Using Commonsense Knowledge and Text Mining for Implicit Requirements Localization

Conference Paper · November 2020

DOI: 10.1109/ICTAI50040.2020.00146

5 authors, including:

Onyeka Emebo
Covenant University Ota Ogun State, Nigeria
18 PUBLICATIONS **96** CITATIONS
SEE PROFILE

Aparna S. Varde
Montclair State University
125 PUBLICATIONS **662** CITATIONS
SEE PROFILE

Vaibhav Anu
Montclair State University
28 PUBLICATIONS **117** CITATIONS
SEE PROFILE

Olawande Daramola
81 PUBLICATIONS **443** CITATIONS
SEE PROFILE

Using Commonsense Knowledge and Text Mining for Implicit Requirements Localization

Emebo Onyeka[1], Aparna S. Varde[2†], Vaibhav Anu[2‡], Niket Tandon[3], Olawande Daramola[4]
Covenant University (Ota, Nigeria)[1], Montclair State University (Montclair, NJ)[††],
Allen Institute for Artificial Intelligence (Seattle, WA)[3], Cape Peninsula University of Technology (Cape Town, South Africa)[4]
onye.emebo@covenantuniversity.edu.ng[1], vardea@montclair.edu[2†], anuv@montclair.edu[2‡],
nikett@allenai.org[3], daramolaj@cput.ac.za[4]

***Abstract*—This paper addresses identification of implicit requirements (IMRs) in software requirements specifications (SRS). IMRs, as opposed to explicit requirements, are not specified by users but are more subtle. It has been noticed that IMRs are crucial to the success of software development. In this paper, we demonstrate a software tool called *COTIR* developed by us as a system that integrates *Commonsense knowledge, Ontology and Text mining for early identification of Implicit Requirements*. This relieves human software engineers from the tedious task of manually identifying IMRs in huge SRS documents. Our evaluation reveals that COTIR outperforms existing IMR tools. This demo paper would be useful to Software Engineers since it deals with automation in the requirements analysis phase, thus contributing to Requirements Engineering. It would interest AI scientists as it entails multi-disciplinary work encompassing text mining, ontology and commonsense knowledge. It makes a broader impact on Smart Cities, because automated identification of IMRs would offer inputs to Smart City Tools, where requirements may often be implicit given that Smart Cities are an emerging and growing paradigm.**

***Index Terms*—AI in Smart Cities; Commonsense Knowledge; Implicit Requirements; Ontology; Requirements Engineering; Software Demo; Text Mining**

I. Introduction

Background and Motivation: Requirements Engineering (RE) is a systematic process with many activities. A very important RE activity is requirements elicitation that harvests requirements (functional and non-functional) from stakeholders. Requirements can be classified into explicit and implicit. Explicit requirements are clearly stated and well-defined ones that a system should execute. Implicit Requirements (IMRs) are assumed or hidden requirements that a software system is expected to fulfill, though not directly captured during elicitation. The success or failure of a software development system highly depends on IMRs [1], [2]. As documented in the literature, there are certain features that can possibly make a natural language text implicit. These are: ambiguity including structural and lexical aspects; vague words and phrases like "to a great extent"; imprecise verbs, e.g. "supported", "handled", "processed", "rejected"; weak phrases such as "normally", "generally" and incomplete knowledge, i.e. lack of further information on any requirement(s). An example of IMR-identification in a requirements case study is as follows [3].

Example IMR Scenario: It was found that the following explicit requirements (R1, R2) were documented for the development of a web-portal for a publishing firm.

i. **R1**: The system shall allow a publisher to create an article, send it for approval and finally publish the article on the portal.

ii. **R2**: The system shall allow a portal user to search for articles published on the portal.

The goal was to develop a web-portal that allows articles to be created and published for users of the system to search and use the articles. However, when the system was developed and installed at the client station, the search displayed results of articles that were in draft, under approval and published. This was unsatisfactory due to an ambiguous phrase ”to search” which was not further elicited. An implicit requirement was missing as further explicated by the requirements engineer below(as analyzed in the literature [3]).

*i. **IMR1**: The system shall only display articles that have been published.*

Issues such as these motivate early identification of IMRs. Since manually identfying IMRs is tedious, infeasible and unscalable, particularly due to huge data in SRS documents and limitations of human working memory, there is need for automation. Existing works addressing IMRs such as [4]–[7] lack intuitive human judgment. Thus, there is a need for incorporating this crucial aspect in automating IMR identification.

