International Journal of
INTELLIGENT SYSTEMS AND APPLICATIONS IN ENGINEERING
ISSN:2147-6799 www.ijisae.org Original Research Paper

# A Critical Study of Pragmatic Ambiguity Detection in Natural Language Requirements

**Ms. Reena S. Satpute, Dr. Avinash Agrawal**

**Submitted**: 02/11/2022 **Accepted**: 01/02/2023

**Abstract:** An approach for pragmatic ambiguity detection in natural language requirements is presented in this paper. Pragmatic ambiguities are determined by the requirements' context, which includes the reader's background knowledge. Readers with different backgrounds may interpret requirements differently. To determine whether a requirement is ambiguous or not, various pragmatic interpretations are compared. In this paper, we will discuss the significance of pragmatic ambiguity detection in NLRs, applications of NLP, ambiguities in NLP, and pragmatic ambiguities, as well as review various techniques used for identifying and resolving ambiguities in natural language requirements. Our objective is to motivate further research in this field by providing a thorough understanding of the difficulties and opportunities related to pragmatic ambiguity detection in NLRs. The tool might be enhanced in the future to support more file types, like PDF. There is ongoing research in the area of pragmatic ambiguity detection, and new approaches and procedures are constantly being developed. It is likely that improvements in pragmatic ambiguity detection and resolution will come as a result of developments in artificial intelligence, machine learning, and natural language processing in the future. Additionally, the growing accessibility of expansive, varied datasets will make it possible to train more reliable and accurate models. Pragmatic ambiguity detection is likely to become a more crucial tool as the field develops in fields like automated language translation, dialogue systems, and natural language understanding.

**Keywords***: Natural Language Processing, Ambiguity, Pragmatic Ambiguity.*

## 1. Introduction

Language is a channel of communication with the help of which we are able talk, write and read. For instance, we use natural language - more specifically, words - to think, decide, plan, and do other things. The key question, though, is whether we can converse similarly with computers in the age of AI [1][2]. In other words, is it possible for people to speak naturally to computers? Because computers require organized data but human speech is unstructured and frequently confusing in nature, it is difficult for us to create NLP applications. There are several advantages to using natural language in software specifications, including improved stakeholder comprehension and communication [3]. [4]. However, ambiguities introduced by natural language can also result in misinterpretations and mistakes during the software development process. In order to increase the overall quality and correctness of the requirements, pragmatic ambiguity detection in NLRs aims to identify and clear out ambiguities in NLRs [5]. Multiple readings of a sentence or phrase depending on the context in which it is used are referred to as pragmatic ambiguity. These misunderstandings can be brought about by a number of things, including a lack of knowledge, inconsistent terminology use, and implicit presumptions. Pragmatic ambiguity detection in NLRs is a difficult task since it calls for knowledge of the domain being produced for the system as well as the context in which the requirements are written [6]. It is impossible to emphasize the significance of pragmatic ambiguity detection in NLRs. The software development process might be delayed and expensive errors can occur as a result of ambiguous specifications. They might also lead to the formation of a system that fails to satisfy the requirements of its users. Therefore, identifying and resolving ambiguities in NLRs is essential to guaranteeing the effectiveness and efficiency of the software development process [7]. The steps in pragmatic ambiguity detection is given in fig 1 can be summarized as follows:

- Identify potential ambiguities: The first step is to identify potential ambiguities in the natural language requirements. This can be done by using natural language processing (NLP) tools to automatically identify and classify ambiguities, or by having domain experts review and validate the requirements.
- Analyze the context: Once potential ambiguities are identified, the next step is to analyze the context in

*Assistant Professor Datta Meghe Institute of Higher Education & Research, Sawangi (M)*
*Email – reenasatpute2017@gmail.com*
*Associate Professor & Dean Shri Ramdeobaba College of Engineering and Management, Nagpur*
*Email – agrawalaj@rknec.edu*

which they appear. This includes understanding the domain knowledge of the system being developed and the intended meaning of the requirements.

- Resolve the ambiguities: After analyzing the context, the ambiguities can be resolved by clarifying the requirements and ensuring that they are unambiguous. This may involve rephrasing, adding additional information, or removing implicit assumptions.
- Validate the resolution: The final step is to validate that the resolved ambiguities are accurate and that they meet the needs of the stakeholders. This may involve testing the requirements with users or conducting a formal review by domain experts.
- Document the resolution: After validating, it's important to document the resolution of the ambiguities, the reason why it was resolved that way and who was involved in the process.


**Fig. 1.** Pragmatic Ambiguities Detection Steps

The significance of pragmatic ambiguity detection in NLRs, applications of NLP, ambiguities in NLP, pragmatic ambiguities, and a review of several strategies used for finding and resolving ambiguities in natural language requirements are all covered in this paper. Our objective is to further this field of research by providing a thorough knowledge of the challenges and opportunities related with pragmatic ambiguity detection in NLRs.

## 2. Overview of NLP

Natural language processing (NLP), which can describe and interpret language and communication digitally, has attracted a lot of attention lately. Machine translation, spam email recognition, information retrieval and summarizing, in addition to healthcare and inquiry answering, are just a few of the many functions it presently serves [8]. The study of how we may use machines to understand and alter natural language for further investigation is known as natural language processing, or NLP. Numerous computer methods for the automatic analysis and interpretation of human language are included in NLP. In NLP, the basic words of language play a significant role. Examples of some basic words include the following: bad, partly, old, magnificent, extremely, etc. A composite is a group of these essential words. [9] [10]. Examples of several composite words are young guy, not really startled, and very nice movie. Simple terminology for atomic and composite terms are words and phrases, respectively. Language's fundamental building component is the word. Words make up human languages, whether they are spoken or written. One of the first stages toward understanding the language is the NLP word-level techniques. The effectiveness of NLP systems, such as automated question answering, information retrieval, and machine translation, relies on the text's intended meaning. Ambiguity, or ambiguous or open meaning depending on use context, is the biggest difficulty. NLP are used in number of applications as presented in fig 2 as presented in [11]-[17].

