹¹ <https://huggingface.co/>

¹² <https://www.sbert.net/docs/publications.html>

5.3.4. Cluster detection and Keywords extraction

The simple cluster detection algorithm we introduce in section 5.3.2. requires two hyperparameters, which are cosine similarity threshold, and minimum group size. The cosine similarity threshold indicates the minimum similarity that two sentences are considered as the same group. In addition, the minimum group size is another threshold that decides how big a group can be considered a cluster. The goal of this framework is to identify core themes from interviews, rather than grouping all semantically similar sentences into the same group.

The clustering algorithms take feature vectors of the sentences from the previous step as input and group the vectors according to semantic similarity. If the cosine similarity score is greater than the hyperparameter "*similarity threshold*", the two sentences are considered to belong to the same group, and a group is identified as a cluster when the number of sentences it contains exceeds the "*minimum group size*".

Once the clusters and their containing sentences are identified, the next step is to extract key concepts to represent the topic. The qualitative researcher can easily explain the key themes of each cluster using the key concepts obtained from each cluster. The key concepts extraction framework is an iterative human-in-loop procedure, which requires the expert with domain knowledge to evaluate the output and adjust the hyperparameter accordingly.

In our experiment, we set the cosine similarity threshold as 0.6, and the minimum group size as 15. As a result, the cluster detection algorithm returns 15 clusters. After we applied the keyword extraction algorithm, five keywords were generated for each cluster to explain its core themes. The extracted keywords are in Table 18 to Table 20.

Cluster	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5	Cluster 6
Keywords	<ul style="list-style-type: none"> • Community • Business goal • Well established community • Marketing team • Surrounding community 	<ul style="list-style-type: none"> • Retailer • Marketing • Store • Partnered • Department 	<ul style="list-style-type: none"> • Accessibility • Location • Community • City • Come here 	<ul style="list-style-type: none"> • Walking • Program • Walking program • Started walk • Shopping center 	<ul style="list-style-type: none"> • Support community • Partnership • Closely • Work close • Charity • 	<ul style="list-style-type: none"> • Responsibilities • Manager • Marketing manager • Property manager • Leasing manager
Expert's interpretation	Connecting community	Retail and business	Accessibility	Walk program	Support community	Mall manager

Table 18. The experiment's results.

Cluster	Cluster 7	Cluster 8	Cluster 9	Cluster 10	Cluster 11	Cluster 12
Keywords	<ul style="list-style-type: none"> • Event • Time • Here • Strategy • Lull • 	<ul style="list-style-type: none"> • Yoga • Pilates • yoga class • Zumba • Instructor • 	<ul style="list-style-type: none"> • Shopping center • Mall situated • C train • Across street • Community surrounding 	<ul style="list-style-type: none"> • Big liability • Managers • Security problem • Senior Overseing 	<ul style="list-style-type: none"> • Diverse • Different • Awesome • City • Culture 	<ul style="list-style-type: none"> • Physical • Physical activity • Activity • Activity just • Gym
Expert's interpretation	Events and activities	Fitness class	Mall location and transit	Liability and security	Cultural diversity	Physical activity

Table 19. The experiment's results (Cont).

Cluster	Cluster 13	Cluster 14	Cluster 15
Keywords	<ul style="list-style-type: none"> • Charity • Work • Community • Nonprofit organization • Library 	<ul style="list-style-type: none"> • Marketing • Medium • Insta famous • Social medium • leasing • 	<ul style="list-style-type: none"> • Dog community • Bring dog • Trick or treat • Certain event • Friendly
Expert's interpretation	Cluster 13	Cluster 14	Cluster 15

Table 20. The experiment's results (Cont).

As the result, a total of eight themes are extracted through the framework, which are “Community”, “Retail”, “Accessibility”, “Mall-walk”, “Shopping center”, “Mall manager”, “Event”, and

“Fitness”. As we introduced in section 5.2.1, in designing the interview questions, the researcher deliberately chose the following six topics as the main interview content (as Table 21).

Research interest from human experts	Related theme extracted
<i>Mall manager’s role</i>	Mall Manager (cluster 6)
<i>Location of the mall</i>	Location and accessibility (Cluster 3), mall location and transit (cluster 9)
<i>Mall’s Business</i>	Business goal connecting community (cluster 1), retail, and business (cluster 2),
<i>Mall & Community</i>	Location and accessibility (Cluster 3), support community (cluster 5), charity and NGO (cluster 13)
<i>In-mall physical program</i>	Walk program (cluster 4), events and activities (cluster 7), fitness class (cluster 8), physical activities (cluster 12)
<i>Sustainable long-term activity support from the mall manager</i>	Walk program (cluster 4), events and activities (cluster 7), fitness class (cluster 8), physical activities (cluster 12)

Table 21. The comparison between research interests and extracted themes.

5.4. Discussion

With the development of artificial intelligence, the most tedious procedures in healthcare qualitative text analysis tasks can be partially replaced or supported by machines. However, the nature of qualitative research never changes, and human experts are still needed to observe and feel the subject’s thoughts, tendencies, feelings, and emotions about a particular topic, theme, or even a word. So, do machine learnings and artificial intelligence have a place in qualitative research? The answer is positive. We believe that AI techniques can support healthcare qualitative research in the following aspects. First, ML and AI techniques can aid qualitative research in the preparation stage. As an example, neural question generation technology has a great deal of potential to assist qualitative researchers in the preparation of interview questions (Zhou et al. 2018). In addition, the machine learning methods can capture the interviewee's mood and attitude

through the analysis of video, voice, and notes during an interview, which can be used as an important reference for qualitative analysis (Fan et al. 2016; Raschka and Mirjalili 2019; Schuller and Schuller 2021). Third, as the framework proposed by this article, ML and AI technologies can help healthcare researchers with coding when analyzing qualitative data, as well as with topic generation.

Qualitative research related to healthcare is a complex activity involving multiple steps and details (Chapman, Hadfield, and Chapman 2015). However, the above-mentioned ML and AI solutions can only support one or a few tasks in qualitative research. Furthermore, deep learning is notorious for its computational complexity, so it is impractical to use multiple neural network-based approaches in one qualitative research. Hence, research and studying how to integrate ML solutions and design a high-quality qualitative research support system in the healthcare domain is essential.

5.5. Limitations and future works

Several limitations can be found in this study, which we perceive can be resolved with further investigation of methodological development. The first limitation comes from pre-trained contextualized word representation. The selected word embedding method is pre-trained for the general-purpose down streaming task. However, due to the insufficient amount of qualitative data available for this study, we did not apply any fine-tuning procedure for domain adaptation. Although the clustering results are informative and intuitive, the performance could be better with customized fine-tuning. The second limitation comes from the evaluation of the purposed method. Since clustering methods lack labels, there is no standard method for evaluating them. Several measures have been proposed for calculating the goodness of text clustering, including manual

evaluation of results and extrinsic evaluation, which are applicable for most cases (Chang et al., 2009; Wallach et al., 2009). Nevertheless, manual analysis of clustering results would add to the qualitative researchers' workload and consume more time, which is contrary to the purpose of this study. To close such a gap, future works should include an interactive visualization user interface and a well-designed system to aid researchers in adjusting the hyperparameters (similarity threshold, and minimum group size) and interpreting the results.

Moreover, unlike conventional text clustering algorithms, the proposed work detects potential themes from the qualitative narrative text, rather than allocating all the data to different clusters. As a result, at this stage, the method cannot automatically code or suggest codes to qualitative researchers. Supervised text classification could be a solution for auto-coding. Since the topics are extracted, these extracted topics can work as supervised labels for text classification tasks. Due to the characteristic of qualitative coding, transfer learning, and few-shot text classification has a vast potential for qualitative healthcare research coding (Geng et al., 2020; Howard & Ruder, 2018; Yogatama et al., 2017).

Furthermore, as we discussed RQ3 in section 5, the textual data analysis and mining in health research contain various sub-categories, such as clinical research, epidemiological research, and health promotion research described in this chapter. As a result, there is a great deal of heterogeneity in the data from different medical studies; therefore, understanding the textual data, and deciding which pre-trained LM should be applied is important. One of the future directions is to introduce new algorithms to examine research datasets to determine whether generic pre-trained models or domain-specific models are better suited to the research dataset.

Last, qualitative research related to healthcare is a complex activity involving multiple steps and details (Chapman et al., 2015). However, ML and AI solutions can only support one or a few tasks in qualitative research. Since deep learning is notorious for its computational complexity, it is impractical to use multiple neural network-based approaches in one qualitative research. Therefore, integrating ML solutions and designing a qualitative research support system to support health care research is crucial.

5.6. Conclusion

In this chapter, we first examined interview transcripts from a real-world health promotion project in Alberta, Canada, which included seven interviews between healthcare researchers and mall managers. In our analysis, the terms used in the interview transcripts are relatively generic, which is not surprising given the fact that most interviewees do not have the necessary healthcare expertise to participate in these projects. Hence, based on the characteristic of our data, we applied pre-trained BERT, which was trained with general articles. With the BERT model, the embeddings containing sentence information can be extracted to represent each sentence. Then, a Cosine similarity score-based cluster detection algorithm is introduced to identify important themes that occur frequently. The degree of similarity and frequency of occurrence are controlled by two hyperparameters of the algorithm as thresholds. As a result, the method extracted twelve mutually exclusive groups. Based on the extracted groups, we applied a TF-IDF score-based keyword (or phrase) extraction algorithm to extract candidate themes, which are the most representative terms to represent each group. The healthcare researchers can easily assign a final theme based on the extracted candidate themes. The extracted themes cover all of the topics that healthcare specialists are interested in while designing the interview questions. The experiment results met the objective of using AI to support qualitative research, particularly when dealing with large amounts of data.

It is also useful to aid qualitative researchers in expediting their theorizing process by reducing the analytic workload.

To conclude, AI-based technologies such as the combination of contextualized text representation techniques, clustering methods, and keyword extraction methods can help qualitative researchers by initiating summative topics on large text documents. The proposed AI-based candidate theme extraction method is significantly effective for analyzing text-based data at the level of line-by-line initial coding and their categorization in focused coding. However, due to ethical and legal considerations, AI should provide sufficient support through recommendations and proposals, rather than making any decisions in human-related research.

Chapter 6. A Design Science Enabled Project-Specific Semantic Networks Construction Framework

6.1. Introduction

Design is a knowledge-intensive activity, which usually involves knowledge from more than one discipline. The nature of the design is to select and integrate atomic knowledge from a finite set of knowledge sources to build candidate design solutions and use principles from design science to select the best design solution to satisfy the design problem (Yong, 2011b). Thus, an understanding of the meaning of concepts is necessary to use, represent, manage, infer, and reason design knowledge.

Engineering design knowledge is a compound knowledge that is closely connected with various domains such as engineering, business, aesthetics, healthcare, etc. (Tatlisu & Kaya, 2017). The compound characteristic of the knowledge brings challenges for communication, collaboration, and knowledge sharing between stakeholders from different backgrounds. Further, the tacit nature of design knowledge introduces a further barrier to engineering design (Wong & Radcliffe, 2000). As a result, designers need to frequently search for information to quickly acquire the knowledge needed for their projects. Graph-based knowledge such as knowledge graphs and semantic networks have a great research value in this regard because they can be used to model, organize, infer, and visualize knowledge (Yoshioka et al., 1998).

A knowledge graph refers to a large network that represents concepts and relationships between them in a linked data structure (Ehrlinger & Wöß, 2016; Fensel et al., 2020). The linked data structure connects concepts in the same context, which facilitate knowledge and information integration, and generate explicit knowledge representation for both human and machine. The term Knowledge Graph has received considerable attention since it was introduced by Google as part of the Search Engine Optimization (SEO) solution (Singhal, 2012). In recent years, this type of approach is widely valued in knowledge-focused fields, such as healthcare, education, ICT, chemistry, biology, engineering, etc. (Abu-Salih, 2021). Research from the engineering design field has also been successful in yielding noticeable achievements on various knowledge graph-related tasks (Han et al., 2022).

With the development of different research fields, domain-specific terms, concepts, and keywords arise, and these terms, acronyms, and jargons become part of the engineering design knowledge barrier since its multidisciplinary characteristic. To close the gap by automated acquisition and modeling of lexical design knowledge, semantic networks are widely adopted by the engineering design domain in recent years (Geum & Park, 2016; He et al., 2019; Sarica et al., 2019, 2020). A semantic network is a lexical knowledge graph in which every node represents a word, and each node represents a specific semantic connection between linked two nodes (Hu & Liu, 2004). The notion of semantic networks is not new to the field of computer science. The construction of a semantic network requires intensive expert involvement, which is costly and time-consuming.

We describe a novel approach RomNet for building engineering design semantic networks that are intended to be used for retrieval of design information, key phrase extraction, and various text-related down-streaming tasks. Our semantic network construction approach is based on several key techniques including Environment-Based Design (EBD), a design science text modeling

technology known as Recursive Object Modelling (ROM), and Neural Network Language Model (NNML) (Bojanowski et al., 2017; Zeng, 2008, 2020). The proposed approach contains 3 main modules which are: 1) Environment analysis; 2) Data collection; 3) RomNet construction.

The rest of the chapter is structured as follows: Section 6.2 introduces the related works and theoretical background of the proposed study; Section 6.3 describes the proposed approach; Section 6.4 demonstrates a case study with the proposed framework, and last, section 6.5 is a conclusion.

6.2. Related works

6.2.1. Design knowledge modeling

Design knowledge formalization, modeling, and representation are an ongoing research hotspot during the past few decades. For design knowledge modeling, ontologies are the most commonly applied frameworks, which conceptualize terms and the relationships among them with formal specifications (Gruber 1993; Noy and McGuinness 2001). The term “ontology” is a controversial concept, which was first proposed in the field of philosophy, and now it is extended to various domains such as Natural Language Processing (NLP) (Miller et al., 1993), design knowledge management (Štorga et al. 2010), and healthcare (El-Sappagh et al. 2018).

Lin et al. (1996) proposed a design knowledge model called requirement ontology to support requirement analysis by modeling the product knowledge into concepts and different types of relationships. With the help of the requirement ontology, the engineer can refine, trace, validate, and change the customer requirement in the product design. Štorga et al. (2010) introduce an ontology framework that is based on a taxonomy of product concepts including objects, processes, attributes, design attributes, propositions, quantities, and relations. Each category is further divided

into subcategories of refinement. Some works proposed ontology to model the design itself, such as FBS ontology (Gero and Kannengiesser 2014), generic design activity ontology (Sim and Duffy 2003), TIPS ontology (Fernandes et al. 2007), PDO (Catalano et al. 2009), and DSO (Rockwell et al., 2009; Rockwell et al., 2010). Ontology and a variety of models of knowledge were used in the early works of knowledge-based design systems, and it is required that domain experts participate in most of the design activities, such as acquiring knowledge (Dixon et al. 1987), knowledge graph construction (Yoshioka et al. 1998), and knowledge validation (O’Leary 1991). However, at that time due to low computational resource building, a semantic network is also expensive for a task. Recently, many works are coming out in this domain to introduce semantic networks for the design knowledge representation.

6.2.2. The EBD methodology

Environmental-based design (EBD) is a systematic design approach that incorporates different environmental factors within the design of the target product to generate the best design solutions. The EBD method is aimed at solving problems caused by the target product’s environment to achieve a balance between the environment and the product (Zeng 2011). In EBD, step-by-step instruction is provided, that guides designers toward designing a product with the minimum amount of knowledge. The EBD method significantly lowers the barriers to system design, allowing novice designers to generate design solutions without sufficient domain knowledge. As an approach to interdisciplinary design, the EBD methodology has been applied successfully in many different fields, such as aviation geometric, education, medical, and traditional product design (Tan et al. 2013, 2011; Yi, Deng, and Zeng 2014).

The EBD methodology includes three major activities: environment analysis, conflict identification, and solution generation (Zeng 2011). The environment analysis helps designers to define the current environment by identifying primary objects and the relations between them. The environment refers to a scope that contains everything related to the product, such as user, parts, materials, input, output, and operational requirements. The objective of the EBD analysis is to collect potential design knowledge related to the production environment. Conflict identification is a process to identify conflicts and dependencies between objects from product knowledge. The solution generation is to make a design decision to find the optimal candidate design solution to the design problem by integrating, extending, and synthesizing atomic design knowledge.