The COTIR Tool: In this work we demonstrate an AI tool called *COTIR*, which stands for *Commonsense knowledge, Ontology and Text mining for Implicit Requirements*. As the name implies, this tool integrates commonsense concepts, ontological aspects and mining of textual data to help identify those areas of explicit requirements where relevant IMRs may be hiding. COTIR is thus an ***IMR-source localization*** tool. To the best of our knowledge, this is the first AI tool embodying commonsense knowledge (CSK) for IMR detection to simulate human reasoning. We find ontology imperative here, as it facilitates formalized semantic description of relevant domain knowledge for IMRs. Text mining involves analyzing text to extract useful information and is thus significant in SRS analysis to understand textual similarities in SRS, identify a basis for analogy and discover knowledge for IMRs. Our proposed system, COTIR, comprises CSK for inferring subtleties, ontology for specifying formalisms and text mining for

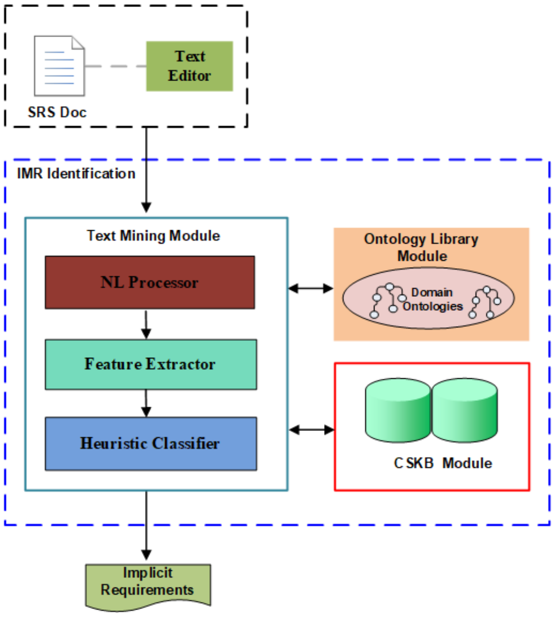


Fig. 1. The COTIR Pipeline

knowledge discovery. COTIR shows significant improvement over state-of-the-art as elaborated later. This demo paper is based on a tool developed using our COTIR approach [8].

II. COTIR System Architecture

The pipeline of the COTIR tool appears in Fig. 1. Its core system functionalities are depicted as rectangular boxes, while the logic, data and knowledge artefacts enabling functionalities are shown in oval boxes.

Data Input: The input to COTIR is a preprocessed SRS document. Preprocessing entails extracting sentences, and depicting images, tables etc. with equivalent textual formats.

NL Processor: This is a core component of the text mining module that facilitates processing of natural language requirements for use by the feature extractor. The implemented natural language processing (NLP) operations are i) Sentence selection, ii) Tokenization, iii) Parts of speech (POS) tagging, iv) Entity detection, and v) Parsing. The Apache OpenNLP library is used to implement NLP operations.

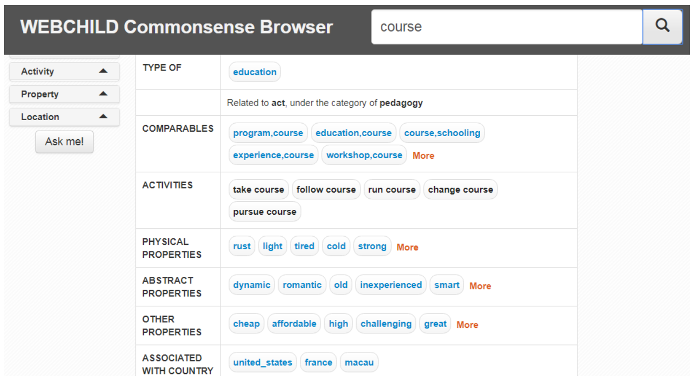
Ontology Library: The Ontology Library (OL) module and the Commonsense Knowledge Base (CSKB) module form the COTIR backbone, serving as knowledge representation (KR) in domain ontology for specific purposes / general business rules. The ontology library has been built using Java Protege 4.1 ontology API. Formally an ontology structure O is defined as O={C, R, A^o} where:

1. C is a set whose elements are called concepts
2. $R \subseteq CXC$ is a set whose elements are called relations [For example, $r = (c1, c2) \in R$ is written $r(c1) = c2)$]
3. A^o is a set of axioms on O

Commonsense Knowledge Base (CSKB): A commonsense knowledge (CSK) source called WebChild [9] and related domain-specific KBs [10] constitute the CSKB used to guide the identification of IMRs for specific domains. CSK is crucial in identification of IMRs, especially for the disambiguation of various concepts in SRS documents (elaborated later).

Feature Extractor: This module offers essential rules to find sources of IMRs in SRS documents. Some features that can make a natural language text implicit are documented in the literature [2], [11]. These are: ambiguity including structural and lexical aspects; vague words and phrases like “to a great extent”; imprecise verbs, e.g. “supported”, “handled”, “processed”, “rejected”; weak phrases such as “normally”, “generally” and incomplete knowledge, i.e. lack of further information on any requirement(s).

Heuristic Classifier: This is the final module, responsible for classifying the actual requirements in SRS using intermediate outputs of the previous modules. Thus it helps identify IMR-sources, i.e. the ultimate output of COTIR.



