

Detecting Cross Domain Ambiguity in Requirements Through Natural Language Processing Approach



By

Ibrahim Khalil

(Registration No: 00000320876)

Supervisor: Dr. Wasi Haider Butt

DEPARTMENT OF COMPUTER & SOFTWARE ENGINEERING,
COLLEGE OF ELECTRICAL & MECHANICAL ENGINEERING,
NATIONAL UNIVERSITY OF SCIENCES AND TECHNOLOGY,
ISLAMABAD

July 2023

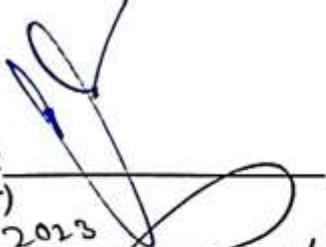
THESIS ACCEPTANCE CERTIFICATE

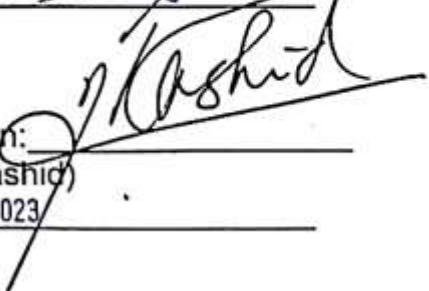
Certified that final copy of MS/MPhil Thesis written by NS Ibrahim Khalil Registration No. 00000320876 of College of E&ME has been vetted by undersigned, found complete in all respects as per NUST Statutes/Regulations, is free of plagiarism, errors and mistakes and is accepted as partial fulfillment for award of MS/MPhil degree. It is further certified that necessary amendments as pointed out by GEC members of the scholars have also been incorporated in the thesis.

Signature : 

Name of Supervisor: Dr Wasi Haider Butt

Date: 01-08-2023

Signature of HOD: 
(Dr Usman Qamar)
Date: 01-08-2023

Signature of Dean: 
(Brig Dr Nasir Rashid)
Date: 01 AUG 2023

Dedicated to a hopeful couple, my mother and my father, whose motivations, and endless prayers led me to this achievement.

Acknowledgment

I am thankful to ALLAH Almighty for his blessings throughout this research work. It was quite a challenging process that could not have been completed without the help of Allah Almighty and the strength that he given to me.

I would like to thank my sincere supervisor '**Dr. Wasi Haider Butt**' for his determined guidance and the entire committee: '**Brig. Dr Farooque-e-Azam**' and '**Dr. Arslan Shaukat**' for their endless support. I cannot thank them enough for their role in the completion of this thesis and report. I also thank my parents, spouse, siblings, and friends who encouraged me and kept me motivated during my master's program.

I am also eternally grateful to the Department of Computer and Software Engineering and the management of College of Electrical and Mechanical Engineering, NUST, who helped me and supported me throughout this journey.

Abstract

Background: In the Requirements elicitation various techniques are adapted to gather the exact needs of the stakeholders which are usually from different background. These techniques are used to clarify the actual problem being solved. There may also be greater chances of ambiguities in the terms used for the requirements. These terms used by stakeholders may vary their meaning domain to domain which may lead to an undesirable interpretation of the requirements.

Aim & Objectives: A project success can be measured/estimated if and only if the initially collected requirements are clear, unambiguous, and well understood. Similarly, the ambiguous or not understandable requirements can lead to the failure or closure of the project in disastrous form. An initial step in the requirement elicitation is usually gathering requirements in natural language. This study analyzes different tools, techniques, and approaches used for detecting ambiguities in natural language requirements, validate the approaches applied for the term's ambiguity among different domain, and to develop and use more precise approach for terms extraction of different domains, similarity finding and ranking of the ambiguities in their semantics.

Methodology: An algorithm 'Word2Vec' was found as majority in use for ambiguous word detection in text. This previously used algorithm was replaced by 'FastText' algorithm on a same data to identify more suitable approach between them. Ambiguity score of the ambiguous terms were calculated and compared scores of the high ambiguous terms produced by Word2Vec with the score produced by FastText.

Results and Conclusion: Data of five different domain were assessed via Word2Vec and FastText algorithms. Ambiguous terms were extracted and then was ranked as per their ambiguity level. The rankings of same term produced by both algorithms were compared and difference in the rankings were calculated.

This approach seeks to disambiguate texts and improve the process of software requirements elicitation in natural language.

Keywords: Ambiguity detection in Natural language requirements, Cross-domain ambiguity, Requirement's engineering, Natural language processing (NLP), Term ambiguity in Domains

Table of Contents

ACKNOWLEDGMENTS	I
ABSTRACT	II
TABLE OF CONTENTS	III
LIST OF FIGURES	IV
LIST OF TABLES	V
CHAPTER 1: INTRODUCTION	1
1.1 MOTIVATION	3
1.2 PROBLEM STATEMENT	3
1.3 AIMS AND OBJECTIVES	3
1.4 THESIS OUTLINE	4
CHAPTER 2: LITERATURE REVIEW	6
2.1 OVERVIEW AND MAJOR OUTCOMES OF SLR	6
2.2 REVIEW METHODOLOGY	8
2.2.1 <i>Research Questions</i>	8
2.2.2 <i>Category Definition</i>	9
2.2.3 <i>Review Protocol</i>	9
2.3 RESULTS, ANALYSIS & ANSWERS TO RESEARCH QUESTIONS	15
2.4 CONCLUSION OF LITERATURE REVIEW	26
2.5 SUMMARY TABLE OF LITERATURE REVIEW	27
2.6 RESEARCH GAP	27
CHAPTER 3: PROPOSED APPROACH	29
3.1 WORD EMBEDDING	30
3.2 APPROACH	33
3.2.1 <i>Wikipedia Crawling</i>	34
3.2.2 <i>Pre-Processing</i>	35
3.2.3 <i>Language Model Generation</i>	36
3.2.4 <i>Elicitation Scenarios</i>	37
3.2.5 <i>Cross-Domain Ambiguity</i>	38
3.2.6 <i>Cross-Domain Term Selection</i>	38
3.2.7 <i>Cross-Domain Ambiguity Ranking</i>	40
CHAPTER 4: IMPLEMENTATION, RESULTS & DISCUSSION	44
4.1 DATA COLLECTION/DATASET	44
4.2 EXPERIMENTAL SETUP	45
4.3 RANKING OF TERMS FOR CROSS-DOMAIN AMBIGUITY USING WORD2VEC MODELS OF THE LITERATURE	46
4.4 CROSS-DOMAIN AMBIGUITY RANKING USING WORD2VEC AND FASTTEXT MODELS ON NEW DATASET	53
4.5 RANKING OF TERMS FOR CROSS-DOMAIN AMBIGUITY ON NEW DATASET USING WORD2VEC AND FASTTEXT MODEL	53
4.6 COMPARISON OF WORD2VEC AND FASTTEXT COMBINED RESULTS ON NEW DATASET ...	62
4.7 SELECTION OF DOMINANT SHARED TERMS BY FASTTEXT	65
4.8 CASE STUDIES OF THE EFFECTIVENESS OF FASTTEXT TERMS	68
4.9 LIMITATIONS	70
CHAPTER 5: CONCLUSION & FUTURE WORK	71
5.1 CONCLUSION	71
5.2 FUTURE WORK	71
REFERENCES	72

List of Figures

Figure 1. Thesis Outline.....	5
Figure 2. Overview & Major Outcomes of SLR.....	7
Figure 3. Search Process	11
Figure 4. Publication Year of cited Research Articles.....	13
Figure 5. Approaches for Cross Domain Ambiguity (I)	21
Figure 6. Approaches for Cross Domain Ambiguity (II).....	21
Figure 7. Overview of Measuring Cross Domain Ambiguity.....	30
Figure 8. FastText 3-gram Representation of Word "string"	32
Figure 9. Training FastText on Domain Texts.....	33
Figure 10. Wikipedia Articles Crawling.....	34
Figure 11. Wikipedia Articles of Domains as Text Files.....	35
Figure 12. Pre-processing of the Text.....	36
Figure 13. Generation of the Language Models	36
Figure 14. Code to Call Function for the Selection of Cross-Domain Terms	40
Figure 15. Dominant Shared Terms Ranking	42
Figure 16. Ambiguity Score of FastText in Comparison to Word2Vec (for I1 & I2).....	56
Figure 17. Ambiguity Score of FastText in Comparison to Word2Vec (for I3 & I4).....	58
Figure 18. Ambiguity Score of FastText in Comparison to Word2Vec (for M1, M2 & M3).....	61
Figure 19. Dominant-Shared Terms Ambiguity Scores Comparison (1)	63
Figure 20. Dominant-Shared Terms Ambiguity Scores Comparison (2)	63
Figure 21. Dominant-Shared Terms Ambiguity Scores Comparison (3)	63
Figure 22. Dominant-Shared Terms Ambiguity Scores Comparison (4)	63
Figure 23. Dominant-Shared Terms Ambiguity Scores Comparison (5)	64
Figure 24. Dominant-Shared Terms Ambiguity Scores Comparison (6)	64
Figure 25. FastText Word2Vec Ambiguity Score Comparison.....	64

List of Tables

Table 1. Selected Research Papers with Catalogue	12
Table 2. Year-wise distribution of Selected Studies	13
Table 3. Data Abstraction and Combination.....	14
Table 4. Identified NLP Approaches for Automated Ambiguity Detection	16
Table 5. Identified Tools for Automated Ambiguity Detection	17
Table 6. Identified Techniques for Automated Ambiguity Detection	18
Table 7. Specific Cross-Domain Ambiguity Approaches.....	20
Table 8. Detail of the Cross-Domain Ambiguity Studies	27
Table 9. Scenarios Considered for Requirement Elicitation.....	37
Table 10. Domain with Wikipedia Articles	38
Table 11. Output Number of Terms from Different Scenarios Using Language Models of Existing Code	46
Table 12. Code Files that Generated Dominant Shared Terms of Different Scenarios	47
Table 13. Terms of the Table 2 of [1] along with its Output File.....	49
Table 14. Terms of the Table 3 of [1] along with its Output File.....	50
Table 15. Terms of the Table 4 of [1] along with its Output Files	51
Table 16. Ambiguity Scores of the Terms of Table 13 Using New Language Models by Word2Vec and FastText.....	54
Table 17. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 16	55
Table 18. Ambiguity Scores of the Terms of Table 14 Using New Language Models by Word2Vec and FastText.....	56
Table 19. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 18	57
Table 20. Ambiguity Scores of the Terms of Table 15 (M1 & M2) Using New Language Models by Word2Vec and FastText.....	58
Table 21. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 20	59
Table 22. Ambiguity Scores of the Terms of Table 15 (M3) Using New Language Models by Word2Vec and FastText.....	60
Table 23. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 22	61
Table 24. FastText Word2Vec Comparison of Ambiguity Score.....	62
Table 25. Dominant-Shared Terms List by FastText for Light Controller and Mechanical CAD Scenarios	65

Table 26. Dominant-Shared Terms List by FastText for Medical Software and Athletes
Network Scenarios66

Table 27. Dominant-Shared Terms List by FastText for Medical Device, Medical Robot and
Sport Rehab Machine Scenarios.....67

CHAPTER 1: INTRODUCTION

Requirement engineering has a vital significance in the development of a software. It is the phase that defines, documents, and maintains the requirements for all the upcoming phases in the process. Commonly software system is built for different areas w.r.t its domain which may be a simple straight forward and generalized field such as sports, construction or a more specific and technical one such as mechatronics engineering. It sometime combines more than one area for a single software development, e.g., sports and mechatronics engineering. For this kind of situation, the requirements experts must meet with the domain specialists to elicit the knowledge of its domain for the development of system [1]. In the requirement elicitation process, even if a little ambiguity is left over, it leads to the major defects in the later phases/steps.

Ambiguities in the system requirement can be of different types; specifically, the requirements written in the natural languages have lexical, syntactic, or structural, semantic, and pragmatic ambiguities in common. Lexical ambiguity occurs if a term used in natural language requirement have un-related meanings due to poor usage of vocabulary [2]. In the syntactic ambiguity the sentences have more tree of syntax with one, with diverse sense. A semantic ambiguity exists if a sentence maybe interpreted into more logic expression [3]. In the pragmatic ambiguity, meaning of sentence is subject to its context [4] and if someone gives reference to a certain entity and that entity points to more than one meaning, this also leads an ambiguous requirement called referential ambiguity [5]. After writing the requirements, requirement analyst track them and remove the ambiguities in it. For the last decade, natural language processing techniques are being applied successfully in the projects wherever natural language is used to write requirements. Software requirements are based on natural language initially from stakeholders' point of view, which are addressed by requirement engineers and domain experts in the iterations. Some of the ambiguities are easy to trace out and some of the ambiguities may requires the meddling of relevant field experts and stakeholders to rectify.

Natural language processing techniques are in use with the combination of other approaches to analyze the requirements text for identifying ambiguities and inconsistencies. More specifically if we talk about term ambiguities across the domain, there are lot of terms which results varies in meaning subject to the domain i.e., the term 'formula' will be a mathematical formula in one domain and a type of car for the other [1]. The study found different

approaches proposed to detect such kind of cross domain ambiguities, which were then analyzed with the implementation and usage point of view.

The requirements in written form have been assumed plenty of times for the identification of ambiguity in it. Some of the studies emphasis on the terms or expression, which may be the source of uncertainties [6]. Various tools, techniques and approaches have projected for uncovering and rectification related to the ambiguities in requirement's document. Tools and approaches in addition to Natural Language Processing techniques have been used for finding uncertainties in the requirements [7]. The identified techniques are proposed for various kind of ambiguities i.e., to address referential ambiguities, pragmatic ambiguities, domain specific ambiguities and other variabilities in requirement documents [8], [9], [10]. Various studies also proposed composite approaches and tools for better detection of anomalies in the requirements and for its redressal.

This **thesis proposes** the identification of ambiguity caused in the requirements by different terms. That is, the same term appears in different domains, but interpretation of the term varies with respect to the domain in which it is used. The term ambiguity can occur when the same term used to map different things. This may be due to feature of the language in use or may be due to absence of or indefinite descriptions [11]. This form of ambiguity is more problematic where there is more than one domain involved in the development of a system. After the identification of these ambiguous terms from one domain based on its semantics, the same term is checked in other domain(s) if it occurs sufficiently in that domain(s). These ambiguous terms are then scored accordingly based on its ambiguity level and ranked upon it. This procedure of ambiguity finding and ranking has done in the article [1]. The same procedure is repeated in this study but via FastText Algorithm, and the results of both the algorithms have been compared to determine which algorithm perform better for the same purpose.

This **chapter summarize** that non-ambiguous requirements are essential for a successful development of system and the focus here is on one of the ambiguity categories which commonly occurs in natural language requirements, that is the terms which has different interpretation in different domains. Word2Vec algorithm is used for detection of these potential ambiguous terms in the literature. We used FastText on the dataset of different domains to detect these ambiguous terms across the domains.

1.1 Motivation

The motivation of this research is to select the more effective approach for the uncovering of vague terms used in the requirements gathered from different areas. The research used text dataset from the Wikipedia articles related to five different domains. These articles were considered to analyze general type data for different selected domains and to identify ambiguous terms in it, which will lead to non-ambiguous requirements for further software development. Also, it will provide an option to choose more appropriate approach to be opt by requirement analysts for uncovering ambiguous terms in a certain domain requirement. These ambiguous terms can then be modified for clarity of the meanings and to ease the interpretation in the design phase.

1.2 Problem Statement

Requirement engineering has a significant role in the whole process of software development. After the feasibility study, usually the requirements are elicited from the domain for which the software is being developed. The requirements are elicited from domain expert via different means such as questionnaires, meetings, interviews etc. These requirements are then analyzed in different ways with a view to proceed further in the right direction in the development cycle of the system. One of the ways among various is to detect ambiguities from these requirements and eliminate or clarify as desired. These ambiguities can be of different types needed to be identified. One of the common ambiguities exists in the requirements is term ambiguity which is focused on this study. To deal with these ambiguities, multiple ways were found in the literature. Most commonly Natural language processing approaches were opted for the said purpose and precisely, skip-gram negative sampling variant of Word2Vec algorithm had been used. In this study we considered combination of approaches along with replacement of the approaches used in the literature with other relevant solutions. Additionally, the outcome of approach must be compared with the previous results got from the same dataset.

1.3 Aims and Objectives

The major objectives of the research are as follows:

- To perform a comprehensive systematic-literature-review of recent articles on cross-domain ambiguity.

- To obtain dataset of different specified domains by crawling Wikipedia articles in text form for further usage.
- To explore algorithms that are used particularly for the detection of ambiguous terms in the corpus.
- To propose an approach that detect and rank the potential vague terms in the dataset of different domains.
- To analyze the results which is obtained via new approach and to compare it with previous results to observe any significance positive change.

1.4 Thesis Outline

The remaining work is structured as follows:

Chapter 2 states a literature review in detail and the important relevant work performed by analysts and researchers in the previous few years, which covers the basics and background of the ambiguity detection and NLP approaches usage for the analysis of requirements. The systematic literature review is composed of three core sections. The ever first is the review-protocol which displays detail upon the procedure using which the literature-review has carried out. Second Section offers detail on research study carried out on this area in the form of research-questions and tables. The section three shows the research-gap that are encountered in the study.

Chapter 3 consists of the proposed approach in detail. It discusses the method in terms of an overview of the algorithm, main components of the approach and depiction of solution.

Chapter 4 includes implementation, validation, and discussion on results together with research-questions and related figures. It also compare results of our work with the state of the art. Moreover, it precisely describes the limitations of this study.

Chapter 5 concludes the thesis and reveals the future work of this research.

The thesis outline is shown in Figure 1.

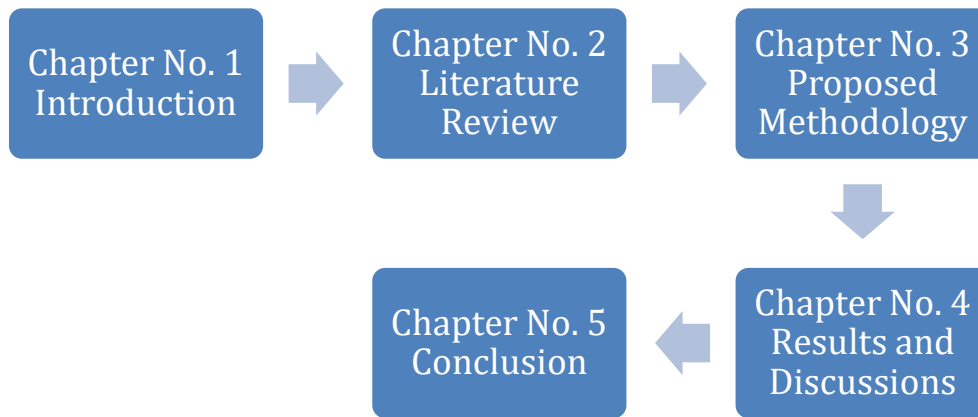


Figure 1. Thesis Outline

CHAPTER 2: LITERATURE REVIEW

This chapter covers the systematic literature review for the area of this research. The Chapter comprises of an overview and major outcomes of Systematic literature, contribution of literature review, review methodology, research questions, category definitions, review protocol of literature review, results, and analysis, answer the research questions for literature and conclusion of the SLR.

2.1 Overview and Major outcomes of SLR

There are several studies in which the authors presented tools, techniques, approaches, and combination of these for the existence of cross-domain ambiguities in the requirements. Most of these studies are covered in this SLR. It also provides an overview of detailed usage of these approaches for the said purpose. Detailed research cover almost all the key features that concerns with the use of natural language processing approach for the detection of ambiguity in natural language requirements.

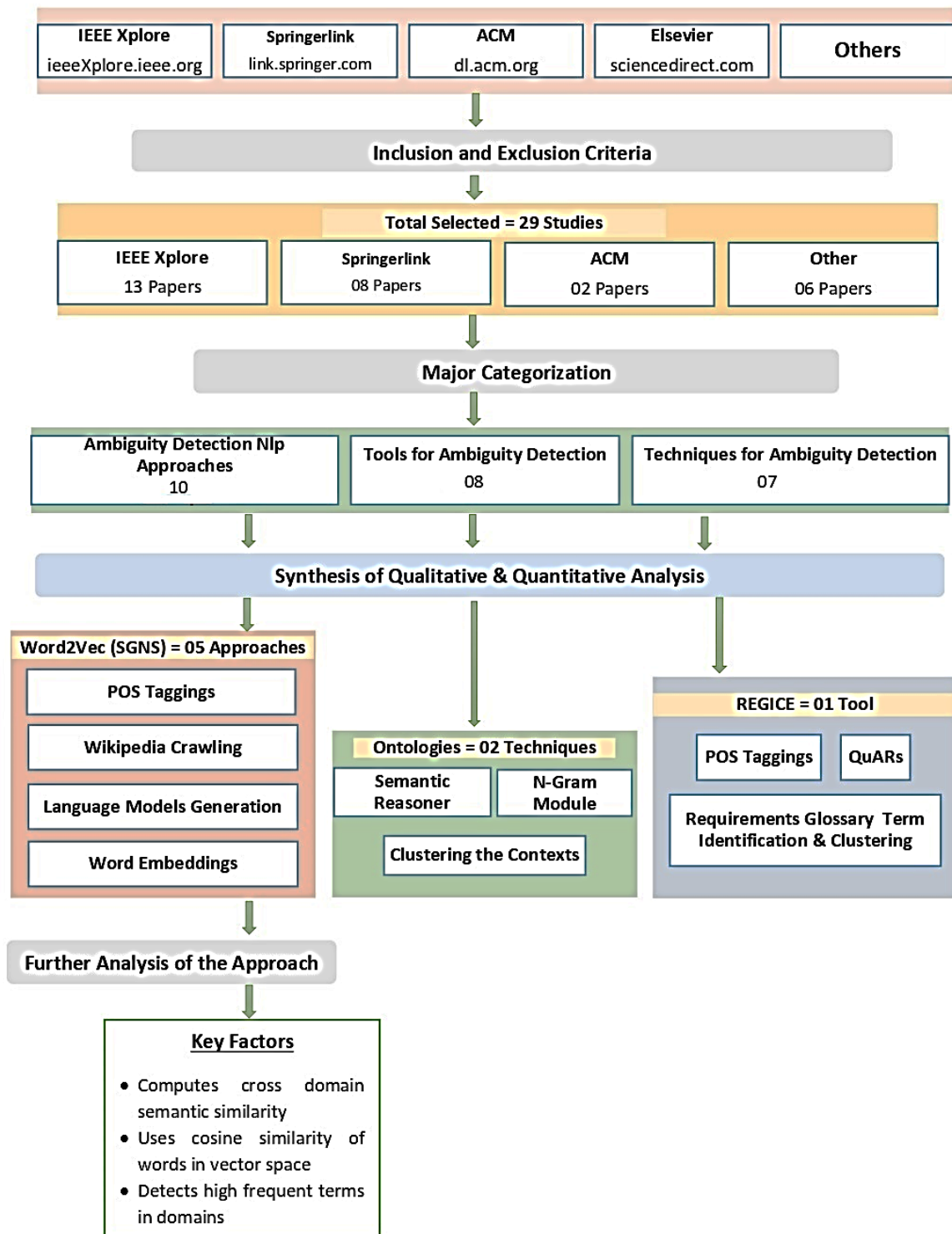


Figure 2. Overview & Major Outcomes of SLR

Overview and key results of SLR is depicted in Figure 2. Twenty-nine landmark articles which were published from 2012 to 2022 in the major repositories were studied and analyzed in detail after needful filtration as per selection criteria. The articles were then categorized in further three groups for the purpose of analysis and to further study it to answer the research

questions. These groups are ‘Ambiguity Detection NLP Approaches – 10 papers’, ‘Tools for Ambiguity Detection - 08 papers’, and ‘Ambiguity Detection Techniques - 07 papers. A comprehensive analyzation of the final selected articles was performed to find the requisite and accurate result. So, the whole synthesis of analysis is divided into three categories. Five approaches were found relevant which were using Word2Vec algorithm with the combination of POS tagging, Wikipedia crawling, language model generation, and Word-embeddings. Ontologies based detection of term ambiguities were found in two studies. Ontologies were developed with which the semantic-reasoner, N-gram module were used along with clustering the contexts. Another approach was adopted for the same purpose using REGICE tool that used QuARs and combine POS tagging.