The EBD methodology explained the nature of design, and the relationships between design requirements, design knowledge, and design solution. The design is initiated with a valid design requirement that requirement is composed of multiple atomic design requirements. The atomic design requirements represent the user's expected individual product facts, which include feature, function, and aspect. In some cases, these product facts will not resemble those of the final product, since they are only viewed from the perspective of the user. The design on the other hand requires designers to understand the design requirements and decompose the requirement into several valid design questions. With the design questions, the design team can decompose, distribute, track, and validate the design tasks. The nature of design solution generation is to collect and integrate design knowledge to satisfy the design questions, and the nature of design knowledge is a combination of facts about the target product (Figure 34). Hence, the nature of design is to collect and intergrade atomic design knowledge to satisfy the design requirements. The explicit atomic design knowledge is mostly represented by textual data, such as textbooks, requirement specifications, or other publications.

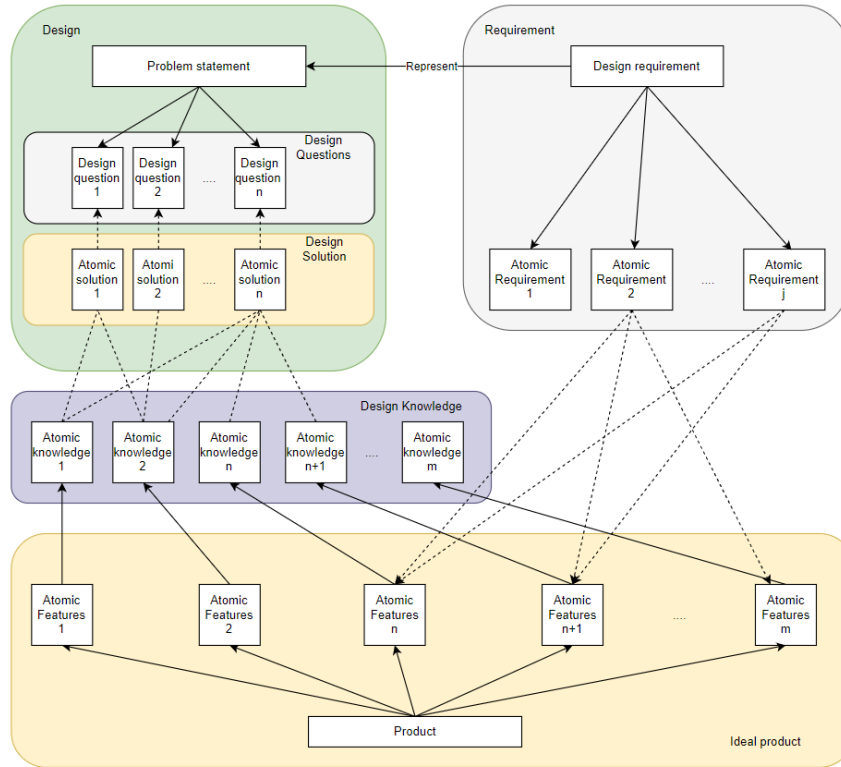


Figure 34. Relation between design, design requirement, design knowledge, and product.

6.2.3. Recursive Object Modeling (ROM) and keyword extraction

ROM is a graphical language model that models the linguistic dependency of words from a sentence (Zeng 2008). The ROM diagram follows the theoretical settings from EBD theory, which believes everything in the universe is an object, and there are relations between two objects (Zeng 2011). Similarly, for natural language, the universe is composed of individual words, and each word is an object in this universe.

For a sentence, the universe is bounded by the sentence, and the words from the sentence become objects in the sentence universe. The ROM diagram is a simple linguistic model that models the sentence-level lexical universe. There are three types of relations between objects in the ROM diagram, which are predicate, constraint, and connection (Zeng 2008).

ROM diagram is a directed cyclic graph, where the direction implies the relationships between objects. The predicate relation indicates an action, a constraint modifies an object, and the connections reflect two or more equally situated objects.

The ROM diagram is an important component in environment analysis by helping designers to identify important environmental components (objects) from the design requirements and clarify the relationships between these objects. For atomic design, the object is the basic building block for atomic design knowledge. Understanding the meaning of each object and the relationships between objects is crucial to understanding atomic knowledge.

6.3. Method

The overall RomNet Framework is applicable for both manual construction and automatic construction of project-specific engineering design semantic networks. The project-specific means the boundaries of a RomNet are scoped by a specific project that requires a semantic network. Unlike other semantic network construction strategies introduced in the previous section, RomNet utilizes EBD methodology during the construction process to analyze the environment of the target project. In general, the proposed framework is composed of three parts, which are 1) Environment analysis, 2) Data collection, and 3) RomNet construction, and the detailed procedure is illustrated in Figure 35.

6.3.1. Environment analysis

The environment analysis started with a statement that describes the desired product (Zeng 2020). The statement should be a description of the requirement, function, action, behavior, expectation, and relationships from the existing project.

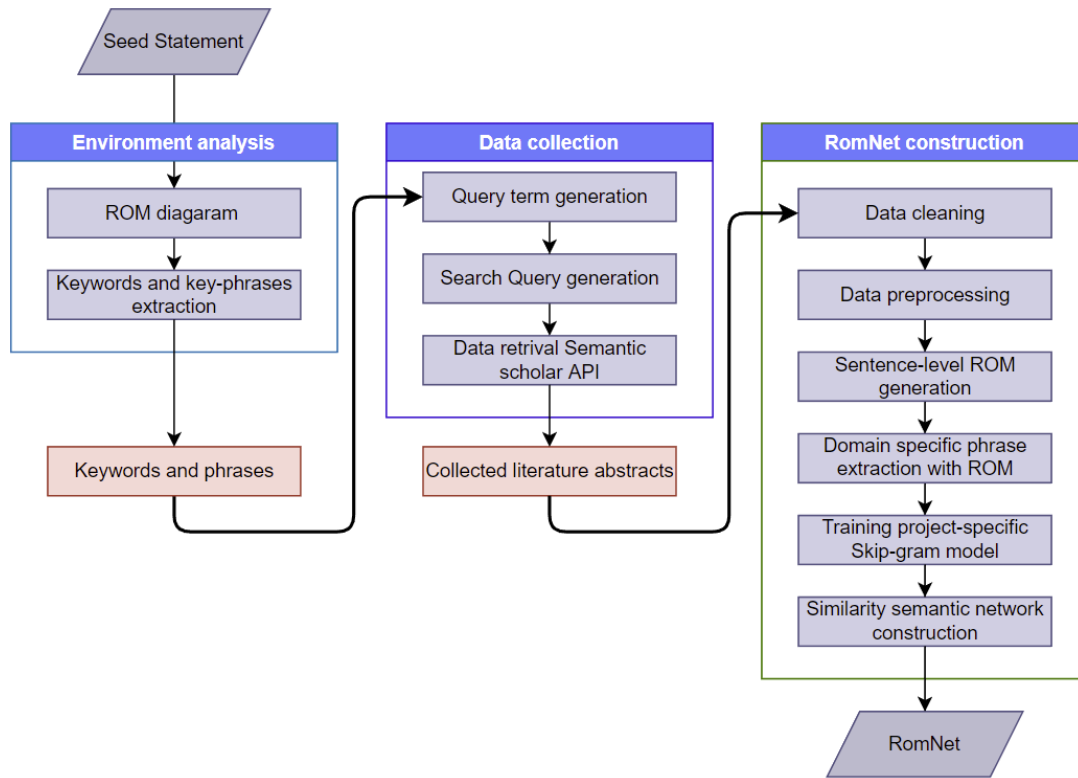


Figure 35. The RomNet framework.

Furthermore, the statement must be valid and accurately provided by an experienced engineer or expert from the target domain. The goal of an environmental analysis is to extract keywords for use in information retrieval. Based on the EBD analysis, a set of querying words can be extracted and prioritized based on their importance in the ROM diagram (Zeng 2020). Based on the existing ROM diagram generation tool named ROMA, which can transform the natural English text into a ROM diagram (Zeng 2008). ROMA can export ROM diagram into XML or JSON format, which contains both ROM objects and ROM relationships.

In the proposed method, we build a wrapper function called *romGenerator* that calls ROMA APIs to generate a ROM diagram for a given sentence. Using the ROM diagram as input, we developed a recursive algorithm to extract keywords and key phrases from the given sentence. The pseudocode of the algorithm is shown in Table 22.

INPUT	<i>S</i> ← <i>Seed Statement</i>
INITIALIZE	noun_set = {}, noun_phrase_set = {}, verb_set = {}, verb_phrase_set = {}
1	rom_diagram = romGenerator (<i>S</i>) //ROMA ROM diagram generation API
2	object_set = rom_diagram.obj // is a set that stores all the ROM objects
3	FUNCTION findModifier (obj)
4	IF type(obj) is <i>ROM Object</i> THEN
5	obj = List(obj) // List() is creating a list data structure
6	ELSE IF type(obj) is <i>Set</i> THEN
7	FOR neighbor in obj.last().neighbors // a.last() return last
8	IF neighbor.exist() THEN // element of list a
9	IF neighbor.pos is <i>NOUN OR ADJ OR RB OR VBN</i> THEN
10	obj.add(neighbor) // a.add(b) will add b at the end of list a
11	RETURN findModifier (obj)
12	ELSE RETURN obj
13	ELSE RETURN obj
14	ELSE RETURN obj
15	FOR obj IN object_set
16	IF (obj.pos is <i>NOUN</i>) THEN
17	noun_set.add(obj)
18	IF findModifier(obj).last() is NOT obj THEN
19	noun_phrase_set.add(findModifier(obj))
20	ELSE IF (obj.pos is <i>VERB</i>) THEN
21	verb_set.add(obj)
22	IF findModifier(obj).last() is NOT obj THEN
23	verb_phrase_set.add(findModifier(obj))
24	END
OUTPUT	noun_set, noun_phrase_set, verb_set, verb_phrase_set

Table 22. Pseudocode Algorithm for extracting keywords and key phrases.

In our study, the seed statement is provided by our industrial partner from Bombardier Inc., which is a leading business jet manufacturer in Canada. The seed statement is:

“The aircraft shall have the braking capability to stop the aircraft during all foreseeable landing conditions within the aircraft defined landing distance and runways”,

which describes the basic requirements of a braking system in aircraft design. Based on the seed statement the ROM diagram is generated in Figure 36.

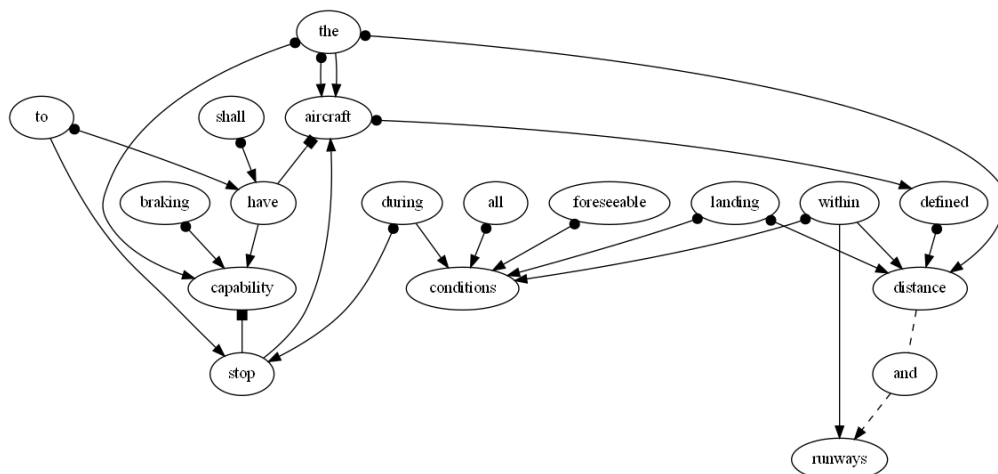


Figure 36. The ROM diagram for seed statement, generated with Graphviz¹³.

With the keyword extraction algorithms, we extracted keywords and phrases for the data collection step. The extracted keywords and phrases are aircraft, brake, land, runway, condition, aircraft braking capability, aircraft runway, aircraft defined runway, braking capability, foreseeable condition, landing condition, landing distance, and landing runway.

6.3.2. Data collection and preparation

The input of this step is a list of key terms, and the output of this step is a collected title-abstract dataset retrieved from Semantic Scholar API¹⁴, which allows users to search papers by keywords. For the search query generation, Query expansion is applied when there are not many search results returned for a basic keyword search. The common method includes replacing or adding the original term with the synonyms, antonyms, meronyms, hyponyms, and hypernyms from available thesaurus WordNet. In addition, a scoping term is needed when the keyword is too generic. Once, the queries for literature search are ready, by calling Semantic Scholar API, the RomNet data

¹³ <https://graphviz.org/>

¹⁴ <https://api.semanticscholar.org/graph/v1/paper/>

collection module would be able to retrieve the title abstract of related papers. The retrieved data is stored in a comma-separated value (CSV) file, which contains the following fields: title, abstract, year (published year), citation count, and fields of study. In total research work from 19 different research fields is retrieved. Among the extracted 12,713 papers, approximately 55% (n=6,993) are from the engineering domain (as Figure 37). The second field that stands out above the others is computer science (23%, n=2,927). The rest of the fields have a relatively lower distribution in our sampled research works. Once retrieved the abstracts of project-related papers are, the first step is to prepare the data into machine digestible format. Some of the articles from Semantic Scholar API does not contain abstract, so the first step of data cleaning is to remove the empty values. Because the retrieved results might contain non-English articles, we applied an off-the-shelf language identification tools solution named LangDetect¹⁵ to keep papers written in English only. The LangDetect library is a Naïve Bayes algorithm that returns a language prediction based on the sequence of spellings (character n-gram). In total, 42 samples are excluded due to missing abstracts or being written in a non-English language.

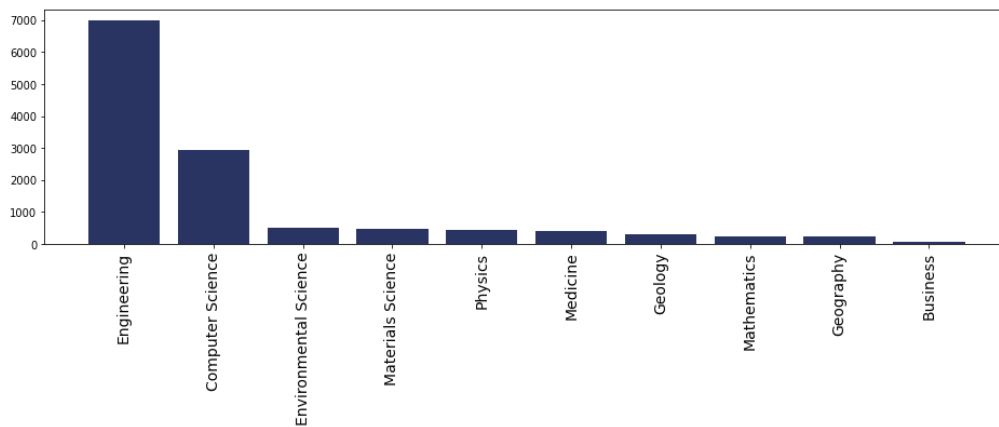
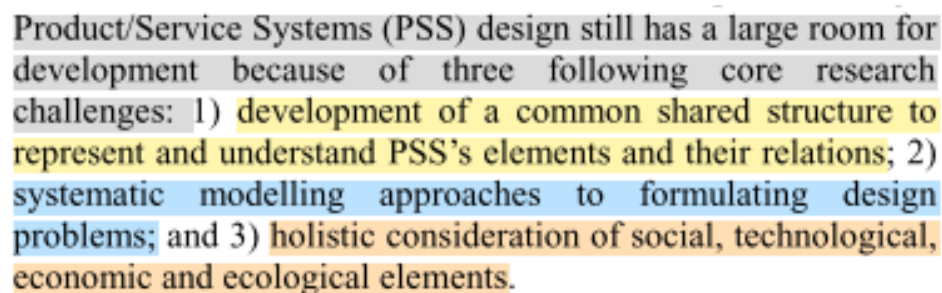


Figure 37. The distribution of top-10 filed of studies.

¹⁵ <https://github.com/fedelopez77/langdetect>

Given that the granularity of our study is at the sentence level, a third step would be to split all abstracts into separate sentences for constructing a sentence dataset. The activity to split a given text by sentence is also called sentence tokenization. For sentence tokenization, we applied Stanza, which is an open-source Python toolkit that supports multiple NLP tasks, such as tokenization, lemmatization, part-of-speech tagging, dependency parsing, and named entity recognition (Qi et al. 2020). The English NLP functionalities of Stanza are trained on the CoNLL-2003 corpus, which is based on the Reuter Corpus which is a collection of newspaper textual documents. However, the format and style of scientific abstracts differ from those of newspapers. For example, in scientific abstract, we detect a list of numbered items that are contained by a single large sentence (Figure 38). Although this type of large sentence is grammatically correct, due to its complexity and the EBD analysis atomic feature, we implement a rule-based method to break this type of long sentence into several chunks. We successfully identified 102,943 sentences, from the collected abstract as the result.