TYPE OF	education
	Related to **act**, under the category of **pedagogy**
COMPARABLES	program,course · education,course · course,schooling · experience,course · workshop,course · More
ACTIVITIES	take course · follow course · run course · change course · pursue course
PHYSICAL PROPERTIES	rust · light · tired · cold · strong · More
ABSTRACT PROPERTIES	dynamic · romantic · old · inexperienced · smart · More
OTHER PROPERTIES	cheap · affordable · high · challenging · great · More
ASSOCIATED WITH COUNTRY	united_states · france · macau

Fig. 2. Relevant Partial Screenshot of WebChild

III. Functioning of COTIR

The COTIR tool functions primarily based on the aspects described herewith: CSK, text mining and ontology.

Role of Commonsense Knowledge: To describe the role played by CSK in COTIR, we provide a screenshot of our main CSK source WebChild in Fig. 2. WebChild [9] is a huge knowledge base of concepts, properties and relationships derived from the web that represent everyday commonsense aspects. Hence, this aids the inferring of subtle knowledge on IMRs (since these refer to subtle requirements not explicitly identified). The screenshot in Fig. 2 refers to a commonsense term “course” in education. Due to commonsense concepts, properties and relationships, WebChild allows us to distinguish this *academic course* from a course in a 3-course meal or course in the route of an aircraft. Likewise, many concepts are resolved with respect to ambiguity, vagueness etc. Hence, WebChild and relevant domain-specific knowledge bases form the CSKB module of COTIR. Actual meanings of terms in SRS documents and their relationships can better be determined with CSKBs. Thus it is a crucial module.

Text Mining using CSKB and Ontology: The text mining functionality of COTIR, part of its NL Processor is illustrated in Fig. 3. Raw text from SRS documents forms the input to this module. It conducts sentence segmentation to output strings,

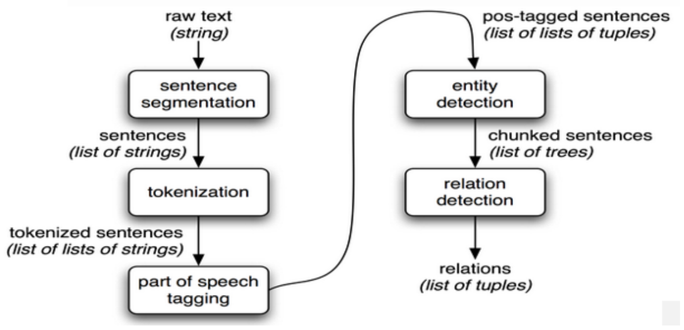


Fig. 3. Text Mining in NL Processor

subject to tokenization. The tokenized sentences undergo POS (part of speech) tagging. These POS tagged sentences are subject to entity detection. This gives chunked sentences as a list of trees that undergo relation detection. The ontology library module plays an important role in identifying these entities and relations, using the ontology structure O defined as O=$\{C, R, A^o\}$ of concepts, relations and axioms respectively. Further, the CSKB module helps resolve ambiguities and vagueness in entities and relations. It augments incomplete knowledge via a probabilistic domain classifier [10]. This is useful since we deal with subtle implicit requirements.

IV. System Demonstration

We provide a system demo of the COTIR tool with various snapshots. Consider an Example Scenario on CMS (Course Management System) [3].

Example CMS Scenario; A software developer is working on the design and implementation of a Course Management System. SRS is based on explicit requirements available. This is the input to COTIR through a screen (see Fig. 4).

The top left part of the screen allows selection of required functions. In this example, consider that a software developer selects "Analysis" in the categories "Lexical (General)", "Lexical (Context Disambiguation)" and "Vagueness". The right side of screen shows all requirements in the given scenario. An excerpt appears in Fig. 5 with a few requirements.

Considering these requirements, the text source for our scenario uses the CMS SRS document from [12]. This project is developed at the University of Twente. The requirements for the CMS describe the basic functionality expected such as course enrollment, course material and roster upload, student grading and e-mails communication. Based on this, an example of a concept class (COURSE) from the CSKBs of COTIR appears in Table I.

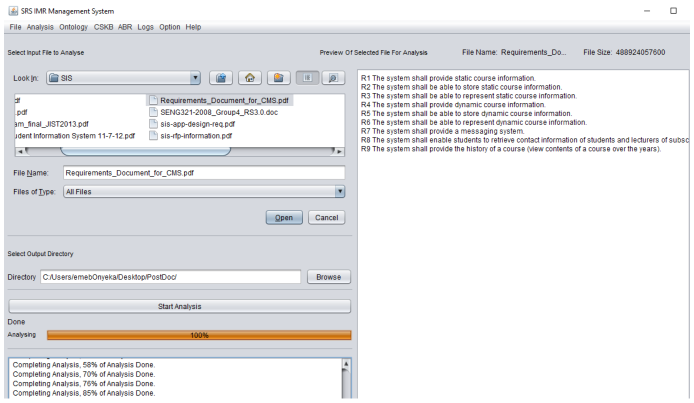
Attributes	Type, Description, Code, Unit, Department, Category, Weight
Relations	Is-offered-by, is-prerequisite-to, belongs-to
Interactions	Register, Evaluate, Teach, Recommend, Service