The **key contributions** of Systematic Literature Review are:

- a) Identifying the approaches of ambiguity detection in requirements.
- b) Finding several automatic tools.
- c) Reporting the usage position of the tool.
- d) Detecting the major validation-techniques over which the rationality of the tools has been confirmed by scholars.
- e) identifying algorithms for ambiguity detection, stating its strategy and the procedure it focused on for the ambiguity detection in natural language requirements.
- f) Classifying the practical usage of several ambiguity finding tools and approaches.
- g) Summarizing NLP approaches for the cross-domain ambiguity.
- h) Finding more accurate and latest approach for the detection of terms potentially cause of ambiguity.

2.2 Review Methodology

This literature review followed the guidelines of Kitchenham guidelines [12]. The key areas of the methodology are planning, conducting and report. Review Protocol of the methodology section is an important step which is further divided in two sub-categories such as Review Protocol Development and Category Definition. More precisely, this segment explains Category Definition and Review Protocol. Furthermore, research questions of this study are also stated in this section.

2.2.1 Research Questions

Research questions have been summarized as below:

RQ1: Which techniques have been proposed in the literature for automated cross-domain ambiguity detection in requirement engineering?

RQ2: Which NLP approaches have been used for automated cross-domain ambiguity detection in requirement engineering?

RQ3: What are popular tools used / developed for automated cross-domain ambiguity?

RQ4: What are advantages and limitations of tools and techniques proposed for automated cross-domain ambiguity detection?

RQ5: Which ambiguity detection approach has better research productivity over the years from 2012 to 2022?

RQ6: How cross-domain ambiguity detection approach may be ranked as per their accuracy?

2.2.2 Category Definition

The research has been divided into three main sections that helps finding answers to the research questions.

- a) **Category 1:** In this category the previous studies concerned with the identification of techniques used for ambiguity detection in requirements, have been considered.
- b) **Category 2:** This category step-in to the identified studies and filter out only those which are very specific to NLP approaches used cross-domain ambiguities caused by the same terms.
- c) **Category 3:** In this category we have compared the mechanism of finding the cross-domain ambiguities via popular and customized tools from the selected concerned studies and identified the most common among them.

2.2.3 Review Protocol

After the category definition, the Review Protocol is formed as per the given procedure of Kitchenham [12]. Review protocol has six stages. Two steps which is background and research-questions are elaborated before, while rest of the steps are described in the below captions:

I. Acceptance and Rejection Criteria

Acceptance and rejection mainly consists of a set of some proper rules and a criterion which make the foundation for inclusion or exclusion of a specific study for the topic. These rules

consist of steps which are necessary to be followed with a view to decide regarding a study for its inclusion or exclusion. The articles which do not follow these certain pre-defined parameters are not considered for the SLR. Whereas the studies which fulfilled these rules have been considered for further work. These parameters for acceptance and rejection of papers are given below:

- a) **Subject:** The papers which closely belong to the ambiguity detection in requirements should be selected. Those studies which dealt with other than cross-domain ambiguities must be dropped as this literature only focused on cross domain ambiguity detections caused via same terms in different domains.
- b) **Publication Year:** This literature deals only with the articles which are published from 2012 onward till date. Even the relevant papers which were published before 2012 were rejected. This is because, the latest research on the topic was focused and the second justification is that almost all this research was based on the results of previous research. Therefore, this literature studies duration was sensibly selected. This decade papers focused the more recent approaches related to the topic and backing by previous studies on the relevant area. As an example, a framework based on ontologies is used for the corrections of requirements with inconsistent state in paper [13] published in 2016, while the same concept of using ontologies for the same purpose was also used in [14] study. Hence the paper [13] covers the approach beyond 2012.
- c) **Publisher:** Papers were chosen from various well-known and authentic scientific databases. The repositories include IEEE, ACM, SPRINGER to conduct this literature review. These repositories are so dependable and reliable and the articles which are selected via these databases endure a rigorous peer review. Thus, majority of the articles for this SLR were selected from these databases.
- d) **Language:** Only English language studies have been selected for this systematic literature review. The studies written other than in English language were not considered for this review.
- e) **Validation:** The articles in which the validation of approaches, tools is done thoroughly with the help of dataset, open-source data or supported via a proper case study are included. As an example, a paper [10] presented the detection of ambiguous terminologies using NLP approach with Word2Vec algorithm and its validation process by presenting dataset and complete project. On the other hand, the articles that missing validation or case studies were excluded from the study.

II. Search Process

The well-known repositories i.e., IEEE, Elsevier, ACM, Springer started to be explored after specifying the criteria of acceptance / rejection of studies. Different keywords and search-items were used for finding the papers in the mentioned databases. Search of the articles was performed using very relevant keywords such as “ambiguity detection in requirements”. Resultantly, hundreds of links were coming up which could not be practically examined. For example, the IEEE showed 19700 results for the search phrase “cross domain ambiguity in requirements” in default search setting.

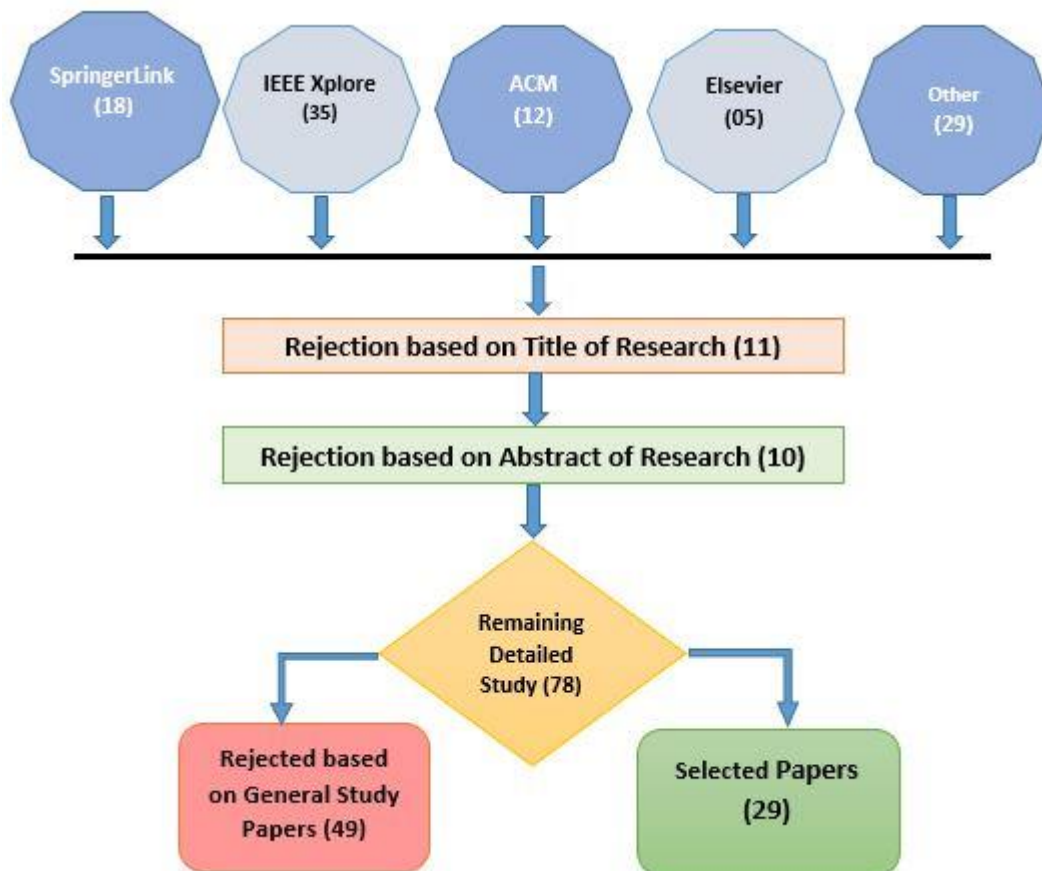


Figure 3. Search Process

The searching results were refined by applying filters like the searching publication range was applied i.e., from 2012 to 2022. The logical operators were applied to extract the process of search on keyword searching. Furthermore, synonyms of the keywords and possible replacement words were tried with a view not to miss out any relevant and important study from the literature. The snowballing guidelines [15] (forward and backward snowballing) were used to search further related studies. We gathered the most related papers after these procedures to consider it for this systematic literature review. With these methods, a detailed

search process was carried out, through which we found 29 research articles to find out precise and correct answers to the research-questions. The whole searching procedure is explained in Figure 3, and the phases are described below:

- At first, 99 papers got from whole databases. Titles of research were checked and analyzed the relevance of it as per criteria. Eleven studies were rejected which shows insignificance to this research in their titles.
- Secondly, we focused on the abstract of the remaining 88 papers. The studies whose abstract went against the defined criteria were dropped from consideration. Total of 10 papers were discarded after the study of paper abstracts.
- The remaining 78 papers were analyzed. Detailed study of the papers was carried out, in which validation of the studies was also considered for the verification of the approach adapted in the papers. Based on the detailed investigation of remaining papers, 49 further articles were excluded from the literature review of this research.
- Finally, the remaining 29 papers were selected for a comprehensive analysis and systematic literature review on the topic.

Table 1. Selected Research Papers with Catalogue

Sr. #	Catalogue	Article Type	Studies References (selected)	No. of Articles
1	IEEE	Conference	[3], [4], [8], [9], [10], [16], [17], [18], [19], [20], [13], [21], [22]	13
3	ACM	Conference	[23], [24]	02
4	Springer	Journal	[1], [25], [26], [27], [28]	08
		Conference	[29]	
		Book Sec.	[30], [31]	
5	Others	Journal	[32], [33], [34], [35], [36], [37]	06

III. Quality Evaluation

Research studies from high impact articles which were authentic and acknowledged internationally were tried to be selected from scientific repositories to ensure the reliable outcomes of the literature review. Major databases were considered to select articles according to the criteria for the selection and rejection as mentioned above. The detail of selected papers and its distribution via concerned publishing scientific databases are given in Table 1. The table explains the details of the selected articles to be referenced, their databases, the paper publication type i.e., conference or journal, and total selected papers from repository. From IEEE 13 papers are selected, from ACM 02 papers, 08 papers from Springer, and 06 articles from other journals.

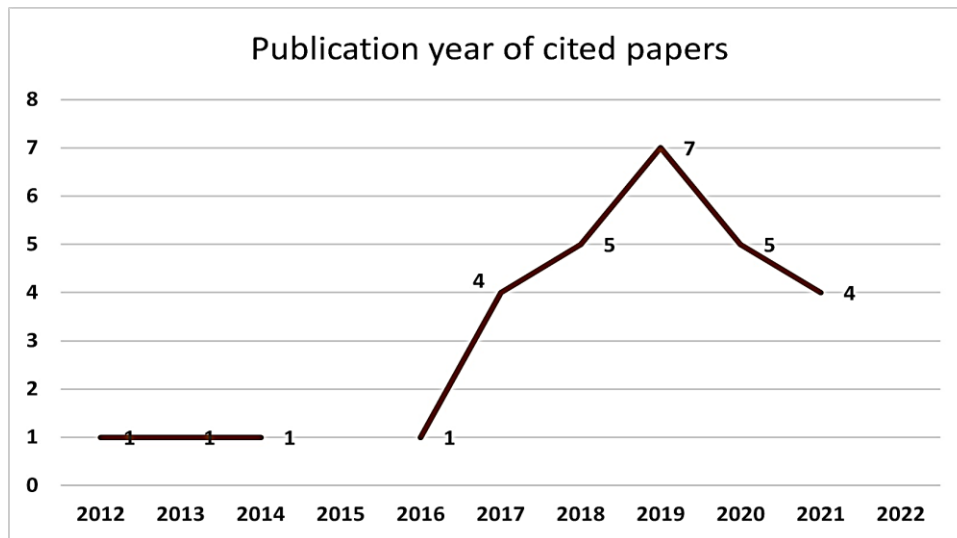


Figure 4. Publication Year of cited Research Articles

Moreover, studies were separated based on its type i.e., conference and journal, against each database in the table. Such as, 13 conference papers from IEEE database are selected, 02 conference papers from ACM repository is added to the SLR, 05 journals, 01 conference paper and 02 book sections are included from the Springer database, and 06 studies which were very associated with this study was also included from other journals.

Table 2. Year-wise distribution of Selected Studies

Sr.	Year	Studies	Contribution percentagewise	Total
1	2012	[8]	3.4%	1
2	2013	[37]	3.4%	1
3	2014	[4]	3.4%	1
4	2015	--	0%	0
5	2016	[13]	3.4%	1
6	2017	[9], [17], [27], [32]	13.8%	4
7	2018	[3], [10], [18], [20], [25]	17.2%	5
8	2019	[16], [19], [24], [26], [30], [34], [1]	24.1%	7
9	2020	[21], [29], [31], [33], [36]	17.2%	5
10	2021	[22], [23], [28], [35]	13.8%	4
11	2022	--	0%	0

Yearly detail of the studies selected for the review is shown in the Figure 4. Vertical axis of the graph show us the maximum number of article(s) per year included in the literature. Similarly, horizontal axis indicates years in which the papers are published that is from 2012 to 2022.

Table 2 depicts the Figure 4 in tabular form which presents the division of the selected papers year-wise. The focus was not to miss any relevant study from the review in these years and to

add the latest possible information from the trusted sites. The papers are mentioned against each year for the reference. Percentage of year wise contribution is listed in the table. The last column of the table shows the total number of selected papers in each year for the literature. An important point here is that this research area started to be explored more from 2017 onward as the number of studies from 2017 onward were found more than the previous years back till 2012. Similarly, IEEE database has more research studies relevant to this research topic as compared to other databases, as a smaller number of studies were found from other repositories. Resultantly, 29 total papers were extracted for this systematic literature review finally.

Types of the publications and sortation is also a significant factor in the SLR demonstration in better way. Therefore 13 journal papers + book sections out of 29 are included in this SLR, which is 44.8% contribution in the study. Similarly, 16 conference papers were included in which calculated about 55.2% of the total studies. These studies qualified the criteria of inclusion in this SLR.

IV. Data Extraction and Synthesis

Selection of studies was done according to the pre-defined criteria after which a pattern was created to extract and synthesize the data. This process is shown in Table 3. Using this pattern, answers to the research questions have automatically been extracted. It also helps us in gathering and synthesizing the required details from the articles. The information obtained from the selected studies have bibliography info, an overview of the study, methodology, description of the implementation, results of the research, limitation of the study, tools, techniques, and approaches adapted in the research papers. With this procedure all answers to the research question were satisfied. The pattern facilitated in gathering outcomes of the numerous unnecessary data.

Table 3. Data Abstraction and Combination

Sr	Type	Specification
1	Bibliography Data	Type of the research-paper i.e., conference or journal, title headings, author, yearly publication, publisher's detail is studied.
Info Abstraction		
2	General-Data	This contains general overview of the SLR
3	Results Validation	Result of the research ideas is validated via formal methods
Data Combination		
5	Categorization	All categories are considered to answer the

		questions, and then outcome is classified
6	Approaches & Techniques	The most relevant approaches & Techniques are separately mentioned in Table 7

2.3 Results, Analysis & Answers to Research Questions

The main purpose of the study is to observe and analyze certain literature to search and find answers to the research questions. In this section we are reporting the outcomes of the extracted data after the detailed examination of data. The important journals which contributed to the recent approaches in ambiguity detection are; ‘Automated Software Eng’, ‘Empirical Software Engineering’, ‘From software Engineering to formal methods, tools, back’, ‘Requirements Engineering: Foundation for Software Quality’, ‘Journal of Telecommunication, Electronic and Computer Engineering (JTEC)’, ‘Association for computational linguistics’. Similarly, some of the conferences that contributed to this systematic literature review are ‘Artificial Intelligence for Requirements Engineering (AIRE)’, ‘International Workshop on Empirical Requirements Engineering (EmpiRE)’, ‘International Requirements Engineering Conference (RE)’, ‘Evaluation and Assessment in Software Engineering’, ‘Human System Interaction (HSI)’, ‘International Conference on Software Engineering (ICSE)’ and some others.

1) NLP Approaches Identified for Ambiguity Detection

One among the research questions includes the NLP approach used in the literature for the detection of ambiguity in requirements. Various NLP approaches have been applied to sense ambiguities in natural language requirements. An approach is precise methodology being followed in which it is described in what way artifacts are formed. This portion discusses the natural language processing approaches used by different researchers in the selected research literature. From this literature 09 different NLP approaches were identified being used by researchers in a number of ways. These approaches and combination of approaches are listed in Table 4. These approaches are mentioned in 10 different research articles published between 2012-22. These approaches are mentioned with their concern abbreviations wherever available. NLP basic techniques like POS tagging, lemmatization, tokenization have been used in majority of the studies with a combination of other approaches to obtain the purpose. Wikipedia crawling, Word embeddings, Language model generations were other commonly used approaches along with these previously mentioned approaches. Different pre-defined and customized algorithms were adapted for ambiguous word detection. Some of

them are Word2Vec, Text mining, Constituency parser, BabelNet and Frame semantics. This combination of approaches along with its custom usage to trace ambiguity are listed against the study referenced.

Table 4. Identified NLP Approaches for Automated Ambiguity Detection

Identified NLP Approaches	Custom Usage of the Approach	Identified Research
Word2vec (SGNS)	Word Embeddings, Language Model Generation	[9], [10]
Linear Transformation of Word Embedding Spaces	Applied linear transformation on Word Embedding spaces via Machine Learning techniques.	[33]
POS tagging and normalization (using Stanford parser)	Graph-based Centrality for word sense disambiguation	[3]
BabelNet (lexical database) for Ambiguity detection.		
N-gram approach lexical ambiguity detection		
POS Tagging	Online English dictionary (for disambiguation)	[18]
Tokenization		
Wordnet		
OpenNLP		
POS Tagging,	Generated & analyze models for detecting ambiguities and inconsistencies. (A Theoretical approach)	[34]
Tokenization,		
Morphological analysis,		
Semantic analysis		
Text mining		
Active learning	Generated via SEMAFOR	[35]
Frame semantics		
NLTK	for POS Tagging	
Wikipedia crawling (through Petscan)	Word Embeddings, Language Model Generation	[19]
POS Tagging		
Tokenization		
word2vec (SGNS)		
POS Tagging	Heuristics applied for the ease of decision making related to a phrase.	[22]
Tokenization		
Constituency parser		
Wikipedia crawling		
Heuristics		
word2vec (SGNS)	Word Embeddings, Language Model Generation,	[1]
Wikipedia crawling		

POS Tagging	Customized algorithms for cross domain term selection and ranking	
-------------	---	--

2) Tools Identified for Ambiguity Detection

In the selected literature studies, some of the authors proposed tools/solutions for identifying ambiguities in requirements. Tools having specific functionalities designed to achieve a targeted aim through some processing. This section represents the modern ambiguity detection tools and are listed in Table 5. Total of 09 tools identified that are precisely used for the identification of ambiguity in requirements. These tools are listed from the selected literature found in 12 research studies published in between 2012-22.

Some of tools are used in single while some have made combination with other tools, NLP approaches or requisite alteration to achieve the goal. Each tool with its combination or alteration if any is listed along with its custom usage design for ambiguity detection in requirements. The last column shows the research reference in which the approach has been identified. The mostly used tool in the literature was Quality-Analyzer for Req Specification (QuARS) and General-Architecture for Text Engineering (GATE) on second.

Table 5. Identified Tools for Automated Ambiguity Detection

Identified Tools	Custom Usage of Tool	Identified Research
Tokenization	Language Model Generation SpaCy-based NLP tool prototype	[23]
POS_Tagging		
Dependency_Parsing		
Lemmatization		
Sentence_Boundary Detection		
Rule-Based-Match		
Syntactic-derivation tree		
GATE	Shallow parsing	
SREE	Gezetter Jape Rule	[25]
QuOD	--	[26]
GATE	Shallow Parsing Gazetteer JAPE Rules	[27]
QuARs	Lexical/syntactical analyzer quality evaluator	[20], [24], [30], [36]

GATE	Text Extraction Boilerplate checking BNF Grammar with JAPE	[32]
Stanford NLP parser		
QuARS	Requirements Glossary term Identification and Clustering (REGICE)	[31]
Part-of-Speech (POS) Tagging		
Requirement Assessment Tool (RAT)	RAT (in comparison with other four tools, i.e., QuOD, QVscribe, Innoslate, RQA)	[16]
QuARS	Compared with Req. Scout, QVscribe	[29]

3) Techniques Identified for Ambiguity Detection

Most of the Techniques are applied to validate different tools. Some of the researchers have used combination of open-source projects and models for validity. Techniques mostly express how to apply or use the tool(s) or other functionality to achieve goals. Six different techniques and combination of techniques were identified from the selected studies that were designed in such a way that detects ambiguities in the requirements. Total 07 studies were identified using these techniques for the above-mentioned purpose. These approaches were applied to sense these inconsistencies from natural language requirements, general listed common in use requirements, while some were applied on controlled natural languages (CNLs). As described above, these techniques were detected from the papers published between 2012-22. These identified techniques are listed in Table 6, along with its custom usage to obtain the purpose. Last column of the table lists the research article references from which these approaches are identified.