Product/Service Systems (PSS) design still has a large room for development because of three following core research challenges: 1) development of a common shared structure to represent and understand PSS's elements and their relations; 2) systematic modelling approaches to formulating design problems; and 3) holistic consideration of social, technological, economic and ecological elements.

Figure 38. An example to illustrate styles in scientific paper abstract

From the previous step, the collected abstracts are divided into individual sentences. Due to certain typesetting and formatting requirements of journals and conferences, some special characters may appear in the sentences. For example, the hyphen is widely used in research publications for

compounding multiple words (e.g., ‘self-rated’ in Figure 39), or using a hyphen to break a word at the end of a line (e.g., ‘men-tal’ in Figure 39).

Abstract This paper investigates the relationships between subjective rating measure and physiological measure of **men-tal** stress and mental effort during design activities. Mental stress and mental effort were captured by skin conductance and EEG beta2 power (20–30 Hz) whereas **self-rated** mental

Figure 39. Example of hyphen in the abstract.

We implement a simple Regular Expression (RE) based algorithm for the hyphen and dash replacement. In this algorithm, we applied an existing Python library PyEnchant¹⁶ for checking spelling, which can detect whether a given word is valid according to the backend English dictionary. The RE algorithm first detects whether a sentence is containing a hyphen, if any hyphen is detected from the sentence, the algorithm simply chops the word before and after hyphen into two pieces and sends them to PyEnchant to check the validity. When both parts are valid words, the algorithms replace the hyphen ‘-’ with an underscore ‘_’ to merge these two words as a phrase, otherwise, it eliminates the hyphen to restore the original word. The word with an underscore is usually treated as a single word by most tokenization algorithms. In addition to the hyphen, special characters (e.g., “/n”), and URLs, can be frequently found in the sentences; however, unlike hyphens, most of the special characters do not influence the structure, meaning, or logic of the sentences; accordingly, they are removed without further consideration.

After cleaning the sentences, the average sentence length is 23 words. The longest sentence consists of 819 words, whereas the shortest sentence consists of only one word. The frequency

¹⁶ <https://pyenchant.github.io/pyenchant/>

distribution of the word counts of the extracted sentences can be fitted to a normal distribution (as Figure 40) with a mean of 23.26 and a standard deviation of 10.98. For the purposes of ensuring the quality of the input sentences, we simply sampled the sentences within 2 standard deviations of the mean, which includes sentences containing more than 3 words and less than 43 words. Thus, 97,487 sentences (94,7%) retained for the following step.

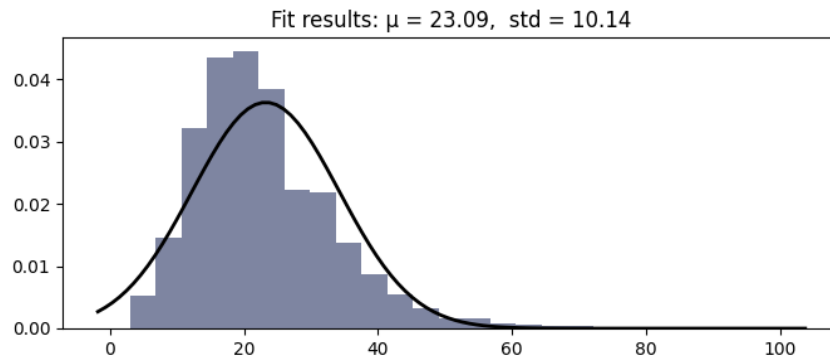


Figure 40. The word count distribution of sentence dataset.

6.3.3. Rom-based Phrase extraction

There are four main activities during the RomNet construction phase, namely ROM diagram generation, domain-specific phrase extraction, fine-tuning of the language model, and building the semantic network. As we introduced in section 6.3.1, the ROM diagram is extracted with ROMA3. In this project, we developed a wrapper function that can invoke ROMA3 API requests to generate ROM diagrams and to receive and store the ROM diagram into an XML file as the output. From 97,487 sentences, 86,726 ROM diagrams were successfully generated, representing approximately 89% of the completed sentence list. The majority of failed sentences are sentences with less than five words, which usually consist of headings, subheadings, phrases, and sequences of words that are not structured in a grammatically correct manner.

With the generated ROM diagram, the next step is to extract a set of key concepts from the collected data. At the EBD analysis phase, a recursive algorithm for keywords and phrase extraction is introduced, in which nouns and verb phrases are recursively found in a ROM graph and continuously added to the phrase list until all nodes have been traversed. It begins by listing all nouns and verbs in the given ROM diagram. The algorithm then searches for other objects that modify the objects in the noun and verb lists.

The modifying relationship is described as “constraints” in ROM. For example, in Figure 41, object 2 and object 3 are the constraint of object 1, and object 4 is the constraint of object 3. The recursive algorithm would start from node 1 to traverse the graph based on the Breadth-First Search (BFS) strategy. As the output, the algorithm will generate a list containing nouns, verbs, noun phrases, and verb phrases.

For example, “fast” and “unmanned” are the constraint of “aircraft”, and “Extremely” is constraining “fast”. As the output, the proposed algorithm extracts the following phrases: “extremely fast”, “fast aircraft”, “unmanned aircraft”, and “extremely fast aircraft”.

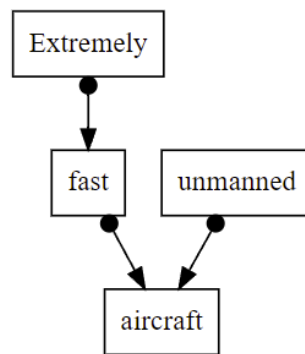


Figure 41. An example of constraint relations in ROM.

With the extracted phrases, the next step is to train a neural network language model to automatically transfer words and phrases with dense, meaningful vector representations which are

also noted as word embeddings in academia and industries. There are various popular neural network-based language models readily available. The latest state-of-the-art language models are based on transformer architecture, such as BERT (Devlin et al. 2019), GPT (Brown et al. 2020), and RoBERTa (Liu et al. 2019) which are much more complex than Word2vec models. These newly developed model can generate contextualized word embeddings, which determines the meaning of a given word by its context. In other words, these models can provide a more meaningful interpretation of a given sequence of text with an excessive amount of training or fine-tuning time. However, in our task, the objective of the training a language model is to learn the relationships between domain-specific terms, which meaning usually does not change with context, such as the concept “*flap setting*” in aerodynamic means a maneuver of a pilot, and the word “*flap*” usually means “*a part of aircraft wing*” instead of “*slap*”. Therefore, since our task is domain-specific and the terms are context-free, we selected the simple, efficient, and effective small neural network-based method word2vec for our language model.

Our proposed method applied the Skip Gram model, which is also known as one of the Word2vec embedding techniques introduced by Google in 2013 (Mikolov et al. 2013). Compared to other language models, this method takes less computational resources, but is faster to train, and produces excellent results (Rong 2014). Skip-gram model is a three-layered neural network that is composed of an input layer, one hidden layer, and an output layer. By employing skip-grams, neural networks are trained to use the center word to predict its surroundings words. The input center word is represented by one-hot encoding, with the context words within a window size treated as the output (as Figure 42).

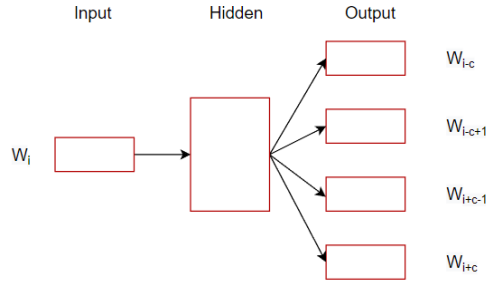


Figure 42. A simple Skip-gram algorithm illustration.

The objective of the model is to maximize the probability of predicting correct context words. Hence, the likelihood of the context words by given center word w_i with window size c and vocabulary size V can be illustrated by the following equation (1).

$$L(\theta) = \prod_{i=1}^V \prod_{\substack{-c \leq j \leq c \\ j \neq 0}} P(w_{i+j} | w_i; \theta) \quad (1)$$

When the length of the word window is C , the input would be a $C \times V$ dimensional matrix. The hidden layer is a neural network layer with N dimensions (N nodes) that also determines the feature size of the word embeddings. Then, after the calculation of the hidden layer, the data become a V -dimensional vector. Each entry of the vector represents a possibility of a word from the vocabulary, and after a Softmax function, the sum of the probabilities is normalized to 1. Through a process of iterative training and back propagation, the weights are eventually updated to their optimal value.

The proposed framework applied an existing NLP tool Gensim¹⁷, which is a Python library for topic modeling, information retrieval, and language modeling. Our hyperparameters were selected from a wide array of candidates and evaluated against a real-world truth set reviewed by an expert.

¹⁷ <https://github.com/RaRe-Technologies/gensim>

The truth set randomly sampled 457 pairs of aircraft-related terms from NASA thesaurus¹⁸, which is an authorized Ontology for information retrieval and document indexing in NASA Technical Reports Server (NTRS, also known as STI repository).

Hyperparameter	Candidate values
Window size (W)	3, 5, 7, 9
Minimum frequency	2, 5, 7, 10
Word embedding dimension	50, 100, 200, 300
Negative samples	5, 9, 14, 20
Learning rate	0.01, 0.015, 0.025, 0.05

Table 23. Hyperparameters for the skip-gram model selection.

The trained skip-gram language model is capable of converting a given term or concept into dense word embeddings. Based on the Cosine similarity algorithm the similarity or relatedness between two-word embeddings is calculated. The cosine similarity between two vectors U , and V is defined as follows:

$$\text{cosine}(U, V) = \frac{\sum_{i=1}^n U_i V_i}{\sqrt{\sum_{i=1}^n U_i^2} \sqrt{\sum_{i=1}^n V_i^2}} \quad (2)$$

Through the Cosine similarity algorithm, it is possible to quantify the relationship between two terms. Further, a Cosine similarity score can be used to determine the top n words related to a given concept. For example, Figure 43 shows the word vectors from the skip-gram model, which is related to the term ‘helicopter’. t-SNE (Maaten and Hinton 2008) algorithms are used for the

¹⁸ <https://sti.nasa.gov/nasa-thesaurus/>

visualization which is responsible for converting the original 300-dimensional word embeddings into 2-dimensional points.

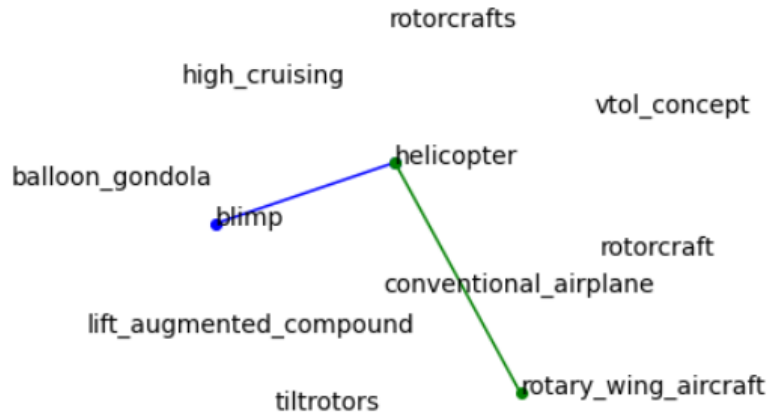


Figure 43. Vector visualization for words and phrases related to 'helicopter'.

6.3.4. Semantic network construction

The last step of the proposed approach is to generate a project-specific semantic network to assist engineers and designers in information retrieval. First, a ROM diagram based on the skip-gram vocabulary is generated (as Figure 44). Second, the keyword extraction algorithm is applied to identify the important concepts from the generated ROM diagram.

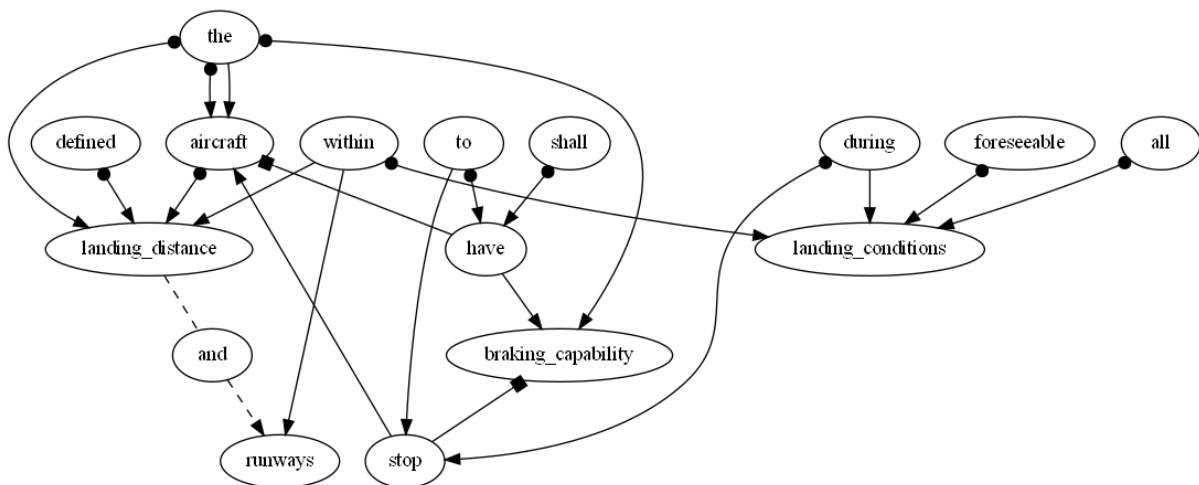


Figure 44. New generated ROM diagram for seed statement.

<i>Concept</i>	<i>Extracted concept</i>
Braking capability	realistic assessment, winter runway, landing airplane, pavement grooving, using ground based, landing controller, gear steering, towed measuring, energy recovery, braking availability, overall energy, antilock, gear test, dynamometer system, NASA, FAA, deformable contaminant, photoelectric, SR radar, landing deceleration
Landing conditions	kinematical parameter, wing tank, safety boundary, occupation time, sink velocity, hydraulic force, flap configuration, gear configuration, elastic aircraft, cruise condition, different angle, full scale model, approach condition, presented model, upwash angle, landing case, takeoff procedure, total noise, engine arrangement, possible range
Stop	towing concept, tail hook, confines, overrunning aircraft, dispatch, mail, electrostatic charge, aerospace engineer, gate capacity, iterative algorithm, anti-skid control, maybe, pulley, mined, shut, huge, cable, fly by wire aircraft, temporarily, slowing, arrestment, feathering, desired location
Landing distance	CRFI table, LDA, following characteristic, mission fuel, takeoff distance, simulation based approach, take off distance, spoiler position, flap position, Canadian runway, climb performance, information publication, domestic airline, field length, different criterion, temperature effect, deceleration equation, climb gradient, take-off run, reference handbook, accelerate stop distance, design mission, friction index, naval carrier, CRFI, ground performance, turn rate, aircraft range, landing length, rejected takeoff
Runways	taxiway, touchdown zone, lighting system, lighted, RWY, slippery, center line, holding position, hard surface, taxiing, practical solution, apron, rainfall intensity, big problem, overall throughput, taking off aircraft, occupation time, KSC, visual range, current position, touchdown area, drifted, computer calculation, localizer beam, arrival runway, undershoot overshoot, runway light, emergency facility

Table 24. The extracted concepts are related to the project statement.

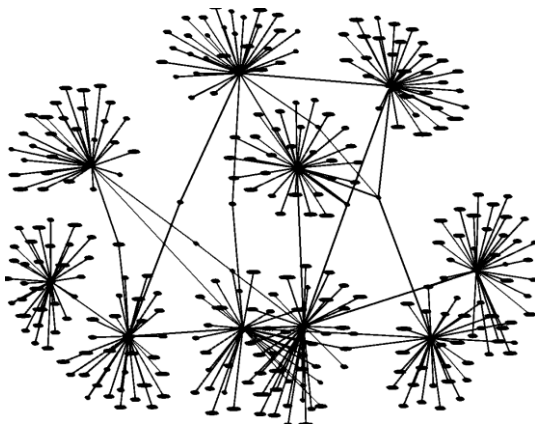


Figure 45. Semantic network for seed statement.

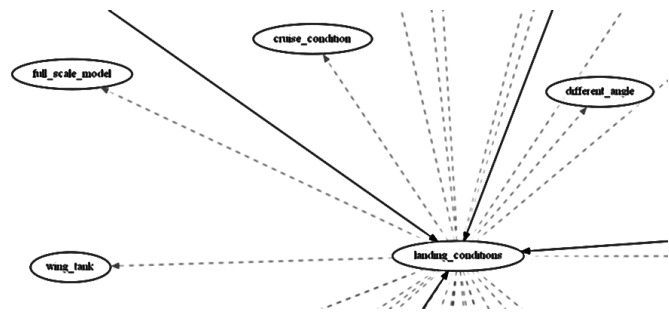


Figure 46. A zoomed part from the semantic network.