TABLE I
Example of COURSE Concept Class

Accordingly, in the given scenario, the following steps would be taken by the COTIR system. (1) **Preprocess:** Source documents are converted to get requirements in text format (without graphics, tables etc.) (2) **Select:** Relevant commonsense knowledge bases (CSKBs) are chosen for the identification of IMRs. (3) **Import:** SRS documents and domain ontology are transferred to the COTIR environment. (4) **Analyze:** Possible sources of IMRs are outlined by the feature extractor module. (5) **Recommend:** Potential IMR-sources are detected with suitable recommendations. (6) **Manage:** The recommendations are used to handle IMRs, this may include expert opinion. A partial snapshot of the output after the analysis, to identify the potential sources of IMRs contained in the SRS, appears in Fig. 6. This refers to a lexical ambiguity report on IMRs. The procedure in COTIR to identify IMR sources is summarized in Algorithm 1 here.

It should be noted here that the CSK source WebChild has over 2 million disambiguated concepts. This aids in improving discovery of ambiguities for augmenting IMR-source identification. To emphasize the importance of CSK, the COTIR framework is experimented without the CSKB module and it is observed that fewer lexical ambiguities are found. If the CSKB module is integrated, extra word/phrases, lexical in nature, are found making the total number of lexical ambiguities discovered greater.

Hence, the presence of CSK enhances the discovery and explication ability of COTIR. In this scenario, the Lexical Am-


SRS IMR Management System
File Analysis Ontology CSKB ABR Logs Option Help
Select Input File to Analyse
Preview Of Selected File For Analysis
Look In: SIS
Open
Cancel
Select Output Directory
Browse
Start Analysis
Done
Analysing
100%

File Name: Requirements_Do... File Size: 488924057600

df
.pdf
am_final_JIST2013.pdf
udent Information System 11-7-12.pdf
Requirements_Document_for_CMS.pdf
SENG321-2008_Group4_RS3.0.doc
sis-app-design-req.pdf
sis-rfp-information.pdf

File Name: Requirements_Document_for_CMS.pdf
Files of Type: All Files
Directory C:/Users/emebOnyeka/Desktop/PostDoc/

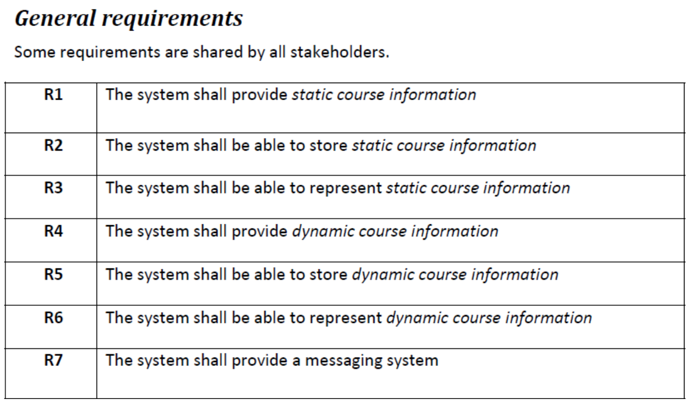
R1 The system shall provide static course information.
R2 The system shall be able to store static course information.
R3 The system shall be able to represent static course information.
R4 The system shall provide dynamic course information.
R5 The system shall be able to store dynamic course information.
R6 The system shall be able to represent dynamic course information.
R7 The system shall provide a messaging system.
R8 The system shall enable students to retrieve contact information of students and lecturers of subsc
R9 The system shall provide the history of a course (view contents of a course over the years).

```
Completing Analysis, 58% of Analysis Done.
Completing Analysis, 70% of Analysis Done.
Completing Analysis, 76% of Analysis Done.
Completing Analysis, 85% of Analysis Done.
```

Fig. 4. Demo Snapshot of COTIR Input and Analysis Screen

```
Algorithm 1: IMR-SOURCE IDENTIFICATION
Input: A vector T of token of words from each Requirement ID $R_{id}$,
Output: A vector IMR containing the list of words recommendation that are Implicit in $R_{id}$.

Initialize T
[Check Req for Vague]
for all tokens $t_i \in T$ ordered by each Requirements $R_{id}$ in ascending order
    seek(POS)
    if POS == Adv || POS == Adj
        Add $t_i$ to IMR [tag as vague]
[check Req for Ambiguity]
    else if POS == RB || POS == RBS || POS == RBR
     checkDict( t ) [return sense]
        if sense > 1
        add $t_i$ to IMR [tag as ambiguous]
    for each IMR in $R_{id}$
        while $Reqid_i < Req_i$
            if IMR > 1
         display Req [with IMR tag]
```


General requirements

Some requirements are shared by all stakeholders.