Table 6. Identified Techniques for Automated Ambiguity Detection

Identified Techniques	Custom Usage of the Technique	Identified Research
Syntax for NL requirement	logic based CNLs (controlled natural language)	[17]
Use artifact i.e., architecture model and connecting it with the texts of requirement.	Effective requirement's language proposition.	
Semantic of Business Vocabulary and Rules (SBVR)	Controlled NL used for ambiguity resolution	[28]

Ontology Based Framework	<ol style="list-style-type: none"> 1. Develop ontology represents domain knowledge. 2. A Semantic reasoner for the deduction of logical inferences from the axioms 	[13]
N-gram module	For non-referential ambiguities	[37]
Ontologies	(Wiktionary, Wikipedia disambiguation pages) for across domain ambiguities	
Clustering the contexts used for either of the above	--	
Score-Based Ambiguity Detector and Resolver (SBADR)	(Using “Stanford Core NLP” API and four filtering pipelines.) Worked on Coordination, attachment, & analytical ambiguities	[21]
Shortest-path (least-cost path) search Algorithm	Constructs Domain Knowledge Graph for identifying ambiguities.	[4], [8]

4) Cross-Domain Ambiguity Detection Approaches

There are several different types of ambiguities which can be found in requirements through proper analysis of these requirements. If the requirements are elicited in natural language can be in inconsistent state due to various reasons. These may be due to syntactic ambiguities, structural or lexical ambiguities, semantic or pragmatic ambiguities in these requirements which leads the requirements interpretation to a different perspective other than stockholders’ actual needs. Lexical ambiguity may occur if a specified terminology has a meaning that is not related to problem due to poor vocabulary usage [2]. If the sentence has more than one syntax trees in different ways, then this is syntactic ambiguity but if a sentence can be interpreted into more than one logical expression; this kind of ambiguity is semantic ambiguity [3]. Pragmatic ambiguity occurs when meaning of an expression depends on its context [4]. Similarly, if a reference points to an entity which has more than a single interpretation resultantly makes a requirement inconsistent and such kind of ambiguity may be referred to as referential ambiguity [5].

The ambiguities as mentioned above make a requirement inconsistent in different ways. Either of these leads to the wrong interpretation of the user needs which can then be costly recovery for both developers and stakeholders. One of them is an ambiguity caused by terms used differently for different domains and meaning of these terms may be changed if the

requirements domain change [9]. Such kind of terms if associated with computer terminologies then it is more likely to be considered in alternate yet diverse meaning as requirement analysts, designers and developers may interpret it certainly as per meaning of their own domain. This specific type of ambiguity was targeted by a very few research studies which is figured out from the specified literature.

Among various tools, techniques, and NLP approaches specifically the target of which is identification of ambiguities lies in the same terms of different domain are separated and listed below in Table 7.

Table 7. Specific Cross-Domain Ambiguity Approaches

Approach/Technique	Number of research	Identification
Word2Vec (SGNS) Word Embeddings	5	[1], [9], [10], [19], [33]
Develop Ontology representing domain knowledge	2	[13], [37]
Requirement Glossary term identification and clustering (REGICE)	1	[31]

Skip Gram Negative Sampling (SGNS) type of Word2Vec perform Word Embeddings which represents words in vectors and using vectors spaces, semantic similarities among the words are computed. Based on the input words, various language models are generated and using documents of different domain, the meaning of words from the different domain are being compared [1], [9], [10], [19], [33]. 05 research studies have been found using this similar approach for detecting ambiguous terms across the domain. For the same purpose, 02 studies using ontology-based approach for the representation of domain knowledge [13], [37]. One of the articles used QuARs tool in combination with NLP basics functionalities and designed a customized tool REGICE tool to detect term ambiguity in requirements [31]. Word2Vec algorithm has used more than any other technique for the detection of ambiguities in different domain as shown in Figure 6 & Figure 5. Both pie and box-and-whisker charts depict the number of research categorized upon specified approach for the said purpose.

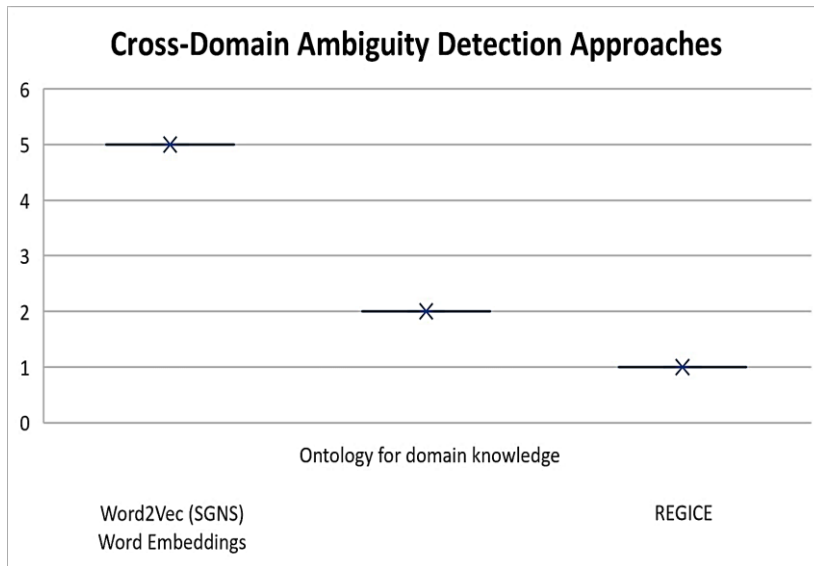


Figure 6. Approaches for Cross Domain Ambiguity (I)

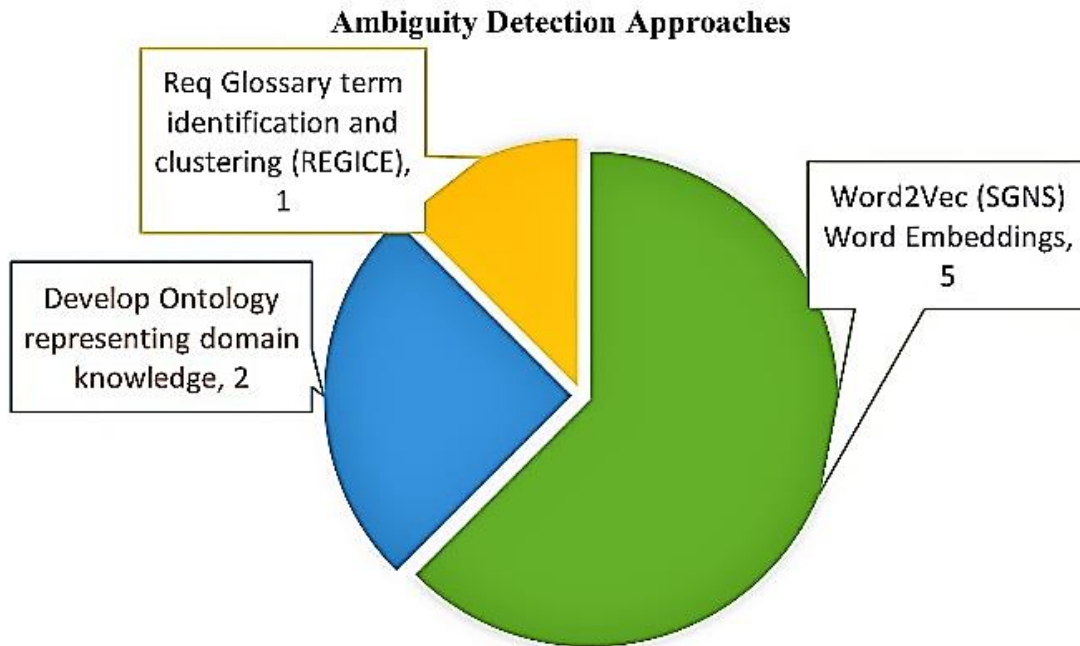


Figure 5. Approaches for Cross Domain Ambiguity (II)

Based on NLP techniques, the domain ontology is supposed to be developed using SRS document. These ontologies and language models represent domain knowledge, act as domain model which helps in removing various types of ambiguities in comparison with other domain and particularly term ambiguities detection [13]. In addition to that n-gram module was also used for referential ambiguity detection followed by two ontologies and clustering. One ontology used Wiktionary for identifying terms having multiple senses, other used Wikipedia for identifying terms having disambiguation pages [37].

For the detection of variabilities and ambiguity in natural language requirements QuARs tool was used and particularly the usage of a tool REGICE which is using the glossary of terms from the requirement document, building clusters of the similar requirement terms. This tool clarifies the use of these terms in relevance to the domain and helps disambiguate the requirement text of the software domain [31].

5) Answers to the Research Questions

1. Which techniques have been proposed in the literature for automated cross-domain ambiguity detection in requirement engineering?

Answer: Total of six techniques have been proposed for ambiguity detection in the requirement documents as shown in Table 6. Two of the techniques proposed controlled natural languages (CNLs) templates for the requirements. One of these focused on the syntax of the requirements using logic based CNLs. Formal languages are used along with these CNLs for removing the ambiguities using syntaxes with additional information [17]. This study combines natural language requirements with the predefined formal syntaxes. Additionally, these natural language requirements are mapped with generated models. Outcome of this process believed to disambiguate the requirements texts. This study considers a requirement as a property, which would be either true or false. It considers the requirement a Boolean valued function. The requirements are divided into the possible smaller parts to make atomic expressions of it. These atomic expressions and implication functions are bonded with conjunctions making tree like structure. This way the requirement text is interpreted to perform reasoning and check it as per consistency criteria.

CNL is also considered as bridge between natural language requirements and the actual formal requirements in the study [28]. It also reports logical representation usage as formalized mathematical expressions in the place of natural language to avoid ambiguity in the software requirement specification (SRS). One significant study related to the cross-domain ambiguity used domain ontology to explore the related domain knowledge [13]. This technique used NLP techniques to discover the terms synonyms that are multilingual. This concept epitomizes an abstract domain model. Resultantly, the SRS document is converted into formal logical form in which each requirement will be interpreted in exactly one way to eliminate ambiguities.

Tyler Baldwin suggested a Term Ambiguity Detection framework having three modules [37]. One of the modules used to sense non-referential ambiguity by examining N gram data from

requirement text. Second module used two ontologies for the detection of cross domain ambiguities. One ontology used Wiktionary to check whether a term has other than one senses, if so, that term will be considered as ambiguous term. Second ontology is to find that if a term has Wikipedia disambiguation page, then mark it ambiguous. The third module used an approach by clustering contexts of words. For this module they used topic modeling method “latent Dirichlet allocation (LDA)” [38]. If a term is not seemed in the most weighted 10 words in a cluster, then it is marked as ambiguous. As a conclusion, a term is marked ambiguous if any of these three modules indicates ambiguity for it.

For the detection of coordination ambiguity, attachment & analytical ambiguities of natural language requirements, a technique namely “Score-based Ambiguity Detector & Resolver (SBADR)” has been used by Mohamed Osama [21]. This technique used “Stanford Core NLP” API with combination of four filtering pipelines. The API is used to obtain maximum possible parsing tree of the sentences in a requirement. These generated trees are then passed through the pipelines to sense and resolve the syntactic ambiguity by suggesting maximum possible interpretation of the given requirement. These interpretations are then analyzed a sentence level for the different types of ambiguity in it. It provides a reliable automated identification process for syntactic ambiguity and not restricted to a particular ambiguity type.

A technique outlined in the research [4] tries to identify pragmatic ambiguity in papers with NL requirements. A requirement has a pragmatic ambiguity, if various readers understand it differently, subject to the requirement context. A requirement's context refers to the other related requirements in document that have an impact on how well the requirement is understood as well as the reader's background knowledge [8], which gives interpretation to the requirement's concept. This study builds graphs for the domain knowledge from the requirement document and using these graphs identifying pragmatic ambiguity in the document.

2. Which NLP Approaches have been used for automated cross-domain ambiguity detection in requirement engineering?

Answer: In the relevant literature of the cited period, Nine NLP approaches were identified that are proposed for the ambiguity detection in the requirement documents as shown in Table 4. Total of five studies were more specific to detect cross-domain ambiguity and all of

them used word embeddings, SGNS variant of Word2Vec algorithm in combination with other techniques for the detection and rectification of this specific type of ambiguity [1], [9], [10], [19], [33].

3. What are popular tools used / developed for automated cross-domain ambiguity?

Answer: Use of nine tools has been identified from the research. As mentioned in Table 5 in detail. Majority of the tools were used for vague terms identification, other types of ambiguities in requirement documents. Only a single tool that was found to be proposed for cross-domain type ambiguities was REGICE [31].

4. What are advantages and limitations of tools and techniques proposed for automated cross-domain ambiguity detection?

Answer: Various tools & techniques and NLP approaches have been identified from the literature which is discussed below:

REGICE Tool: As the focus is on term ambiguity in the requirements across the domain, therefore REGICE (Requirement Glossary term identification and clustering) is suggested in the [31] in detail which basically extract terms, computes similarity in them, and divide these terms into relative terms clusters as presented in [39], but it works on one requirement document only at a time.

Developing Ontologies: One of the modules proposed for cross domain ambiguity detection employs ontologies. The terms that have more than one senses in Wiktionary or pages in Wikipedia were considered as ambiguous. Issue in this methodology is the reason of specific term selection from the large text, i.e., all the appeared terms cannot be processed, and this approach was recommended for limited number of terms.

Word2Vec (SGNS)/Word Embeddings: Majority of the study used this approach for determining cross-domain ambiguities from requirements which generate language models and computes semantic similarity between words of different domains. Word2Vec model uses cosine similarity for the computation of similarity among words in vector space. The prominent feature of this approach is that it selects those terms which occurs highly frequent in one domain and that appears sufficiently in other domain(s).

5. Which ambiguity detection approach has the better research productivity over the years from 2012 to 2022?

The literature has revealed several tools, strategies, and NLP approaches, which are addressed below:

One of the ambiguities type in the requirements brought on by several terminologies used in the natural language requirements. In other words, although a term may exist in several different contexts, its meaning will change depending on the context. When two different concepts are mapped to the same term, ‘term ambiguity’ could arise. This could be because of the language being used, the lack of descriptions, or both [11]. In situations where more than one domain is involved in the creation of a system, this type of ambiguity is particularly difficult. The same phrase is examined in other domains if it occurs frequently enough there after such ambiguous terms in one domain are identified based on their semantics. All the ambiguous terms were then ranked and graded in accordance with their degree of ambiguity. This method of ambiguity detection and rating was used to create the article [1].

6. How cross-domain ambiguity detection approach may be ranked as per their accuracy?

REGICE (Requirement Glossary term identification and clustering), which basically extracts terms, computes their similarity, and divides those terms into relative terms clusters as presented in [39], is suggested in the [31] in detail because term ambiguity in the requirements across the domain is the priority. However, it only works on one requirement document at a time. Ontologies are used by one module suggested for cross-domain ambiguity identification [37]. Wiktionary terms and Wikipedia entries with several meanings were seen as ambiguous. The main issue with this methodology is that it only works for a small number of terms because it is impossible to process all the terms that exist in the immense text.

An approach was employed by most of the articles to identify cross-domain ambiguity from requirements that produce language models and compute semantic similarity between terms of various domains [1], [9], [10], [19], [33]. The Word2Vec model calculates word similarity in vector space using cosine similarity. This approach's standout characteristic is that it chooses words that are used frequently in one domain and that sufficiently exist in other domains.

2.4 Conclusion of Literature Review

The existence of cross-domain ambiguity in the requirements covered by this SLR has been addressed in several research articles that have reported methodologies, approaches, and combinations of these approaches. It also gives a general overview of how these methods are used thoroughly for the intended purpose.

The use of a natural language processing technique for the identification of inconsistency within natural language requirements was extensively studied, covering nearly all the significant concerns in the limited yearly frame. After the necessary filtration in accordance with the selection criteria, 29 landmark publications that were published in the main repositories between 2012 and 2022 were studied and thoroughly analyzed. The articles were subsequently separated into three further groups for assessment and further investigation to get the answers to the research questions. These groups are 'Ambiguity Detection NLP Approaches - 10 papers', 'Tools for Ambiguity Detection - 08 papers', and 'Ambiguity Detection Techniques - 07 papers'.

To obtaining the necessary and precise result, a thorough analysis of the final chosen articles was carried out. The entire synthesis is thus separated into three groups. The Word2Vec algorithm with the integration of POS tagging, Wikipedia crawling, language model development, and Word-embeddings were found in five relevant studies. Two research studies found that term ambiguities might be detected using ontologies. Semantic-reasoner, N-gram module, and context clustering were employed in the development of ontologies. Another strategy employing the REGICE tool that combined POS tagging with QuARs was employed for the same objective.

The identified approaches analyzed each one in detail with an aim to carry out the techniques which work on automated cross-domain ambiguities. As compared to other identified tools, techniques and approaches which are proposed for cross-domain ambiguities, 'word embedding through SGNS variant of Word2Vec used in combination with other algorithms' is implemented more than any other proposed techniques successfully for the required purpose.

2.5 Summary Table of Literature Review

Table 8. Detail of the Cross-Domain Ambiguity Studies

Tool / Approach	Reference No.	Author(s)	Publication Year	Combination Tool(s) / Algorithm(s)	Comparison
Req Glossary term identification and clustering (REGICE)	[31]	S. Jarzabek	2020	QuARS	Works on one requirement document only at a time
				Part-of-Speech (POS) Tagging	
Develop Ontology representing domain knowledge	[13]	M.p.s Bhatia	2016	Ontology Based Framework	Recommended for limited number of terms
				Semantic reasoner	
	[37]	T. Baldwin	2013	N-gram module	
				Clustering the contexts used for either of the above	
Word2Vec (SGNS) Word Embeddings	[1]	A. Ferrari	2019	Word2Vec (SGNS) NLP Techniques Language Models	Not any of the Above limitations. Works with multiple Domains
	[9]	A. Ferrari	2017		
	[10]	A. Ferrari	2018		
	[19]	S. Mishra	2019		
	[33]	V. Jain	2020		

Table 8 presents a summary of the literature on tools/frameworks that are proposed by worthy researchers. These approaches are used for cross-domain ambiguity detection / resolution. It also provides to-the-point knowledge about tools/frameworks as the Algorithm used, the comparison in these approaches/results, author's information, publication year and cited reference number.

2.6 Research gap

In this section area for the improvement in the existing research literature is discussed. A detailed analysis of the selected articles was carried out in which tools, techniques, frameworks, and other NLP approaches were used. After a comprehensive screening

procedure filtered the research that stipulates an endorsement for the detection of the ambiguities caused by terms in requirements across different domains.

The gap found in our selected studies was that research focused on using different approaches like using ontologies, QuARs tool with NLP approaches, and Word2Vec algorithm with NLP approaches. The first two approaches have some deficiencies as mentioned in Table 8 in the comparison column. However, Word2Vec model is featured as it selects ambiguous yet highly frequent terms in more than one domain. Data of different domains was taken from the Wikipedia articles for which the Wikipedia crawling of the relevant domain articles was performed. After which language models are generated through Word2Vec model based on the text corpus. The model uses cosine similarity for the computation of similarity among words in vector space.

The words that sense ambiguous based on the context of the word are marked as ambiguous and assigned dissimilarity score. But some of the words in the text that have more than a single meaning in different domains are scored less by the model as compared to other words. It is therefore the target of this research is to apply some other algorithm / model using the same parameters and analyze ambiguity sense in terms from the given corpus or other analyzer as well to find an improved solution to the gap.

CHAPTER 3: PROPOSED APPROACH

This Chapter presents approach for the most suitable alternative of the Word2Vec algorithm for the detection of ambiguities caused by the terms used in natural language requirements that are domain dependent. The approach has a pipeline of data collection, pre-processing of data, building language model, applying alternative algorithm(s), and resultant terms score of dissimilarity.

The alternate algorithm used in this approach is FastText. The potential ambiguous terms were scored as per their dissimilarity score. The score is then compared with the previously generated dissimilarity score of the previous approach. Figure 7 shows the basic procedure of this approach.

The approach starts with domains of the software requirements. Data considered for the domain is taken from Wikipedia via Wikipedia crawling. The process gives us data for each domain to be processed further. Wikipedia articles for predefined domains are crawled and resultant domain documents are collected. The domain documents become the input for the generation of language models. Language models for each of the domains are separately generated. The cross-domain terms are selected from all domain documents. Those terms are selected which appeared enough in at least two of the domains and ambiguous by meaning with respect to its domain. This process produces dominant-shared terms of the domain. The dominant-shared terms are then scored as per its ambiguity level called ambiguity score. This score is generated through language models of the domains. Higher score indicates that the term has more different interpretations as compared to another domain and vice versa. The terms are then ranked as per their ambiguity score.