Thirdly, the proposed approach adds the top-N most relevant concepts to the statement, based on a comparison of the semantic similarity between the extracted key concepts and words from the skip-gram vocabulary. We extracted the top 30 concepts for each core concept from the seed statement with a domain-specific skip-gram model to construct the semantic network. Due to some

concepts having fewer related terms, we applied a threshold which is a normalized average semantic score to each group of related concepts. When the semantic similarity score exceeds the threshold, the term would be considered a related concept. The extracted results are shown in Table 24.

$$threshold = \frac{\sum_{i=1}^m \sum_{j=i}^n sim_{i,j} - \min(sim)}{m \times n(\max(sim) - \min(sim))} \quad (3)$$

Figure 45 shows the final RomNet for the given seed sentences, and Figure 46 shows a part of the RomNet, as the figure illustrated the concept “*landing condition*” is connected with solid lines and dashed lines, the solid lines indicate the ROM relationships, and the dashed lines connect the terms extracted by trained skip-gram model.

6.4. Discussion, limitations, and future works

In this chapter, the framework is presented in a methodological framework and illustrated with an actual project from the construction of aircraft semantic networks. The project team has no experience in aircraft design and does not have any prior knowledge of aviation terms. Based on the EBD design methodology and ROM tool, the RomNet framework can identify domain-specific keywords and phrases with relatively small data. This is entirely due to the sentence-level lexical knowledge provided by the ROM diagram. With the use of pre-defined relationships and part-of-speech of each term, the proposed algorithm can extract a set of potential keywords and phrases. In addition, when the sample size is insufficient, the ROM key phrase extraction method may result in a higher frequency of rare possible phrases compared to the traditional co-occurrence-based method. For innovative activities, especially for design in aviation, the number of available research publications is relatively fewer than in other domains. For example, when we search

“aircraft braking system” from the Web of Science search engine, only 487 results would be retrieved; however, when we change the query to “car braking system”, there are 2,115 related research returned. Hence, using the proposed framework for the identification of domain-specific phrases would help engineers and designers collect project-related concepts effectively and efficiently. Training a skip-gram word embedding model allows us to convert these extracted project-related terms and phrases into dense vectors, which allows the framework to compute the semantic relatedness based on the Cosine similarity score. With the project-specific language model, the initial ROM diagram for the seed statement can be expanded with semantically related terms from language model vocabulary. These extracted terms and RomNet semantic networks have a great potential for various down streaming tasks such as information retrieval, communication, and idea generation.

During the study, we identified several open issues and challenges that need to be addressed in the future. First, the study focuses solely on extracting project-specific terms and training language models to retrieve relevant terms but does not provide a method to provide the meaning for these terms. By combining available resources, such as aerodynamic ontology (NASA thesaurus), generic semantic networks (such as WordNet), open encyclopedia (Wikipedia), online dictionary API (such as Dictionary API), etc., both common sense and domain-specific definitions can be retrieved.

Second, in this work, we only focus on the semantic similarities between project-related terms. In addition to the similarity measurement, the embeddings trained on the project-related literature abstracts can express a certain degree of understanding of analogy. A famous example:

$$\text{“}vector(\textit{King}) - vector(\textit{Man}) + vector(\textit{Woman}) \approx vector(\textit{Queen})\text{”}$$

by Mikolov et al. (2013) perfectly illustrated the potential of the language model. In engineering design and design science studies, Design-by-Analogy (DbA) is a growing research field, which encourages designers to seek related areas, concepts, experiences, and artifacts to solve the design problem (Moreno et al., 2014). The word-based ideation is a proven effective design methodology in modern engineering design and using word embedding that is specifically trained on the domain-related publication has a great potential to support the early stage of the design idea generation. For example, when a designer wants to design a wing for a flying car, the analogy can be represented by “Aircraft + wing – car = ?”. Our trained skip-gram model inferred several concepts that might be related to the problem, such as “movable flap”, “jet flap”, “racing car”, “unconventional configuration”, “composite wing” and many others.

As part of the future work, we will continue to analyze design concept extraction from textual data and implement state-of-the-art architecture to represent design concepts accurately. Methods and frameworks can leverage vector semantics in DbA studies.

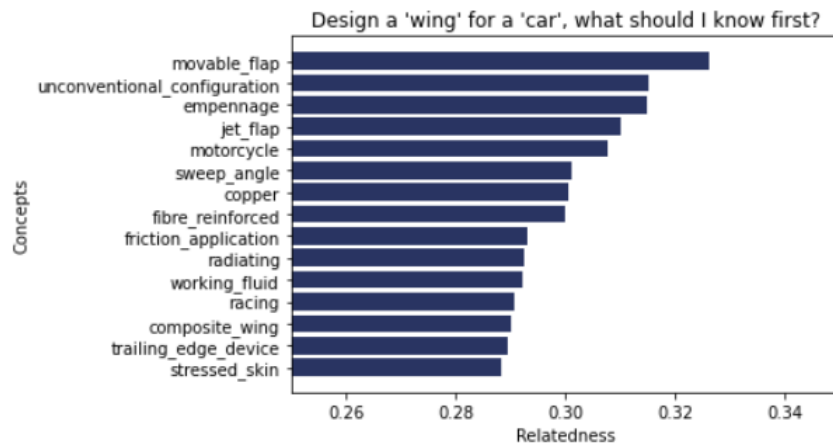


Figure 47. The related terms to the given analogy question.

Third, this chapter only focuses on technical feasibility and integration of NLP techniques and design science. However, due to the purpose of this study, this chapter is restricted to discussing

the application of down streaming. It is important to conduct follow-up studies in order to explain how engineers and designers will benefit from the RomNet, and how networks will be visualized to fill the knowledge gap.

Forth, there are many challenges associated with training new employees to be able to actively participate in existing roles (Akondy & Murthy, 2016). In light of the complex nature of organizational design knowledge, there are usually few beneficial structured training resources that cover the broadest range of concepts. Future works also can combine RomNet and EBD methodology to aid organization training and education by locating and retrieving project-related documents and information.

6.5. Conclusion

In this chapter, we proposed a design semantic network construction framework named RomNet. The framework leverages design science methodology EBD, design text analysis tool ROM, and neural network language model Word2vec techniques to extract key concepts from published paper abstracts related to the current project. The paper used a case study from the aviation domain, more specifically an aircraft braking system design project as a usage scenario to illustrate and explain the framework. The proposed RomNet is an automated framework that does not requires domain experts' supervision or collaboration during the entire lexical knowledge extraction and construction phases. In the case study, we use Semantic Scholar API for information collection. To collect information, we developed a simple rule-based algorithm and user interface that takes seed statements as input and automatically generates search queries and collects title abstracts from Semantic Scholar API. Based on the EBD methodology, the proposed key phrase extraction method can recursively search from the ROM diagram to extract concepts related to the design

project. By training a skip-gram model with the extracted phrases, the framework can compute and measure the interrelatedness between concepts extracted from the related publication abstracts. Last, by ranking the relatedness scores for a given term, the framework can construct a small semantic network for seed statements provided by experts.

Chapter 7. Conclusion

The content of this thesis includes 1) conducting a literature review on the current state of data-driven requirement elicitation; 2) proposing a Natural Language Processing (NLP) based qualitative data analysis framework, which could assist the researcher in extracting knowledge narrative data; 3) to propose an EBD-based lexical knowledge acquisition framework that helps designers to extract project-specific semantic network from available textual data.

Through these works, we have studied the nature of design knowledge and the relationships between design knowledge, design requirement, design solution, stakeholders, and the designer. The requirement reflects stakeholders' expectation, understanding, imagination, or, more broadly and abstractly -- knowledge of the desired product or system. Stakeholder ideas and opinions can be gathered through various sources, such as interviews, questionnaires, meetings, documents, social networks, and product websites. By analyzing these available sources, the designer can have an initial idea about desired product or system. In this thesis, the first two works studied the early stage of the design, more specifically, requirement elicitation and analysis and its influence on design. Through the literature review, we identified the most current data-driven methods that extract design-related knowledge from massive textual resources, such as user feedback, design specification, research articles, and other written materials. The data-driven methods help the designers to have a holistic overview of the product from big data; however, due to the nature of

the computational analysis, the results tend to be biased and sometimes misleading. Hence, applying a human-computer collaborative analysis of the requirement is required. The second proposed work identified such a problem and aimed to propose a method that can fully use ML and NLP techniques features and let the machine do the tedious, laborious tasks. In addition to ML/NLP pipeline, we introduced a supplementary pipeline that allows qualitative researchers to do manual inspection and quality control.

Apart from requirements, trustworthy resources from the internet also can provide valuable knowledge in the early stage of the design. The third study proposed a project-specific EBD-enabled design knowledge framework, which includes project-specific knowledge scoping, key concept extracting, knowledge retrieving, and design knowledge modeling. The extracted design knowledge is stored in a graph database (Neo4j in our case study), which can be used to support communication, collaboration, concept generation, and initial design ideation.

Design is complicated and knowledge-intensive. Researchers are trying to find a way to model design itself and other essential concepts in design, such as design knowledge, design techniques, and design tasks. Design, as one of the most natural human activities, sometimes happens without purpose. Hence, it is very challenging to model design activity correctly. Environment-based design (EBD) provides a complete theory about design, design modeling, and the designer's capability. The designer's capability is modeled as knowledge, skills, and affect. The knowledge and skills represent the designer's education and experience, and the effect refers to the designer's emotional state, such as confidence and belief. Hence, modeling and quantifying an individual's knowledge will greatly contribute to analyzing an individual's potential capability for a specific task, which helps in estimating the project design budget and schedule to project timeline.

One branch of our future works is related to design knowledge management, such as modeling individual design knowledge, organizational (or group, team) posed design knowledge, and managing design knowledge, including design knowledge retention and design knowledge education.

The other branch of our future work relates to EBD-based decision support system design. Through the first work (literature review), we identified that most works try to build an end-to-end artificial intelligence solution that takes whatever textual data they have as input and group, categorize, generate, summarize, or rewrite it. However, the drawbacks are obvious. First, the end-to-end machine learning algorithms are challenging to be debugged. When an error occurs, the system cannot be tuned as quickly as a rule-based system. Second, the end-to-end system generates design decisions without human permission, which makes design governance difficult. As a result, there would be potential liability issues occurs. Third, most current research related to NLP and ML in design studies uses classification and clustering methods to group similar requirements together, but only a few papers demonstrate what other follow-up procedures are required to complete the requirement analysis. As a result, most of the current research is not explicitly focused on providing alternative traditional demand elicitation solutions but on demonstrating how artificial intelligence can potentially play a role in demand engineering. Hence, to fill this gap, in the future, we will continue the design decision support system design, which combines both machine learning and human. The system should be a pipeline with multiple procedures instead of end-to-end and allow a human expert to override any possible checkpoint.

The second work introduces an NLP and ML-based system that supports qualitative research. However, the work only supports the initial coding stage by extracting the main topics from the interview transcripts. In the future, we will continue this study to expand the coverage of the ML

algorithms by analyzing the linguistic feature, semantic feature, and other potential features. In addition, the framework's current stage targets and supports the initial coding stage of the Grounded Theory (GT); in the future, interactive coding and theme-building tools should be built to support qualitative researchers. In this scenario, automated code recommendation, topic generation, and sample generation have great potential to support GT and other qualitative research activities.

References

- Abad, Zahra Shakeri Hossein, Oliver Karras, Parisa Ghazi, Martin Glinz, Guenther Ruhe, and Kurt Schneider. 2017. "What Works Better? A Study of Classifying Requirements." Pp. 496–501 in *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*.
- Abualhaija, Sallam, Chetan Arora, Mehrdad Sabetzadeh, Lionel C. Briand, and Eduardo Vaz. 2019. "A Machine Learning-Based Approach for Demarcating Requirements in Textual Specifications." Pp. 51–62 in *Proceedings of the IEEE International Conference on Requirements Engineering*. Vols. 2019-Sept. IEEE.
- Ahmad, Arshad, Chong Feng, Muzammil Khan, Asif Khan, Ayaz Ullah, Shah Nazir, and Adnan Tahir. 2020. "A Systematic Literature Review on Using Machine Learning Algorithms for Software Requirements Identification on Stack Overflow." *Security and Communication Networks* 2020. doi: 10.1155/2020/8830683.
- Ahmed, Saeema. 2005. "Encouraging Reuse of Design Knowledge: A Method to Index Knowledge." *Design Studies* 1–360. doi: 10.1142/7114.
- Ahn, EunJin, and Huyn Kang. 2018. "Introduction to Systematic Review and Meta-Analysis: A Health Care Perspective." *Korean Journal of Anesthesiology* 71(2):1–38.
- Al-Subaihin, A. A., F. Sarro, S. Black, L. Capra, M. Harman, Y. Jia, and Y. Zhang. 2016. "Clustering Mobile Apps Based on Mined Textual Features." in *International Symposium on Empirical Software Engineering and Measurement*. Vols. 08-09-Sept.
- Almfelt, Lars, Fredrik Berglund, Patrik Nilsson, and Johan Malmqvist. 2006. "Requirements Management in Practice: Findings from an Empirical Study in the Automotive Industry." *Research in Engineering Design* 17(3):113–34. doi: 10.1007/s00163-006-0023-5.
- Arora, Simran, Avner May, Jian Zhang, and Christopher Ré. 2020. "Contextual Embeddings: When Are They Worth It?" 2650–63. doi: 10.18653/v1/2020.acl-main.236.
- Asif, Muhammad, Ishfaq Ali, Muhamad Sheraz Arshed Malik, Muhammad Hasanain Chaudary, Shahzadi Tayyaba, and Muhammad Tariq Mahmood. 2019. "Annotation of Software Requirements Specification (SRS), Extractions of Nonfunctional Requirements, and Measurement of Their Tradeoff." *IEEE Access* 7:36164–76. doi: 10.1109/ACCESS.2019.2903133.
- Azadegan, Aida, K. Nadia Papamichail, and Pedro Sampaio. 2013. "Applying Collaborative Process Design to User Requirements Elicitation: A Case Study." *Computers in Industry* 64(7):798–812. doi: 10.1016/j.compind.2013.05.001.