R1	The system shall provide *static course information*
R2	The system shall be able to store *static course information*
R3	The system shall be able to represent *static course information*
R4	The system shall provide *dynamic course information*
R5	The system shall be able to store *dynamic course information*
R6	The system shall be able to represent *dynamic course information*
R7	The system shall provide a messaging system

Fig. 5. Excerpt from CMS Requirements Specification

biguity Report output by COTIR (based on functions selected by the software developer), serves to identify ambiguities in the SRS, constituting IMRs that need to be further addressed. These ambiguities, highlighted in red, indicate that the corresponding requirements are implicit and must be clarified to have a better SRS before proceeding with development. This helps the software developer contact respective consultants who can provide expert opinion for enhancing the SRS. In the absence of such IMR-source detection, the SRS would have retained ambiguities which at a later stage would cause problems through bugs and/or client dissatisfaction. Software developers using our COTIR system have confirmed that it is useful for early identification and management of IMRs [3].

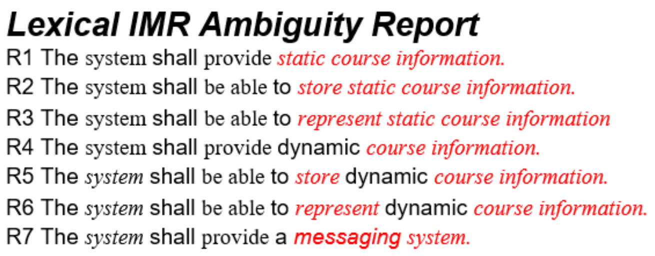
V. Experimental Evaluation

The COTIR tool is evaluated with real data for software development. Besides CMS, it is evaluated in Smart Cities and Tactical Control Systems [3], [8]. Ground truth is annotated by experts. Evaluation metrics are Recall $R = TP/(TP+FN)$, Precision $P = TP/(TP+FP)$ and F-score $F = 2P \times R/(P+R)$ where *TP, TN, FP, FN* are true positives, true negatives, false positives, false negatives respectively. (TP: requirements judged by expert and COTIR as implicit, TN: both as explicit, FP: requirements judged by COTIR as implicit and expert as explicit, FN: vice versa). Evaluation reveals average $R = 73.7\%, P = 68.22\%, F = 70.3\%$, pursuant with best practices in software engineering. Comparative assessment of COTIR is conducted with related tools in the literature, namely, NAI, SR-Elicitor (abbreviated as SR-E) and ARUgen (abbreviated as ARU) [4], [5], [11]. The summarized assessment results are in Table II where "T" and"C" stand for any other "Tool" and "COTIR" respectively, "L.A" and "S.A" stand for "Lexical Ambiguity" and "Structural Ambiguity" respectively, while "V" stands for "Vagueness". Comparison shows that for Lexical and Structural Ambiguity, COTIR is better than NAI and SR-Elicitor (Recall, F-Score); and is almost at par in Precision. For Vagueness, COTIR does better that ARUgen across all metrics. Hence, we can infer from evaluation that COTIR outperforms the state-of-the-art.

On the practical side, it is observed that COTIR can augment software implementation, by reducing software defects at the

Tools	Test	Recall (%)		Precision (%)		F-Score (%)	
		T	C	T	C	T	C
NAI	L.A.	70	74.2	85.4	84.36	77.73	78.28
	S.A.	82.4	85.75	84.2	83.91	82.7	83.34
SR-E	L.A.	80.12	84.22	85.76	85.1	79.4	81.23
ARU	V.	87.51	89.63	91.12	93.51	89.28	90.71

TABLE II
Comparative Performance of COTIR


Lexical IMR Ambiguity Report

R1 The system shall provide *static course information.*
R2 The system shall be able to *store static course information.*
R3 The system shall be able to *represent static course information*
R4 The system shall provide dynamic *course information.*
R5 The *system* shall be able to *store* dynamic *course information.*
R6 The *system* shall be able to *represent* dynamic *course information.*
R7 The *system* shall provide a ***messaging*** *system.*

Fig. 6. Demo Snapshot with Example of COTIR Output

RE phase by around 20% thus enhancing overall software development efforts by around 10% on an average, as seen in real developmental phases embodying the COTIR tool. [3]. Also, evaluation is conducted with and without the CSK module. Evaluation with CSK discovers IMR to the extent of 33% more on an average [3]. This highlights usefulness of early IMR identification with COTIR from a real-world standpoint, and significance of CSK within.

VI. Related Work

The work in [5] addresses IMRs through a natural language analytical approach. In [4], systems such as NAI, SR-elicitor and ARUgen that address IMRs are compared. Efforts such as [6], [7] engage analogy reasoning and related approaches with respect to IMRs. Andresel et al. propose "ontology-mediated queries (OMQs) that use domain knowledge to infer missing facts" [13]. Samer et al. propose methods to identify dependencies among requirements using AI with high "prediction quality" [14].