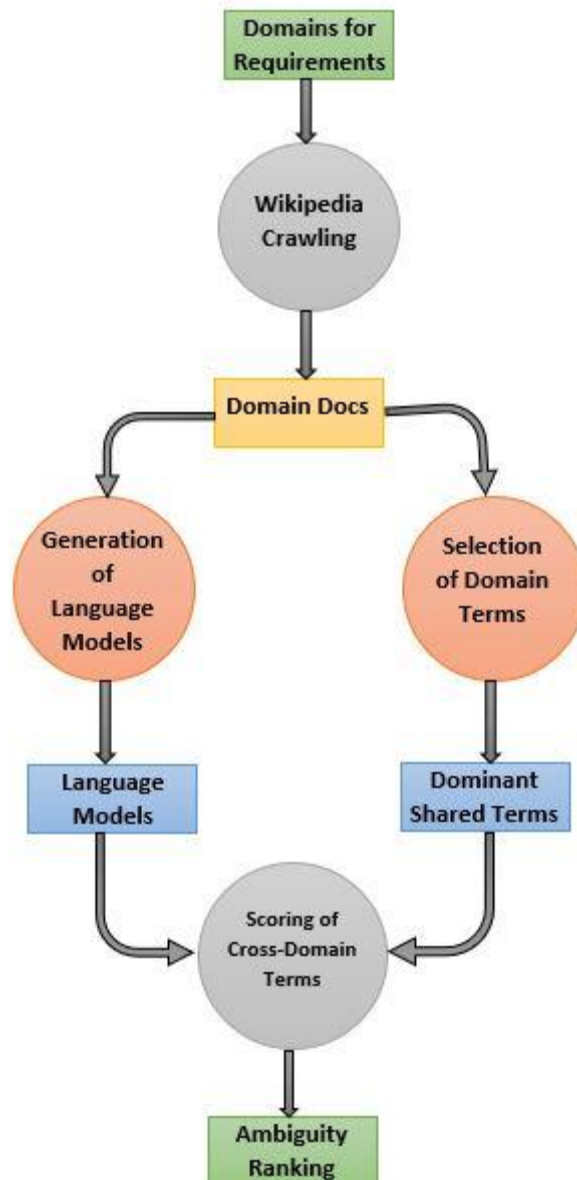


Figure 7. Overview of Measuring Cross Domain Ambiguity

3.1 Word Embedding

Word embedding comprehend combination of approaches to represent the words in a vector space as numerical vector. This representation of words enables to measure similarities in these vectors and thus the semantic similarity in the words can be calculated [9]. It is automatic word representations that incorporate semantic information from a specified corpus of natural language (NL). Specifically, a vector space is built based on given input called word embeddings, which are vector representations of words. If the matching words in the input corpus are more semantically similar, the distance between word embeddings will be closer. Consequently, the distance between vectors will be less for

more similar words (given a domain specific corpus) than the distance in vectors for less similar words.

The base of word embedding is distributional structure [40] where the author stated about the distributional facts that a speech can possibly be divided into discrete segments which has distribution in speech yet independent in its own. These segments are considered as elements. Other distributional facts stated in some elements are same as per their distribution. These similar terms are combined in a set called "similarity groupings". The degree of dependance of same set elements can be measured upon the utterances of the elements. The sets interpretation of these elements varies domain to domain but if an element appear in the same context may have similar meanings.

Collobert and Weston proposed to train word embeddings using a deep neural network expects words based on two words on the right and two words on left [41]. One of the most well-known word-embedding approaches was presented in the study [42]. The term "skip-gram with negative sampling" (SGNS) refers to this technique. The log-bilinear models were suggested to efficiently learn continuous word representations from very large datasets. Some of the vector space models and their classes are studied in the Turney and Pantel article [43].

The "Word2vec" model created by Google researchers [44] learns and creates word-embedding from corpus of natural language text. The Word2vec implementation of skip gramme negative sampling (SGNS) [42] predicts a set of words $w \in V_W$ and their contexts $c \in V_C$, where V_C and V_W are respectively the vocabularies of the context-words and input-words. A word's context words w_i are a group of words $(w_{i-win}, \dots, w_{i-1}, w_{i+1}, \dots, w_{i+win})$ for a given window size win . Each $w, c \in \mathbb{R}^d$ is the d dimensional word embedding w word and c context. Each word/context vector formed by Word2Vec from text corpus and analyzed to compare them for semantic similarity [45]. Negative sampling's (NS) primary goal is to develop accurate word-vector representations from a corpus. Word vectors represented in the paper [19] from the different domain's requirements using Word2Vec and the author aimed to use FastText for the same purpose in the future.

The study of Fabiano and Nan [46] demonstrated that Word2Vec in comparison with FastText is less suitable for the representation of high dimensional word vectors specifically when transitory data, multiword combinations, spelling mistakes or if there is multilingual data exists. FastText specifically makes advantage of the sub-word information to

make effective representation of rare words when embedded [47]. The author of article [48] trained different models for vector representation from text corpus (tweets), demonstrated in his study that FastText had the higher precision among other models i.e., Word2Vec and GloVe and effectively handle different disparities in linguistic styles which is a part of natural language. FastText is a useful way to represent sentences as it helps in utilizing word morphology, which allows words with similar radicals to share training [49].

The Word2Vec extension FastText proposed by AI research laboratory Facebook in 2016 [50], [51]. FastText divides words into multiple n-grams called sub-words [52], in contrast to Word2vec, which input a single word to the neural network. Example of the n-grams for a word *string* is *str*, *tri*, *rin*, *ing*. The total of these n-grams will be embedding vector to represent the word *string*, as shown in Figure 8. We receive word-embedding for all n-grams provided the training dataset after the neural network is fully trained. Due to the high likelihood that a certain n-grams will also exist in other words, FastText accurately depicts unusual words. It's important to note that Word2vec does not offer any vector representations for words that are not present in the corpus.

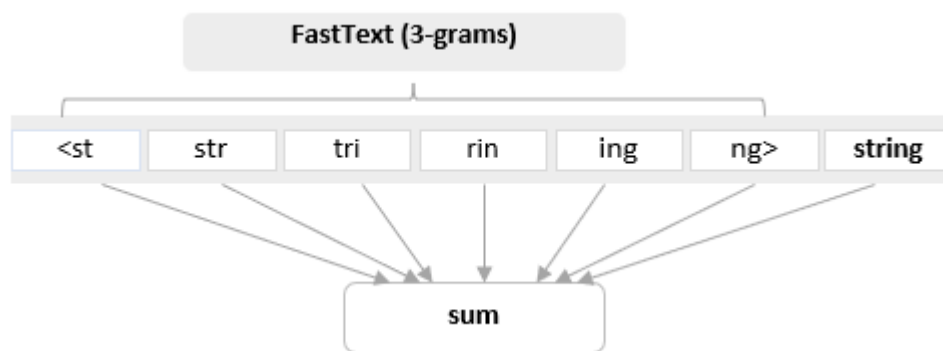


Figure 8. FastText 3-gram Representation of Word "string"

The cosine similarity is used by the FastText model to determine the semantic relationship between two distinct words in a vector space. If we assume that there are two words embedding vectors w' and w'' , the following equation can be used to compute cosine angle of these vectors [45].

$$\cos (w', w'') = \frac{w' \cdot w''}{|w'| |w''|}$$

The scoring range of a word is in between 0-1. The words are employed in nearly a different context and are more semantically dissimilar if the score is close to 1. The

opposite is true only if the score is nearer to 0, which indicates that the terms are not so related. Figure 9. Training FastText on Domain Texts shows training of the FastText on the input domain's corpora.

```
def generate_N_grams(words, ngram=1):
    temp = zip(*[words[i:] for i in range(0, ngram)])
    ans = ['_'.join(ngram) for ngram in temp]
    return ' '.join(ans)

def train_model_fasttext(dir, language, lemmatizer, size, window, min_count, iter, ngram=1, threads=12):
    corpus_name = dir.split("/")[-1]
    if os.path.exists(os.path.join(INTERMID_DIR, corpus_name)):
        os.remove(os.path.join(INTERMID_DIR, corpus_name))
        # combine_all_files(, corpus_name)

    list_of_files = [(file_name, corpus_name) for file_name in glob(dir + "/*.txt")]

    with concurrent.futures.ThreadPoolExecutor() as executor:
        executor.map(combine_all_files, list_of_files)

    lemmas = clean(os.path.join(INTERMID_DIR, corpus_name), lemmatizer)
    lemmas = generate_N_grams(lemmas, ngram=1)
    open(os.path.join(INTERMID_DIR, corpus_name), mode="w").write(lemmas)
    model = fasttext.train_unsupervised(os.path.join(INTERMID_DIR, corpus_name), dim=size, minCount=min_count,
                                       thread=threads)

    return model
```

Figure 9. Training FastText on Domain Texts

3.2 Approach

The suggested method for creating a prioritized list of possibly ambiguous terms is shown in Figure 7. To limit the scope of our process, we concentrate on nouns rather than the terms as a whole. However, the strategy is transferable to other linguistic categories.

The strategy is as stated below. To extract domain-specific documents for a given domain, we first crawl Wikipedia (Wikipedia Crawling). Then, we use the FastText model to train the word embeddings (Generating Language Models) from the corpus formed up of domain-specific documents.

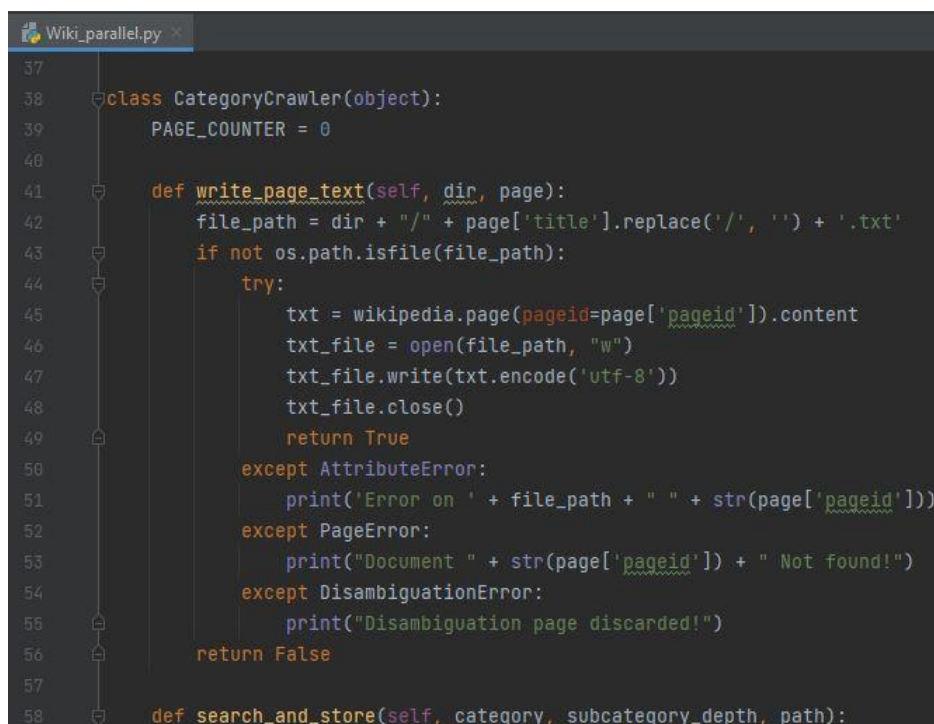
The next step is to look for the most common nouns that appear across texts (Cross-domain Term Selection). We compare the meanings of each of the nouns, that we refer to as dominant shared terms, across the various domains. As was previously said, these are words that are frequently used in different domains, and as a result, when stakeholders use them in different contexts, they may frequently lead to misconceptions. The dominating shared terms' degree of ambiguity is measured in the final step, and a score (Cross-domain Ambiguity

rating) is provided based on this measurement. This is accomplished by using vector space using FastText embeddings to represent a similarity space, where the two related words would be considered as similar if its embeddings (through cosine similarity) found to be closer to each other. FastText can provide a ranked list of the given words that are most similar in the language model's embeddings space along with a measure of how similar they are. Therefore, to determine the degree of ambiguity of a dominating shared term, we evaluate the list of more similar words generated by various domain-specific language models. The dominant shared term is considered as less ambiguous if the two lists have more common words and if the similarity-values of the exact same terms in two lists are closer to each other.

The following subsections provide comprehensive descriptions of the various steps.

3.2.1 Wikipedia Crawling

The Wikipedia Crawling stage makes a corpus C_i for every domain D_i for a set $D = (D_i: i = 1 \dots n)$ of n domains. Each C_i contains pages from a Wikipedia portal that belong to a particular area domain. Every Wikipedia portal is organized as a tree, with categories acting as the nodes and pages acting as the leaves.



```
Wiki_parallel.py x
37
38 class CategoryCrawler(object):
39     PAGE_COUNTER = 0
40
41     def write_page_text(self, dir, page):
42         file_path = dir + "/" + page['title'].replace('/', ' ') + '.txt'
43         if not os.path.isfile(file_path):
44             try:
45                 txt = wikipedia.page(pageid=page['pageid']).content
46                 txt_file = open(file_path, "w")
47                 txt_file.write(txt.encode('utf-8'))
48                 txt_file.close()
49                 return True
50             except AttributeError:
51                 print('Error on ' + file_path + " " + str(page['pageid']))
52             except PageError:
53                 print("Document " + str(page['pageid']) + " Not found!")
54             except DisambiguationError:
55                 print("Disambiguation page discarded!")
56             return False
57
58     def search_and_store(self, category, subcategory_depth, path):
```

Figure 10. Wikipedia Articles Crawling

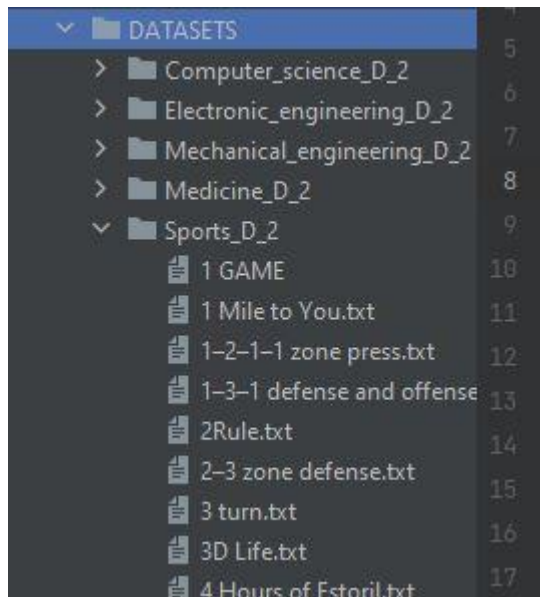


Figure 11. Wikipedia Articles of Domains as Text Files

One can visit a base category of *mechanical engineering*¹ domain to see a visual sample. We developed an algorithm that, assuming the base category of the portal, conducts breadth-first search upon subclasses of the base category, and gets all the Wikipedia articles that are available through the search, to access the pages classified by a Wikipedia portal. We established a maximum limit of 10,000 articles to be retrieved from each portal because the total number of pages that could be accessed might be very large. In addition, we restricted the depth of the subcategories of a domain to a maximum of 2. Reaching articles with more general content in comparison to deep subcategories is possible by concentrating on higher-level subcategories. Figure 10 shows the code of extracting Wikipedia pages as text files, while Figure 11 shows extracted files view of different domains.

3.2.2 Pre-Processing

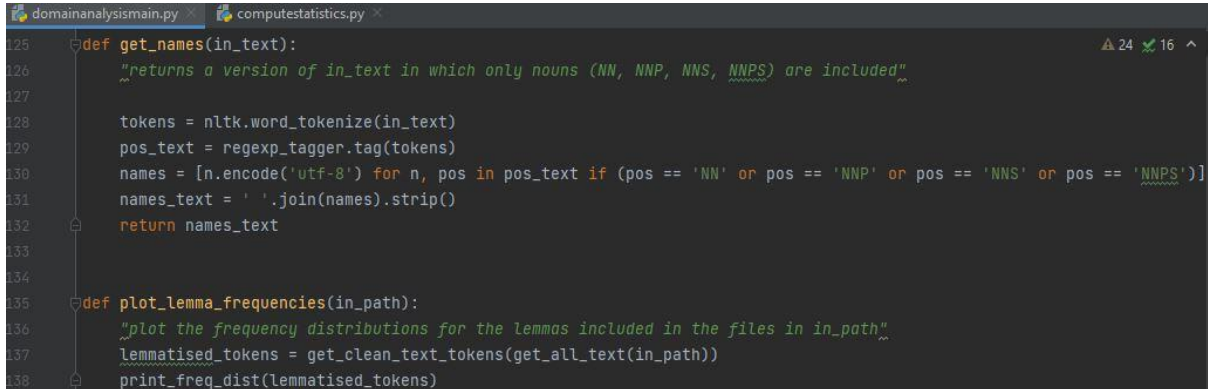
Each of the corpus is pre-processed by:

- a) Changing each word's case to lowercase.
- b) Eliminating stop words such as common terms like pronouns, articles and conjunctions are avoided because they do not have a special meaning in this context.
- c) Lemmatizing by converting every single term to its corresponding lemma, which enables each word inflections (i.e., processes, process) to be treated as a single word

¹ https://en.wikipedia.org/wiki/Category:Mechanical_engineering

(i.e., process). We employ the WordNet Lemmatizer from NLTK Package² Python in this implementation.

Figure 12 shows some of the pre-processing code used for the input text.



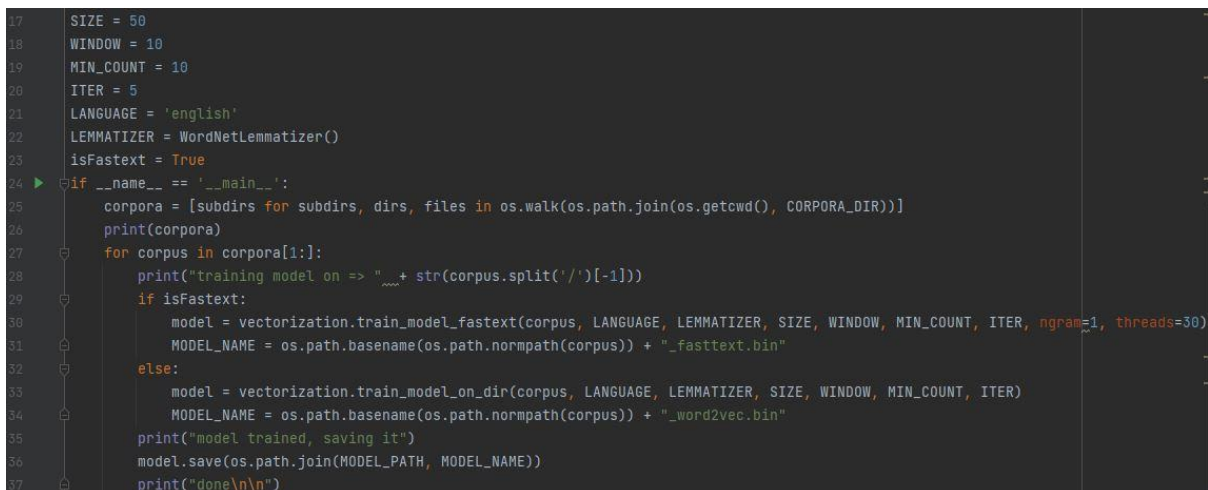
```
domainanalysismain.py | computestatistics.py
125 def get_names(in_text):
126     """returns a version of in_text in which only nouns (NN, NNP, NNS, NNPS) are included"""
127
128     tokens = nltk.word_tokenize(in_text)
129     pos_text = rexp_tagger.tag(tokens)
130     names = [n.encode('utf-8') for n, pos in pos_text if (pos == 'NN' or pos == 'NNP' or pos == 'NNS' or pos == 'NNPS')]
131     names_text = ' '.join(names).strip()
132     return names_text
133
134
135 def plot_lemma_frequencies(in_path):
136     """plot the frequency distributions for the lemmas included in the files in in_path"""
137     lemmatised_tokens = get_clean_text_tokens(get_all_text(in_path))
138     print_freq_dist(lemmatised_tokens)
```

Figure 12. Pre-processing of the Text

As an example, the phrase "meetings are arranged for requirements analysis" becomes "meeting arrange requirement analysis" after pre-processing.

3.2.3 Language Model Generation

The FastText algorithm [50] is used in this stage to develop language models M_i specific to its domain using each input corpus C_i . The amount of L , or the total length of context that is to be observed, the measurement of d , or the dimensionality of the embedding, and the m value, or the smallest amount of occurrence that a word must have for the algorithm to take it into consideration, must be specified. We used $L = 10$, $d = 50$, and $m = 10$ for our illustration.



```
17 SIZE = 50
18 WINDOW = 10
19 MIN_COUNT = 10
20 ITER = 5
21 LANGUAGE = 'english'
22 LEMMATIZER = WordNetLemmatizer()
23 isFasttext = True
24 if __name__ == '__main__':
25     corpora = [subdirs for subdirs, dirs, files in os.walk(os.path.join(os.getcwd(), CORPORA_DIR))]
26     print(corpora)
27     for corpus in corpora[1:]:
28         print("training model on => " + str(corpus.split('/')[-1]))
29         if isFasttext:
30             model = vectorization.train_model_fasttext(corpus, LANGUAGE, LEMMATIZER, SIZE, WINDOW, MIN_COUNT, ITER, ngram=1, threads=30)
31             MODEL_NAME = os.path.basename(os.path.normpath(corpus)) + "_fasttext.bin"
32         else:
33             model = vectorization.train_model_on_dir(corpus, LANGUAGE, LEMMATIZER, SIZE, WINDOW, MIN_COUNT, ITER)
34             MODEL_NAME = os.path.basename(os.path.normpath(corpus)) + "_word2vec.bin"
35         print("model trained, saving it")
36         model.save(os.path.join(MODEL_PATH, MODEL_NAME))
37         print("done\n\n")
```

Figure 13. Generation of the Language Models

² <https://www.nltk.org/>

The values have been chosen based on preliminary inspections on data. Figure 13 is the code preview of language models generation using FastText.

3.2.4 Elicitation Scenarios

A total of five domains are represented in the elicitation scenarios. That is:

- i. Computer Science (CS)
- ii. Electronic Engineering (EEN)
- iii. Mechanical Engineering (MEN)
- iv. Medicine (MED)
- v. Sport (SPO)

Each scenario takes a portion of the domains into account. The initial four can be viewed as an interview between an analyst with computer science expertise and a domain expert. The final three scenarios consist of a group meeting for elicitation with related domain specialists. The situations are briefly explained below, together with a code (“Int” will be used for interviews and “Mee” for the group-meetings), and the abbreviations of the domains as mentioned above is taken into consideration. The details have just a narrative purpose by offering examples of real-world settings in which our technique might be applied. Detail of the scenario are given below in Table 9. Scenario abbreviations are listed against each scenario that is considered for the interview and meeting for the sake of eliciting requirements for the specific domain.

Table 9. Scenarios Considered for Requirement Elicitation

Sr.	Scenario	Scenario Code	Detail
1.	Light Controller (CS-EEN)	Int1	A controller with a generic piece of software incorporated in it that controls the room illumination.
2.	Mechanical CAD (CS-MEN)	Int2	A program that helps in the designing of mechanical parts.
3.	Medical Software (CS-MED)	Int3	A program that helps in the diagnosis of specific diseases according to symptoms.
4.	Athletes Network (CS-SPO)	Int4	An online community of athletes.
5.	Medical Device (CS-EEN-MED)	Mee1	A medical device connected with a mobile app, used to track the patient's heart rate.