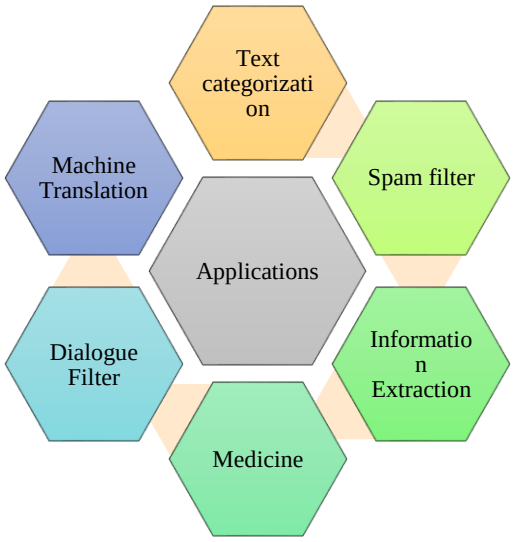


**Fig. 2.** Application of NLP

**3.**

## 4. Ambiguities in NLP

Language is a method of communication with the help of which we can speak, read and write. For example, we think, we use natural language—more specifically, words—to make choices, plans, and other things. The key issue, however, is whether humans can converse similarly with machines in the age of AI. In other words, is it possible for people to speak naturally to computers? Because computers need organized data but human speech is unstructured and often confusing in nature, it is difficult for humans to create NLP applications [18]. This makes it possible to define Natural Language Processing (NLP) as the area of computer science, particularly Artificial Intelligence (AI) that deals with teaching computers how to comprehend and use human language. According to technical definitions, the primary function of NLP would be to train computers to process and analyze vast amounts of natural language data [19]. One of the main problems in NLP is ambiguity. We take into account a number of various factors when attempting to interpret the meaning of a term, including the context in which it is used, our own worldview, and the way a word is typically used in society. Words may imply various things in different contexts and can also alter their meaning over time. Homographs, which are two words with the same spelling but distinct etymologies, and polysemy, which is one word with many meanings, are examples of this phenomena. Ambiguity is a term often used in natural language processing and is defined as the capacity to be interpreted in several contexts. Uncertainty is the capacity to be comprehended in more than one manner, to put it simply. Language in general is highly pragmatic. Fig 3 represents the ambiguities in NLP.

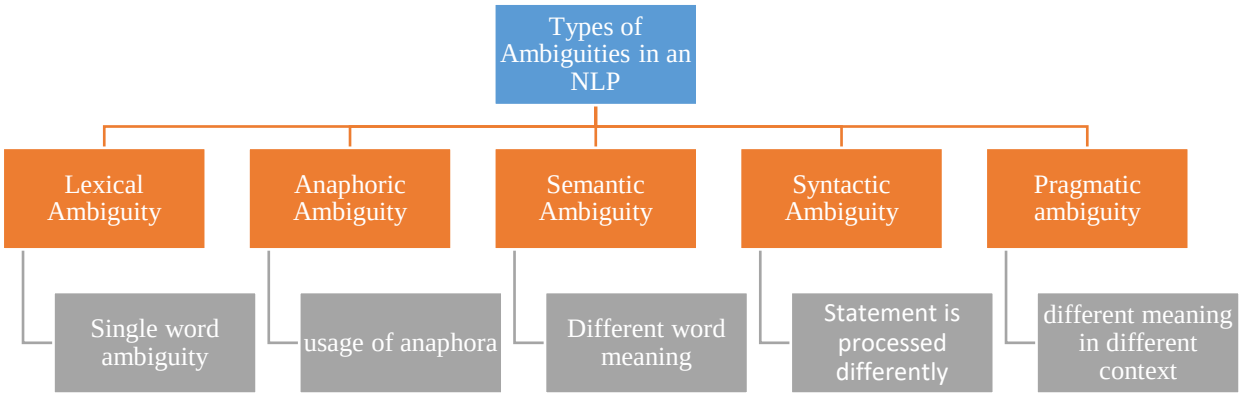


**Fig. 3.** Types of Ambiguities in an NLP

Ambiguity in Natural Language Processing can be removed using:

- Word Sense Disambiguation
- Part of Speech Tagger
- HMM (Hidden Markov Model) Tagger
- Hybrid combination of taggers with machine learning techniques.

In order to clarify the intended meanings of words (word senses) in a certain context, word sense disambiguation (WSD) is used. WSD classifies an occurrence of a word in context into one or more of its sense classes for a particular word and its potential meanings [20]. The characteristics of the context, such as words nearby, serve as the basis for categorization. Concepts of Part of Speech (POS) tagging [21] and the use of probabilistic grammars in parsing are examples of statistical strategies for ambiguity resolution. Large corpora of word use data, such as the Brown Corpus, WordNet, and SentiWordNet, have been used to develop statistical techniques for reducing sentence ambiguity. The POS, grammatical structure, and frequency of use for words and sentences obtained from a large sample of written language are pre-tagged on the words and sentences in these various corpora. The process of choosing the most probable part of speech from among the potential meanings for each word in a phrase is known as POS tagging. The notion of grammar rules and rules connected to part-of-speech tags are explored by probabilistic grammar [22]. Each rule in a probabilistic language is assigned a probability depending on how often it is used in the corpus. When a sentence is syntactically confusing, this method helps choose the optimal option (that is, there are more than one different parse trees for the text).