- Baker, Cody, Lin Deng, Suranjan Chakraborty, and Josh Dehlinger. 2019. "Automatic Multi-Class Non-Functional Software Requirements Classification Using Neural Networks." *Proceedings - International Computer Software and Applications Conference* 2:610–15. doi: 10.1109/COMPSAC.2019.10275.
- Bakiu, Elsa, and Emitza Guzman. 2017. "Which Feature Is Unusable? Detecting Usability and User Experience Issues from User Reviews." *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017* 182–87. doi: 10.1109/REW.2017.76.
- Barbosa, Ricardo, Daniele Januario, Ana Estela Silva, Regina Moraes, and Paulo Martins. 2015. "An Approach to Clustering and Sequencing of Textual Requirements." Pp. 39–44 in *Proceedings - 2015 45th Annual IEEE/IFIP International Conference on Dependable Systems and Networks Workshops, DSN-W 2015*.
- Baxter, David, James Gao, Keith Case, Jenny Harding, Bob Young, Sean Cochrane, and Shilpa Dani. 2007. "An Engineering Design Knowledge Reuse Methodology Using Process Modelling." *Research in Engineering Design* 18(1):37–48. doi: 10.1007/s00163-007-0028-8.
- Bengio, Yoshua, Réjean Ducharme, and Pascal Vincent. 2001. "A Neural Probabilistic Language Model." *Journal of Machine Learning Research* 3:1137–55.
- Binkhonain, Manal, and Liping Zhao. 2019. "A Review of Machine Learning Algorithms for Identification and Classification of Non-Functional Requirements." *A Review of Machine Learning Algorithms for Identification and Classification of Non-Functional Requirements Expert Systems with Applications: X* 1. doi: 10.1016/j.eswax.2019.100001.
- Birch, David, Alvise Simondetti, and Yi ke Guo. 2018. "Crowdsourcing with Online Quantitative Design Analysis." *Advanced Engineering Informatics* 38(June):242–51. doi: 10.1016/j.aei.2018.07.004.
- Blei, David M., Andrew Y. Ng, and Michael I. Jordan. 2003. "Latent Dirichlet Allocation." *Journal of Machine Learning Research* 3(4–5):993–1022. doi: 10.1016/b978-0-12-411519-4.00006-9.
- Bourque, Pierre, and Richard E. Fairley. 2014. *Guide to the Software Engineering Body of Knowledge Version 3.0 (SWEBOK Guide V3.0)*. IEEE Computer Society.
- Brace, William, and Vincent Cheutet. 2012. "A Framework to Support Requirements Analysis in Engineering Design." *Journal of Engineering Design* 23(12):876–904. doi: 10.1080/09544828.2011.636735.
- Brereton, Pearl, Barbara A. Kitchenham, David Budgen, Mark Turner, and Mohamed Khalil. 2007. "Lessons from Applying the Systematic Literature Review Process within the Software Engineering Domain." *Journal of Systems and Software* 80(4):571–83. doi: 10.1016/j.jss.2006.07.009.
- Brown, Tom B., Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal,

- Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. “Language Models Are Few-Shot Learners.” *Advances in Neural Information Processing Systems 2020-Decem*.
- Canedo, Edna Dias, and Bruno Cordeiro Mendes. 2020. “Software Requirements Classification Using Machine Learning Algorithms.” *Entropy* 22(9). doi: 10.3390/E22091057.
- Carreno, Laura V. Galvi., and Kristina Winbladh. 2013. “Analysis of User Comments: An Approach for Software Requirements Evolution.” Pp. 582–91 in *Proceedings - International Conference on Software Engineering*.
- Casamayor, Agustin, Daniela Godoy, and Marcelo Campo. 2009. “Semi-Supervised Classification of Non-Functional Requirements: An Empirical Analysis.” *Inteligencia Artificial* 13(44):35–45. doi: 10.4114/ia.v13i44.1044.
- Casamayor, Agustin, Daniela Godoy, and Marcelo Campo. 2010. “Identification of Non-Functional Requirements in Textual Specifications: A Semi-Supervised Learning Approach.” *Information and Software Technology* 52(4):436–45. doi: 10.1016/j.infsof.2009.10.010.
- Casamayor, Agustin, Daniela Godoy, and Marcelo Campo. 2012. “Knowledge-Based Systems Functional Grouping of Natural Language Requirements for Assistance in Architectural Software Design.” *Knowledge-Based Systems* 30:78–86. doi: 10.1016/j.knosys.2011.12.009.
- Cascini, G., G. Fantoni, and F. Montagna. 2013. “Situating Needs and Requirements in the FBS Framework.” *Design Studies* 34(5):636–62. doi: 10.1016/j.destud.2012.12.001.
- Catalano, Chiara E., Elena Camossi, Rosalinda Ferrandes, Vincent Cheutet, and Neyir Sevilmis. 2009. “A Product Design Ontology for Enhancing Shape Processing in Design Workflows.” *Journal of Intelligent Manufacturing* 20(5):553–67. doi: 10.1007/s10845-008-0151-z.
- Chai, T., and R. R. Draxler. 2014. “Root Mean Square Error (RMSE) or Mean Absolute Error (MAE)?-Arguments against Avoiding RMSE in the Literature.” *Geosci. Model Dev* 7:1247–50. doi: 10.5194/gmd-7-1247-2014.
- Chapman, A. L., M. Hadfield, and C. J. Chapman. 2015. “Qualitative Research in Healthcare: An Introduction to Grounded Theory Using Thematic Analysis.” *Journal of the Royal College of Physicians of Edinburgh* 45(3):201–5. doi: 10.4997/JRCPE.2015.305.
- Chen, Jie, Chunxia Zhang, and Zhendong Niu. 2016. “Identifying Helpful Online Reviews with Word Embedding Features.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 9983 LNAI:123–33. doi: 10.1007/978-3-319-47650-6_10.
- Chen, Nan Chen, Margaret Drouhard, Rafal Kocielnik, Jina Suh, and Cecilia R. Aragon. 2018. “Using Machine Learning to Support Qualitative Coding in Social Science: Shifting the

- Focus to Ambiguity.” *ACM Transactions on Interactive Intelligent Systems* 8(2). doi: 10.1145/3185515.
- Chen, Ning, Jialiu Lin, Steven C. H. Hoi, Xiaokui Xiao, and Boshen Zhang. 2014. “AR-Miner: Mining Informative Reviews for Developers from Mobile App Marketplace.” Pp. 767–78 in *Proceedings - International Conference on Software Engineering*.
- Cho, Kyunghyun, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. “Learning Phrase Representations Using RNN Encoder-Decoder for Statistical Machine Translation.” *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference* 1724–34. doi: 10.3115/v1/d14-1179.
- Chowdhury, Gobinda G. 2003. “Natural Language Processing.” *Annual Review of Information Science and Technology* 37(1):51–89. doi: 10.1145/234173.234180.
- Chun Tie, Ylona, Melanie Birks, and Karen Francis. 2019. “Grounded Theory Research: A Design Framework for Novice Researchers.” *SAGE Open Medicine* 7:205031211882292. doi: 10.1177/2050312118822927.
- Cleland-Huang, Jane, Raffaella Settini, Xuchang Zou, and Peter Solc. 2007. “Automated Classification of Non-Functional Requirements.” *Requirements Engineering* 12(2):103–20. doi: 10.1007/s00766-007-0045-1.
- Cleland-Huang, Jane, Raffaella Settini, Xuchang Zou, and Peter Sole. 2006. “The Detection and Classification of Non-Functional Requirements with Application to Early Aspects.” *Proceedings of the IEEE International Conference on Requirements Engineering* 36–45. doi: 10.1109/RE.2006.65.
- Cooper, Harris M. 1988. “Organizing Knowledge Syntheses: A Taxonomy of Literature Reviews.” *Knowledge in Society* 1(1):104–26. doi: 10.1007/BF03177550.
- Cooper, Harris M., Larry V Hedges, and Jeffrey C. Valentine. 2019. *HANDBOOK OF RESEARCH SYNTHESIS AND META-ANALYSIS*. New York: Russell Sage Foundation.
- Coughlan, Jane, and Robert D. Macredie. 2002. “Effective Communication in Requirements Elicitation: A Comparison of Methodologies.” *Requirements Engineering* 7(2):47–60. doi: 10.1007/S007660200004.
- Dadzie, A. S., R. Bhagdev, A. Chakravarthy, S. Chapman, J. Iria, V. Lanfranchi, J. Magalhães, D. Petrelli, and F. Ciravegna. 2009. “Applying Semantic Web Technologies to Knowledge Sharing in Aerospace Engineering.” *Journal of Intelligent Manufacturing* 20(5):611–23. doi: 10.1007/s10845-008-0141-1.
- Dalpiazz, Fabiano, Davide Dell’Anna, Fatma Başak Aydemir, and Sercan Çevikol. 2019. “Requirements Classification with Interpretable Machine Learning and Dependency Parsing.” *Proceedings of the IEEE International Conference on Requirements Engineering 2019-Septe*:142–52. doi: 10.1109/RE.2019.00025.

- Davis, Randall, Shrove Howard, and Szolovits Peter. 1993. "What Is a Knowledge Representation?" *The Knowledge Frontier* 14(1):1–43. doi: 10.1007/978-1-4612-4792-0_1.
- El Dehaibi, Nasreddine, Noah D. Goodman, and Erin F. MacDonald. 2019. "Extracting Customer Perceptions of Product Sustainability from Online Reviews." *Journal of Mechanical Design, Transactions of the ASME* 141(12). doi: 10.1115/1.4044522.
- Dekhtyar, Alex, and Vivian Fong. 2017. "RE Data Challenge: Requirements Identification with Word2Vec and TensorFlow." Pp. 484–89 in *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*.
- Deocadez, Roger, Rachel Harrison, and Daniel Rodriguez. 2017. "Automatically Classifying Requirements from App Stores: A Preliminary Study." *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference Workshops, REW 2017* 367–71. doi: 10.1109/REW.2017.58.
- Deshpande, Gouri, Chahal Arora, and Guenther Ruhe. 2019. "Data-Driven Elicitation and Optimization of Dependencies between Requirements." Pp. 416–21 in *Proceedings of the IEEE International Conference on Requirements Engineering*. Vols. 2019-Septe. IEEE.
- Devlin, Jacob, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." (Mlm).
- Devlin, Jacob, Ming Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. "BERT: Pre-Training of Deep Bidirectional Transformers for Language Understanding." in *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference*.
- Dhinakaran, Venkatesh T., Raseshwari Pulle, Nirav Ajmeri, and Pradeep K. Murukannaiah. 2018. "App Review Analysis via Active Learning: Reducing Supervision Effort without Compromising Classification Accuracy." Pp. 170–81 in *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018*. IEEE.
- Dixon, John R., Adele Howe, Paul R. Cohen, and Melvin K. Simmons. 1987. "Dominic I: Progress toward Domain Independence in Design by Iterative Redesign." *Engineering with Computers* 2(3):137–45. doi: 10.1007/BF01201261.
- Dreisbach, Caitlin, Theresa A. Koleck, Philip E. Bourne, and Suzanne Bakken. 2019. "A Systematic Review of Natural Language Processing and Text Mining of Symptoms from Electronic Patient-Authored Text Data." *International Journal of Medical Informatics* 125(December 2018):37–46. doi: 10.1016/j.ijmedinf.2019.02.008.
- Edunov, Sergey, Alexei Baevski, and Michael Auli. 2019. "Pre-Trained Language Model Representations for Language Generation." *NAACL HLT 2019 - 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1:4052–59. doi: 10.18653/v1/n19-1409.
- El-Sappagh, Shaker, Francesco Franda, Farman Ali, and Kyung Sup Kwak. 2018. "SNOMED CT

- Standard Ontology Based on the Ontology for General Medical Science.” *BMC Medical Informatics and Decision Making* 18(1):1–19. doi: 10.1186/s12911-018-0651-5.
- Elbattah, Mahmoud, Émilien Arnaud, Maxime Gignon, and Gilles Dequen. 2021. “The Role of Text Analytics in Healthcare: A Review of Recent Developments and Applications.” *HEALTHINF 2021 - 14th International Conference on Health Informatics; Part of the 14th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2021* 5(Biostec):825–32. doi: 10.5220/0010414508250832.
- EzzatiKarami, Mahtab, and Nazim H. Madhavji. 2021. *Automatically Classifying Non-Functional Requirements with Feature Extraction and Supervised Machine Learning Techniques: A Research Preview*. Vol. 12685 LNCS. Springer International Publishing.
- Falessi, Davide, Giovanni Cantone, and Gerardo Canfora. 2010. “A Comprehensive Characterization of NLP Techniques for Identifying Equivalent Requirements.” in *ESEM 2010 - Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*.
- Fan, Yin, Xiangju Lu, Dian Li, and Yuanliu Liu. 2016. “Video-Based Emotion Recognition Using CNN-RNN and C3D Hybrid Networks.” *ICMI 2016 - Proceedings of the 18th ACM International Conference on Multimodal Interaction* 445–50. doi: 10.1145/2993148.2997632.
- Fantoni, G., E. Coli, F. Chiarello, R. Apreda, F. Dell’Orletta, and G. Pratelli. 2021. “Text Mining Tool for Translating Terms of Contract into Technical Specifications: Development and Application in the Railway Sector.” *Computers in Industry* 124:103357. doi: 10.1016/j.compind.2020.103357.
- Fernandes, Rui, Ian Grosse, Sundar Krishnamutry, and Jack Wileden. 2007. “DESIGN AND INNOVATIVE METHODOLOGIES IN A SEMANTIC FRAMEWORK.” Pp. 0- in *International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*. ASME.
- Ferrari, Alessio, Gloria Gori, Benedetta Rosadini, Iacopo Trotta, Stefano Bacherini, Alessandro Fantechi, and Stefania Gnesi. 2018. “Detecting Requirements Defects with NLP Patterns: An Industrial Experience in the Railway Domain.” *Empirical Software Engineering* 23(6):3684–3733. doi: 10.1007/s10664-018-9596-7.
- Ferrari, Alessio, Giorgio Oronzo Spagnolo, and Stefania Gnesi. 2017. “PURE: A Dataset of Public Requirements Documents.” *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017* 502–5. doi: 10.1109/RE.2017.29.
- Friedman, Ken. 2000. “Creating Design Knowledge: From Research into Practice.” *IDATER 2000 Conference* 05–32.
- Frisch, Alan M. 1982. “What ’ s in a Semantic Network ?” 19–27.
- Fu, Bin, Jialiu Lin, Lei Liy, Christos Faloutsos, Jason Hong, and Norman Sadeh. 2013. “Why People Hate Your App - Making Sense of User Feedback in a Mobile App Store.” Pp. 1276–

84 in *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Vol. Part F1288.

Fu, Yuan Qiu, Ping Yoon Chui, and Martin G. Helander. 2006. “Knowledge Identification and Management in Product Design.” *Journal of Knowledge Management* 10(6):50–63. doi: 10.1108/13673270610709215.

Ganegedara, Thushan. 2018. *Natural Language Processing with TensorFlow*.

Gao, Fei, M. Meng, and Steve Clarke. 2008. “Knowledge, Management, and Knowledge Management in Business Operations.” *Journal of Knowledge Management* 12(2):3–17. doi: 10.1108/13673270810859479.

García, Ivan, Carla Pacheco, Andrés León, and Jose A. Calvo-Manzano. 2020. “A Serious Game for Teaching the Fundamentals of ISO/IEC/IEEE 29148 Systems and Software Engineering – Lifecycle Processes – Requirements Engineering at Undergraduate Level.” *Computer Standards and Interfaces* 67(September 2019):103377. doi: 10.1016/j.csi.2019.103377.

Garzoli, Francesco, Danilo Croce, Manuela Nardini, Francesco Ciambra, and Roberto Basili. 2013. “Robust Requirements Analysis in Complex Systems through Machine Learning.” *Communications in Computer and Information Science* 379 CCIS:44–58. doi: 10.1007/978-3-642-45260-4_4.

Gavrilova, Tatiana, and Tatiana Andreeva. 2012. “Knowledge Elicitation Techniques in a Knowledge Management Context.” *Journal of Knowledge Management* 16(4):523–37. doi: 10.1108/13673271211246112.

Gero, John S., and Udo Kannengiesser. 2014. “The Function-Behaviour-Structure Ontology of Design.” Pp. 263–83 in *An Anthology of Theories and Models of Design*.

Gnanasekaran, Rajesh Kumar, Suranjan Chakraborty, Josh Dehlinger, and Lin Deng. 2021. “Using Recurrent Neural Networks for Classification of Natural Language-Based Non-Functional Requirements.” *CEUR Workshop Proceedings* 2857.

Goldberg, Yoav. 2017. *Neural Network Methods for Natural Language Processing*.

Goldberg, Yoav, and Omer Levy. 2014. “Word2vec Explained: Deriving Mikolov et Al.’s Negative-Sampling Word-Embedding Method.” (2):1–5.

Gough, David, Sandy Oliver, and James Thomas. 2012. *An Introduction to Systematic Reviews*.

Gruber, Thomas R. 1993. “A Translation Approach to Portable Ontology Specifications.” *Knowledge Acquisition* 5(2):199–220.

Guetterman, Timothy C., Tammy Chang, Melissa DeJonckheere, Tanmay Basu, Elizabeth Scruggs, and V. G. V. Vidyiswaran. 2018. “Augmenting Qualitative Text Analysis with Natural Language Processing: Methodological Study.” *Journal of Medical Internet Research* 20(6):e231. doi: 10.2196/jmir.9702.

- Gulle, Kim Julian, Nicholas Ford, Patrick Ebel, Florian Brokhausen, and Andreas Vogelsang. 2020. "Topic Modeling on User Stories Using Word Mover's Distance." *Proceedings - 7th International Workshop on Artificial Intelligence and Requirements Engineering, AIRE 2020* 52–60. doi: 10.1109/AIRE51212.2020.00015.
- Guptill, Janet. 2005. "Knowledge Management in Health Care." *Journal of Health Care Finance* 31(3):10–14. doi: 10.4018/978-1-61520-670-4.ch023.
- Gururangan, Suchin, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. "Don't Stop Pretraining: Adapt Language Models to Domains and Tasks." 8342–60. doi: 10.18653/v1/2020.acl-main.740.
- Guzman, Emitza, Mohamed Ibrahim, and Martin Glinz. 2017. "A Little Bird Told Me: Mining Tweets for Requirements and Software Evolution." Pp. 11–20 in *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017*.
- Guzman, Emitza, and Walid Maalej. 2014. "How Do Users like This Feature? A Fine Grained Sentiment Analysis of App Reviews." Pp. 153–62 in *2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings*. IEEE.
- Hahn, Udo, and Klemens Schnattinger. 1998. "Towards Text Knowledge Engineering." *Proceedings of the National Conference on Artificial Intelligence* 524–31.
- Halim, Fahrizal, and Daniel Siahaan. 2019. "Detecting Non-Atomic Requirements in Software Requirements Specifications Using Classification Methods." *2019 1st International Conference on Cybernetics and Intelligent System, ICORIS 2019 (August):269–73*. doi: 10.1109/ICORIS.2019.8874888.
- Hall, M., E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten. 2009. "The WEKA Data Mining Software: An Update." *ACM SIGKDD Explorations Newsletter* 11(1):10–18.
- Han, Ji, Serhad Sarica, Feng Shi, and Jianxi Luo. 2022. "Semantic Networks for Engineering Design: State of the Art and Future Directions." *Journal of Mechanical Design, Transactions of the ASME* 144(2):1–47. doi: 10.1115/1.4052148.
- Han, Yi, and Mohsen Moghaddam. 2021. "Eliciting Attribute-Level User Needs from Online Reviews with Deep Language Models and Information Extraction." *Journal of Mechanical Design, Transactions of the ASME* 143(6). doi: 10.1115/1.4048819.
- Haque, Md Ariful, Md Abdur Rahman, and Md Saeed Siddik. 2019. "Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study." *1st International Conference on Advances in Science, Engineering and Robotics Technology 2019, ICASERT 2019* 2019(Icasert). doi: 10.1109/ICASERT.2019.8934499.
- Henry, David, Allison B. Dymnicki, Nathaniel Mohatt, James Allen, and James G. Kelly. 2015. "Clustering Methods with Qualitative Data: A Mixed-Methods Approach for Prevention Research with Small Samples." *Prevention Science* 16(7):1007–16. doi: 10.1007/s11121-015-0561-z.