6.	Medical Robot (CS-EEN-MEN-MED)	Mee2	A surgical robot operated by a computer.
7.	Sport Rehab Machine (CS-EEN-MEN-MED-SPO)	Mee3	A technologically advanced rehabilitation device designed especially for athletes.

3.2.5 Cross-Domain Ambiguity

The approach was put into practice in Python with the help of several other necessary libraries, including the Wikipedia Python API ³, gensim ⁴ for FastText implementation, and spaCy library⁵ for tasks related to NLP [53]. Such as it is used for POS tagging for the identification of nouns.

Table 10. Domain with Wikipedia Articles

Sr.	Domain Name	Wikipedia Articles	Words	Vocabulary
1.	Computer_Science	10,000	6971198	140757
2.	Electronic_Engineering	4,901	3427293	82098
3.	Mechanical_Engineering	6,513	4081887	97122
4.	Medicine	10,000	6737874	183625
5.	Sports	10,000	7326483	187145

Table 10 lists the downloaded articles of Wikipedia for given domains as well as the size of vocabulary and number of words in total pages related to a domain. Since we crawled all the pages contained in the corresponding Wikipedia portals, several corpora of the domain (such as EEN & MEN) contain fewer than 10,000 documents because articles in these portals were less than the mentioned threshold.

There are five separate language models produced against each domain-specific corpus.

3.2.6 Cross-Domain Term Selection

In this step we will select the terms that will be checked for ambiguity occurrence in the domain corpora *Corp* of each domain. Such terms will be considered as dominant shared

³ <https://pypi.org/project/wikipedia/>.

⁴ <https://radimrehurek.com/gensim/>.

⁵ <https://spacy.io>.

terms T_{Dom} . These terms will be extracted from the corpus through proper procedure using Algorithm 1 [1].

Algorithm 1: Selection of Cross Domain Terms

Select_Dominant_Shared_Terms ($Corp, m, n$)

1. $T_{Dom} = Empty$
2. **for** $Corp_a \in Corp$ **do**
3. **for** $term \in Vocab(Corp_a)$ **do**
4. **if** Frequency ($term, Corp_a$) $\geq m$ **do**
5. **if** POS ($term$) == Noun **do**
6. **if** $term \notin T_{Dom}$ **do**
7. **for** $Corp_b \in (Corp - Corp_a)$ **do**
8. **if** Frequency ($term, Corp_b$) $\geq n \times$ Frequency ($term, Corp_a$) **do**
9. $T_{Dom} = T_{Dom} \cup \{term\}$
10. **Return** T_{Dom}

There are three inputs to this algorithm upon which it operates. The first input parameter is corpora $Corp = [Corp_a : a = 1 \dots 5]$ of a specific domain. Second parameter is m , which is the minimum number of existences of a $term$ in the Vocabulary “**Vocab**” of a specific domain Corpora $Corp_a$. If a term is m or more then m number of time appeared in one specific domain, then this term would be recommended as candidate for dominant shared terms T_{Dom} inclusion. Similarly, the candidate term will be checked for its “**Frequency**” in at least on other domain corpora $Corp_b$. The third parameter is n , if candidate term is n or more than n -time frequent in another domain then it will be considered for further checks to include it in T_{Dom} . Thus, m and n are the frequency ratio of a term in two different domains. If a term is more frequent in the corpora of one domain but not sufficiently occur in any other domain, then it will not be considered as part of T_{Dom} . Moreover, the term will be checked if it is not already the part of T_{Dom} and the part-of-speech “POS” tag of the term is “Noun”. If all the conditions are satisfied, then the term will be selected to be the part of T_{Dom} .

This procedure shall be repeated for each term of the corpus belongs to a domain to produce list of dominant shared terms. Figure 14 shows the code used to call function for cross-domain terms selection.


```

20 ▶ if __name__ == '__main__':
21     models = dict()
22     models['cs'] = Word2Vec.load(os.path.join(MODEL_PATH, "Computer_Science_D_2.bin"))
23     models['med'] = Word2Vec.load(os.path.join(MODEL_PATH, "Medicine_D_2.bin"))
24     models['sport'] = Word2Vec.load(os.path.join(MODEL_PATH, "Sports_D_2.bin"))
25     models['ele'] = Word2Vec.load(os.path.join(MODEL_PATH, "Electronic_Engineering_D_2.bin"))
26     models['mec'] = Word2Vec.load(os.path.join(MODEL_PATH, "Mechanical_Engineering_D_2.bin"))
27     models['lit'] = Word2Vec.load(os.path.join(MODEL_PATH, "Literature_D_2.bin"))
28
29     # models['cs'] = FastText.load_fasttext_format(os.path.join(MODEL_PATH, "Computer_Science_D_2_fasttext.bin"))
30     # models['med'] = FastText.load_fasttext_format(os.path.join(MODEL_PATH, "Medicine_D_2_fasttext.bin"))
31     # models['sport'] = FastText.load_fasttext_format(os.path.join(MODEL_PATH, "Sports_D_2_fasttext.bin"))
32     # models['ele'] = FastText.load_fasttext_format(os.path.join(MODEL_PATH, "Electronic_Engineering_D_2_fasttext.bin"))
33     # models['mec'] = FastText.load_fasttext_format(os.path.join(MODEL_PATH, "Mechanical_Engineering_D_2_fasttext.bin"))
34     for model in models.keys():
35         print(model)
36         pprint(models[model].wv.most_similar(TERM, topn=20))
37         #pprint(models[model].fasttext.most_similar(TERM, topn=20))

```

Figure 14. Code to Call Function for the Selection of Cross-Domain Terms

3.2.7 Cross-Domain Ambiguity Ranking

This step generates ambiguity ranked list A_{Dom} of the dominant shared terms T_{Dom} . As per the language models $\acute{M} = [M_{\vartheta} : \vartheta = 1 \dots 5]$, the cross-domain ambiguity calculated, and the degree of ranking is measured.

Algorithm 2 given below is used to calculate the ambiguity ranking. Detail of the algorithm is as follows. Every dominant shared term may have similar words in each domain. The concept of similar words is that a word will be considered as similar with the dominant shared term if it has the same linguistic context, such that they have similar neighbor words in the input corpora of the domain upon which the language models are generated. The similar word's lists of a dominant shared term in each domain is generated using the FastText Model, called similarity lists. The similarity lists are then compared to evaluate the meaning variation of dominant-shared term. The dissimilarity score of each dominant shared term is computed on comparing the similarity lists, as this comparison of the list is indirectly linked with it. The term will be considered more ambiguous if the dissimilarity score is high. This dissimilarity score can be reflected as the **ambiguity score** to a given dominant shared term.

Algorithm 2: Ambiguity Valuation of Cross Domain Terms

Ambiguity-Ranking (T_{Dom}, \acute{M}, h)

1. $A_{Dom}, simL_{\vartheta}, simV_{\vartheta}, Union, Rnk, Var, \sigma = Empty$
2. **for** $term \in T_{Dom}$ **do**
3. **for** $M_{\vartheta} \in \acute{M}$ **do**
- 4.

```

5.       $simL_a[term], simV_a[term] \leftarrow \text{MostSimilar}(term, M_a, h)$ 
6.
7.       $Union[term] \leftarrow simL_1[term] \cup \dots \cup simL_5[term]$ 
8.
9.      for  $word \in Union[term]$  do
10.
11.      $Rnk[term][word] \leftarrow \text{BestRank}(simL_1[term][word] \dots simL_5[term][word])$ 
12.      $\sigma[term][word] \leftarrow \text{Variance}(simV_1[term][word] \dots simV_5[term][word])$ 
13.
14.      $Var[term] \leftarrow \sum_{word \in Union[term]} \frac{\sigma[term][word]}{Rnk[term][word]}$ 
15.
16.  $A_{Dom} = \text{Sort}(T_{Dom}, Var)$ 
17. Return  $A_{Dom}$ 

```

While comparing the words in the similarity lists, if a word is not available in other similarity lists, then the word will obtain a zero-similarity value. For example, the word *player* lists in the similarity-list for *loop* in the domain of sports for the Tug-of-war, but it would not appear in the computer domain. Similarly, if the lists have more similar words against a dominant shared term and the words possess more close similarity values to each other, then the dominant-shared term is having a consistent interpretation in various domains and is not considered as ambiguous. On the other hand, if the lists have a rare similar word against a dominant shared term, then that term is likely to have diverse interpretation in different domains and would be considered as more ambiguous. A word will be weighted more if it is most similar to a dominant shared term as per the language model. The maximum rank of a word in all the similarity lists will be count for the ambiguity score computation. More concisely, for each word, its sum of best-ranked variance of the similarity-values is associated with the ambiguity score.

```

88 def ambiguity_mse_rank(domains, vocab_size=100, w2v_topn=100):
89     words = get_frequent_shared_words_spacy(domains, vocab_size)
90     output = list()
91     for word in words:
92         sorted_tops = list()
93         sorted_words = list()
94         tops = list()
95         for domain in domains:
96             sorted_tops.append(domain.wv.most_similar(word, topn=w2v_topn))
97             sorted_words.append([word for word, score in sorted_tops[-1]])
98             tops.append(dict(sorted_tops[-1]))
99         shared = set()
100        for top_word in tops:
101            shared.update(top_word.keys())
102        mse = 0
103        for shared_word in shared:
104            min_rank = w2v_topn + 1
105            for sorted_word in sorted_words:
106                try:
107                    min_rank = min(min_rank, sorted_word.index(shared_word) + 1)
108                except: pass
109            scores = list()
110            for top in tops:
111                scores.append(top.get(shared_word, 0))
112            mse += np.var(scores) / min_rank
113        counts = list()
114        for domain in domains:
115            counts.append(domain.wv.vocab[word].count)
116        output.append((word, len(shared), mse, counts))
117    return sorted(output, key=lambda x: -x[2])

```

Figure 15. Dominant Shared Terms Ranking

Input to the Algorithm 2 is set T_{Dom} of dominant-shared term, the \hat{M} language model and the length h of a similarity list [10]. The ambiguity score is calculated against each $term \in T_{Dom}$. This process is accomplished by determining the similarity-lists $simL_a[term]$ along with the similarity-values $simV_a[term]$ against each of the language model. In the next step union of the similar words $Union[term]$ in the similarity list are taken. The maximum similarity of a word in the $Union[term]$ is considered as the best one $Rnk[term][word]$ and it is calculated by BestRank function. For every word belong to the $Union[term]$, the variance of its similarity values $\sigma[term][word]$ is calculated against all the similarity lists using Variance function. Rank weights are then assigned by dividing the word variance $\sigma[term][word]$ with the best-ranked value $Rnk[term][word]$. Finally, sum of all the rank weighted variances $Var[term]$ produces the ambiguity score to a term belongs to T_{Dom} .

All the terms in the T_{Dom} are then sorted based on ambiguity score and a sorted list of ambiguous terms A_{Dom} is returned by the algorithm. Figure 15 shows the code used for ranking of the dominant-shared terms.

The number of dominant shared terms m that were intended to rank in our scenarios is 800. The frequency ratio n is set at 0.3, and the total amount of words h for the similarity lists which we compare using the Ambiguity-Ranking process is set at 100. The values for these factors were taken from the previously applied approach [1]. With these parameters, we regenerate lists of dominant shared terms in each scenario, sorted according to the ambiguity level.

CHAPTER 4: IMPLEMENTATION, RESULTS & DISCUSSION

In this chapter, implementation of the approach as stated in the chapter 3 is discussed, and results of the approach are analyzed in detail. The implementation consist of data collection of different fields and results preparation and examination. Basic information of the approach implementation, use of algorithms and programs is also discussed in this section.

The research model of previous literature and approach by researchers were configured successfully and the same approach with different model algorithm was applied. Common code of the literature [54] was reused in which necessary modification was performed for the new models and results generation.

The dataset consists of Wikipedia articles and the previous implementation was performed in 2018 [1]. As the Wikipedia articles are modifiable, therefore articles of the domains were crawled again, and the same approach was applied using old model and then used the new model on the same data. The reason behind was to accurately get results for the same data and perform comparison among different model results. As if we get results on new model using new corpora, it would not be effectively compared with the results of old model on old dataset and it is more likely to have a different corpus for the application of model.

4.1 Data Collection/Dataset

Data of the five domains was considered for the language models generation, and data for the domain is obtained from Wikipedia articles. For the requirement elicitation, 07 scenarios were made for interviews and meetings. Each scenario considered a subset of the domains. The first four can be seen as an interview between a domain expert and an analyst with computer science knowledge. The remaining three scenarios included a group of experts from related domains for elicitation meeting. The interviews were between two domain experts while the meetings were among a group of domain experts as the remaining three scenarios were a composite of more than two domains (detail is given in the Table 9).

Interviews of the Electronic Engineering, Mechanical Engineering, Medical, and Sports domain experts were considered in the elicitation scenarios to be conducted with requirement's analyst of the computer science domain. On the other hand, for the meeting of *Medical Device* domain, there involved three domains computer science, electronic engineering, and medical in the meeting. Likewise for the *Medical Robot* scenario, four

domains involved in the meeting; that were computer science, electronic engineering, mechanical engineering and medical. Similarly, in the *Sport Rehab Machine*, all the five domains i.e., computer science, electronic engineering, mechanical engineering, medical, and sports involved in the meeting of the domain expert of the requirement's elicitation.

For each domain, maximum ten thousand Wikipedia articles as text files were used as the dataset for the approach implementation. The language models of the dataset generated using different models, and the data was pre-processed using NLP approaches and APIs. Different terms that were frequent in more than one domain were selected called dominant shared terms. The context of each dominant-shared terms are analyzed using the language models and most-similar words from the context of these dominant-shared terms were gathered, analyzed in such a way that those terms that shared more similar words in their context are considered as less ambiguous and the terms that had little similar words across its contexts tended to be more ambiguous in their interpretation in different domains. These terms were scored and ranked as per their ambiguity level and then sorted. In this way the results were generated. This procedure is discussed in detail in chapter 03.

As discussed earlier the language models were generated using Word2Vec algorithm using Gensim library. According to the literature most of the cross-domain ambiguity detection was performed using NLP approaches and Word2Vec algorithm for word vector representation and comparison of these vectors. We found in the literature that the same algorithm Word2Vec extension FastText was compared and used by many researchers for word vector representation on natural language text, and they demonstrated that FastText brought effective results in comparison to Word2Vec [19], [46], [47], [48], [49]. So, for the same text corpus, we used FastText model to generate language models, and then these language models were used for the selection of dominant-shared terms and their ambiguity rankings. Result of both the models will be compared in this chapter.

4.2 Experimental Setup

The proposed methodology was employed in Python language version 3.10 using PyCharm community edition 2.2.2022 various libraries as mentioned in section 3.2. Description of the algorithms is given in the section 3.2.6 and 3.2.7. Results to compare with state of the art were extracted using 06 code snippets that implement the algorithm for generating the results.

4.3 Ranking of Terms for Cross-Domain Ambiguity Using Word2Vec Models of the literature

The ambiguity ranking of dominant-shared terms are listed in table 2, 3, 4 of article [1]. The author listed terms of the scenario-based domains such as each scenario has different domains involved in it (detail is given in Table 9). The author listed 20 terms higher in rank and 20 terms that are bottom in rank list of dominant-shared term against each scenario. High rankings suggest a greater likelihood of ambiguity because words with higher rankings may have multiple meanings depending on the domains involved.

The approach of the state-of-the-art article was re-configured, and the output of the code was generated using six files as given in the code files [54] using the same dataset (language models). Four of the code files were marked significant as per result generation according to the scenarios stated above. Detail of these files given below:

Table 11. Output Number of Terms from Different Scenarios Using Language Models of Existing Code

File Name	Output Terms (Dominant-Shared-Term) Detail of Different Scenarios				
	CS-Ele	CS-Mec	CS-Med	CS-Sport	Total
ambiguity_tests_merge	30	30	30	30	120
ambiguity_tests_interviews_combi	30	30	30	30	120
ambiguity_tests_multi	Med-Sw	Med-Dev	Med-Robot	Sport-Rehab-Machine	Total
	40	40	40	40	160

ambiguity_tests_multi_combi	Med-Sw	Med-Dev	Med-Robot	Sport-Rehab-Machine	Total										
	30	30	30	30	120										
ambiguity_tests_merge_file_or_AMT	CS-Een	CS-Men	CS-Med	CS-Sport	Med-Sw	Med-Dev	Med-Robot	Sport-Rhb	Total						
	414	302	328	261	328	513	606	731	3483						
ambiguity_tests_pairs	CS-Ele	CS-Lit	CS-Mec	CS-Med	CS-Sport	Ele-Lit	Ele-Mec	Ele-Med	Ele-Sport	Lit-Mec	Lit-Med	Lit-Sport	Mec-Med	Mec-Sport	Total
	200	200	200	200	200	200	200	200	200	200	200	200	200	200	3000

Different abbreviations of domain name are used in the Table 11. CS for Computer Science, Ele/Een for Electronic Engineering, Men/Mec for Mechanical Engineering, Med for Medical, Sw for software, Lit for Literature, Rhb for Rehab, Dev for Device (the abbreviations mentioned here as it is in the output file). Similarly, there are different scenarios given having different combination of domains i.e., Med-Sw has two domains (CS-MED), Med-Dev has three domains (CS-EEN-MED), Medical Robot has four domains (CS-EEN-MEN-MED) and Sport Rehab Machine has five domains (CS-EEN-MEN-MED-SPO) in it.

Different files have output for different scenarios, detail of the file output is given below in Table 12:

Table 12. Code Files that Generated Dominant Shared Terms of Different Scenarios

Sr.	File Name	Output Detail
1.	ambiguity_tests_merge (ATM)	Lists terms for the Interviews related combination of domains used in requirement elicitation.
2.	ambiguity_tests_interviews_combi (ATIC)	Lists terms for the Interviews related combination of domains used in requirement elicitation.
3.	ambiguity_tests_multi (ATMu)	Lists terms for the Meetings related combination of domains for requirement elicitation.

4. ambiguity_tests_multi_combi (ATMC)	Lists terms for the Meetings related combination of domains for requirement elicitation.
5. ambiguity_tests_merge_for_AMT (ATMFA)	Lists terms for the Interviews and Meetings related combination of domains for requirement elicitation.
6. ambiguity_tests_pairs (ATP)	Lists dominant shared terms found in inter-domain combination

Most of the terms with the same ambiguity score as mentioned by the authors were generated and analyzed in detail. The output of each file was printed in the console that were saved as text file for further analysis.

Table 2 in the paper [1] listed the top 19 and bottom 20 ranked terms (total 39) for each of the scenario's interviews I1 & I2, along with ambiguity scores. I1 has been given the scenario name as Light Controller having two domains involved in it, that is: Computer Science and Electronic Engineering. While I2 has been given the scenario name of Mechanical CAD, having two domains involved in it, that is: Computer Science and Mechanical Engineering. Similarly, Table 3 listed the top 19 and bottom 20 ranked terms (total 39) for each of the two scenario's interviews I3 & I4, along with ambiguity scores. I3 has been given the scenario name as Medical Software having two domains involved in it, that is: Computer Science and Medical. While I4 has been given the scenario name of Athletes Network, having two domains involved in it, that is: Computer Science and Sports.

Table 4 of the base paper listed top 20 and bottom 20 rank terms (total 40) the meetings of 03 scenarios. The meetings are given name as M1, M2, M3, and the scenarios names are Medical device, Medical robot, and Sport rehab machine respectively. In the M1, the listed terms are basically the dominant-shared terms of three domains that are: Computer Science, Electronic Engineering, and Medical. In the M2, the listed terms are the dominant-shared terms of four domains that are: Computer Science, Electronic Engineering, Mechanical Engineering and Medical. Likewise, In the M3, the listed terms are the dominant-shared terms of five domains that are: Computer Science, Electronic Engineering, Mechanical Engineering, Medical and sports.

Locally generated output of the files was analyzed in which some of the terms were not found despite of using the language models already provided. Each term was searched in the output thoroughly and recorded the file name in which the term and its similar score was found.

Similar kind of tables were generated in which the file name was mentioned against the term, in the output of which the term and its score existed. The tables are mentioned below as Table 13, Table 14, and Table 15.

Table 13. Terms of the Table 2 of [1] along with its Output File

I1 CS, EEN	Score	Output File	I2 CS, MEN	Score	Output File
news	1.475026		hull	1.654419	
formula	1.466022	ATM	house	1.447376	
relation	1.452569	ATM	argument	1.391507	
surface	1.428484		bar	1.361833	
motor	1.406120		option	1.339622	ATM
flash	1.405892		room	1.336570	
studio	1.375377	ATM	disk	1.328546	
contact	1.375058		expression	1.317302	ATM
interpretation	1.343498	ATM	interpretation	1.316122	ATM
bell	1.343395		reduction	1.314786	ATM
reduction	1.292722	ATM	respect	1.306615	
head	1.282602		relation	1.295833	ATIC, ATM, ATP
deal	1.246001		representation	1.286481	ATM
link	1.199083		formula	1.270888	ATM
ion	1.178597		institute	1.245143	
desktop	1.171765	ATM	port	1.241405	ATM
pair	1.171018		rest	1.229141	
profile	1.141438		statement	1.215488	
particle	1.139154		string	1.214932	
school	0.238366		october	0.385298	
performance	0.235937	ATIC, ATP	state	0.366343	ATIC, ATP
term	0.233936	ATIC, ATM, ATP	category	0.360713	
article	0.226150	ATM	december	0.360142	
september	0.225291		period	0.359295	
conference	0.222977	ATIC, ATM, ATP	hour	0.355321	
number	0.221349	ATIC, ATM, ATP	cost	0.354388	ATIC, ATM, ATP
example	0.219640	ATIC, ATM, ATP	test	0.352514	
computer	0.216365	ATIC, ATM, ATP	space	0.346210	ATIC, ATM, ATP
range	0.214473		advantage	0.345200	ATM
student	0.212899		september	0.343362	
march	0.210152		day	0.338486	ATM
system	0.203739	ATIC, ATM, ATP	minute	0.331166	
december	0.202965		time	0.316961	ATIC, ATM, ATP
variety	0.201474	ATM	market	0.316713	ATM

point	0.200544	ATIC, ATM, ATP	range	0.289602	
science	0.197188	ATIC, ATP	variety	0.270675	ATM
april	0.196663		term	0.260514	ATIC, ATM, ATP
october	0.176143		year	0.245205	ATIC, ATM, ATP
june	0.159893		example	0.220515	ATIC, ATM, ATP
<i>Total Found</i>		<i>17</i>	<i>Total Found</i>		<i>19</i>

Total 17 terms of the I1 and 19 terms of I2 were traced in the output of all six code files. Some of the terms existed in more than one file output.