Commonsense Knowledge (CSK) is of much interest in AI [15]. WordNet [16] is a classical CSK source. WebChild [9] extracts commonsense concepts, properties and relationships from the web, comprising everyday aspects. CSK-SNIFFER [17] generates adversarial images using spatial commonsense helpful in object detection, especially in smart mobility [18].

Smart city research entails significant work in recent years. The work in [19] addresses optimality in smart grids, considering electricity costs and signal prediction. Research on neural networks and deep learning pertinent to smart mobility is presented in [20], leveraging CSK as well.

Given this literature study, COTIR fits within the areas of such work. It stands out since it brings together CSK with text mining and ontology for early identification of IMR sources.

VII. Applications: Smart City Perspectives

A. Ontology for CMS useful in Smart Cities

COTIR has been useful in the RE phase of a CMS (Course Management System). The course registration ontology in its

SRS document has comprised the following steps [3].

1) Outline the relevant classes in the ontology
2) Organize the classes in a taxonomic manner such that they form a subclass–superclass hierarchy
3) Define the necessary slots and accordingly describe the allowed values for these slots
4) Enter the actual values for slots based on instances
5) Explain the relationships among various classes

An important aspect is that *Student Matriculation Number* in the CMS "has been made a functional requirement since no two students can have the same matriculation number" [3]. This is an example of early IMR identification.

COTIR has been helpful in such ontology design and refinement in the context of CMS. It has saved time and effort in software development as a whole (approximately by 10 % as mentioned in the evaluation [3]) and has produced better results in the outcome of system development. Thus it makes broader impact on the *smart people* characteristic of Smart Cities [18] which entails advances in 21st century education. COTIR impacts this due to its usefulness in CMS, thus aiding automation in 21st century education.

B. RE for Smart Cities and Tactical Control

COTIR has been useful in the RE phase in software development for Smart Cities. There has been a project called EMMON (EMbedded MOnitoring) [21], a European Research and Development initiative. Europe has been very strong in the Smart Cities paradigm [18] with many leading Smart Cities being from there, e.g. Copenhagen (Denmark), Barcelona (Spain), Amsterdam (The Netherlands) etc. The inspiration for EMMON has derived its roots from much-growing emphasis on goals here, e.g. smart locations and ambient intelligent environments. These entail smart homes, smart public spaces, smart forests etc. In the achievement of such goals, the development of software systems leveraging embedded technology is critical. This promotes thriving of smart environments and makes contributions to scalable digital services that augment the overall quality of life. Since Smart Cities are fairly recent and growing, not all requirements may be available upfront while creating SRS documents as inputs for designing Smart City tools with embedded technology. This is where COTIR has been helpful in early identification of IMRs. It has been evaluated in the context of Smart City tools and has been found to reduce data collection time and enhance the overall software development process while also helping to reduce potential software defects [3]. It thus fits into the arena of other such research [19], [20].

Yet another context where COTIR has been useful is in the RE phase of a Tactical Control System (TCS). An interesting project here [22] has been an initiative by the Naval Surface Warfare Center (NSWC) Dahlgren Division and Joint Technology Center / System Integration Laboratory. This initiative has been pursued their Research Development and Engineering Center in the USA. COTIR has been useful in IMR identification here. Evaluation in TCS has yielded positive results [3], analogous to its evaluation in CMS. Hence COTIR's contribution to Smart Cities and Tactical Control can be perceived as a broader impact.

C. Envisaged Applications and Extension

We anticipate that COTIR has various application areas:

1) Automation of security requirements identification process can be conducted. Security Requirements frequently remain hidden and this causes insecure software release [23]. The COTIR tool can be used to resolve this issue through early detection of security related IMRs.
2) Supporting requirements elicitation in *Agile* environments can be explored. Research has shown Agile Software Development (ASD) frequently suffers from requirements omission [24]. Hence we anticipate that the COTIR tool can alleviate this issue in ASD via early identification of non-functional IMRs.
3) Deep learning can be integrated to enhance the current COTIR tool so that IMRs can be extracted from tables and graphs in the SRS documents. We briefly explain this application with respect to our proposed extension / enhancement to the COTIR tool herewith.

Envisaged COTIR Extension: Motivated by the success of COTIR's development and evaluation, we are in the process of proposing an extension to COTIR. This extension *(Enhanced COTIR)* aims to extract big data from SRS contents that go beyond plain text, i.e. data in tabular formats, figures entailing images in addition to text and numbers etc, These complex data items constitute a significant part of many SRS documents and deserve attention. Our short vision paper in this direction [25] summarizes our early efforts in the area.

We anticipate that Enhanced COTIR can make a better impact on Smart Cities. This is illustrated in Fig. 7 where we depict the proposed framework with its usefulness. Enhanced COTIR can identify IMRs from text, images, graphs and tables. Thus, outputs from Enhanced COTIR can form inputs to the RE phase of Smart City tools. This would obviously augment their development with better requirements.