- Hochreiter, Sepp, and Jürgen Schmidhuber. 1997. “Long Short-Term Memory.” *Neural Computation* 9(8):1735–80. doi: 10.1162/NECO.1997.9.8.1735.
- Houmb, Siv Hilde, Shareeful Islam, Eric Knauss, Jan Jürjens, and Kurt Schneider. 2010. “Eliciting Security Requirements and Tracing Them to Design: An Integration of Common Criteria, Heuristics, and UMLsec.” *Requirements Engineering* 15(1):63–93. doi: 10.1007/s00766-009-0093-9.
- Huang, Jingwei, Adrian Gheorghe, Holly Handley, Pilar Pazos, Ariel Pinto, Samuel Kovacic, Andrew Collins, Charles Keating, Andres Sousa-Poza, Ghaith Rabadi, Resit Unal, Teddy Cotter, Rafael Landaeta, and Charles Daniels. 2020. “Towards Digital Engineering: The Advent of Digital Systems Engineering.” *Int. J. System of Systems Engineering* 10(3):234–61. doi: 10.1504/IJSSE.2020.109737.
- Hussain, Ishrar, Leila Kosseim, and Olga Ormandjieva. 2008. “Using Linguistic Knowledge to Classify Non-Functional Requirements in SRS Documents.” Pp. 287–98 in *Lecture Notes in Computer Science*. Vol. 5039 LNCS.
- James, G., D. Witten, T. Hastie, and R. Tibshirani. 2013. *An Introduction to Statistical Learning - with Applications in R*.
- Jeon, Kahyun, Ghang Lee, and H. David Jeong. 2021. “Classification of the Requirement Sentences of the US DOT Standard Specification Using Deep Learning Algorithms.” *Lecture Notes in Civil Engineering* 98:89–97. doi: 10.1007/978-3-030-51295-8_8.
- Jha, Nishant, and Anas Mahmoud. 2019. “Mining Non-Functional Requirements from App Store Reviews.” *Empirical Software Engineering* 24(6):3659–95. doi: 10.1007/s10664-019-09716-7.
- Jia, Wenjun, and Yong Zeng. 2021. “EEG Signals Respond Differently to Idea Generation, Idea Evolution and Evaluation in a Loosely Controlled Creativity Experiment.” *Scientific Reports* 11(1):1–20. doi: 10.1038/s41598-021-81655-0.
- Jie, Pan, Huang Jingwei, Wang Yunli, Cheng Gengdong, and Zeng Yong. 2021. “A Self-Learning Finite Element Extraction System Based on Reinforcement Learning.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM*.
- Jindal, Rajni, Ruchika Malhotra, and Abha Jain. 2016. “Automated Classification of Security Requirements.” *2016 International Conference on Advances in Computing, Communications and Informatics, ICACCI 2016* 2027–33. doi: 10.1109/ICACCI.2016.7732349.
- Jo, Yohan, and Alice Oh. 2011. “Aspect and Sentiment Unification Model for Online Review Analysis.” *Proceedings of the 4th ACM International Conference on Web Search and Data Mining, WSDM 2011* 815–24. doi: 10.1145/1935826.1935932.
- Johnson, Mark. 2009. “How the Statistical Revolution Changes (Computational) Linguistics.” 3–11. doi: 10.3115/1642038.1642041.

- Jones, Isaac A., and Kyoung Yun Kim. 2015. "Systematic Service Product Requirement Analysis with Online Customer Review Data." *Journal of Integrated Design and Process Science* 19(2):25–48. doi: 10.3233/jid-2015-0011.
- Joulin, Armand, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. 2017. "Bag of Tricks for Efficient Text Classification." *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference 2*:427–31. doi: 10.18653/v1/e17-2068.
- Joung, Junegak, and Harrison M. Kim. 2021. "Automated Keyword Filtering in Latent Dirichlet Allocation for Identifying Product Attributes from Online Reviews." *Journal of Mechanical Design, Transactions of the ASME* 143(8):1–6. doi: 10.1115/1.4048960.
- Jung, Daekyo, Vu Tran Tuan, Dai Quoc Tran, Minsoo Park, and Seunghee Park. 2020. "Conceptual Framework of an Intelligent Decision Support System for Smart City Disaster Management." *Applied Sciences (Switzerland)* 10(2).
- Jurafsky, Daniel, and James H. Martin. 2018. "Speech and Language Processing." 1. doi: 10.1162/089120100750105975.
- Kengphanphanit, Natthaphon, and Pornsiri Muenchaisri. 2020. "Automatic Requirements Elicitation from Social Media (ARESM)." *PervasiveHealth: Pervasive Computing Technologies for Healthcare* 57–62. doi: 10.1145/3418994.3419004.
- Khelifa, Amani, Mariem Haoues, and Asma Sellami. 2018. "Towards a Software Requirements Change Classification Using Support Vector Machine." *CEUR Workshop Proceedings* 2279:1–10.
- Kitchenham, Barbara A., Tore Dybå, and Magne Jorgensen. 2004. "Evidence-Based Software Engineering." in *Proceedings. 26th International Conference on Software Engineering*.
- Kitchenham, Barbara, O. Pearl Brereton, David Budgen, Mark Turner, John Bailey, and Stephen Linkman. 2009. "Systematic Literature Reviews in Software Engineering - A Systematic Literature Review." *Information and Software Technology* 51(1):7–15. doi: 10.1016/j.infsof.2008.09.009.
- Knauss, Eric, Siv Houmb, Kurt Schneider, Shareeful Islam, and Jan Jürjens. 2011. "Supporting Requirements Engineers in Recognising Security Issues." *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 6606 LNCS:4–18. doi: 10.1007/978-3-642-19858-8_2.
- Ko, Youngjoong, Sooyong Park, Jungyun Seo, and Soonhwang Choi. 2007. "Using Classification Techniques for Informal Requirements in the Requirements Analysis-Supporting System." *Information and Software Technology* 49(11–12):1128–40. doi: 10.1016/j.infsof.2006.11.007.
- Kobilica, Armin, Mohammed Ayub, and Jameleddine Hassine. 2020. "Automated Identification of Security Requirements: A Machine Learning Approach." *PervasiveHealth: Pervasive*

- Computing Technologies for Healthcare* (1):475–80. doi: 10.1145/3383219.3383288.
- Kolpondinos, Martina Z., and Martin Glinz. 2020. “GARUSO: A Gamification Approach for Involving Stakeholders Outside Organizational Reach in Requirements Engineering.” *Requirements Engineering* 25(2):185–212. doi: 10.1007/s00766-019-00314-z.
- Kurtanovic, Zijad, and Walid Maalej. 2017a. “Automatically Classifying Functional and Non-Functional Requirements Using Supervised Machine Learning.” *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017* 490–95. doi: 10.1109/RE.2017.82.
- Kurtanovic, Zijad, and Walid Maalej. 2017b. “Mining User Rationale from Software Reviews.” *Proceedings - 2017 IEEE 25th International Requirements Engineering Conference, RE 2017* 61–70. doi: 10.1109/RE.2017.86.
- Lacity, Mary C., and Marius A. Janson. 1994. “Understanding Qualitative Data: A Framework of Text Analysis Methods.” *Journal of Management Information Systems* 11(2):137–55. doi: 10.1080/07421222.1994.11518043.
- Lame, Guillaume. 2019. “Systematic Literature Reviews: An Introduction.” *Proceedings of the International Conference on Engineering Design, ICED 2019-Augus(AUGUST)*:1633–42. doi: 10.1017/dsi.2019.169.
- Lange, Douglas S. 2008. “Text Classification and Machine Learning Support for Requirements Analysis Using Blogs.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 5320 LNCS:182–95. doi: 10.1007/978-3-540-89778-1_14.
- Leeson, William, Adam Resnick, Daniel Alexander, and John Rovers. 2019. “Natural Language Processing (NLP) in Qualitative Public Health Research: A Proof of Concept Study.” *International Journal of Qualitative Methods* 18:1–9. doi: 10.1177/1609406919887021.
- Lensu, Anssi. 2002. *Computationally Intelligent Methods for Qualitative Data Analysis*.
- Li, Chuanyi, Liguang Huang, Jidong Ge, Bin Luo, and Vincent Ng. 2018. “Automatically Classifying User Requests in Crowdsourcing Requirements Engineering.” *Journal of Systems and Software* 138:108–23. doi: 10.1016/j.jss.2017.12.028.
- Li, Jing, Xinwei Zhang, Keqin Wang, Chen Zheng, Shurong Tong, and Benoit Eynard. 2020. “A Personalized Requirement Identifying Model for Design Improvement Based on User Profiling.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 34(1):55–67. doi: 10.1017/S0890060419000301.
- Li, Tong. 2018. “Identifying Security Requirements Based on Linguistic Analysis and Machine Learning.” *Proceedings - Asia-Pacific Software Engineering Conference, APSEC 2017-Decem*:388–97. doi: 10.1109/APSEC.2017.45.
- Li, Zhanjun, and Karthik Ramani. 2007. “Ontology-Based Design Information Extraction and

- Retrieval.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 21(2):137–54. doi: 10.1017/S0890060407070199.
- Liang, Hong, Xiao Sun, Yunlei Sun, and Yuan Gao. 2017. “Text Feature Extraction Based on Deep Learning: A Review.” *Eurasip Journal on Wireless Communications and Networking* 2017(1):1–12. doi: 10.1186/s13638-017-0993-1.
- Lim, Sachiko, Aron Henriksson, and Jelena Zdravkovic. 2021. *Data-Driven Requirements Elicitation: A Systematic Literature Review*. Vol. 2. Springer Singapore.
- Lim, Soo Ling, and Anthony Finkelstein. 2012. “StakeRare: Using Social Networks and Collaborative Filtering for Large-Scale Requirements Elicitation.” *IEEE Transactions on Software Engineering* 38(3):707–35. doi: 10.1109/TSE.2011.36.
- Lin, Jinxin, Mark S. Fox, and Taner Bilgic. 1996. “A Requirement Ontology for Engineering Design.” *Concurrent Engineering Research and Applications* 4(3):279–91. doi: 10.1177/1063293x9600400307.
- Liu, Ying, Jian Jin, Ping Ji, Jenny A. Harding, and Richard Y. K. Fung. 2013. “Identifying Helpful Online Reviews: A Product Designer’s Perspective.” *CAD Computer Aided Design* 45(2):180–94. doi: 10.1016/j.cad.2012.07.008.
- Liu, Ying, Wen Feng Lu, and Han Tong Loh. 2007. “Knowledge Discovery and Management for Product Design through Text Mining - A Case Study of Online Information Integration for Designers.” *Proceedings of ICED 2007, the 16th International Conference on Engineering Design* DS 42(August):1–12.
- Liu, Yinhan, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. “RoBERTa: A Robustly Optimized BERT Pretraining Approach.” (1).
- Loper, Edward, and Steven Bird. 2002. “NLTK: The Natural Language Toolkit.” doi: 10.3115/1225403.1225421.
- Lu, Mengmeng, and Peng Liang. 2017. “Automatic Classification of Non-Functional Requirements from Augmented App User Reviews.” Pp. 344–53 in *Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*. Vol. Part F1286.
- Lund, Kevin, and Curt Burgess. 1996. “Producing High-Dimensional Semantic Spaces from Lexical Co-Occurrence.” *Behavior Research Methods, Instruments, and Computers* 28(2):203–8. doi: 10.3758/BF03204766.
- Ljutov, Alexey, Yilmaz Uygun, and Marc Thorsten Hütt. 2019. “Managing Workflow of Customer Requirements Using Machine Learning.” *Computers in Industry* 109:215–25. doi: 10.1016/j.compind.2019.04.010.
- Maalej, Walid, Zijad Kurtanović, Hadeer Nabil, and Christoph Stanik. 2016. “On the Automatic

- Classification of App Reviews.” *Requirements Engineering* 21(3):311–31. doi: 10.1007/s00766-016-0251-9.
- Maalej, Walid, Maleknaz Nayebi, Timo Johann, and Gunther Ruhe. 2015. “Towards Data - Driven Requirements Engineering.” 1–6.
- Maalej, Walid, Maleknaz Nayebi, and Guenther Ruhe. 2019. “Data-Driven Requirements Engineering-Aan Update.” *Proceedings - 2019 IEEE/ACM 41st International Conference on Software Engineering: Software Engineering in Practice, ICSE-SEIP 2019* 289–90. doi: 10.1109/ICSE-SEIP.2019.00041.
- Maaten, Lanuerns van der, and Geoffrey Hinton. 2008. “Visualizing Data Using T-SNE.” *Journal of Machine Learning Research* 9(11). doi: 10.1007/s10479-011-0841-3.
- Mahmoud, Anas. 2015. “An Information Theoretic Approach for Extracting and Tracing Non-Functional Requirements.” *2015 IEEE 23rd International Requirements Engineering Conference, RE 2015 - Proceedings* 36–45. doi: 10.1109/RE.2015.7320406.
- Maiti, Richard, and Frank Mitropoulos. 2017. “Prioritizing Non-Functional Requirements in Agile Software Engineering.” *Proceedings of the SouthEast Conference, ACMSE 2017* 212–14. doi: 10.1145/3077286.3077565.
- Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*.
- Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schütze. 2008. *Introduction to Information Retrieval Introduction*. Cambridge University Press.
- Manning, Christopher D., and Hinrich Schütze. 1999. *Foundations of Statistical Natural Language Processing*. Massachusetts Institute of Technology.
- Martens, Daniel, and Walid Maalej. 2019. “Towards Understanding and Detecting Fake Reviews in App Stores.” *Empirical Software Engineering* 24(6):3316–55. doi: 10.1007/s10664-019-09706-9.
- Massey, Aaron K., Jacob Eisenstein, Annie I. Anton, and Peter P. Swire. 2013. “Automated Text Mining for Requirements Analysis of Policy Documents.” Pp. 4–13 in *2013 21st IEEE International Requirements Engineering Conference, RE 2013 - Proceedings*.
- McMahon, Chris, Alistair Lowe, and Steve Culley. 2004. “Knowledge Management in Engineering Design: Personalization and Codification.” *Journal of Engineering Design* 15(4):307–25. doi: 10.1080/09544820410001697154.
- Meng, Jie, Gaofa He, Xiang Li, Liancheng Ren, Chaoqun Dong, and Min Liu. 2021. “Using Environment Based Design Method to Improve Mechanical Curriculum Teaching.” *Innovations in Education and Teaching International*. doi: 10.1080/14703297.2021.1882328.
- Meng, Lingling, Runqing Huang, and Junzhong Gu. 2013. “A Review of Semantic Similarity

- Measures in WordNet.” *International Journal of Hybrid Information Technology* 6(1):1–12.
- Meth, Hendrik, Manuel Brhel, and Alexander Maedche. 2013. “The State of the Art in Automated Requirements Elicitation.” *Information and Software Technology* 55(10):1695–1709. doi: 10.1016/j.infsof.2013.03.008.
- Mikolov, Tomas, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. “Efficient Estimation of Word Representations in Vector Space.” *1st International Conference on Learning Representations, ICLR 2013 - Workshop Track Proceedings* 1–12.
- Mikolov, Tomas, Martin Karafiat, Lukas Burget, and Sanjeev Cernock’y, Jan “Honza”, Khudanpur. 2010. “Recurrent Neural Network Based Language Model.” Pp. 271–87 in *INTERSPEECH*. Vol. 25.
- Milhim, Hamzeh Bani, and Andrea Schiffauerova. 2013. “Towards Formalizing and Formulating the Successful Organizational Innovation Process.” *Journal of Integrated Design and Process Science* 17(2):5–21. doi: 10.3233/jid-2013-0001.
- Miller, G., R. Beckwith, C. Fellbaum, D. Gross, and K. Miller. 1993. “Introduction to WordNet: On-Line Database.” *International Journal of Lexicography* 3(4):235–44.
- Miller, George A. 1995. “WordNet: A Lexical Database for English.” *Communications of the ACM* 38(11):39–41. doi: 10.1145/219717.219748.
- Mohammad, Hossin, and Sulaiman Md Nasir. 2015. “A Review on Evaluation Metrics for Data Classification Evaluations.” *International Journal of Data Mining & Knowledge Management Process* 5(2):01–11. doi: 10.5121/ijdkp.2015.5201.
- Morkos, Beshoy, James Mathieson, and Joshua D. Summers. 2014. “Comparative Analysis of Requirements Change Prediction Models: Manual, Linguistic, and Neural Network.” *Research in Engineering Design* 25(2):139–56. doi: 10.1007/s00163-014-0170-z.
- Muller, Michael, Shion Guha, Eric P. S. Baumer, David Mimno, and N. Sadat Shami. 2016. “Machine Learning and Grounded Theory Method.” 3–8. doi: 10.1145/2957276.2957280.
- Murphy, Kevin. 2012. *Machine Learning: A Probabilistic Perspective*.
- Myllynen, Santeri, Ilpo Suominen, Tapani Raunio, Rasmus Karell, and Jussi Lahtinen. 2021. “Developing and Implementing Artificial Intelligence-Based Classifier for Requirements Engineering.” *Journal of Nuclear Engineering and Radiation Science* 7(4):1–9. doi: 10.1115/1.4049722.
- Navarro-Almanza, Raul, Reyes Juarez-Ramirez, and Guillermo Licea. 2018. “Towards Supporting Software Engineering Using Deep Learning: A Case of Software Requirements Classification.” *Proceedings - 2017 5th International Conference in Software Engineering Research and Innovation, CONISOFT 2017* 2018-Janua:116–20. doi: 10.1109/CONISOFT.2017.00021.