Table 14. Terms of the Table 3 of [1] along with its Output File

I3 CS, MED	Score	Output File	I4 CS, SPO	Score	Output File
mouse	1.599703		michael	1.849829	
matrix	1.542257		protein	1.677702	
argument	1.478305		statement	1.619878	ATM
client	1.430145		reduction	1.617916	
pair	1.423335		loop	1.535914	
editor	1.419535		string	1.522536	
arm	1.418809		founder	1.503727	ATM
strength	1.409200		formula	1.489577	ATM
house	1.396889		washington	1.484234	
relation	1.369563		effect	1.480257	
formula	1.356015		edge	1.447221	ATM
layer	1.348549	ATM	mechanism	1.435133	
loop	1.340321		layer	1.430993	
symbol	1.316503		corner	1.424862	
reduction	1.316299	ATM	threat	1.418862	
room	1.311479		driver	1.418780	
statement	1.311244	ATM	fire	1.412927	
expression	1.297684	ATM	surface	1.411994	ATIC, ATM, ATP
surface	1.296393	ATIC, ATM, ATMu, ATMC, ATP	wave	1.411662	ATM
report	0.383933		category	0.503321	ATP
concern	0.379293		level	0.497828	ATP
publication	0.346782	ATM	sale	0.488006	ATM
article	0.336694	ATIC, ATM, ATMu, ATMC, ATP	book	0.485710	ATIC, ATM, ATP
issue	0.333983	ATIC, ATM, ATMu, ATMC, ATP	term	0.481157	ATIC, ATM, ATP
history	0.330724	ATIC, ATM, ATMu, ATMC, ATP	company	0.456744	ATIC, ATM, ATP
student	0.321891	ATIC, ATM, ATMu, ATMC, ATP	market	0.448333	ATM
award	0.319464	ATIC, ATM, ATMu, ATMC, ATP	april	0.409362	
time	0.308453	ATIC, ATM, ATMu, ATMC, ATP	history	0.401996	ATIC, ATM, ATP
december	0.299433		child	0.394809	
april	0.283379		june	0.377780	

october	0.279610		september	0.365755	
march	0.279493		student	0.355196	ATIC, ATM, ATP
september	0.272048		time	0.353464	ATIC, ATM, ATP
june	0.264172		award	0.350350	ATIC, ATM, ATP
category	0.263526	ATM	article	0.343666	ATM
term	0.214670	ATIC, ATM, ATMu, ATMC, ATP	march	0.341266	
variety	0.201947	ATM	october	0.335467	
range	0.187092		december	0.331697	
year	0.187065	ATIC, ATMu, ATMC, ATP	range	0.329518	
<i>Total Found</i> 16			<i>Total Found</i> 18		

Total 16 terms of the I3 and 18 terms of I4 were traced in the output of all six code files. Some of the terms existed in more than one file output.

Table 15. Terms of the Table 4 of [1] along with its Output Files

M1 CS, EEN, MED	Score	Output File	M2 CS, EEN, MEN, MED	Score	Output File	M3 CS, EEN, MEN, MED, SPO	Score	Output File
argument	2.023164	ATMFFA	argument	2.363125	ATMFFA	consequen ce	2.626065	
relation	1.921059	ATMFFA	respect	2.180597		respect	2.596219	ATMFFA
formula	1.915565	ATMFFA	expression	2.180273	ATMFFA , ATM	statement	2.555115	ATMFFA
interpretati on	1.904085	ATMFFA, ATM	consequen ce	2.173204		michael	2.497249	
consequenc e	1.863527		statement	2.094200	ATMFFA	story	2.398440	ATMFFA
expression	1.849218	ATMFFA, ATM	ion	2.089713	ATMFFA	argument	2.363125	ATMFFA
arm	1.838583		father	1.952681	ATMFFA	brother	2.304480	ATMFFA
surface	1.811224	ATMFFA, ATMu, ATMC	institution	1.933082	ATMFFA	founder	2.238028	ATMFFA
house	1.802686	ATMFFA	relation	1.921059	ATMFFA	end	2.236240	
client	1.741765	ATMFFA	formula	1.915560	ATMFFA	ray	2.228692	
strength	1.730889	ATMFFA	interpretati on	1.904085	ATMFFA , ATM	relation	2.216734	ATMFFA
mouse	1.716812	ATMFFA	career	1.890641	ATMFFA	stability	2.213691	ATMFFA
appearance	1.681019	ATMFFA	option	1.875132	ATMFFA , ATM	institution	2.201938	ATMFFA
ion	1.626415	ATMFFA	office	1.868301	ATMFFA	sense	2.199332	
statement	1.622403	ATMFFA	appearance	1.864991	ATMFFA	surface	2.193154	ATMFFA, ATMu

discovery	1.615736	ATMFA	man	1.863109	ATMFA	robert	2.185128	
differential	1.615159		compression	1.845858	ATMFA	expression	2.180273	ATMFA, ATM
sense	1.600477		symbol	1.834537	ATMFA	angle	2.164515	
gap	1.598580	ATMFA	piece	1.823973		option	2.157274	ATMFA, ATM
segment	1.580185	ATMFA, ATM	house	1.816749	ATMFA	bill	2.147097	ATMFA
range	0.288589		keyboard	0.350738	ATMFA	polygon	0.400052	ATMFA
purpose	0.286876	ATMFA	spin	0.350316		organization	0.398878	ATMFA
capability	0.284362	ATMFA	architecture	0.338673	ATMFA	processing	0.395477	ATMFA
april	0.283379		quantum	0.324911		project	0.379329	ATMFA
phone	0.282070	ATMFA	instruction	0.322033	ATMFA	battery	0.373586	ATMFA
code	0.281997	ATMFA	time	0.308453	ATMFA	geometry	0.360184	ATMFA
october	0.279610		testing	0.305869	ATMFA	keyboard	0.350738	ATMFA
march	0.279493	ATMFA	decrease	0.297651		architecture	0.338673	ATMFA
book	0.277109	ATMFA	case	0.296671	ATMFA	quantum	0.324911	ATMFA
september	0.272048		test	0.292459		instruction	0.322033	ATMFA
publication	0.266428	ATMFA	electron	0.288590	ATMFA	time	0.308453	ATMFA
june	0.264172		photon	0.284026		test	0.292459	
group	0.243118	ATMFA, ATM	phone	0.282070	ATMFA	electron	0.288590	ATMFA
school	0.238366	ATMFA, ATM	code	0.281997	ATMFA, ATM	photon	0.284026	
term	0.233936	ATMFA, ATM, ATMC	term	0.233936	ATMFA, ATM, ATMC	phone	0.282070	ATMFA, ATM
article	0.226150	ATMFA, ATM	conference	0.222977	ATMFA, ATM, ATMC	code	0.281997	ATMFA, ATM
conference	0.222977	ATMFA, ATM, ATMC	computer	0.216365	ATMFA, ATM, ATMC	computer	0.216365	ATMFA, ATM, ATMC
computer	0.216365	ATMFA, ATM, ATMC	student	0.212899	ATMFA	student	0.212899	ATMFA
student	0.212899	ATMFA	variety	0.201474	ATMFA, ATM	variety	0.201474	ATMFA, ATM
variety	0.201474	ATMFA, ATM	century	0.179093	ATMFA, ATM, ATMC	century	0.179093	ATMFA, ATM, ATMC
<i>Total Found</i> 31			<i>Total Found</i> 32			<i>Total Found</i> 31		

Total 31 terms of the M1, 32 terms of M2 and 31 terms of M3 were traced in the output of all six code files. Some of the terms existed in more than one file output.

Output of the six files was analyzed further and it was observed that three files were not significant for the above table terms as its output score for a certain term also existed in other file's result. So, the most significant files with respect to the above table terms are three files, that are `ambiguity_tests_merge` (ATM), `ambiguity_tests_merge_for_AMT` (ATMFA), and `ambiguity_tests_pairs` (ATP). These files have all the available outputs of the above table terms.

4.4 Cross-Domain Ambiguity Ranking Using Word2Vec and FastText Models on New Dataset

The previous approach used Wikipedia articles of 2018 (as the dataset) for all the domains. It is obvious that these articles are kept being modified by the experts of relevant domain. Also, only the ready language models were available in the literature code, and the dataset (Wikipedia articles) of that time was not available. Therefore, before the application of FastText model, the dataset (Wikipedia articles) of the relevant domains were again crawled using the same parameters as mentioned in the section 3.2.1. Detail of the newly crawled dataset is given in Table 10.

After obtaining the articles from Wikipedia, further procedure was performed as given in chapter 3, i.e., the next step was to perform pre-processing and build language models to train Word2Vec and FastText on the given text corpora. So, by following the proposed approach, language models were built on new dataset, and then we applied the algorithms on new language models to select dominant shared terms rank it as per its ambiguity level. We used the same six code files as with the old models for the sake of obtaining same scenario's output as defined. Different Output of the code files were generated by using the same parameters and stored in the .csv files for further analysis and comparison.

4.5 Ranking of Terms for Cross-Domain Ambiguity on New Dataset Using Word2Vec and FastText Model

The Word2Vec model is applied on re-crawled Wikipedia article's text. On the same dataset the FastText was also applied and language models from both models were generated. For the application of algorithms, selection of cross domain dominant shared terms and its ranking,

the six files (mentioned in Table 12) were executed with the same parameters and with a little due modification. The results are stored in .csv files for further analysis. Each term mentioned in the Table 13, Table 14, and Table 15 was searched in the generated results and the ambiguity scores of both models were recorded. It is necessary to mention that these terms were listed in the state of the art as top and bottom 20-terms as per its ambiguity level against each scenario.

Table 16. Ambiguity Scores of the Terms of Table 13 Using New Language Models by Word2Vec and FastText

I1 CS, EEN	WV score	FT score	Difference	I2 CS, MEN	WV score	FT score	Difference
news	1.4786	1.5707	0.0921	hull			
formula	1.3131	1.2184	-0.0947	house	1.0279	1.4661	0.4382
relation	1.3109	0.9607	-0.3502	argument			
surface	1.1939	1.5910	0.3970	bar	1.2062	0.8578	-0.3484
motor	0.5379	1.0334	0.4955	option	0.7359	1.0474	0.3115
flash				room	1.0105	0.8660	-0.1444
studio	1.3122	1.5154	0.2033	disk			
contact	1.0793	0.9794	-0.0999	expression			
interpretation				interpretation			
bell	0.8897	0.9050	0.0152	reduction	0.7359	0.6805	-0.0555
reduction	0.6601	0.6059	-0.0542	respect	1.0766	1.3274	0.2508
head	0.5295	1.1513	0.6217	relation	1.1647	1.2043	0.0396
deal	1.2544	1.3505	0.0961	representation			
link				formula	1.1621	1.0994	-0.0627
ion				institute			
desktop				port			
pair				rest			
profile	0.8120	1.0252	0.2133	statement			
particle	1.3158	1.2053	-0.1105	string			
school				october			
performance	0.1709	0.7012	0.5303	state	0.2382	0.7153	0.4771
term	0.2826	0.7563	0.4738	category	0.4272	0.9496	0.5224
article	0.4300	0.4227	-0.0073	december			
september				period	0.2845	0.6784	0.3939
conference				hour	0.2809	0.4923	0.2114
number	0.2426	0.6241	0.3816	cost			
example	0.3201	0.4080	0.0879	test	0.3783	0.5671	0.1888
computer	0.2492	0.8274	0.5782	space	0.3926	0.6600	0.2674
range	0.2394	0.5316	0.2922	advantage	0.2409	0.4304	0.1895
student				september			

march				day	0.3536	0.7404	0.3868
system	0.3695	0.7396	0.3701	minute	0.2120	0.5648	0.3528
december				time	0.2247	0.8429	0.6182
variety	0.2454	0.4462	0.2008	market	0.4466	0.6027	0.1561
point				range	0.2889	0.4749	0.1860
science				variety	0.2040	0.3807	0.1767
april				term	0.2825	0.6598	0.3773
october				year	0.1956	0.8400	0.6443
june				example	0.2534	0.3855	0.1322

Table 16 lists the terms of Table 13 with ambiguity scores generated by new Word2Vec and FastText models against each term. The difference of both model's scores are calculated and listed with a view to find which term is marked more ambiguous as per the FastText score. For this, the Word2Vec ambiguity score was subtracted from the FastText score. The positive value in the difference column indicates that the term is marked more ambiguous by FastText, likewise the negative value shows that the term is marked less ambiguous by FastText model.

Table 17. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 16

I1 Light Controller (CS, EEN)			I2 Mechanical CAD (CS, MEN)		
Model	Terms Score	Total Terms	Model	Terms Score	Total Terms
FastText	Positive	16	FastText	Positive	20
	Negative	6		Negative	4
	Not Found	17		Not Found	15
Total		39	Total		39

In the 39 listed terms of I1 scenario Light Controller, 16 terms marked positive which means these terms are considered more ambiguous, 6 terms values are negative indicates that these terms are computed as less ambiguous by FastText word-vector representation, in comparison with Word2Vec model (as shown in Table 17). The 17 terms were not found in the new outputs. Similarly, in the I2 scenario Mechanical CAD, total 20 terms calculated positive, 4 were negative, and 15 terms were not found. Comparison of both the models is given in Figure 16.

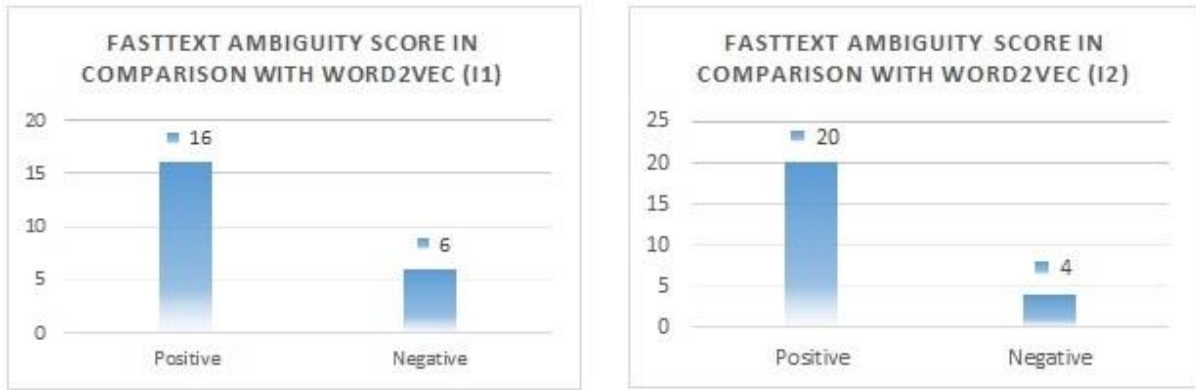


Figure 16. Ambiguity Score of FastText in Comparison to Word2Vec (for I1 & I2)

Table 18 includes terms from Table 14 along with the ambiguity scores determined for each term using new models. To determine which term is identified as more ambiguous according to the FastText, the difference between the scores of the models is calculated and listed. To account for this, the FastText score was deducted from the Word2Vec ambiguity score. The difference column's positive value indicates that the term has been identified as being more ambiguous by the FastText model, while the negative value indicates the term is classified as being less ambiguous in comparison with Word2Vec.

Table 18. Ambiguity Scores of the Terms of Table 14 Using New Language Models by Word2Vec and FastText

I3 CS, MED	WV score	FT score	Difference	I4 CS, SPO	WV score	FT score	Difference
mouse				michael			
matrix				protein			
argument	1.1495	1.6139	0.4644	statement	1.6386	1.6291	-0.0094
client				reduction			
pair				loop			
editor	1.1431	0.8626	-0.2805	string			
arm				founder			
strength	1.1629	1.4413	0.2784	formula	1.5427	1.5903	0.0476
house	1.2062	1.3571	0.1509	washington			
relation	1.0317	0.7766	-0.2551	effect	1.2423	1.4687	0.2264
formula	1.2051	1.3617	0.1567	edge	1.3502	1.4599	0.1098
layer				mechanism			
loop				layer			
symbol				corner	1.0777	1.1400	0.0623
reduction	0.4981	0.6410	0.1430	threat	1.5018	1.3462	-0.1556
room	0.7852	0.9174	0.1322	driver	1.3229	1.2959	-0.0270

statement	1.3254	1.3452	0.0198	fire			
expression	1.2320	1.2160	-0.0160	surface	1.1264	1.6873	0.5609
surface	1.2366	1.5338	0.2972	wave	1.3199	1.5637	0.2438
report	0.3539	0.8266	0.4726	category	0.4274	1.0885	0.6612
concern	0.3644	1.2511	0.8866	level	0.4349	1.0773	0.6424
publication	0.3185	0.8023	0.4837	sale	0.4648	0.5826	0.1178
article	0.3595	0.4867	0.1272	book	0.6974	0.7708	0.0734
issue	0.4573	1.3039	0.8466	term	0.4349	0.8855	0.4507
history	0.4008	0.9536	0.5529	company	0.4525	0.7291	0.2765
student	0.3065	0.5812	0.2747	market			
award	0.2247	0.6297	0.4050	april			
time	0.2884	0.9699	0.6815	history	0.4759	1.1165	0.6406
december				child	0.3406	0.5253	0.1847
april				june			
october				september			
march				student	0.6424	0.8079	0.1655
september				time	0.3427	1.2673	0.9245
june				award	0.5621	0.7553	0.1933
category	0.2768	0.4056	0.1288	article	0.4979	0.6069	0.1090
term	0.2132	0.8121	0.5989	march			
variety	0.1788	0.8812	0.7024	october			
range	0.2339	0.5453	0.3113	december			
year	0.3863	0.9448	0.5585	range	0.6968	1.0417	0.3450

Table 19. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 18

I3 Medical Software (CS, MED)			I4 Athletes Network (CS, SPO)		
Model	Terms Score	Total Terms	Model	Terms Score	Total Terms
FastText	Positive	22	FastText	Positive	19
	Negative	3		Negative	3
	Not Found	14		Not Found	17
Total		39	Total		39

The 39 terms listed in the I3 scenario Medical Software, in which 22 terms have positive values, indicate that they are more ambiguous, and 3 terms have negative markings, pointing that FastText as opposed to Word2Vec model, computes these terms as less ambiguous (given in Table 19). The new outputs did not contain any of the 14 terms. Likewise, results were obtained in the I4 scenario Athletes Network, where a total of 19

terms were calculated as positive, 3 as negative, and 17 as not found. The two models are compared as shown in Figure 17.

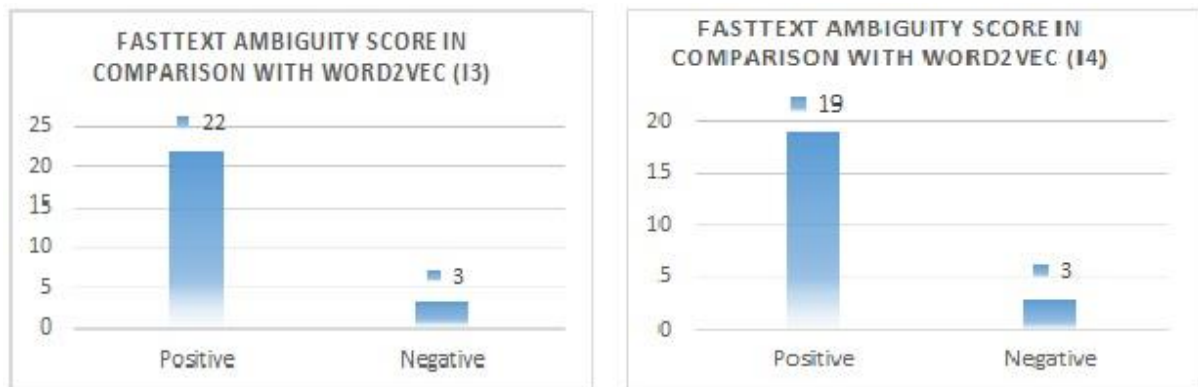


Figure 17. Ambiguity Score of FastText in Comparison to Word2Vec (for I3 & I4)

Table 22 contain the terms from Table 15, with the newly calculated ambiguity scores for each term using the new models. The difference column shows which term is ambiguous by the FastText. The Word2Vec ambiguity score is subtracted from the FastText for the evaluation. The term is considered as being more ambiguous by the FastText model if the difference column has positive value, while negative values indicate that the term has been determined to be less ambiguous in comparison to Word2Vec.