Methods for using a knowledge dictionary are proposed by Kato et al. [23] recommended by analyzing requirement specifications and supporting documentation for transitive verbs and their objects. TaskLint is a system that was developed by Manam et al. [24], and its purpose is to automatically discover errors in task instructions. TaskLint is able to recognize words and sentences that may indicate uncertainty among workers by making use of a wide variety of previously developed NLP

technologies. This is similar to the way that code analysis tools ("linters") detect probable traits in code that might signal the presence of flaws. In our evaluation of TaskLint, we used task instructions that were generated by inexperienced users. This confirmed that static tools have the ability to increase task clarity and the correctness of outputs, but it also brought to light a number of difficulties. In order to deal with anaphoric ambiguity in requirements, Ezzini et al.[25] created a comprehensive automated solution that takes into account both ambiguity detection and anaphora interpretation. When it comes to the problem of ambiguity identification, supervised machine learning performs better than both a large-scale language model known as SpanBERT and a solution that is put together using off-the-shelf NLP coreference re-solvers. Roop et al. [26] presented a supervised neural network model that makes use of numerous approaches in order to achieve the highest possible accuracy in sense detection. Kaddoura et al. [27] gave a comprehensive assessment of research works that seeks to resolve Arabic word meaning disambiguation using the AWSD datasets that are already in existence. Saxena et al. [28] compiled a list of SMT systems for five different translation tasks, using English and Indian languages as the test dataset and evaluating the systems using the BLEU and METEOR criteria. The investigations were conducted on the En-Kn, En-Ta, En-Te tasks, En-Hi, and they demonstrated an increase in BLEU points by 2.68, 0.78, 2.32, 2.3, respectively, and an increase in METEOR points by 1.34, 0.72, 0.693,1.07, respectively, over the baseline model. Yadav et al. [29] provided a comprehensive analysis of a variety of disambiguation techniques in their presentation. Certain tools are still in the process of being developed, and it is possible that in the future they will be able to eliminate ambiguity. Formal language is required to be used when writing a requirement document so that there is no room for ambiguity. However, consumers do not appreciate formal language because it is not as clear and is more difficult to comprehend. Handling ambiguity in the requirements is addressed by Ezzini et al. [30] proposed automated technique, which makes use of natural language processing. Our strategy involves using Wikipedia to automatically generate a corpus of content that is specific to a given domain. When domain knowledge is integrated, the accuracy of ambiguity detection and interpretation sees a large improvement, both of which are favorable. Our work is focused on coordination ambiguity, sometimes known as CA, as well as prepositional-phrase attachment ambiguity (PAA). Osama et al. [31] presented an effective and versatile automatic syntactic ambiguity detection solution for NL needs. The proposed method involves sifting through all of the many scoring interpretations that can be produced for a particular sentence by using the Core NLP library. When we tested our method on a collection of datasets including 126 requirements, we were able to achieve an average precision of 65% and recall of 99%. Mishra et al. [32] measured the ambiguity potential of the words that are most commonly used in computer science when those words were employed in various subdomains of engineering, such as civil, petroleum, biomedical, and environmental engineering, aerospace etc. This research makes use of a natural language processing method called word embedding in order to identify ambiguous terms. In Ferrari et al. [33] evaluated the degree to which natural language processing (NLP) may be practically applied to the task of finding errors in the requirements papers of a railway signaling manufacturer. In order to search for frequently ambiguous terms, the SREE tool was utilized, and it was applied to the criteria. Based on the results of the trials, it appears that SREE and our patterns may play roles in the detection of requirements deficiencies that are complementary to one another. Below table 1 presents the summary of the research contributions presented above:

**Table 1.** Research Contribution for Ambiguity Detection in NLP

| **Ref** | **Year** | **Methodology** | **Technologies** | **Ambiguity** | **Results** |
|---|---|---|---|---|---|
| [23] | 2022 | Knowledge Dictionary | - | Pragmatic | Recall 99% |
| [24] | 2018 | TaskLint | - | - | - |
| [25] | 2022 | BERT | - | Anaphoric | Precision 60% |
| [26] | 2022 | Artificial Neural Network | - | Grammar | - |
| [27] | 2022 | Word Segmentation Based Dictionary | - | Grammar | - |
| [28] | 2022 | Unsupervised Learning | - | Grammar | - |

| | | | | | |
|---|---|---|---|---|---|
| [29] | 2021 | Controlled Language | POS Tagging | Lexical | Recall 80.12% and Precision 85.76% |
| | | Knowledge Based & Ontology | Stanford | Lexical | Recall 92.85% Precision 92.85% |
| | | Knowledge based & Ontology | WordNet | Lexical | Precision 83.4% |
| [30] | 2021 | Domain-Specific Corpora | - | Domain | Precision 80% recall 89% |
| [31] | 2020 | Filtering pipelines | Stanford | Syntactic | Recall 99% precision 65% |
| [32] | 2019 | | Word embedding | Domain | - |
| [33] | 2018 | NLP | Word embedding | - | Precision 83% and recall 85% |

## 5. Pragmatic Ambiguity Detection

Whenever a statement is pragmatic ambiguous and the context lacks the details necessary to make it clear, pragmatic ambiguity results. An example of pragmatic ambiguity is presented in table 2. The lack of data necessitates inference. When a statement is read or written sarcastically and the context lacks the details necessary to make it clear, pragmatic ambiguity results [34]. A requirement has a pragmatic ambiguity, roughly speaking, if various readers understand it differently, depending on the context of the demand. The data of a requirements encompasses both the requirements of the particular file that have an impact on how well the demand is understood as well as the reader's prior knowledge, which gives the ideas described in the requirement meaning.