VIII. Conclusions and Roadmap

This paper demonstrates an AI tool called COTIR to automate early identification of IMR sources in SRS documents. It embodies commonsense knowledge with ontology and text mining to detect IMR sources. As noticed in real software projects [3], the use of COTIR reduces software defects by approximately 10% and thereby enhances overall software quality by approximately 20%, on an average. COTIR makes a broader impact on AI in Smart Cities, analogous to other such works, e.g. [19], [20], since it is evaluated within Smart City tools (in addition to CMS shown here). In Smart City tools, there is great likelihood of implicitness, since Smart Cities are recently gaining importance and many tools are in their inception. As ongoing work [25], we would replace the heuristic classifier with a CNN based classifier entailing deep learning. Thereby, we would aim to find IMRs from images, tables etc. in SRS documents. This is expected to be highly

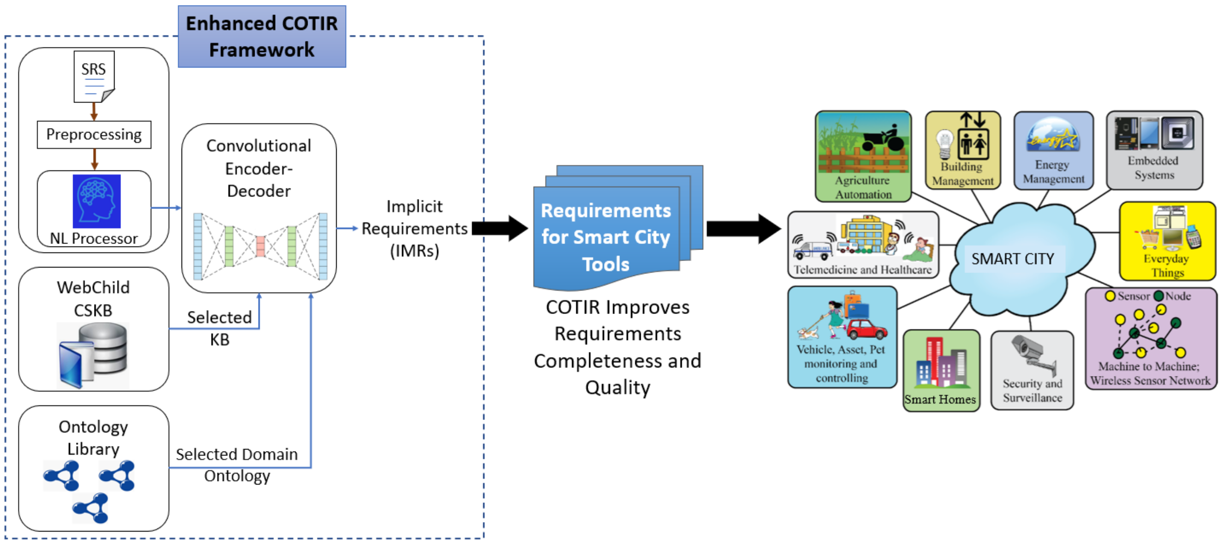

Enhanced COTIR Framework
SRS
Preprocessing
NL Processor
Convolutional Encoder-Decoder
Implicit Requirements (IMRs)
WebChild CSKB
Selected KB
Ontology Library
Selected Domain Ontology
Requirements for Smart City Tools
COTIR Improves Requirements Completeness and Quality
SMART CITY
Agriculture Automation
Building Management
Energy Management
Embedded Systems
Everyday Things
Telemedicine and Healthcare
Sensor
Node
Machine to Machine; Wireless Sensor Network
Vehicle, Asset, Pet monitoring and controlling
Smart Homes
Security and Surveillance

Fig. 7. Enhanced COTIR Framework and its Smart City Impacts

beneficial in software development and would make stronger impacts on the paradigm of AI in Smart Cities.

Acknowledgments

This research was supported by a Fulbright Scholarship Program for Onyeka Emebo when he was a Visiting PhD Candidate at Montclair State University. Early efforts in this area started when Aparna Varde was a Visiting Researcher at Max Planck Institute for Informatics (Saarbruecken, Germany) working on a project on commonsense knowledge. The rest of the funding is from internal sources in the authors' respective institutions. There is recent funding, NSF MRI Award Number 2018575, related to this work.

References

[1] H. Dreyer, M. Wynn, and G. Bown, "Tacit and Explicit Knowledge in Software Development Projects: Towards a Conceptual Framework for Analysis," in *eKnow*, 2015, pp. 49–52.

[2] F. Fabbrini, M. Fusani, S. Gnesi, and G. Lami, "An automatic quality evaluation for natural language requirements," in *Intl. Conf. on Requirements Engineering: Foundation for Software Quality (RE-FSQ)*, 2001, pp. 4–5.