Table 20. Ambiguity Scores of the Terms of Table 15 (M1 & M2) Using New Language Models by Word2Vec and FastText

M1 CS, EEN, MED	WV score	FT score	Difference	M2 CS, EEN, MEN, MED	WV score	FT score	Difference
argument	1.1495	1.6139	0.4644	argument	1.1495	1.6139	0.4644
relation	1.7698	1.2323	-0.5375	respect	2.1368	2.2208	0.0840
formula	1.7813	1.7706	-0.0107	expression	2.2029	1.5736	-0.6293
interpretation				consequence	1.4678	1.3797	-0.0882
consequence	0.5591	1.1151	0.5560	statement	1.3254	1.3452	0.0198
expression	1.2320	1.2160	-0.0160	ion			
arm				father			
surface	1.8182	2.1538	0.3356	institution	0.5461	1.0328	0.4867
house	1.2062	1.8194	0.6133	relation	2.0889	1.8586	-0.2302
client	1.4509	1.2633	-0.1877	formula	1.9784	1.9287	-0.0498
strength	1.4320	1.6707	0.2387	interpretation			
mouse				career	0.9498	0.8639	-0.0859

appearance	1.0677	0.9733	-0.0944	option	1.5689	1.6228	0.0538
ion				office	1.7145	2.0938	0.3792
statement	1.3254	1.3452	0.0198	appearance	1.8794	1.3751	-0.5044
discovery	0.8578	0.8638	0.0059	man	2.0139	2.0425	0.0287
differential				compression	1.4077	0.7231	-0.6846
sense	1.4915	1.4170	-0.0746	symbol			
gap	1.4768	1.6187	0.1418	piece	1.8707	1.7651	-0.1056
segment	1.4565	1.3899	-0.0666	house	1.6825	1.8321	0.1496
range	0.3460	0.9463	0.6003	keyboard			
purpose	0.4379	1.0515	0.6136	spin			
capability	0.2946	0.4188	0.1242	architecture			
april				quantum	0.7796	1.0898	0.3102
phone	0.2424	0.4190	0.1766	instruction	1.3436	1.1762	-0.1674
code	0.9078	1.1172	0.2094	time	0.4403	1.4424	1.0021
october				testing	0.4486	0.7526	0.3040
march				decrease			
book	0.4022	0.6902	0.2880	case	0.7083	0.8238	0.1155
september				test	0.4486	0.7526	0.3040
publication	0.3185	0.8023	0.4837	electron	1.1818	1.5797	0.3978
june				photon			
group	0.5350	0.8317	0.2967	phone	0.2424	0.4190	0.1766
school	0.2961	0.6151	0.3189	code	1.0073	1.5125	0.5051
term	0.4991	0.7563	0.2573	term	0.5757	0.9316	0.3559
article	0.4300	0.4227	-0.0073	conference	0.4705	0.8391	0.3686
conference	0.4705	0.8391	0.3686	computer	0.7425	0.8510	0.1085
computer	0.2492	0.8274	0.5782	student	0.3065	0.5812	0.2747
student	0.3065	0.5812	0.2747	variety	0.3354	0.6292	0.2937
variety	0.2454	0.4462	0.2008	century	0.1945	0.4912	0.2967

Table 21. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 20

M1 Medical Device (CS, EEN, MED)			M2 Medical Robot (CS, EEN, MEN, MED)		
Model	Terms Score	Total Terms	Model	Terms Score	Total Terms
FastText	Positive	22	FastText	Positive	22
	Negative	8		Negative	9
	Not Found	10		Not Found	9
Total		40	Total		40

Table 22. Ambiguity Scores of the Terms of Table 15 (M3) Using New Language Models by Word2Vec and FastText

M3 CS, EEN, MEN, MED, SPO	WV score	FT score	Difference
consequence	1.4678	1.3797	-0.0882
respect	2.5993	2.5128	-0.0864
statement	1.9414	1.8920	-0.0495
michael			
story	1.9306	2.5789	0.6483
argument	1.1495	1.6139	0.4644
brother	1.8379	1.2399	-0.5980
founder			
end			
ray			
relation	2.0889	1.8586	-0.2302
stability	1.7865	0.9160	-0.8706
institution	0.5461	1.0328	0.4867
sense	1.8001	1.6886	-0.1114
surface	1.1264	1.6873	0.5609
robert			
expression	2.2029	1.5736	-0.6293
angle	1.6802	1.4746	-0.2055
option	1.5689	1.6228	0.0538
bill	2.0789	2.2631	0.1843
polygon			
organization	0.6251	1.0004	0.3753
processing	0.4417	1.0235	0.5818
project	0.3966	0.6587	0.2620
battery	0.5983	0.7987	0.2004
geometry	1.0715	0.7578	-0.3137
keyboard			
architecture			
quantum	0.7796	1.0898	0.3102
instruction	1.3436	1.1762	-0.1674
time	0.5354	1.8147	1.2793
test	0.8820	1.3883	0.5063
electron	1.1818	1.5797	0.3978
photon			
phone	0.2424	0.4190	0.1766
code	1.4340	1.9659	0.5319
computer	0.7425	0.8510	0.1085
student	0.6424	0.8079	0.1655

variety	0.9956	1.1567	0.1611
century	0.2640	0.4912	0.2272

Table 23. Comparison of FastText with Word2Vec Term's Ambiguity Score in Table 22

M3 Sports Rehab Machine (CS, EEN, MEN, MED, SPO)		
Model	Terms Score	Total Terms
FastText	Positive	20
	Negative	11
	Not Found	9
	Total	40

The 40 terms stated in the M1 scenario Medical Device, 8 terms have negative value, indicating that FastText rather than the Word2Vec model computes these terms as less ambiguous, while 22 terms have positive values, indicating that they are more ambiguous as per FastText Model. None of the 10 terms were present in the revised outputs.



Figure 18. Ambiguity Score of FastText in Comparison to Word2Vec (for M1, M2 & M3)

The M2 scenario Medical Robot produced results such that, with a total of 22 terms calculated as positive, 9 as negative, and 9 as not found (shown in Table 21). Moreover, M3 scenario Sport Rehab Machine given the result as, difference of 20 terms found as positive,

11 was negative and 9 was not found in the total of 40 terms (given in Table 23). In Figure 18, the two models are contrasted.

4.6 Comparison of Word2Vec and FastText Combined Results on New Dataset

Different files were used to produce the ambiguous words in the different scenarios. Detail of these files' usage are given in Table 12. Output of these files is combined in such a way that common terms from both files were searched and ambiguity score of both Word2Vec and FastText models are written in a third file. For example, output of file ambiguity_tests_merge (ATM) are total 2878 terms. Common terms of these two files (ATM output for Word2Vec and for FastText) are 2806 in the total 2878 ranked terms. In these common terms, after the comparison of all terms, the ambiguity score of 2399 terms is high in the FastText ATM result and Word2Vec score high for 407 terms. Similar comparison for each scenario in the result of the given files are given in Table 24 and Figure 19, Figure 20, Figure 21, Figure 22, Figure 23, and Figure 24.

As the dominant-shared terms for each scenario were so large in quantity against each model, and the manual analysis might be time consuming and error prone, therefore we built *Excel Macros* for finding, comparing, listing, and organizing the terms. Also, these macros were used to extract the uncommon terms of both models in each pair of domains.

Table 24. FastText Word2Vec Comparison of Ambiguity Score

File	Common Terms	Positive Ambiguity Score		Terms Not Found in FT	Total
		FastText	Word2Vec		
ATM	2806	2399	407	72	2878
ATIC	385	362	23	15	400
ATMu	371	363	8	29	400
ATMC	755	728	27	44	799
ATMFA	4124	3384	740	38	4162
ATP	942	898	44	58	1000

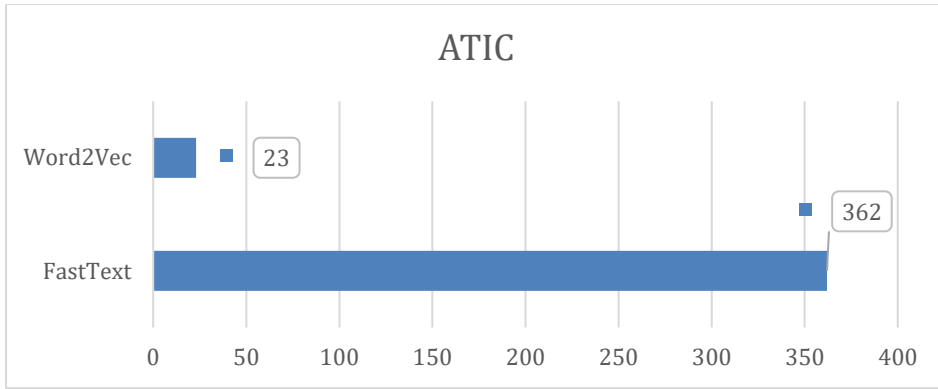


Figure 19. Dominant-Shared Terms Ambiguity Scores Comparison (1)

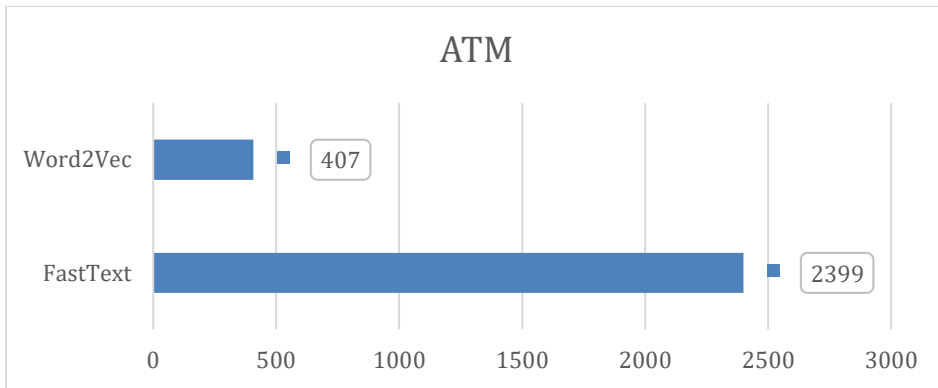


Figure 20. Dominant-Shared Terms Ambiguity Scores Comparison (2)

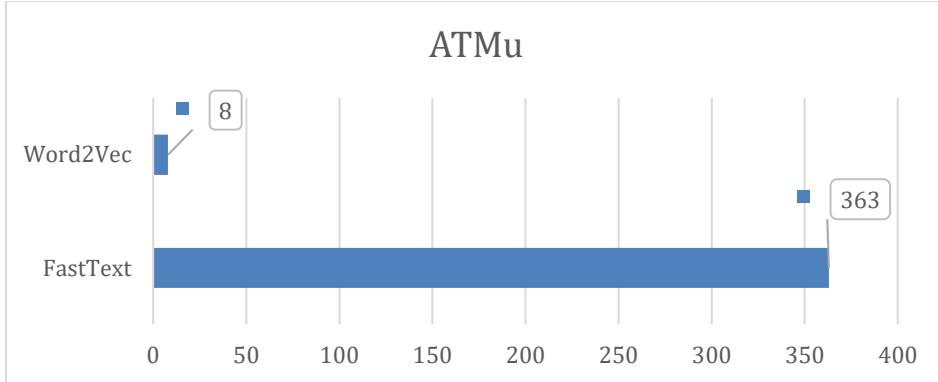


Figure 21. Dominant-Shared Terms Ambiguity Scores Comparison (3)

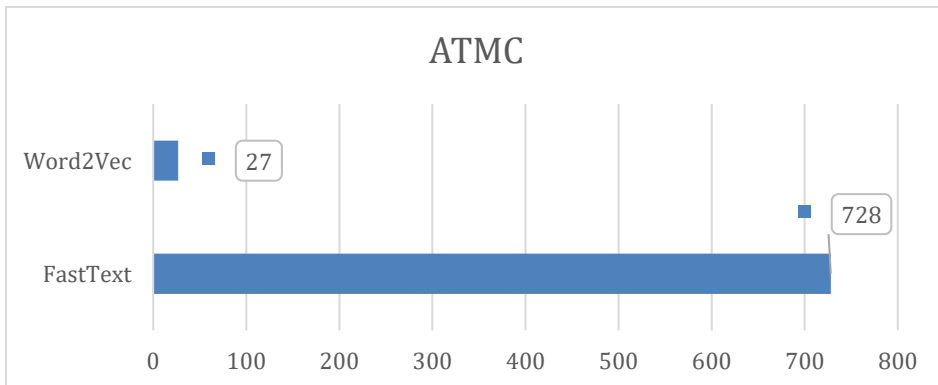


Figure 22. Dominant-Shared Terms Ambiguity Scores Comparison (4)

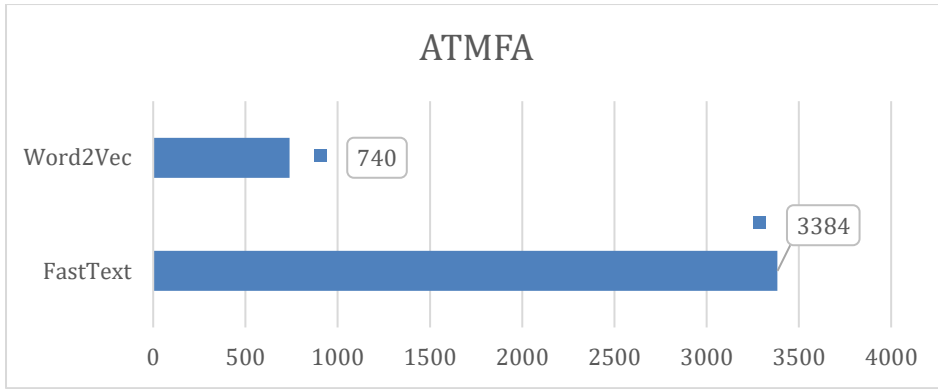


Figure 23. Dominant-Shared Terms Ambiguity Scores Comparison (5)

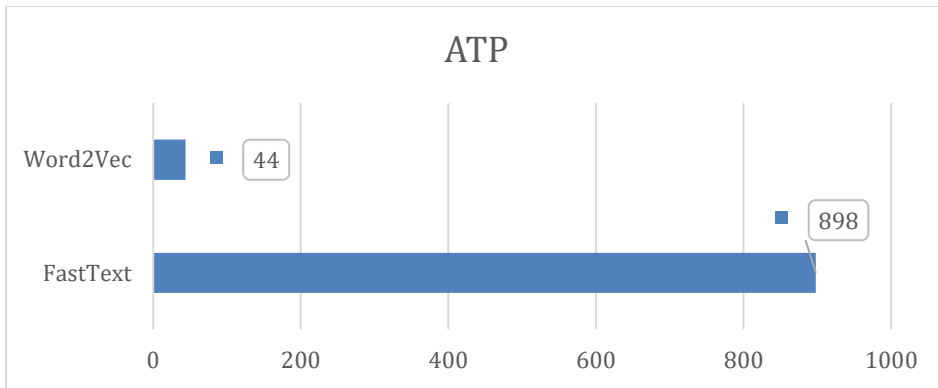


Figure 24. Dominant-Shared Terms Ambiguity Scores Comparison (6)

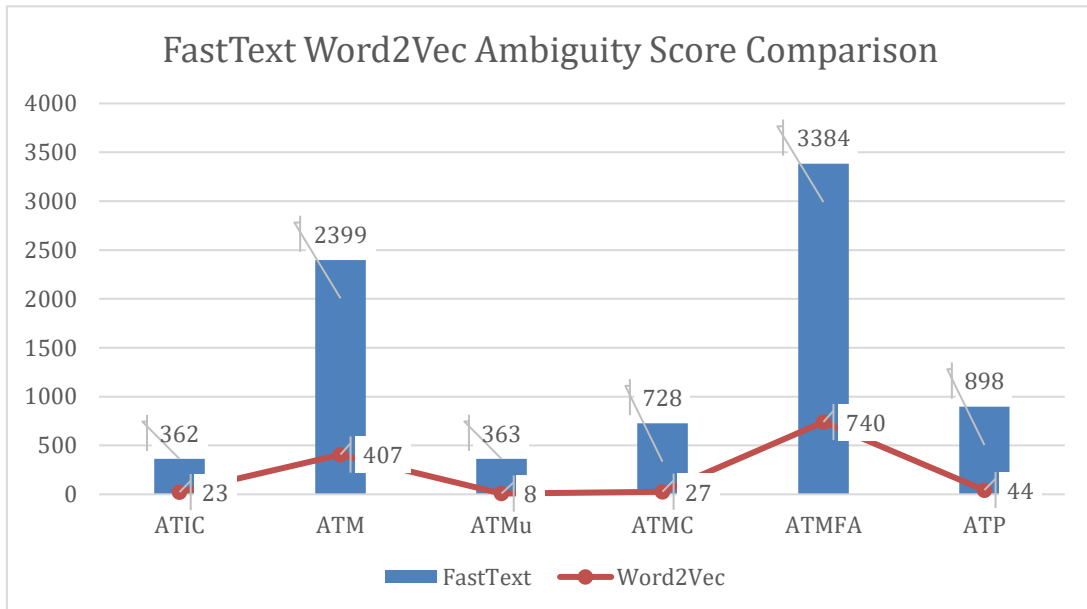


Figure 25. FastText Word2Vec Ambiguity Score Comparison

Figure 25 shows visual representation of Table 24 in which the clustered columns represents the high FastText output, and the horizontal brown line represent high values of Word2Vec model.

4.7 Selection of Dominant Shared Terms by FastText

As in the literature, from the result of Word2Vec dominant-shared terms the top 20 and bottom 20 terms were listed [1], expressing the 20 most and 20 least ambiguous terms. In the same way from the result of FastText dominant-shared terms, the 20 most ambiguous and 20 least scored terms are listed against each scenario and given in the Table 25 Table 26 and Table 27 below:

Table 25. Dominant-Shared Terms List by FastText for Light Controller and Mechanical CAD Scenarios

Sr.	Light controller I1 (CS, EEN)		Sr.	Mechanical CAD I2 (CS, MEN)	
	Term	Score		Term	Score
1.	surface	1.59096	1.	machinery	1.646123
2.	news	1.570741	2.	park	1.615587
3.	man	1.552746	3.	bell	1.474922
4.	studio	1.515439	4.	appliance	1.471294
5.	potential	1.452937	5.	house	1.466081
6.	contrast	1.440053	6.	gradient	1.459927
7.	aspect	1.428504	7.	potential	1.450888
8.	ground	1.423502	8.	calculator	1.445849
9.	air	1.360362	9.	class	1.439836
10.	channel	1.358456	10.	stock	1.433542
11.	deal	1.350508	11.	field	1.418269
12.	water	1.344624	12.	piece	1.41196
13.	game	1.336707	13.	ground	1.388091
14.	energy	1.3231	14.	clock	1.362174
15.	coverage	1.316237	15.	parallel	1.360973
16.	cell	1.313304	16.	action	1.353842
17.	field	1.299738	17.	type	1.339001
18.	piece	1.293262	18.	respect	1.327407
19.	gap	1.253745	19.	hybrid	1.323835
20.	carrier	1.245965	20.	mark	1.321176
21.	amplitude	0.467667	21.	experiment	0.539596
22.	amd	0.466124	22.	measure	0.536097
23.	sale	0.465218	23.	measurement	0.530514
24.	addition	0.464722	24.	location	0.526131
25.	color	0.459062	25.	manufacturer	0.524683
26.	mobile	0.458707	26.	flight	0.523489
27.	test	0.458335	27.	reference	0.522834

28.	total	0.446947	28.	academy	0.511415
29.	telephone	0.420993	29.	hour	0.492302
30.	phone	0.418956	30.	degree	0.488334
31.	advantage	0.41661	31.	order	0.481053
32.	today	0.394948	32.	range	0.474885
33.	sin	0.386986	33.	calculation	0.467964
34.	president	0.368968	34.	case	0.461196
35.	price	0.364017	35.	college	0.44772
36.	increase	0.350908	36.	advantage	0.430415
37.	clock	0.343635	37.	increase	0.429334
38.	combination	0.337399	38.	master	0.407038
39.	modulation	0.327778	39.	north	0.391403
40.	court	0.255111	40.	variety	0.380674

Table 26. Dominant-Shared Terms List by FastText for Medical Software and Athletes Network Scenarios

Sr.	Medical software I3 (CS, MED)		Sr.	Athletes network I4 (CS, SPO)	
	Term	Score		Term	Score
1.	fiber	1.640556	1.	boot	1.695729
2.	motor	1.606068	2.	surface	1.687301
3.	type	1.599276	3.	movement	1.598654
4.	pulse	1.585226	4.	agency	1.581641
5.	library	1.560013	5.	art	1.571298
6.	resource	1.548058	6.	wave	1.563652
7.	idea	1.546585	7.	card	1.562775
8.	failure	1.535262	8.	material	1.53141
9.	surface	1.53385	9.	activity	1.521687
10.	program	1.531147	10.	course	1.497328
11.	property	1.52729	11.	spin	1.495257
12.	alternative	1.521999	12.	route	1.492943
13.	host	1.519336	13.	bill	1.491617
14.	air	1.510406	14.	goal	1.491474
15.	case	1.509322	15.	equipment	1.479725
16.	friend	1.508765	16.	relay	1.472861
17.	name	1.483718	17.	family	1.456204
18.	mission	1.470916	18.	piece	1.44645
19.	scientist	1.461924	19.	variation	1.43019
20.	availability	1.461715	20.	story	1.428106
21.	protection	0.607006	21.	hour	0.889138

22.	chair	0.602771	22.	military	0.841241
23.	job	0.600881	23.	television	0.831664
24.	measurement	0.600296	24.	partner	0.811091
25.	money	0.583641	25.	participant	0.789333
26.	student	0.581199	26.	success	0.772783
27.	experiment	0.552848	27.	music	0.762495
28.	range	0.545283	28.	school	0.760408
29.	travel	0.521072	29.	report	0.75292
30.	increase	0.507215	30.	position	0.748885
31.	sale	0.494142	31.	member	0.716138
32.	decade	0.488249	32.	direction	0.712265
33.	article	0.486702	33.	protection	0.707374
34.	biology	0.473625	34.	total	0.702056
35.	reference	0.465292	35.	president	0.669979
36.	north	0.435227	36.	brand	0.658793
37.	executive	0.422421	37.	regulation	0.644789
38.	director	0.418565	38.	chief	0.619433
39.	category	0.405581	39.	list	0.559163
40.	price	0.347632	40.	child	0.525307

Table 27. Dominant-Shared Terms List by FastText for Medical Device, Medical Robot and Sport Rehab Machine Scenarios

Sr.	Medical device M1 (CS, EEN, MED)		Sr.	Medical robot M2 (CS, EEN, MEN, MED)		Sr.	Sport rehab machine M3 (CS, EEN, MEN, MED, SPO)	
	Term	Score		Term	Score		Term	Score
1.	motor	2.054434	1.	park	2.419028	1.	ability	2.610864
2.	type	1.996133	2.	type	2.268756	2.	story	2.578883
3.	air	1.969982	3.	respect	2.220806	3.	respect	2.51284
4.	phenomenon	1.919298	4.	surface	2.153767	4.	type	2.454103
5.	process	1.89895	5.	device	2.150813	5.	future	2.425817
6.	particle	1.89878	6.	property	2.14334	6.	park	2.419028
7.	material	1.896871	7.	solution	2.134443	7.	order	2.415292
8.	gamma	1.891737	8.	process	2.12899	8.	thing	2.373934
9.	liquid	1.884634	9.	barrier	2.115433	9.	film	2.335974
10.	solution	1.867904	10.	thing	2.105908	10.	motor	2.334503
11.	board	1.855416	11.	office	2.093762	11.	spot	2.313416
12.	film	1.847767	12.	story	2.091625	12.	formula	2.293928
13.	property	1.835732	13.	magazine	2.090205	13.	peak	2.281734
14.	pattern	1.834297	14.	team	2.070382	14.	magazine	2.281644
15.	respect	1.824345	15.	approach	2.068058	15.	stock	2.264412
16.	house	1.819449	16.	glass	2.061337	16.	bill	2.263122
17.	beam	1.811696	17.	film	2.059012	17.	deal	2.255644
18.	mark	1.806572	18.	motor	2.054434	18.	field	2.242056
19.	glass	1.798863	19.	atom	2.052603	19.	spin	2.239163

20.	barrier	1.795809	20.	ability	2.043992	20.	sheet	2.23529
21.	executive	0.422421	21.	usb	0.48659	21.	actuator	0.50265
22.	telephone	0.420993	22.	omega	0.482839	22.	century	0.491201
23.	phone	0.418956	23.	biology	0.473625	23.	usb	0.48659
24.	capability	0.418835	24.	prediction	0.46769	24.	omega	0.482839
25.	laptop	0.417104	25.	amd	0.466124	25.	biology	0.473625
26.	category	0.405581	26.	mpeg	0.459843	26.	prediction	0.46769
27.	chief	0.404338	27.	color	0.459062	27.	amd	0.466124
28.	today	0.394948	28.	mobile	0.458707	28.	mpeg	0.459843
29.	compatibility	0.39377	29.	fi	0.448872	29.	color	0.459062
30.	director	0.391759	30.	pc	0.435289	30.	mobile	0.458707
31.	sin	0.386986	31.	executive	0.422421	31.	fi	0.448872
32.	wireless	0.384006	32.	phone	0.418956	32.	pc	0.435289
33.	president	0.368968	33.	capability	0.418835	33.	phone	0.418956
34.	increase	0.350908	34.	laptop	0.417104	34.	capability	0.418835
35.	clock	0.343635	35.	compatibility	0.39377	35.	laptop	0.417104
36.	combination	0.337399	36.	sin	0.386986	36.	compatibility	0.39377
37.	modulation	0.327778	37.	wireless	0.384006	37.	sin	0.386986
38.	degree	0.32216	38.	increase	0.350908	38.	wireless	0.384006
39.	storage	0.321692	39.	clock	0.343635	39.	clock	0.343635
40.	court	0.255111	40.	modulation	0.327778	40.	modulation	0.327778

4.8 Case Studies of the Effectiveness of FastText Terms

In the dominant-shared terms of FastText, some of the terms have a high ambiguity score but was not listed in the resultant terms of Word2Vec model. Based on various scenarios, we discuss some important cases below:

The term *net* is ranked higher in the domain of CS-Sports domain, which may be interpreted as computer network or internet, while in the Sport domain it may be expressed as fabric, bag, or a mesh for ball. The term *press* in the same domain combination also ranked high for ambiguity level; it can be seen as the news, media related to the sports events in sport domain, and ‘a click’, ‘a press down on a key’ like meaning in the CS domain. The term *port* given in the list against CS-Electronics also has dual interpretation in both domains, such as it can be a physical terminal for an electronic machine to connect with an external physical circuit. While the port can also be seen as the logical port numbers declared in a software and associated with network protocols to allow transfer of data between two systems. The term *park* given as ambiguous term in CS-Sport domain is intended as to adjust read/write head of the hard disk to its default location, or it may be an area name where computers / technology

company exists. On the other hand, the park may be a venue for playing, presentation, or exercises in the sports domain.