A key component of natural language processing is pragmatic ambiguity identification, which aids in deciphering the intended meaning of a sentence or phrase. Language ambiguities may come from a variety of places, including homonyms, synonyms, and idiomatic expressions [35]. It is challenging for robots to comprehend and interpret human language because of these ambiguities, which may cause misunderstanding and confusion. We may increase the precision and dependability of natural language processing systems by spotting and resolving pragmatic ambiguities. This is necessary for a number of applications, including conversation systems, automatic language translation, and natural language comprehension. Additionally, pragmatic ambiguity detection may assist natural language based systems become more effective and efficient by enhancing their overall usability and user experience [36].

**Table 2.** Meaning of Pragmatic Ambiguities

| **Sentence** | **Direct Meaning** | **Other meaning** |
|---|---|---|
| Do you know what time it is? | requesting the time | angering someone for not doing anything in a timely manner |
| Will you crack open the door? I'm getting hot | For breaking the door | For door opening |
| He chicken is ready to eat the breakfast | Small hen is ready to have its food | The prepared chicken is ready to eat |

It's crucial to remember that pragmatic ambiguity detection is a continuous process that must be carried out at every stage of software development to guarantee that the specifications are accurate and clear. For pragmatic ambiguity detection in NLRs, various methods are employed. The most cutting-edge method uses artificial intelligence in tools for natural language processing (NLP) [37]. These instruments are capable of automatically locating and categorizing ambiguities in NLRs. Utilizing subject-matter experts to examine and

confirm the requirements is another tactic. To find and resolve ambiguities in the requirements, this method depends on the domain experts' knowledge. There are several advantages to detecting and resolving pragmatic ambiguities in natural language requirements (as presented in fig 4), some of them are:

1) Improved understanding: By identifying and resolving ambiguities, stakeholders have a clearer understanding of the requirements and what the system being developed is supposed to do.
2) Reduced errors: Ambiguities in requirements can lead to misunderstandings and errors in the software development process. By detecting and resolving ambiguities, the likelihood of these errors is greatly reduced.
3) Increased efficiency: Detecting and resolving ambiguities in the requirements allows for a more efficient software development process. This is because ambiguities can cause delays and rework if they are not identified and resolved early on.
4) Better communication: Pragmatic ambiguities detection helps in better communication between stakeholders, and ensures that the requirements are clearly understood by all parties involved in the development process.
5) Improved user satisfaction: By ensuring that the requirements are clear and accurate, the likelihood of developing a system that meets the needs of its users is increased, which leads to improved user satisfaction.
6) Reduced costs: By detecting and resolving ambiguities early on in the development process, the costs associated with rework and delays can be reduced, which can lead to significant cost savings for the project.
7) Better traceability: By documenting the resolution of the ambiguities, it's easier to track any changes in the requirements and to understand the reason why certain decisions were made

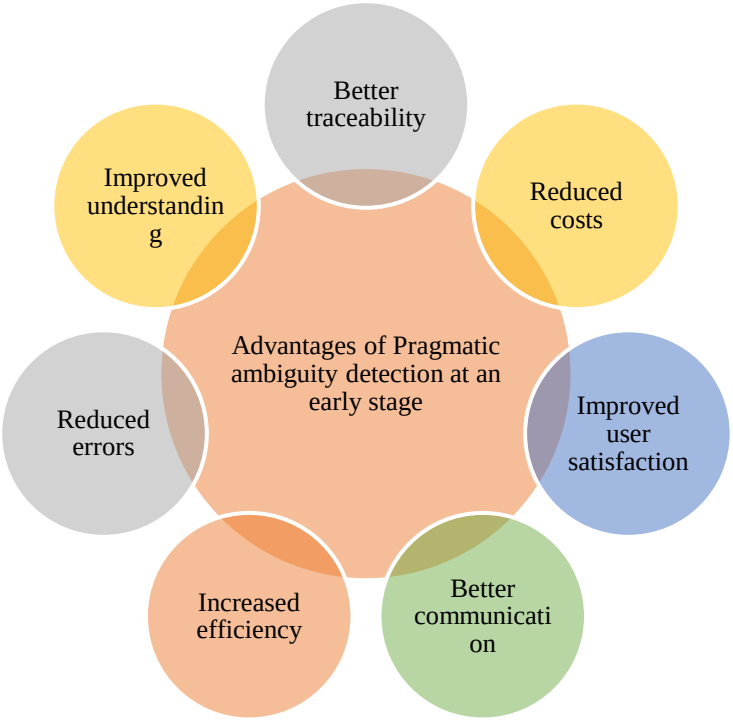


**Fig. 4.** Advantages of Pragmatic Ambiguity Detection

In [39] because natural language is often confusing, so are software needs. Errors in software development are caused by the requirements' ambiguity. As a consequence, several methods and procedures for identifying ambiguity in software requirements have been developed. In order to identify linguistic ambiguity in software requirements, this research employed three supervised machine learning techniques. With an accuracy of 86.66%, RF was the most accurate, followed by SVM and KNN with 80 and 70 percent respectively. In [40], author provided a method for locating these ambiguities depends on the concept of bidirectional encoder called BERT representations from and clustering. This method gives a list of terms that are most comparable to each situation in which a phrase has been used in the document, as well as some sample sentences from the corpus that illustrate the term's context-specific meaning. Uses a corpus that is specialized to computer science (CS) and a multi-domain corpus that contains material from eight distinct application areas. Our test results demonstrate how successful this technique is in locating and identifying intra-domain ambiguities. In [41] author provided an effective and adaptable automated method for NL requirements syntactic ambiguity identification. The