- Nayebi, Maleknaz, Henry Cho, and Guenther Ruhe. 2018. *App Store Mining Is Not Enough for App Improvement*. Vol. 23. Empirical Software Engineering.
- Nelson, Laura K. 2020. "Computational Grounded Theory: A Methodological Framework." *Sociological Methods and Research* 49(1):3–42. doi: 10.1177/0049124117729703.
- Nguyen, Thanh An, and Yong Zeng. 2012. "A Theoretical Model of Design Creativity: Nonlinear Design Dynamics and Mental Stress-Creativity Relation." *Journal of Integrated Design and Process Science* 16(3):65–88. doi: 10.3233/jid-2012-0007.
- Nguyen, Thanh An, and Yong Zeng. 2017. "Effects of Stress and Effort on Self-Rated Reports in Experimental Study of Design Activities." *Journal of Intelligent Manufacturing* 28(7):1609–22. doi: 10.1007/s10845-016-1196-z.
- Nigam, Kamal, Andrew Kachites Mccallum, Sebastian Thrun, and Tom Mitchell. 2000. "Text Classification from Labeled and Unlabeled Documents Using EM." *Machine Learning* 39:2 39(2):103–34. doi: 10.1023/A:1007692713085.
- Noei, Ehsan, Feng Zhang, and Ying Zou. 2021. "Too Many User-Reviews! What Should App Developers Look at First?" *IEEE Transactions on Software Engineering* 47(2):367–78. doi: 10.1109/TSE.2019.2893171.
- Noy, Natalya F., and Deborah L. McGuinness. 2001. "Ontology Development 101: A Guide to Creating Your First Ontology." *Knowledge Systems Laboratory Tech Report*. doi: 10.3390/su9122317.
- Nyamawe, Ally S., Hui Liu, Nan Niu, Qasim Umer, and Zhendong Niu. 2019. "Automated Recommendation of Software Refactorings Based on Feature Requests." *Proceedings of the IEEE International Conference on Requirements Engineering 2019-Sept*:187–98. doi: 10.1109/RE.2019.00029.
- O’Leary, Daniel E. 1991. "Design, Development and Validation of Expert Systems: A Survey of Developers." *Validation, Verification and Test of Knowledge-Based Systems* 3–19.
- O’Leary, Daniel E. 2013. "Artificial Intelligence and Big Data What Is Big Data?" *IEEE Intelligent Systems* 28(2):66–69.
- Okoli, Chitu. 2015. "A Guide to Conducting a Standalone Systematic Literature Review." *Communications of the Association for Information Systems* 37(1):879–910. doi: 10.17705/1cais.03743.
- Oriol, Marc, Melanie Stade, Farnaz Fotrousi, Sergi Nadal, Jovan Varga, Norbert Seyff, Alberto Abello, Xavier Franch, Jordi Marco, and Oleg Schmidt. 2018. "FAME: Supporting Continuous Requirements Elicitation by Combining User Feedback and Monitoring." *Proceedings - 2018 IEEE 26th International Requirements Engineering Conference, RE 2018* 217–27. doi: 10.1109/RE.2018.00030.
- Ormandjieva, Olga, Ishrar Hussain, and Leila Kosseim. 2007. "Toward a Text Classification

- System for the Quality Assessment of Software Requirements Written in Natural Language.” *SOQUA '07: Fourth International Workshop on Software Quality Assurance - In Conjunction with the 6th ESEC/FSE Joint Meeting* 39–45. doi: 10.1145/1295074.1295082.
- Ott, Daniel. 2013. “Automatic Requirement Categorization of Large Natural Language Specifications at Mercedes-Benz for Review Improvements.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 7830 LNCS:50–64. doi: 10.1007/978-3-642-37422-7_4.
- Palacio-Niño, Julio-Omar, and Fernando Berzal. 2019. “Evaluation Metrics for Unsupervised Learning Algorithms.”
- Pan, Jie, Jingwei Huang, Yunli Wang, Gengdong Cheng, and Yong Zeng. 2021. “A Self-Learning Finite Element Extraction System Based on Reinforcement Learning.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 35(2):180–208. doi: 10.1017/S089006042100007X.
- Panichella, Sebastiano, Andrea Di Sorbo, Emitza Guzman, Corrado A. Visaggio, Gerardo Canfora, and Harald C. Gall. 2015. “How Can i Improve My App? Classifying User Reviews for Software Maintenance and Evolution.” *2015 IEEE 31st International Conference on Software Maintenance and Evolution, ICSME 2015 - Proceedings* (September):281–90. doi: 10.1109/ICSM.2015.7332474.
- Parra, Eugenio, Christos Dimou, Juan Llorens, Valentín Moreno, and Anabel Fraga. 2015. “A Methodology for the Classification of Quality of Requirements Using Machine Learning Techniques.” *Information and Software Technology* 67:180–95. doi: 10.1016/j.infsof.2015.07.006.
- Patrício, Lia, Raymond P. Fisk, João Falcão e Cunha, and Larry Constantine. 2011. “Multilevel Service Design: From Customer Value Constellation to Service Experience Blueprinting.” *Journal of Service Research* 14(2):180–200. doi: 10.1177/1094670511401901.
- Pedregosa, Fabian, Gael Varoquaux, Gramfort Alexandre, and Bertrand Thirion. 2011. “Scikit-Learn: Machine Learning in Python.” *Journal of Machine Learning Research* 12:2825–30. doi: 10.1289/EHP4713.
- Pennington, Jeffrey, Richard Socher, and Christopher D. Manning. 2014. “GloVe: Global Vectors for Word Representation.” in *Proceedings of the 2014 conference on empirical methods in natural language processing*.
- Perez-Verdejo, J. Manuel, Angel J. Sanchez-Garcia, and Jorge Octavio Ocharan-Hernandez. 2020. “A Systematic Literature Review on Machine Learning for Automated Requirements Classification.” *Proceedings - 2020 8th Edition of the International Conference in Software Engineering Research and Innovation, CONISOFT 2020* 21–28. doi: 10.1109/CONISOFT50191.2020.00014.
- Perini, Anna. 2018. “Data-Driven Requirements Engineering. The SUPERSEDE Way.” *Information Management and Big Data* 1(Annual International Symposium on Information

- Management and Big Data):13–18. doi: 10.1007/978-3-030-11680-4.
- Petcușin, Felix, Liana Stănescu, and Costin Bădică. 2020. “An Experiment on Automated Requirements Mapping Using Deep Learning Methods.” *Studies in Computational Intelligence* 868:86–95. doi: 10.1007/978-3-030-32258-8_10.
- Peters, Matthew E., Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. “Deep Contextualized Word Representations.” *NAACL HLT 2018 - 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies - Proceedings of the Conference* 1:2227–37. doi: 10.18653/v1/n18-1202.
- Polpinij, Jantima, and Khanista Namee. 2021. *Automatically Retrieving of Software Specification Requirements Related to Each Actor*. Vol. 251. Springer International Publishing.
- Prasetyo, Philips K., David Lo, Palakorn Achananuparp, Yuan Tian, and Ee Peng Lim. 2012. “Automatic Classification of Software Related Microblogs.” Pp. 596–99 in *IEEE International Conference on Software Maintenance, ICSM*. IEEE.
- Provost, Foster, and Tom Fawcett. 2013. “Data Science and Its Relationship to Big Data and Data-Driven Decision Making.” *Big Data* 1(1):51–59. doi: 10.1089/big.2013.1508.
- Qi, Jiayin, Zhenping Zhang, Seongmin Jeon, and Yanquan Zhou. 2016. “Mining Customer Requirements from Online Reviews: A Product Improvement Perspective.” *Information and Management* 53(8):951–63. doi: 10.1016/j.im.2016.06.002.
- Qi, Peng, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. “Stanza: A Python Natural Language Processing Toolkit for Many Human Languages.” Pp. 101–8 in.
- Rahimi, Nouf, Fathy Eassa, and Lamiaa Elrefaei. 2020. “An Ensemble Machine Learning Technique for Functional Requirement Classification.” *Symmetry (MI)*:1–25.
- Rahman, M. Abdur, M. Ariful Haque, M. Nurul Ahad Tawhid, and M. Saeed Siddik. 2019. “Classifying Non-Functional Requirements Using RNN Variants for Quality Software Development.” Pp. 25–30 in *MaLTeSQuE 2019 - Proceedings of the 3rd ACM SIGSOFT International Workshop on Machine Learning Techniques for Software Quality Evaluation, co-located with ESEC/FSE 2019*.
- Raschka, Sebastian, and Vahid Mirjalili. 2019. *Python Machine Learning: Machine Learning and Deep Learning with Python, Scikit-Learn, and TensorFlow 2*.
- Rashwan, Abderahman, Olga Ormandjieva, and Rene Witte. 2013. “Ontology-Based Classification of Non-Functional Requirements in Software Specifications: A New Corpus and SVM-Based Classifier.” Pp. 381–86 in *Proceedings - International Computer Software and Applications Conference*.
- Reimers, Nils, and Iryna Gurevych. 2020. “Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks.” *EMNLP-IJCNLP 2019 - 2019 Conference on Empirical Methods in*

Natural Language Processing and 9th International Joint Conference on Natural Language Processing, Proceedings of the Conference 3982–92. doi: 10.18653/v1/d19-1410.

Reinhartz-Berger, Iris, and Mark Kemelman. 2020. “Extracting Core Requirements for Software Product Lines.” *Requirements Engineering* 25(1):47–65. doi: 10.1007/s00766-018-0307-0.

Riaz, Maria, Jason King, John Slankas, and Laurie Williams. 2014a. “Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts.” Pp. 183–92 in *2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings*. IEEE.

Riaz, Maria, Jason King, John Slankas, and Laurie Williams. 2014b. “Hidden in Plain Sight: Automatically Identifying Security Requirements from Natural Language Artifacts.” *2014 IEEE 22nd International Requirements Engineering Conference, RE 2014 - Proceedings* 183–92. doi: 10.1109/RE.2014.6912260.

Ridley, Diana. 2012. *The Literature Review: A Step-by-Step Guide for Students*.

Rockwell, J., I. R. Grosse, S. Krishnamurty, and J. C. Wileden. 2009. “A Decision Support Ontology for Collaborative Decision Making in Engineering Design.” *2009 International Symposium on Collaborative Technologies and Systems, CTS 2009* 1–9. doi: 10.1109/CTS.2009.5067456.

Rockwell, Justin A., Ian R. Grosse, Sundar Krishnamurty, and Jack C. Wileden. 2010. “A Semantic Information Model for Capturing and Communicating Design Decisions.” *Journal of Computing and Information Science in Engineering* 10(3). doi: 10.1115/1.3462926.

Rodeghero, Paige, Siyuan Jiang, Ameer Armaly, and Collin McMillan. 2017. “Detecting User Story Information in Developer-Client Conversations to Generate Extractive Summaries.” Pp. 49–59 in *Proceedings - 2017 IEEE/ACM 39th International Conference on Software Engineering, ICSE 2017*. IEEE.

Rogers, K. J. (Jamie.), John W. Priest, and Gail Haddock. 1995. “The Use of Semantic Networks to Support Concurrent Engineering in Semiconductor Product Development.” *Journal of Intelligent Manufacturing* 6(5):311–19. doi: 10.1007/BF00124675.

Rong, Xin. 2014. “Word2vec Parameter Learning Explained.” 1–21.

Rothe, Sascha, Shashi Narayan, and Aliaksei Severyn. 2020. “Leveraging Pre-Trained Checkpoints for Sequence Generation Tasks.” *Transactions of the Association for Computational Linguistics* 8:264–80. doi: 10.1162/tacl_a_00313.

Rotmensch, Maya, Yoni Halpern, Abdulhakim Tlimat, Steven Horng, and David Sontag. 2017. “Learning a Health Knowledge Graph from Electronic Medical Records.” *Scientific Reports* 7(1):1–11. doi: 10.1038/s41598-017-05778-z.

Rustan, Edhy, and Hasriani Hasriani. 2020. “Communication Pattern between Nurses and Elderly Patients through a Neuro-Linguistic Programming Approach.” *Jurnal Studi Komunikasi*

- (*Indonesian Journal of Communications Studies*) 4(1):75. doi: 10.25139/jsk.v4i1.2180.
- Rusu, Laura Irina, Wenny Rahayu, Torab Torabi, Florian Puersch, William Coronado, Andrew Taylor Harris, and Karl Reed. 2012. "Moving towards a Collaborative Decision Support System for Aeronautical Data." *Journal of Intelligent Manufacturing* 23(6):2085–2100. doi: 10.1007/s10845-011-0549-x.
- Sabir, Maliha, Ebad Banissi, and Mike Child. 2021. *A Deep Learning-Based Framework for the Classification of Non-Functional Requirements*. Vol. 1366 AISC. Springer International Publishing.
- Sampada, G. C., Tende Ivo Sake, and Megha Chhabra. 2020. "A Review on Advanced Techniques of Requirement Elicitation and Specification in Software Development Stages." *PDGC 2020 - 2020 6th International Conference on Parallel, Distributed and Grid Computing* 215–20. doi: 10.1109/PDGC50313.2020.9315741.
- Schuller, Dagmar M., and Björn W. Schuller. 2021. "A Review on Five Recent and Near-Future Developments in Computational Processing of Emotion in the Human Voice." *Emotion Review* 13(1):44–50. doi: 10.1177/1754073919898526.
- Shabestari, Seyed Sina, Michael Herzog, and Beate Bender. 2019. "A Survey on the Applications of Machine Learning in the Early Phases of Product Development." *Proceedings of the International Conference on Engineering Design, ICED 2019-Augus(AUGUST):2437–46*. doi: 10.1017/dsi.2019.250.
- Sharda, Ramesh, Dursun Delen, Efraim Turban, J. Aronson, and T. Liang. 2014. *Business Intelligence and Analytics. System for Decesion Support*.
- Sim, Siang Kok, and Alex H. B. Duffy. 2003. *Towards an Ontology of Generic Engineering Design Activities*. Vol. 14.
- Singh, Abhinav, and Conrad S. Tucker. 2017. "A Machine Learning Approach to Product Review Disambiguation Based on Function, Form and Behavior Classification." *Decision Support Systems* 97:81–91. doi: 10.1016/j.dss.2017.03.007.
- Singh, Jang Bahadur, M. Vimalkumar, Chandwani, Rajesh, and Biju Varkkey. 2020. "Machine Learning and Grounded Theory: New Opportunities for Mixed-Design Research." *Americas Conference on Information Systems (AMCIS) 2020* 0–3.
- Smith, Justin G., and Reid Tissing. 2018. "Using Computational Text Classification for Qualitative Research and Evaluation in Extension." *Journal of Extension* 56(2).
- Sommerville, Ian, and Pete Sawyer. 2003. "Requirements Engineering - A Good Practice Guide." 404.
- Song, Kaitao, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. "MPNet: Masked and Permuted Pre-Training for Language Understanding." (NeurIPS):1–14.