The term *star* is ranked high in Electronic-SPO domain can be taken as a key player of a game, or in the electronics domain, it refers to an energy star, with high star, an electronic appliance considers as least efficient and vice versa. Similarly, in the CS domain, the *star* is a star topology for computers connectivity, or referring a pointer variable, or a product symbol. The term *man* in the medical domain may refer to a patient or a doctor, in sports it may be referred to a coach, a sportsman. If we see this term in CS domain, it may be a type of network. The term *boot* ranked high in the domain CS-Mec and CS-Sport, can be interpreted as in sport footwears, while in the CS the term means loading an operating system to primary memory or startup of a system. The same term is used for a piece of pipe in a three-phase separator's bottom, often upstream of the weir, or a steering wheel boot of a vehicle. The term *book* in the sport domain is a bet on the game events, and the study books in all other domains.

A common term *code* in the computer domain is considered as set of instructions that is written in a programming language for a specific task, while in the electronics it is group-of-symbols for the representation of letters or numbers. In all other domains the code may be interpreted as the rules, law, and standards to be followed for the achievement of a specific goal. Moreover, the term *art* is one of the high scored terms, can be read in the medical domain as the treatment of HIV (antiretroviral), while in the other domain it will lead to a default meaning as a skill, or creativity related to something. The term *spot* is considered as a mark, area, or a specific location in general domains, and in the mechanical domain it can also be interpreted as a satellite name that an imaging satellite (*spot*) for observational purpose. The term *parameter* ambiguity score given high in the domain of CS-electronics which can be elaborated in the electronics field as the values that shows the performance of circuits and components, while in the computer domain the same term is considered as in the meaning of argument which passes the values to the methods or procedures in the programming.

These were some of those terms which are only listed in FastText, and we have observed its ambiguity level through examples. There were other important terms listed in the FastText output that were absent from the list of Word2Vec.

4.9 Limitations

Amongst all the dominant-shared terms listed by FastText, usage of terms in its context was not checked manually and the case study of each term couldn't be analyzed for another verification of the effectiveness of the approach. The reason is that the produced terms needed to be compared with the result of Word2Vec model, which was two time generated due to the expected variation in the internet contents.

As the author of study [19] applied Word2Vec for word vector representation also mentioned FastText and GloVe to be applied for the same purpose, we missed the GloVe model application due to the implementation of FastText and detailed comparison of the terms.

We also did not apply multiple n-grams with the FastText for the detailed examination of the effectiveness of n-grams on the given text corpora. As if n-grams change the language models must be re-build for it and the algorithms must be re applied on the models which needed a lot of time for processing.

One major limitation of the FastText usage is that it requires more time to train on data and generate language models as compared to Word2Vec.

CHAPTER 5: CONCLUSION & FUTURE WORK

5.1 Conclusion

In this research, we have proposed a problem to detect cross-domain ambiguity caused by multi-meaning terms of different domains. We performed a detailed literature review related to the cross-domain ambiguity. The dataset was obtained from the Wikipedia pages for the selected domains. We applied NLP approaches for pre-processing on the text corpus and on the resultant text of different domains, we applied Word2Vec and FastText on the data and built language models. The language models were generated separately against each of the domain. Total seven scenarios were supposed to combine the domains for a particular purpose. Cross-domain ambiguity was calculated by the selection of dominant-shared terms in each domain combination and ranking of the set of terms as per its ambiguity level. Ambiguity score of the terms was calculated based on the similar wording they share in its context across the domains. The process was done twice, once using Word2Vec and secondly using FastText language models. Output of both models were stored and compared with the state-of-the-art article results. Ambiguous terms generated by both models were also compared in which the ambiguity score of most of the terms was high in the FastText results, specifically in the scenarios where more than two domains were involved. Furthermore, we observed that some of the most ambiguous terms listed by FastText were not found in the Word2Vec generated results. We also observed that the model training time of Word2Vec was less than FastText. We concluded that the use of FastText by requirements analyst will be more beneficial for the track of cross-domain ambiguity terms in requirements if there is no strict time constraint for the implementation of approach.

5.2 Future Work

As future work, the proposed approach can be extended to implement by increasing the number of n-grams in the implementation of FastText. Similarly GloVe and BERT can also be used to see its effectiveness in the similar methodology.

REFERENCES

- [1] A. Ferrari and A. Esuli, “An NLP approach for cross-domain ambiguity detection in requirements engineering,” *Autom. Softw. Eng.*, vol. 26, no. 3, pp. 559–598, Sep. 2019, doi: 10.1007/s10515-019-00261-7.
- [2] O. Al-Harbi, S. Jusoh, and N. M. Norwawi, “Lexical Disambiguation in Natural Language Questions (NLQs).” arXiv, Sep. 26, 2017. [Online]. Available: <http://arxiv.org/abs/1709.09250>
- [3] F. Zait and N. Zarour, “Addressing Lexical and Semantic Ambiguity in Natural Language Requirements,” in 2018 Fifth International Symposium on Innovation in Information and Communication Technology (ISIICT), Oct. 2018, pp. 1–7. doi: 10.1109/ISIICT.2018.8613726.
- [4] A. Ferrari, G. Lipari, S. Gnesi, and G. O. Spagnolo, “Pragmatic ambiguity detection in natural language requirements,” in *2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, Aug. 2014, pp. 1–8. doi: 10.1109/AIRE.2014.6894849.
- [5] N. Kiyavitskaya, N. Zeni, L. Mich, and D. M. Berry, “Requirements for tools for ambiguity identification and measurement in natural language requirements specifications,” *Requir. Eng.*, vol. 13, no. 3, pp. 207–239, Sep. 2008, doi: 10.1007/s00766-008-0063-7.
- [6] S. F. Tjong and D. M. Berry, “The Design of SREE — A Prototype Potential Ambiguity Finder for Requirements Specifications and Lessons Learned,” in *Requirements Engineering: Foundation for Software Quality*, J. Doerr and A. L. Opdahl, Eds., in Lecture Notes in Computer Science. Berlin, Heidelberg: Springer, 2013, pp. 80–95. doi: 10.1007/978-3-642-37422-7_6.
- [7] M. Bano, “Addressing the challenges of requirements ambiguity: A review of empirical literature,” in *2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE)*, Aug. 2015, pp. 21–24. doi: 10.1109/EmpiRE.2015.7431303.
- [8] A. Ferrari and S. Gnesi, “Using collective intelligence to detect pragmatic ambiguities,” in *2012 20th IEEE International Requirements Engineering Conference (RE)*, Sep. 2012, pp. 191–200. doi: 10.1109/RE.2012.6345803.
- [9] A. Ferrari, B. Donati, and S. Gnesi, “Detecting Domain-Specific Ambiguities: An NLP Approach Based on Wikipedia Crawling and Word Embeddings,” in *2017 IEEE 25th International Requirements Engineering Conference Workshops (REW)*, Lisbon, Portugal: IEEE, Sep. 2017, pp. 393–399. doi: 10.1109/REW.2017.20.

- [10] A. Ferrari, A. Esuli, and S. Gnesi, "Identification of Cross-Domain Ambiguity with Language Models," in *2018 5th International Workshop on Artificial Intelligence for Requirements Engineering (AIRE)*, Banff, AB: IEEE, Aug. 2018, pp. 31–38. doi: 10.1109/AIRE.2018.00011.
- [11] V. Gervasi, A. Ferrari, D. Zowghi, and P. Spoletini, "Ambiguity in Requirements Engineering: Towards a Unifying Framework," in *From Software Engineering to Formal Methods and Tools, and Back*, M. H. ter Beek, A. Fantechi, and L. Semini, Eds., in Lecture Notes in Computer Science, vol. 11865. Cham: Springer International Publishing, 2019, pp. 191–210. doi: 10.1007/978-3-030-30985-5_12.
- [12] B. Kitchenham, "Procedures for Performing Systematic Reviews".
- [13] M. P. S. Bhatia, A. Kumar, and R. Beniwal, "Ontology based framework for detecting ambiguities in software requirements specification," 2016.
- [14] F. Siddiqui and M. A. Alam, "An Ontology Based Approach for Requirement Inconsistency Detection," no. 1, 2011.
- [15] C. Wohlin, "Guidelines for snowballing in systematic literature studies and a replication in software engineering," in *Proceedings of the 18th International Conference on Evaluation and Assessment in Software Engineering*, London England United Kingdom: ACM, May 2014, pp. 1–10. doi: 10.1145/2601248.2601268.
- [16] A. Naeem, Z. Aslam, and M. A. Shah, "Analyzing Quality of Software Requirements; A Comparison Study on NLP Tools," in *2019 25th International Conference on Automation and Computing (ICAC)*, Lancaster, United Kingdom: IEEE, Sep. 2019, pp. 1–6. doi: 10.23919/IConAC.2019.8895182.
- [17] B. Lebeaupin, A. Rauzy, and J.-M. Roussel, "A language proposition for system requirements," in *2017 Annual IEEE International Systems Conference (SysCon)*, Montreal, QC, Canada: IEEE, Apr. 2017, pp. 1–8. doi: 10.1109/SYSCON.2017.7934808.
- [18] J. Kuchta and P. Padhiyar, "Extracting Concepts from the Software Requirements Specification Using Natural Language Processing," in *2018 11th International Conference on Human System Interaction (HSI)*, Gdansk, Poland: IEEE, Jul. 2018, pp. 443–448. doi: 10.1109/HSI.2018.8431221.
- [19] S. Mishra and A. Sharma, "On the Use of Word Embeddings for Identifying Domain Specific Ambiguities in Requirements," in *2019 IEEE 27th International Requirements Engineering Conference Workshops (REW)*, Jeju Island, Korea (South): IEEE, Sep. 2019, pp. 234–240. doi: 10.1109/REW.2019.00048.

- [20] A. Fantechi, A. Ferrari, S. Gnesi, and L. Semini, “Requirement Engineering of Software Product Lines: Extracting Variability Using NLP,” in *2018 IEEE 26th International Requirements Engineering Conference (RE)*, Banff, AB: IEEE, Aug. 2018, pp. 418–423. doi: 10.1109/RE.2018.00053.
- [21] M. Osama, A. Zaki-Ismail, M. Abdelrazek, J. Grundy, and A. Ibrahim, “Score-Based Automatic Detection and Resolution of Syntactic Ambiguity in Natural Language Requirements,” in *2020 IEEE International Conference on Software Maintenance and Evolution (ICSME)*, Adelaide, Australia: IEEE, Sep. 2020, pp. 651–661. doi: 10.1109/ICSME46990.2020.00067.
- [22] S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh, and L. C. Briand, “Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements,” in *2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, Madrid, ES: IEEE, May 2021, pp. 1485–1497. doi: 10.1109/ICSE43902.2021.00133.
- [23] A. Fantechi, S. Gnesi, S. Livi, and L. Semini, “A spaCy-based tool for extracting variability from NL requirements,” in *Proceedings of the 25th ACM International Systems and Software Product Line Conference - Volume B*, Leicester United Kindom: ACM, Sep. 2021, pp. 32–35. doi: 10.1145/3461002.3473074.
- [24] A. Fantechi, S. Gnesi, and L. Semini, “Applying the QuARS Tool to Detect Variability,” in *Proceedings of the 23rd International Systems and Software Product Line Conference - Volume B*, Paris France: ACM, Sep. 2019, pp. 29–32. doi: 10.1145/3307630.3342388.
- [25] A. Ferrari *et al.*, “Detecting requirements defects with NLP patterns: an industrial experience in the railway domain,” *Empir. Softw. Eng.*, vol. 23, no. 6, pp. 3684–3733, Dec. 2018, doi: 10.1007/s10664-018-9596-7.
- [26] A. Ferrari, G. O. Spagnolo, A. Fiscella, and G. Parente, “QuOD: An NLP Tool to Improve the Quality of Business Process Descriptions,” in *From Software Engineering to Formal Methods and Tools, and Back*, M. H. ter Beek, A. Fantechi, and L. Semini, Eds., in Lecture Notes in Computer Science, vol. 11865. Cham: Springer International Publishing, 2019, pp. 267–281. doi: 10.1007/978-3-030-30985-5_17.
- [27] B. Rosadini *et al.*, “Using NLP to Detect Requirements Defects: An Industrial Experience in the Railway Domain,” in *Requirements Engineering: Foundation for Software Quality*, P. Grünbacher and A. Perini, Eds., in Lecture Notes in Computer Science, vol. 10153. Cham: Springer International Publishing, 2017, pp. 344–360. doi: 10.1007/978-3-319-54045-0_24.

- [28] F. Ashfaq and I. S. Bajwa, “Natural language ambiguity resolution by intelligent semantic annotation of software requirements,” *Autom. Softw. Eng.*, vol. 28, no. 2, p. 13, Nov. 2021, doi: 10.1007/s10515-021-00291-0.
- [29] M. Shepperd, F. Brito e Abreu, A. Rodrigues da Silva, and R. Pérez-Castillo, Eds., *Quality of Information and Communications Technology: 13th International Conference, QUATIC 2020, Faro, Portugal, September 9–11, 2020, Proceedings*, vol. 1266. in *Communications in Computer and Information Science*, vol. 1266. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-58793-2.
- [30] M. H. ter Beek, A. Fantechi, and L. Semini, Eds., *From Software Engineering to Formal Methods and Tools, and Back: Essays Dedicated to Stefania Gnesi on the Occasion of Her 65th Birthday*, vol. 11865. in *Lecture Notes in Computer Science*, vol. 11865. Cham: Springer International Publishing, 2019. doi: 10.1007/978-3-030-30985-5.
- [31] S. Jarzabek, A. Poniszewska-Marańda, and L. Madeyski, Eds., *Integrating Research and Practice in Software Engineering*, vol. 851. in *Studies in Computational Intelligence*, vol. 851. Cham: Springer International Publishing, 2020. doi: 10.1007/978-3-030-26574-8.
- [32] A. Mustafa, W. M. W. Kadir, and N. Ibrahim, “Automated Natural Language Requirements Analysis using General Architecture for Text Engineering (GATE) Framework,” *Journal of Telecommunication, Electronic and Computer Engineering (JTEC)**JTEC*, vol. 9, no. 3–4, pp. 97–101, 2017, [Online]. Available: <https://jtec.utem.edu.my/jtec/article/view/2925>
- [33] V. Jain, R. Malhotra, S. Jain, and N. Tanwar, “Cross-Domain Ambiguity Detection using Linear Transformation of Word Embedding Spaces.” arXiv, Mar. 29, 2020. Accessed: Apr. 11, 2023. [Online]. Available: <http://arxiv.org/abs/1910.12956>
- [34] S. Çevikol and F. B. Aydemir, “Detecting Inconsistencies of Natural Language Requirements in Satellite Ground Segment Domain,” *REFSQ Workshops*, 2019, [Online]. Available: https://ceur-ws.org/Vol-2376/NLP4RE19_paper15.pdf
- [35] A. Chattopadhyay, N. Niu, Z. Peng, and J. Zhang, “Semantic Frames for Classifying Temporal Requirements: An Exploratory Study,” *REFSQ Workshops*, pp. 1–9, 2021, [Online]. Available: <https://homepages.uc.edu/~niunn/papers/NLP4RE21.pdf>
- [36] M. Arrabito, A. Fantechi, S. Gnesi, and L. Semini, “A comparison of NLP Tools for RE to extract Variation Points,” *REFSQ Workshops*, 2020, [Online]. Available: <https://ceur-ws.org/Vol-2584/NLP4RE-paper1.pdf>

- [37] T. Baldwin, Y. Li, B. Alexe, and I. R. Stanoi, "Automatic Term Ambiguity Detection," *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pp. 804–809, Aug. 2013.
- [38] D. M. Blei, Andrew Y. Ng, and Michael I. Jordan, "Latent Dirichlet Allocation," *Journal of Machine Learning Research* 3 (2003) 993-1022, pp. 993–1022, Jan. 2003.
- [39] C. Arora, M. Sabetzadeh, L. Briand, and F. Zimmer, "Automated Extraction and Clustering of Requirements Glossary Terms," *IEEE Trans. Softw. Eng.*, vol. 43, no. 10, pp. 918–945, Oct. 2017, doi: 10.1109/TSE.2016.2635134.
- [40] Z. S. Harris, "Distributional Structure," *WORD*, vol. 10, no. 2–3, pp. 146–162, Aug. 1954, doi: 10.1080/00437956.1954.11659520.
- [41] R. Collobert and J. Weston, "A Unified Architecture for Natural Language Processing: Deep Neural Networks with Multitask Learning".
- [42] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed Representations of Words and Phrases and their Compositionality".
- [43] P. D. Turney and P. Pantel, "From Frequency to Meaning: Vector Space Models of Semantics," *J. Artif. Intell. Res.*, vol. 37, pp. 141–188, Feb. 2010, doi: 10.1613/jair.2934.
- [44] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient Estimation of Word Representations in Vector Space." arXiv, Sep. 06, 2013. Accessed: May 05, 2023. [Online]. Available: <http://arxiv.org/abs/1301.3781>
- [45] K. Bhatia, S. Mishra, and A. Sharma, "Clustering Glossary Terms Extracted from Large-Sized Software Requirements using FastText," in *Proceedings of the 13th Innovations in Software Engineering Conference on Formerly known as India Software Engineering Conference*, Jabalpur India: ACM, Feb. 2020, pp. 1–11. doi: 10.1145/3385032.3385039.
- [46] F. Dalpiaz and N. Niu, "Requirements Engineering in the Days of Artificial Intelligence," *IEEE Softw.*, vol. 37, no. 4, pp. 7–10, Jul. 2020, doi: 10.1109/MS.2020.2986047.
- [47] B. Wang, A. Wang, F. Chen, Y. Wang, and C.-C. J. Kuo, "Evaluating word embedding models: methods and experimental results," *APSIPA Trans. Signal Inf. Process.*, vol. 8, no. 1, 2019, doi: 10.1017/ATSIP.2019.12.
- [48] P. Vora, M. Khara, and K. Kelkar, "Classification of Tweets based on Emotions using Word Embedding and Random Forest Classifiers," *Int. J. Comput. Appl.*, vol. 178, no. 3, pp. 1–7, Nov. 2017, doi: 10.5120/ijca2017915773.
- [49] S. Mishra and A. Sharma, "A Generalized Semantic Filter for Glossary Term Extraction from Large-Sized Software Requirements," in *14th Innovations in Software*

Engineering Conference (formerly known as India Software Engineering Conference), Bhubaneswar, Odisha India: ACM, Feb. 2021, pp. 1–9. doi: 10.1145/3452383.3452387.

[50] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information.” arXiv, Jun. 19, 2017. Accessed: May 02, 2023. [Online]. Available: <http://arxiv.org/abs/1607.04606>

[51] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov, “Bag of Tricks for Efficient Text Classification.” arXiv, Aug. 09, 2016. Accessed: May 05, 2023. [Online]. Available: <http://arxiv.org/abs/1607.01759>

[52] “Word representations · fastText.” <https://fasttext.cc/index.html> (accessed May 17, 2023).

[53] C. D. Manning, “Part-of-Speech Tagging from 97% to 100%: Is It Time for Some Linguistics?,” in *Computational Linguistics and Intelligent Text Processing*, A. F. Gelbukh, Ed., in Lecture Notes in Computer Science, vol. 6608. Berlin, Heidelberg: Springer Berlin Heidelberg, 2011, pp. 171–189. doi: 10.1007/978-3-642-19400-9_14.

[54] “GitHub - isti-fmt-nemis/Domain-specific-ambiguity: Tool for comparing terms of different domains.” <https://github.com/isti-fmt-nemis/Domain-specific-ambiguity/tree/master> (accessed May 14, 2023).