suggested method is based on filtering the Stanford Core NLP library's potential scored interpretations of a given text. Additionally, it gives the user feedback along with potential accurate interpretations in order to clear up any misunderstanding. In order to offer the most probable right interpretations of a need. Tested proposed methodology using a collection of datasets with 126 criteria, and on average were able to obtain 65% accuracy and 99% recall. For locating and recognizing ambiguous terms in a requirement, word embedding was utilized by [42]. The research then use connected data to resolve ambiguity in words. In order to assess the performance of the suggested tool, open-source software specification documents were used to compare human detection and correction capabilities. The results show that, in comparison to the suggested tool, humans struggle to identify and correct ambiguity in software requirement specifications (SRS) that span multiple domains. The outcome demonstrates that, in comparison to humans, the created tool was accurate in identifying and resolving pragmatic ambiguities. In [43], author explored the possibility of discovering near-synonyms rapidly by combining natural language processing and human analytical ability. The results point to a manual examination combined with an enhanced version of our technology as the most efficient strategy. Accuracy is 66%. The goal of [44] is to locate and find ambiguous terms in natural language text that are peculiar to a certain area. This study uses a word embedding-based NLP approach to identify ambiguous terms. More precisely, when terminology from computer science (CS) are employed in other application fields or engineering subdomains. Proposed thorough and in-depth tests on several different subdomains demonstrate that word embedding-based approaches are particularly good at spotting ambiguities that are peculiar to a certain domain. Additionally, our studies show that this method may be used with papers of various sizes. Minimum and maximum similarity scores are both 16.6%. Finally, provide suggestions for more study. In [45] author suggested delving deeper to pragmatic and socio-pragmatic levels of analysis to clear up ambiguity and prevent erroneous interpretations of texts and social media postings, particularly in the sub-tasks of identifying hate speech, in order to avoid making mistakes.

## 6. Current Research Challenges and Future Research Directions

Despite a lot of recent effort, the applications of NLP have been expanding daily, and with that growth come new obstacles. Because people use a variety of words to convey the same idea, there are also challenges associated with dealing with synonyms in language. Designing models to handle ironic and sarcastic statements is a highly difficult problem in natural language processing (NLP), since ironic and sarcastic sentences may sometimes be received in the opposite manner by humans. Additionally, statements in the language that are ambiguous in the sense that they may be understood in several ways are another area that needs improvement so that greater accuracy can be attained. In reality, these kinds of problems also arise when dealing with several fields, for example, when phrases or sentences may have one meaning in the education sector but another in the fields of health, law, or defense. Therefore, although NLP models may be effective for a specific topic or region, these difficulties must be overcome for widespread application. In addition to the difficulties mentioned above, misspelled or improperly used words can also cause issues. Although autocorrect and grammar checkers have greatly advanced as a result of ongoing research, determining the writer's intent by taking into account sarcasm, expressions, informal language, etc. is still a very difficult task. There is no denying that NLP models for the majority of commonly spoken languages have been performing well and developing daily, but models for all people rather than specialized knowledge of a particular language and technology are still needed. Although there has been much discussion of ambiguity detection recommendations in the literature, this process has not yet been automated. One of the objectives of the proposed work is the automation of ambiguity detection. To be more precise, the methodology and tool described in this paper have three objectives: (1) automate ambiguity detection; (2) convince the analyst that the ambiguities it has found are real issues with the document under study; and (3) inform the analyst by outlining the origins of the ambiguities it has found. [Principal Concepts/Outcomes] In that it finds four times as many real ambiguities as the typical human analyst, the provided method offers trustworthy ambiguity identification. The program also provides very accurate ambiguity identification and does not overburden the human analyst with false positives. The instrument that is being used may both identify ambiguities and explain their causes. As a result, in addition to ambiguity identification, it may also be used to train analysts. Additionally, it offers a major opportunity for time and money savings while also improving the quality of industrial requirements engineering. Some of the challenges observed throughout the study are summarized in fig 5.

There are several challenges faced when developing a pragmatic ambiguity detector:

1. Natural language understanding: One of the main challenges is understanding the meaning of natural language text, as it can be affected by factors such as idiomatic expressions, synonyms, and antonyms, making the detection of ambiguities a complex task.

2. Domain-specific knowledge: Another challenge is the need for domain-specific knowledge to accurately detect and resolve ambiguities. This can be difficult to obtain and may require the involvement of domain experts.
3. Inconsistency: The use of natural language in requirements can lead to inconsistencies and variations in the way that similar concepts are expressed. This can make it difficult for a detector to accurately identify ambiguities.
4. Annotation: Annotating a large dataset of natural language requirements for training a detector is a challenging and time-consuming task.
5. Evaluation: It can be difficult to evaluate the performance of a pragmatic ambiguity detector, as it may require a human expert to review and validate the results.
6. Scalability: The detector should be able to handle large amount of data and to analyze it in a short period of time, otherwise it's not useful for real-world scenarios.
7. Adaptability: The detector should be able to adapt to different types of requirements, written in different languages and from different domains, without the need for extensive retraining.
8. False positives: The detector may produce false positives, i.e, it may identify certain parts of the requirements as ambiguous when they are not. This can lead to additional work and can be frustrating for stakeholders.