- Sousa, Fernando, Manuela Aparicio, and Carlos J. Costa. 2010. "Organizational Wiki as a Knowledge Management Tool." *SIGDOC 2010 - Proceedings of the 28th ACM International Conference on Design of Communication* 33–39. doi: 10.1145/1878450.1878457.
- Sowa, John F. 1987. "Semantic Networks." in *Encyclopedia of Artificial Intelligence*.
- Stanik, Christoph, Marlo Haering, and Walid Maalej. 2019. "Classifying Multilingual User Feedback Using Traditional Machine Learning and Deep Learning." Pp. 220–26 in *Proceedings - 2019 IEEE 27th International Requirements Engineering Conference Workshops, REW 2019*. IEEE.
- Starks, Helene, and Susan Brown Trinidad. 2007. "Choose Your Method: A Comparison of Phenomenology, Discourse Analysis, and Grounded Theory." *Qualitative Health Research* 17(10):1372–80. doi: 10.1177/1049732307307031.
- Stone, Thomas, and Seung Kyum Choi. 2013. "Extracting Consumer Preference from User-Generated Content Sources Using Classification." Pp. 1–9 in *Proceedings of the ASME Design Engineering Technical Conference*. Vol. 3 A.
- Storey, Vede C. 1993. "Understanding Semantic Relationships." *The VLDB Journal* 2(4):455–88. doi: 10.1007/BF01263048.
- Štorga, Mario, Mogens Myrup Andreasen, and Dorian Marjanović. 2010. "The Design Ontology: Foundation for the Design Knowledge Exchange and Management." *Journal of Engineering Design* 21(4):427–54. doi: 10.1080/09544820802322557.
- Studer, Rudi, V. Richard Benjamins, and Dieter Fensel. 1998. "Knowledge Engineering: Principles and Methods." *Data and Knowledge Engineering* 25(1–2):161–97. doi: 10.1016/S0169-023X(97)00056-6.
- Suryadi, Dedy, and Harrison M. Kim. 2019. "A Data-Driven Approach to Product Usage Context Identification from Online Customer Reviews." *Journal of Mechanical Design, Transactions of the ASME* 141(12):1–13. doi: 10.1115/1.4044523.
- Sutton, Richard S., and Andrew G. Barto. 2018. *Reinforcement Learning: An Introduction 2nd*.
- Taj, Soonh, Qasim Arain, Imran Memon, and Asma Zubedi. 2019. "To Apply Data Mining for Classification of Crowd Sourced Software Requirements." *PervasiveHealth: Pervasive Computing Technologies for Healthcare* 42–46. doi: 10.1145/3328833.3328837.
- Tamai, Tetsuo, and Taichi Anzai. 2018. "Quality Requirements Analysis with Machine Learning." *ENASE 2018 - Proceedings of the 13th International Conference on Evaluation of Novel Approaches to Software Engineering 2018-March*(Enase 2018):241–48. doi: 10.5220/0006694502410248.
- Tan, Suo, Yong Zeng, Greg Huet, and Clément Fortin. 2013. "Effective Reverse Engineering of Qualitative Design Knowledge: A Case Study of Aerospace Pylon Design." *Proceedings of the ASME Design Engineering Technical Conference* 4(November 2017). doi:

10.1115/DETC2013-13006.

- Tan, Suo, Yong Zeng, and Aram Montazami. 2011. "Medical Devices Design Based on Ebd: A Case Study." *Biomedical Engineering* 3–15. doi: 10.1007/978-1-4614-0116-2_1.
- Tang, Min, Jian Jin, Ying Liu, Chunping Li, and Weiwen Zhang. 2019. "Integrating Topic, Sentiment, and Syntax for Modeling Online Reviews: A Topic Model Approach." *Journal of Computing and Information Science in Engineering* 19(1):1–12. doi: 10.1115/1.4041475.
- Targowski, Andrew. 2005. "From Data to Wisdom." *Dialogue and Universalism* 15(5):55–71. doi: 10.5840/du2005155/629.
- Tatlisu, Enver, and Çiğdem Kaya. 2017. "The Reflection of Experiential Knowledge Into Professional Practice: Case of Industrial Design Education." *Design Journal* 20(sup1):S1415–29. doi: 10.1080/14606925.2017.1352667.
- Thanh An, and Yong Zeng. 2012. "A Theoretical Model of Design Creativity: Nonlinear Design Dynamics and Mental Stress-Creativity Relation." *Journal of Integrated Design and Process Science* 16(3):65–88. doi: 10.3233/jid-2012-0007.
- Timoshenko, Artem, and John R. Hauser. 2019. "Identifying Customer Needs from User-Generated Content." *Marketing Science* 38(1):1–20. doi: 10.1287/mksc.2018.1123.
- Tiwana, Amrit, and Balasubramaniam Ramesh. 2001. "A Design Knowledge Management System to Support Collaborative Information Product Evolution." *Decision Support Systems* 31(2):241–62. doi: 10.1016/S0167-9236(00)00134-2.
- Torres-Carrión, Pablo Vicente, Silvana Aciar, Carine Soledad González-González, and Germanía Rodríguez-Morales. 2018. "Methodology for Systematic Literature Review Applied to Engineering and Education." *2018 IEEE Global Engineering Education Conference (EDUCON)* 1364–73.
- Tóth, László, and László Vidács. 2018. "Study of Various Classifiers for Identification and Classification of Non-Functional Requirements." *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10964 LNCS:492–503. doi: 10.1007/978-3-319-95174-4_39.
- Turner, Stephen P. 2015. "What Do We Mean by Data-Driven?" Pp. 1–17 in *Creating a Data-Driven Organization : Practical Advice From the Trenches*.
- Ur-Rahman, N., and J. A. Harding. 2012. "Textual Data Mining for Industrial Knowledge Management and Text Classification: A Business Oriented Approach." *Expert Systems with Applications* 39(5):4729–39. doi: 10.1016/j.eswa.2011.09.124.
- US DoD. 2018. "Digital Engineering Strategy."
- Verner, June, Jennifer Sampson, and Narciso Cerpa. 2008. "What Factors Lead to Software Project Failure?" *Proceedings of the 2nd International Conference on Research Challenges in*

Information Science, RCIS 2008 71–79. doi: 10.1109/RCIS.2008.4632095.

- Vinyals, Oriol, Igor Babuschkin, Wojciech M. Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H. Choi, Richard Powell, Timo Ewalds, Petko Georgiev, Junhyuk Oh, Dan Horgan, Manuel Kroiss, Ivo Danihelka, Aja Huang, Laurent Sifre, Trevor Cai, John P. Agapiou, Max Jaderberg, Alexander S. Vezhnevets, Rémi Leblond, Tobias Pohlen, Valentin Dalibard, David Budden, Yury Sulsky, James Molloy, Tom L. Paine, Caglar Gulcehre, Ziyu Wang, Tobias Pfaff, Yuhuai Wu, Roman Ring, Dani Yogatama, Dario Wünsch, Katrina McKinney, Oliver Smith, Tom Schaul, Timothy Lillicrap, Koray Kavukcuoglu, Demis Hassabis, Chris Apps, and David Silver. 2019. “Grandmaster Level in StarCraft II Using Multi-Agent Reinforcement Learning.” *Nature* 575(7782):350–54. doi: 10.1038/s41586-019-1724-z.
- Wang, Chong, Fan Zhang, Peng Liang, Maya Daneva, and Marten Van Sinderen. 2018. “Can App Changelogs Improve Requirements Classification from App Reviews?: An Exploratory Study.” *International Symposium on Empirical Software Engineering and Measurement*. doi: 10.1145/3239235.3267428.
- Wang, Min, and Yong Zeng. 2009. “Asking the Right Questions to Elicit Product Requirements.” *International Journal of Computer Integrated Manufacturing* 22(4):283–98. doi: 10.1080/09511920802232902.
- Wang, Wenxin, Yi Feng, and Wenqiang Dai. 2018. “Topic Analysis of Online Reviews for Two Competitive Products Using Latent Dirichlet Allocation.” *Electronic Commerce Research and Applications* 29(April):142–56. doi: 10.1016/j.elerap.2018.04.003.
- Wang, Xiaoying, and Yong Zeng. 2017. “Organizational Capability Model: Toward Improving Organizational Performance.” *Journal of Integrated Design and Process Science*. doi: 10.3233/jid-2017-0005.
- Wang, Yinglin. 2016. “Automatic Semantic Analysis of Software Requirements through Machine Learning and Ontology Approach.” *Journal of Shanghai Jiaotong University (Science)* 21(6):692–701. doi: 10.1007/s12204-016-1783-3.
- Wang, Zuoxu, Chun Hsien Chen, Pai Zheng, Xinyu Li, and Li Pheng Khoo. 2019. “A Novel Data-Driven Graph-Based Requirement Elicitation Framework in the Smart Product-Service System Context.” *Advanced Engineering Informatics* 42(September):100983. doi: 10.1016/j.aei.2019.100983.
- Winkler, Jonas, and Andreas Vogelsang. 2017. “Automatic Classification of Requirements Based on Convolutional Neural Networks.” Pp. 39–45 in *Proceedings - 2016 IEEE 24th International Requirements Engineering Conference Workshops, REW 2016*. IEEE.
- Wohlin, Claes. 2014. “Guidelines for Snowballing in Systematic Literature Studies and a Replication in Software Engineering.” *ACM International Conference Proceeding Series*. doi: 10.1145/2601248.2601268.
- Wong, Lenis R., David Mauricio, and Glen D. Rodriguez. 2017. “A Systematic Literature Review

- about Software Requirements Elicitation.” *Journal of Engineering Science and Technology* 12(2):296–317.
- Wong, W. L. ..., and D. F. Radcliffe. 2000. “The Tacit Nature of Design Knowledge.” *Technology Analysis and Strategic Management* 12(4):493–512. doi: 10.1080/713698497.
- Wong, Wilson, Wei Liu, and Mohammed Bennamoun. 2012. “Ontology Learning from Text: A Look Back and into the Future.” *ACM Computing Surveys* 44(4):1–36. doi: 10.1145/2333112.2333115.
- Wu, Chien Hsing, and K. Srihari. 1996. “Automated Knowledge Acquisition for the Surface-Mount PWB Assembly Domain.” *Journal of Intelligent Manufacturing* 7(5):393–98. doi: 10.1007/BF00123916.
- Wu, Zhibiao, and Martha Palmer. 1994. “Verbs Semantics and Lexical Selection.” 133–38. doi: 10.3115/981732.981751.
- Xiao, Yu, and Maria Watson. 2019. “Guidance on Conducting a Systematic Literature Review.” *Journal of Planning Education and Research* 39(1):93–112. doi: 10.1177/0739456X17723971.
- Xu, Zhaoguang, Yanzhong Dang, Peter Munro, and Yuhang Wang. 2020. “A Data-Driven Approach for Constructing the Component-Failure Mode Matrix for FMEA.” *Journal of Intelligent Manufacturing* 31(1):249–65. doi: 10.1007/s10845-019-01466-z.
- Yan, Wei, Chun Hsien Chen, and Li Pheng Khoo. 2002. “An Integrated Approach to the Elicitation of Customer Requirements for Engineering Design Using Picture Sorts and Fuzzy Evaluation.” *Artificial Intelligence for Engineering Design, Analysis and Manufacturing: AIEDAM* 16(2):59–71. doi: 10.1017/S0890060402020061.
- Yang, Bai, Ying Liu, Yan Liang, and Min Tang. 2019. “Exploiting User Experience from Online Customer Reviews for Product Design.” *International Journal of Information Management* 46(May 2018):173–86. doi: 10.1016/j.ijinfomgt.2018.12.006.
- Yang, Jiami, Lin Yang, Hude Quan, and Yong Zeng. 2021. “Implementation Barriers: A TASKS Framework.” *Journal of Integrated Design and Process Science* 25(X):1–14. doi: 10.3233/jid-210011.
- Yi, Hong, Xiaoguang Deng, and Yong Zeng. 2014. “Curriculum Design Using EBD Methodology: Preliminary Study of English Education in Mid-West University of China.” *Proceedings of the 2014 International Conference on Innovative Design and Manufacturing, ICIDM 2014* 17:282–87. doi: 10.1109/IDAM.2014.6912708.
- Yoshioka, M., T. Sekiya, and T. Tomiyama. 1998. “Design Knowledge Collection by Modeling.” *Globalization of Manufacturing in the Digital Communications Era of the 21st Century* 287–98. doi: 10.1007/978-0-387-35351-7_23.
- Zeng, Yong., and P. Gu. 1999. “A Science-Based Approach to Product Design Theory. Part II:

- Formulation of Design Requirements and Products.” *Robotics and Computer-Integrated Manufacturing* 15(4):341–52. doi: 10.1016/S0736-5845(99)00029-0.
- Zeng, Yong. 2002. “Axiomatic Theory of Design Modeling.” *Journal of Integrated Design and Process Science* 6(3):1–28.
- Zeng, Yong. 2004. “ENVIRONMENT-BASED FORMULATION OF DESIGN PROBLEM.” *Journal of Integrated Design and Process Science* 8(4):45–63.
- Zeng, Yong. 2008. “Recursive Object Model (ROM)-Modelling of Linguistic Information in Engineering Design.” *Computers in Industry* 59(6):612–25. doi: 10.1016/j.compind.2008.03.002.
- Zeng, Yong. 2011. “Environment-Based Design (EBD).” Pp. 1–14 in *Proceedings of the ASME 2011 International Design Engineering Technical Conferences & Computers and Information in Engineering Conference*.
- Zeng, Yong. 2020. “Environment: The First Thing to Look at in Conceptual Design.” *Journal of Integrated Design and Process Science* 24(1):45–66. doi: 10.3233/JID200005.
- Zeng, Yong, and G. .. Cheng. 1991. “On the Logic of Design.” *Design Studies* 137–41.
- Zeng, Yong, Lingyu Wang, Xiaoguang Deng, Xinlin Cao, and Nafisa Khundker. 2012. “Secure Collaboration in Global Design and Supply Chain Environment: Problem Analysis and Literature Review.” *Computers in Industry* 63(6):545–56. doi: 10.1016/j.compind.2012.05.001.
- Zhan, Jiaming, Han Tong Loh, and Ying Liu. 2009. “Gather Customer Concerns from Online Product Reviews - A Text Summarization Approach.” *Expert Systems with Applications* 36(2 PART 1):2107–15. doi: 10.1016/j.eswa.2007.12.039.
- Zhang, Dongmin, Dachao Hu, Yuchun Xu, and Hong Zhang. 2012. “A Framework for Design Knowledge Management and Reuse for Product-Service Systems in Construction Machinery Industry.” *Computers in Industry* 63(4):328–37. doi: 10.1016/j.compind.2012.02.008.
- Zheng, Guanjie, Fuzheng Zhang, Zihan Zheng, Yang Xiang, Nicholas Jing Yuan, Xing Xie, and Zhenhui Li. 2018. “DRN: A Deep Reinforcement Learning Framework for News Recommendation.” *The Web Conference 2018 - Proceedings of the World Wide Web Conference, WWW 2018* 2:167–76. doi: 10.1145/3178876.3185994.
- Zhou, Feng, Jackie Ayoub, Qianli Xu, and X. Jessie Yang. 2020. “A Machine Learning Approach to Customer Needs Analysis for Product Ecosystems.” *Journal of Mechanical Design, Transactions of the ASME* 142(1):1–14. doi: 10.1115/1.4044435.
- Zhou, Qingyu, Nan Yang, Furu Wei, Chuanqi Tan, Hangbo Bao, and Ming Zhou. 2018. “Neural Question Generation from Text: A Preliminary Study.” *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)* 10619 LNAI:662–71. doi: 10.1007/978-3-319-73618-1_56.

- Zimmerman, Philomena. 2017. "DoD Digital Engineering Strategy." in *20th Annual NDIA Systems Engineering Conference, Springfield, VA, US, October 23-26, 2017*.
- Zins, Chaim. 2007. "Conceptual Approaches for Defining Data, Information, and Knowledge." *Journal of the American Society for Information Science and Technology* 58(4):479–93. doi: 10.1002/asi.
- Zowghi, Didar, and Chad Coulin. 2005. "Requirements Elicitation: A Survey of Techniques, Approaches, and Tools." *Engineering and Managing Software Requirements* 19–46. doi: 10.1007/3-540-28244-0_2.