[3] O. Emebo, "*A Process Framework for Managing Implicit Requirements Using Analogy-based Reasoning* - PhD Thesis, Covenant University, Ota, Nigeria," 2017.

[4] U. Shah and D. Jinwala, "Resolving ambiguities in natural language software requirements: a comprehensive survey," *ACM SIGSOFT Software Engineering Notes*, pp. 1–7, 2015.

[5] A. Umber, I. S. Bajwa, and M. Asif Naeem, "NL-based automated software requirements elicitation and specification," in *Communications in Computer and Information Science*, 2011, pp. 30–39.

[6] O. Daramola, T. Moser, G. Sindre, and S. Biffl, "Managing implicit requirements using semantic case-based reasoning research preview," in *Intl. Conf. on Requirements Engineering: Foundation for Software Quality*, 2012, pp. 172–178.

[7] O. Emebo, O. Daramola, and A. Charles, "An automated tool support for managing implicit requirements using Analogy-based Reasoning," in *Intl. Conf. on Research Challenges in Information Science (RCIS)*, 2016, pp. 1–6.

[8] O. Emebo and A. S. Varde, "Early identification of implicit requirements with the COTIR approach using common sense, ontology and text mining," Fulbright Scholarship Program, Montclair State University, Department of Computer Science, Montclair, NJ, Tech. Rep., 2016.

[9] N. Tandon, G. de Melo, F. Suchanek, and G. Weikum, "WebChild: Harvesting and organizing commonsense knowledge from the web," in *ACM Intl. Conf. on Web Search and Data Mining (WSDM)*, 2014, pp. 523–532.

[10] A. Varde, N. Tandon, S. Nag Chowdhury, and G. Weikum, "*Common Sense Knowledge in Domain Specific Knowledge Bases* - Technical Report, Max Planck Institute for Informatics, Saarbruecken, Germany," August 2015.

[11] W. Wilson, L. Rosenberg, and L. Hyatt, "Automated analysis of requirement specifications," in *Intl. Conf. on Software Engineering (ICSE)*, 1997, pp. 161–171.

[12] B. Abma, "*Evaluation of requirements management tools with support for traceability-based change impact analysis* - MS Thesis, University of Twente, Enschede, The Netherlands," 2009.

[13] M. Andresel, M. Ortiz, and M. Simkus, "Query rewriting for ontology-mediated conditional answers," in *AAAI Conf.*, 2020, pp. 2734–2741.

[14] R. Samer, M. Stettinger, M. Atas, A. Felfernig, G. Ruhe, and G. Deshpande, "New approaches to the identification of dependencies between requirements," in *IEEE ICTAI Conf.*, 2019, pp. 1265–1270.

[15] N. Tandon, A. Varde, and G. de Melo, "Commonsense knowledge in machine intelligence," *ACM SIGMOD Record*, vol. 46, pp. 49–52, 2017.

[16] G. Miller, "WordNet: A Lexical Database for English," *Communications of the ACM*, vol. 38, no. 11, pp. 39–41, 1995.

[17] A. Garg, N. Tandon, and A. S. Varde, "I am guessing you can't recognize this: Generating adversarial images for object detection using spatial commonsense (student abstract)," in *AAAI Conference*, 2020, pp. 13 789–13 790.

[18] R. Giffinger, H. Kramar, G. Haindlmaier, and F. Strohmayer, "European Smart Cities 4.0," Department of Spatial Planning, Vienna University of Technology, Austria, Tech. Rep., 2015.

[19] M. Alamaniotis, D. T. Bargiotas, N. G. Bourbakis, and L. H. Tsoukalas, "Genetic optimal regression of relevance vector machines for electricity pricing signal forecasting in smart grids," *IEEE Trans. Smart Grid*, vol. 6, no. 6, pp. 2997–3005, 2015.

[20] A. Pandey, M. Puri, and A. S. Varde, "Object detection with neural models, deep learning and common sense to aid smart mobility," in *IEEE ICTAI*, 2018, pp. 859–863.

[21] CORDIS, "EMMON: EMbedded MONitoring Project," European Commission, EU, Tech. Rep., 2012.

[22] NAVSEA, "Naval sea systems command: The force behind the fleet - NSWC Dahlgren division," 2000.

[23] M. Riaz, J. King, J. Slankas, and L. Williams, "Hidden in plain sight: Automatically identifying security requirements from natural language artifacts," in *2014 IEEE 22nd International Requirements Engineering Conference (RE)*, 2014, pp. 183–192.

[24] A. Yagüe, P. Rodríguez, and J. Garbajosa, "Optimizing agile processes by early identification of hidden requirements," in *International Conference on Agile Processes and Extreme Programming in Software Engineering*. Springer, 2009, pp. 180–185.

[25] O. Emebo, V. K. Anu, and A. S. Varde, "Identifying implicit requirements in SRS big data," in *IEEE Big Data Conf.*, 2019, pp. 6169–6171.