To overcome these challenges, it's essential to develop a detector that uses both NLP techniques and domain-specific knowledge, that is well-trained, and that can be evaluated and improved over time.

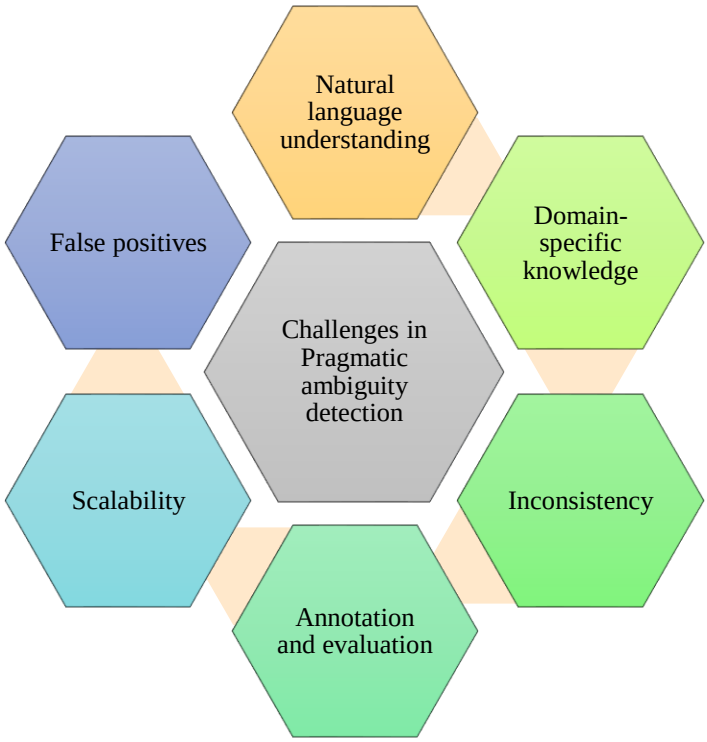


**Fig. 5.** Challenges in Pragmatic Ambiguity Detection

## 7. Conclusion

The software development process gains a lot from natural language requirements, but they also include ambiguities that might cause misunderstandings and mistakes. To ensure the overall quality and correctness of the requirements, pragmatic ambiguity identification in natural language requirements is essential. In this research, we have evaluated numerous methods for detecting and resolving pragmatic ambiguities in natural language requirements and we have addressed the significance of pragmatic ambiguity detection in NLRs. Our objective is to motivate additional study in this field by providing a thorough knowledge of the difficulties and possibilities related to pragmatic ambiguity identification in NLRs. This research aims to identify and resolve ambiguities in software requirements. A remark that is ambiguous has more than one meaning. Semantic, syntactic, lexical, syntax, and pragmatic ambiguities are the most prevalent kinds of ambiguity. Therefore, the goal of this research is to assess how ambiguous common computer science terms like "system," "application," and "database" are when used to various contexts. The outcome demonstrates that, in comparison to humans, the

created tool was accurate in identifying and resolving pragmatic issues. Research in the area of pragmatic ambiguity identification is advancing quickly. Future improvements in technology, such as those made in NLP, ML, and AI, will probably result in more effective methods for locating and resolving ambiguities. Additionally, the utilization of extensive and varied datasets will result in the creation of models that are more accurate and trustworthy. Pragmatic ambiguity detection will be used increasingly often in conversation systems, automated language translation, and natural language comprehension as the area develops. The utility could be enhanced in the future to support new file types, such PDF. The program does not take into account other forms of ambiguity, such as semantic, syntactic, lexical, and syntax, and it can only identify one type of ambiguity. Future updates to the tool will allow for the addition of the additional ambiguities.

## References


[1] Ferrari, G. Lipari, S. Gnesi and G. O. Spagnolo, "Pragmatic ambiguity detection in natural language requirements," 2014 IEEE 1st International Workshop on Artificial Intelligence for Requirements Engineering (AIRE), Karlskrona, Sweden, 2014, pp. 1-8, doi: 10.1109/AIRE.2014.6894849.

[2] Mishra, S., & Sharma, A. (2019). On the use of word embeddings for identifying domain specific ambiguities in requirements. Paper presented at the 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW).

[3] M. Q. Riaz, W. H. Butt and S. Rehman, "Automatic Detection of Ambiguous Software Requirements: An Insight," 2019 5th International Conference on Information Management (ICIM), Cambridge, UK, 2019, pp. 1-6, doi: 10.1109/INFOMAN.2019.8714682.

[4] Sabriye, A. O. J. a., & Zainon, W. M. N. W. (2017). A framework for detecting ambiguity in software requirement specification. Paper presented at the 2017 8th International Conference on Information Technology (ICIT).

[5] Ali, S. W., Ahmed, Q. A., & Shafi, I. (2018). Process to enhance the quality of software requirement specification document. Paper presented at the 2018 International Conference on Engineering and Emerging Technologies (ICEET).

[6] D. Kokane, S. D. Babar and P. N. Mahalle, "Word Sense Disambiguation for Large Documents Using Neural Network Model," 2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT), Kharagpur, India, 2021, pp. 1-5, doi: 10.1109/ICCCNT51525.2021.9580101.

[7] Riaz, M. Q., Butt, W. H., & Rehman, S. (2019). Automatic detection of ambiguous software requirements: An insight. Paper presented at the 2019 5th International Conference on Information Management (ICIM).

[8] Nazir, F., Butt, W.H., Anwar, M.W., Khan Khattak, M.A. (2017). The Applications of Natural Language Processing (NLP) for Software Requirement Engineering - A Systematic Literature Review. In: Kim, K., Joukov, N. (eds) Information Science and Applications 2017. ICISA 2017. Lecture Notes in Electrical Engineering, vol 424. Springer, Singapore. https://doi.org/10.1007/978-981-10-4154-9_56

[9] Mathews, S.M. (2019). Explainable Artificial Intelligence Applications in NLP, Biomedical, and Malware Classification: A Literature Review. In: Arai, K., Bhatia, R., Kapoor, S. (eds) Intelligent Computing. CompCom 2019. Advances in Intelligent Systems and Computing, vol 998. Springer, Cham. https://doi.org/10.1007/978-3-030-22868-2_90

[10] Shankar, V., Parsana, S. An overview and empirical comparison of natural language processing (NLP) models and an introduction to and empirical application of autoencoder models in marketing. J. of the Acad. Mark. Sci. 50, 1324–1350 (2022). https://doi.org/10.1007/s11747-022-00840-3

[11] Yue Kang, Zhao Cai, Chee-Wee Tan, Qian Huang & Hefu Liu (2020) Natural language processing (NLP) in management research: A literature review, Journal of Management Analytics, 7:2, 139-172, DOI: 10.1080/23270012.2020.1756939

[12] Dhar, A., Mukherjee, H., Dash, N.S. et al. Text categorization: past and present. Artif Intell Rev 54, 3007–3054 (2021). https://doi.org/10.1007/s10462-020-09919-1

[13] P. Garg and N. Girdhar, "A Systematic Review on Spam Filtering Techniques based on Natural Language Processing Framework," 2021 11th International Conference on Cloud Computing, Data Science & Engineering (Confluence), Noida, India, 2021, pp. 30-35, doi: 10.1109/Confluence51648.2021.9377042.

[14] S. Alves, J. Costa and J. Bernardino, "Information Extraction Applications for Clinical Trials: A Survey," 2019 14th Iberian Conference on Information Systems and Technologies (CISTI), Coimbra, Portugal, 2019, pp. 1-6, doi: 10.23919/CISTI.2019.8760639.

[15] Abualigah, L., Bashabsheh, M.Q., Alabool, H., Shehab, M. (2020). Text Summarization: A Brief Review. In: Abd Elaziz, M., Al-qaness, M., Ewees, A., Dahou, A. (eds) Recent Advances in NLP: The Case of Arabic Language. Studies in Computational

Intelligence, vol 874. Springer, Cham. https://doi.org/10.1007/978-3-030-34614-0_1

[16] Ni, J., Young, T., Pandelea, V. et al. Recent advances in deep learning based dialogue systems: a systematic survey. Artif Intell Rev (2022). https://doi.org/10.1007/s10462-022-10248-8

[17] Zhou, G. Yang, Z. Shi and S. Ma, "Natural Language Processing for Smart Healthcare," in IEEE Reviews in Biomedical Engineering, 2022, doi: 10.1109/RBME.2022.3210270.

[18] M. Bano, "Addressing the challenges of requirements ambiguity: A review of empirical literature," 2015 IEEE Fifth International Workshop on Empirical Requirements Engineering (EmpiRE), Ottawa, ON, Canada, 2015, pp. 21-24, doi: 10.1109/EmpiRE.2015.7431303.

[19] Chowdhary, K.R. (2020). Natural Language Processing. In: Fundamentals of Artificial Intelligence. Springer, New Delhi. https://doi.org/10.1007/978-81-322-3972-7_19

[20] R. Sahoo, B. R. Das and B. Kishore Mishra, "Analysis and Implementation of Odia Part of Speech Tagger in recent IoT based devices through Chatbot: A review," 2020 International Conference on Computer Science, Engineering and Applications (ICCSEA), Gunupur, India, 2020, pp. 1-4, doi: 10.1109/ICCSEA49143.2020.9132940.

[21] Das Dawn, D., Shaikh, S.H. & Pal, R.K. A comprehensive review of Bengali word sense disambiguation. Artif Intell Rev 53, 4183–4213 (2020). https://doi.org/10.1007/s10462-019-09790-9

[22] Harish, B.S., Rangan, R.K. A comprehensive survey on Indian regional language processing. SN Appl. Sci. 2, 1204 (2020). https://doi.org/10.1007/s42452-020-2983-x

[23] T. Kato and K. Tsuda, "A Method of Ambiguity Detection in Requirement Specifications by Using a Knowledge Dictionary," Procedia Comput. Sci., vol. 207, pp. 1482–1489, 2022, doi: https://doi.org/10.1016/j.procs.2022.09.205.

[24] Manam, VK Chaithanya, Joseph Divyan Thomas, and Alexander J. Quinn. "TaskLint: Automated Detection of Ambiguities in Task Instructions." Proceedings of the AAAI Conference on Human Computation and Crowdsourcing. Vol. 10. No. 1. 2022.

[25] S. Ezzini, S. Abualhaija, C. Arora and M. Sabetzadeh, "Automated Handling of Anaphoric Ambiguity in Requirements: A Multi-solution Study," 2022 IEEE/ACM 44th International Conference on Software Engineering (ICSE), Pittsburgh, PA, USA, 2022, pp. 187-199, doi: 10.1145/3510003.3510157.

[26] Roopa, H. R., and S. Panneer Arockiaraj. "The Role of Artificial Neural Network in Word Sense Disambiguation (WSD)—A Survey." Rising Threats in Expert Applications and Solutions. Springer, Singapore, 2022. 221-227.

[27] Kaddoura, Sanaa, and Rowanda D. Ahmed. "A comprehensive review on Arabic word sense disambiguation for natural language processing applications." Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (2022): e1447.

[28] Saxena, Shefali, et al. "Improved unsupervised statistical machine translation via unsupervised word sense disambiguation for a low-resource and Indic languages." IETE Journal of Research (2022): 1-11.

[29] Yadav, Apurwa, Aarshil Patel, and Manan Shah. "A comprehensive review on resolving ambiguities in natural language processing." AI Open 2 (2021): 85-92.

[30] S. Ezzini, S. Abualhaija, C. Arora, M. Sabetzadeh and L. C. Briand, "Using Domain-Specific Corpora for Improved Handling of Ambiguity in Requirements," 2021 IEEE/ACM 43rd International Conference on Software Engineering (ICSE), Madrid, ES, 2021, pp. 1485-1497, doi: 10.1109/ICSE43902.2021.00133.

[31] 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," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, SA, Australia, 2020, pp. 651-661, doi: 10.1109/ICSME46990.2020.00067.

[32] Mishra, Siba, and Arpit Sharma. "On the use of word embeddings for identifying domain specific ambiguities in requirements." 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW). IEEE, 2019.

[33] Ferrari, Alessio, et al. "Detecting requirements defects with NLP patterns: an industrial experience in the railway domain." Empirical Software Engineering 23.6 (2018): 3684-3733.

[34] (Walton D. (1996) A Pragmatic Synthesis. In: Fallacies Arising from Ambiguity. Applied Logic Series, vol 1. Springer, Dordrecht).

[35] R. Sonbol, G. Rebdawi and N. Ghneim, "The Use of NLP-Based Text Representation Techniques to Support Requirement Engineering Tasks: A Systematic Mapping Review," in IEEE Access, vol. 10, pp. 62811-62830, 2022, doi: 10.1109/ACCESS.2022.3182372.

[36] Khurana, D., Koli, A., Khatter, K. et al. Natural language processing: state of the art, current trends and challenges. Multimed Tools Appl 82, 3713–3744 (2023). https://doi.org/10.1007/s11042-022-13428-4

[37] Apurwa Yadav, Aarshil Patel, Manan Shah, A comprehensive review on resolving ambiguities in natural language processing, AI Open, Volume 2, 2021, Pages 85-92, ISSN 2666-6510, https://doi.org/10.1016/j.aiopen.2021.05.001.

[38] T. P. Nagarhalli, V. Vaze and N. K. Rana, "Impact of Machine Learning in Natural Language Processing: A Review," 2021 Third International Conference on Intelligent Communication Technologies and Virtual Mobile Networks (ICICV), Tirunelveli, India, 2021, pp. 1529-1534, doi: 10.1109/ICICV50876.2021.9388380.

[39] Oo, K.H. (2023). Comparing Accuracy Between SVM, Random Forest, K-NN Text Classifier Algorithms for Detecting Syntactic Ambiguity in Software Requirements. In: Al-Emran, M., Al-Sharafi, M.A., Shaalan, K. (eds) International Conference on Information Systems and Intelligent Applications. ICISIA 2022. Lecture Notes in Networks and Systems, vol 550. Springer, Cham. https://doi.org/10.1007/978-3-031-16865-9_4

[40] Moharil and A. Sharma, "Identification of Intra-Domain Ambiguity using Transformer-based Machine Learning," 2022 IEEE/ACM 1st International Workshop on Natural Language-Based Software Engineering (NLBSE), Pittsburgh, PA, USA, 2022, pp. 51-58, doi: 10.1145/3528588.3528651.

[41] 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," 2020 IEEE International Conference on Software Maintenance and Evolution (ICSME), Adelaide, SA, Australia, 2020, pp. 651-661, doi: 10.1109/ICSME46990.2020.00067.

[42] Khalid Abdikarim Mohamed, Jamilah Din, & Salmi Baharom. (2022). A Tool to Detect Pragmatic Ambiguity with Possible Interpretations Suggestion in Software Requirement Specifications. International Journal of Synergy in Engineering and Technology, 3(2), 52-60. Retrieved from https://tatiuc.edu.my/ijset/index.php/ijset/article/view/141

[43] Fabiano Dalpiaz, Ivor van der Schalk, Sjaak Brinkkemper, Fatma Başak Aydemir, Garm Lucassen, Detecting terminological ambiguity in user stories: Tool and experimentation, Information and Software Technology, Volume 110, 2019, Pages 3-16, ISSN 0950-5849, https://doi.org/10.1016/j.infsof.2018.12.007.

[44] S. Mishra and A. Sharma, "On the Use of Word Embeddings for Identifying Domain Specific Ambiguities in Requirements," 2019 IEEE 27th International Requirements Engineering Conference Workshops (REW), Jeju, Korea (South), 2019, pp. 234-240, doi: 10.1109/REW.2019.00048.

[45] S. Assem and S. Alansary, "Sentiment Analysis From Subjectivity to (Im)Politeness Detection: Hate Speech From a Socio-Pragmatic Perspective," 2022 20th International Conference on Language Engineering (ESOLEC), Cairo, Egypt, 2022, pp. 19-23, doi: 10.1109/ESOLEC54569.2022.10009298.